應用 Tensorflow 深度學習機制於影像辨識之研究 - 以中共軍機影像辨識為例

傅振華 [1*]　　齊立平 [2]　　莫雅婷 [1]

[1] 國防大學管理學院資訊管理學系

[2] 國家中山科學研究院航空研究所

摘要

本研究主要探討於 TensorFlow 平台，運用現行 Object Detection 演算法 Faster R-CNN 建立深度學習模型，辨識標的為中共常見戰機圖片，並研究所訓練之模型於物件影像辨識中的辨識率，探究可以提升辨識率的機器學習模型。研究發現 Faster R-CNN 演算法搭配 ResNet-101 神經網路層辨識率較搭配 Inception V2 神經網路層高，另 ResNet-101 神經網路層辨識率較 ResNet-50 神經網路層高，最後提高訓練次數，只有對單一機型的辨識率有提升，對多種機型的辨識率則有待討論。

關鍵詞：機器學習、深度學習、TensorFlow、中共戰機辨識

A Study on Image Detection with TensorFlow Deep Learning Mechanism – a Case of P.R.O.C. Fighters Recognition

Fu, Chen-Hua[1*]　　Chi, Li-Pin[2]　　Mo, Ya-Ting[1]

[1]Department of Information Management, College of Management, National Defense University

[2]Aeronautical System Research Division, National Chung-Shan Institute of Science and Technology

Abstract

This study discusses a deep learning model implementation on the TensorFlow platform with the fast r-CNN object detection algorithm. The identification target is the pictures of the P.R.O.C. main fighters. It also studies the recognition rate of the trained model in object image recognition and explores the machine learning model that can improve the recognition rate. It is found that the recognition rate of fast r-CNN algorithm combined with resnet-101 neural network layer is higher than that of concept V2 neural network layer. In addition, the recognition rate of the resnet-101 neural network layer is higher than that of the resnet-50 neural network layer. Finally, improve the training times. Only the recognition rate of a single model is improved, and the recognition rate of multiple models remains to be discussed.

words: Machine Learning, Deep Learning, TensorFlow, P.R.O.C fighters Recognition

一、緣起

時代轉變，人工智慧（Artificial Intelligence, AI）取代人們相關作業的議題逐漸受到大眾的關注，從家庭到社會，從校園到企業，甚至於躍上軍事戰備的舞台。美軍發佈的《2013-2017 年國防部科學技術投資優先項目》，將「從數據到決策」專案列為第一，表明在指揮決策上對於人工智慧應用於大數據技術的重視程度。Ashley, K., & Gordon, T.（2005）認為機器學習是屬於弱人工智慧（Weak AI），它嘗試透過資料中找出其間的關聯性，加以學習，然後建構演算方法（或關聯規則），並利用所建構的演算方法／關聯規則進行預測。Michalski, et al.（2013）將其廣泛定義如下：在電腦學習的過程中，會根據與其有關的資訊分門別類，並在分類的過程中累積經驗，並產生執行分類的方法。電腦系統在持續進行分類的過程當中會運用到其執行方法，並且會根據經驗去不斷修正分類的行為。而在機器學習中，模仿大腦的類神經網路（Artificial neural network）所建構出的數學模型，訓練了多層的神經網路，又稱為深度學習（Deep learning）。深度學習將藉由許多處理層所組成的運算模型來學習多種抽象級別的數據。深度學習大大地改進了語音識別、視覺對象辨識、物體偵測、藥物開發和基因組學等許多其他領域的最新技術。人工智慧、機器學習與深度學習此三者之間的關係如圖 1 所示。

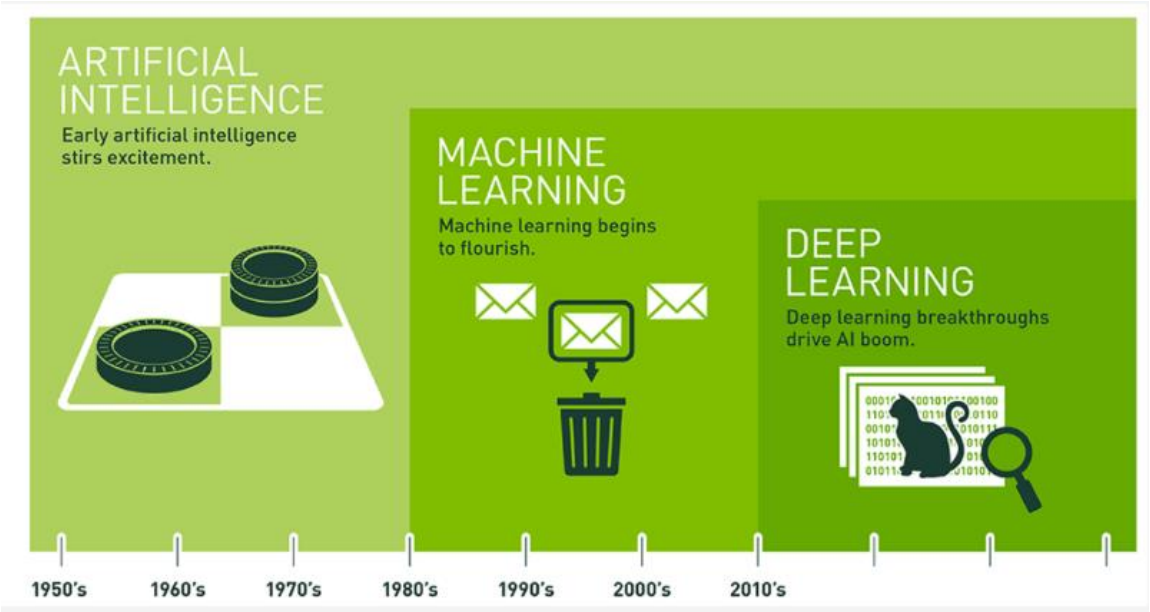


圖 1 人工智慧、機器學習與深度學習的關係

資料來源：林敬恆(2019)

從西元 1980 年到西元 2010 年這三十年間，開發出許多不同層級、架構和初始化設定的類神經網路類型，如卷積類神經網路（Convolutional Neural Network, CNN）、遞歸神經網路（Recurrent Neural Network, RNN）等。其相對應的建置平台也獲得許多重要學者的投入與研究，獲得突破性的進展。TensorFlow 就是目前最知名的機器學習平台，它可以支援許多種神經網路算法（郭利敏，2017）。TensorFlow 是由 Google 研發團隊於西元 2015 年 11 月開發的第二代開放原始碼的機器學習平台，它可以廣泛應用於自然語言處理、語音辨識、影像處理等不同領域。深深受到世界各國深度學習學者的歡迎，Google、Uber、OPenAI、eBay 等大型科技公司的研發團隊均應用它進行深度學習相關的研究。與其他深度學習框架（如：Torch、Caffe、MXnet、Theano 及 Keras 等），Tensorflow 具備下列幾個優點：可透過 Python 程式語言進行開發、提供良好視覺化效果、高運算效能、強大運算功能、高度靈活性。

本研究將以 TensorFlow Object Detection API 所提供的物件影像辨識模型為基礎，運用 TensorFlow 平台建立計算圖，設計張量運算流程，研究模型於物件影像辨識中的辨識率，並建構可以提升辨識率的機器學習模型。藉以探究 TensorFlow 現有物件影像辨識模型應用於軍事飛行器辨識之可行性，並透過相關物件影像辨識模型建構軍事飛行器辨識的機器學習模型；並能建構研究所提軍事飛行器辨識機器學習模型所需的實作環境，期望瞭解應用圖形處理硬體裝置加速機器學習模型訓練與辨識效能；最後能深入探究訓練資料特性對於研究所提軍事飛行器辨識機器學習模型辨識效能之影響。

二、文獻探討

本研究主要探討應用TensorFlow平台於物件影像辨識中的辨識率，發展出一套提升辨識率的影像辨識模型，因此本章節針對本篇研究所應用之相關技術及發展做介紹，以便後續實驗探討。

2.1 人工智慧、機器學習與深度學習

2.1.1 人工智慧

「人工智慧」係由美國科學家John McCarthy於西元1955年提出此一想法與概念，人工智慧主要的目標為了讓電腦能夠具備類似人類具有的能力，如：學習及解決複雜問題、抽象思考、展現創意等能力，能夠進行推理、規劃、學習、交流、感知和操作物體。早年受限於電腦硬體技術的限制，電腦硬體裝置體積十分巨大且電腦硬體計算能力有限，因此，需要高速運算能力的人工智慧，其發展在早年面臨了發展的瓶頸，因而日趨式微。

後來因為電腦硬體效能的突破，高速運算性能與儲存空間的大幅提升，使得人工智慧在沉寂一陣子之後，得以成為資訊技術的重點發展項目之一，並開展「機器學習」這塊新興的人工智慧範疇，並獲得相當不錯的成果；最近數年因資訊技術與相關演算法的持續精進，藉由機器學習的根基，人工智慧又朝向「深度學習」這個領域邁進。

近年來，人工智慧應用的範疇十分地廣泛，最近相當火紅的人工智慧應用： 1.個人語音助理，如：蘋果公司的Siri與微軟公司的Cortana、 2. 打敗人類西洋棋的IBM Deep Blue與 3.勝過人類圍棋高手的Google DeepMind AlphaGo，這些都是近年來人工智慧領域的研究成果。

2.1.2 機器學習

機器學習（Machine Learning, ML）係指經由蒐整以往大量的資料及分析處理經驗中，嘗試搜尋相關事務處理法則，加以學習，進而能夠藉由資訊技術達到具備人工智慧的方法。通常，機器學習包含透過所蒐整的樣本進行訓練，進而能夠發掘其運作模式；機器學習不是依據特定的處理規則撰寫程式。一般而言，機器學習的樣本是可以藉由彙整大量的資料中發現。簡而言之，機器學習將以弱人工智慧（Narrow AI） 的形式呈現，它藉由彙整大量資料，進而能夠找出複雜的處理法則（或是訓練樣本），加以學習，以期能創造處理的演算法（或一組規則），並可以利用它來進行後續的預測。

機器學習可以區分成三大類：監督式學習（Supervised learning）、非監督式學習（Unsupervised learning）及半監督式學習（Semi-supervised learning）。

- 監督式學習是指在機器學習過程中，藉由人工的方式將資料貼上標籤（Label），告訴機器相對應的值，像是告訴機器正確的答案，以協助機器學習於輸出處理時做為評斷誤差的依據（Chapelle, et al.,2009）。這種機器學習方法對於電腦而言相對簡單，但對人們的事先處理作業卻是相對辛苦，但是正確性相對較高。例如：要訓練機器辨識貓和狗，任意選出 100 張貓和狗的照片，然後透過「標註」（Labeling）的方式說明哪些

是貓的照片與哪些是狗的照片，經標註的貓狗照片輸入電腦，讓電腦學習辨識貓及狗的外觀特徵，電腦會依據先前標註的照片去偵測貓和狗的「特徵」（Feature），然後依照所學習的特徵即可辨識出貓和狗，並能夠進行後續的預測工作。

- 非監督式學習是指在機器學習過程中，不以人工方式將資料貼上標籤，所有機器學習的資料都不具特定的標準答案，不能做為機器學習時果輸出時評斷誤差的依據，機器必須透過其他的方式加以判斷尋找答案，因此非監督式學習的預測結果相對地比較不準（Hastie, et al., 2009）。此一機器學習方法因其不必經由人工得事先處裡，進行標註分類，對於人類而言，相對簡單，但是對於電腦的學習處理而言，卻是相對困難，而且此種機器學習結果的誤判可能性較大。舉例來說：隨機挑選出 100 張照片，讓電腦辨識出照片中的貓狗，因沒有事前對照片進行人工判別標註，電腦必須自行嘗試將照片中的特徵加以萃取，然後加以分類處理，以期能辨識出照片中的貓與狗。
- 半監督式學習係將資料區分為二大部份，一部分資料具有標準答案，可做為機器學習輸出時評斷誤差的依據，另一部分的資料則不具備標準答案，電腦必須藉由已具標準答案的資料，評斷出這些資料的答案；因此，半監督式學習是等於監督式學習與非監督式學習優點的結合。舉例來說，隨機選出 100 張照片，將其中的 30 張照片加以標註，說明這 30 張照片中那些是貓的照片及那些是狗的照片，然後讓電腦學習貓與狗的外觀特徵；當電腦獲得貓狗外觀的特徵資訊後，即可對剩餘未經標註的照片進行辨識分類，找出 70 張照片中具有貓與狗的照片。半監督式學習只須經由是前少量的人工分類處理，即能讓機器學習預測的成果較為準確，使目前機器學習最常使用的方式。

2.1.3 深度學習

在以往的機器學習機制中，資料的特徵多經由人為演算法加以萃取出來的，此一方式需經由相關領域專家對標的資料進行多次的研析探究，瞭解標的資料的特殊性質後，方能萃取出有效良好的特徵，此一處理過程即所謂的特徵工程（Feature engineering）（Cheng, et al., 2016）。以往的機器學習演算法會遭遇所謂的天花板效應，亦即當處裡的資料超過某一數量時，其機器學習就會遭遇瓶頸，學習的正確率就無法再提昇。

深度學習此一概念於西元 2006 年由加拿大多倫多大學的 G. E. Hinton 等學者提出，深度學習為機器學習領域的一個分支，屬於機器學習演算法的一種，也是機器學習領域一個新興的研究範疇；深度學習係指一個機器學習過程，其將學習的樣本資料經由相關的訓練方法，藉以獲得一個包括多個階層深度的網路結構；通常，以往的類神經網路會隨機地初始化類神經網路單元的設定權值，此一方式將容易造成類神經網路收斂為局部最小值的結果，為了避免產生是類結果，Hinton 等人提出使用非監督式預先訓練模式，對類神經網路的網路權值進行初始值設定，然後再利用權值微調方式加以優化處理。「深度學習」相對於「淺層學習」方法（如：最大熵方法、支撐向量機（Support Vector Machine, SVM）（Noble, 2006）及 boosting（Schapire, 2003））而言，深度學習模型所學習獲得的非線性處理層級的數量更多，經由原始資料過逐一層級的特徵轉換處理方式，將樣本資料在既有空間的特徵轉換至新的特徵空間中，逐層地自動學習特徵的表示方法，進而能夠更便於後續分類處理或特徵視覺化處理。

深度學習的運作原理系植基於人類大腦的神經網路連接構造，藉由多個階層的類神經網路對各類型資料（如：文字、影像及語音）訊息進行料特徵描述處理，並對所處理的資料進行解釋說明；希望將樣本資料經由多個處理層（Layer）中的線性轉換（Linear transform）或非線性轉換（Non-linear transform），自動萃取出足以呈現資料特性的特徵；同時藉由序列映射模型，將一個一個輸入序列映射到另一個輸出序列，此一序列映射模

型稱之為序列到序列（Sequence-to-sequence）的模型（Nallapati, et al., 2016）。以影像資料為例，人們在處理此類的資料，通常會找出簡單物件的外型，然後逐漸地建構出複雜的物件視覺圖像；深度學習亦是透過此一處理模式，首先經由底層原始特徵資訊逐次地形成高層的抽象特徵及屬性類別，進而顯示資料分層的特徵資訊。

深度學習是一種用於監督式與非監督式特徵表示的演算法，它可用於處理多重線性或非線性轉換的問題，透過多個神經網路處理層，對資料進行相關處理，藉以對資料建立高層抽象模式，達到監督式與非監督式的特徵學習及分層特徵萃取，取代人工取得特徵，因此它具有能夠自動萃取特徵（Feature extraction）的功能，可以被當成是一種特徵學習（Feature learning）。它能在底層對非標籤的資料進行非監督式學習的預先訓練，如此一來能有助於後續監督式學習進行微調（楊宗恩，2017）。

深度學習可以取代專家在樣本資料特徵萃取所耗費的時間，藉由超強的資料自動特徵萃取的能力，深度學習能夠突破以往機器學習應用的瓶頸，進而能夠獲得許多令人驚訝的研究成果。深度學習可成功地克服天花板效應，因此已經在許多領域帶來了顯著的效果，著名的應用包括機器翻譯（Machine translation）、語音識別（Speech recognition）（Chorowski, et al., 2015）及圍棋程式 AlphaGo 等。

甚麼是深度學習？簡而言之，深度學習是一種應用需要整合具有多個深層的類神經網路的學習模型，其中，應用了需多不同類型與層級的類神經網路，如卷積類神經網路、遞歸類神經網路等。因此，某些深度學習架構多被稱為深度類神經網路（Deep Neural Network, DNN）。

2.2 類神經網路、深度類神經網路與卷積類神經網路

2.2.1 類神經網路

類神經網路（Neural Network, NN）是植基於生物神經系統運作方式的數學模型；通常一個類神經網路會有數個層級，而每個層級理會包含數量不一（數十到數百）的神經元（Neuron），每一層級上的神經元會將上個層級神經元的輸出做為其輸入，然後予以加總處理，並經由激活函式的運算處理，加以轉換，做為此一神經元的輸出；此外，在二個不同層級之間的神經元節點，它們之間需透過權值（Weight）做為記憶強度的設定。類神經網路透過一定層級的神經元節點深度運算後，可以獲得一系列的輸出結果，此一輸出結果將與預期結果進行比較，以做為調整類神經網路應用函式與權重值設定的依據，期使類神經網路的運算結果能夠更趨近期望的結果。如此一來，即可將經由學習過程所獲得的類神經網路模型應用於新的資料樣本，進行相關的預測；由於，所獲得的類神經網路模型往往能夠得到近似預期的預測結果，進而能夠達到機器學習的效果與目的。

經由上述得知，類神經網路的輸出主要是依據類神經網路的層級數、各層之間神經元連接方式、權值與激活函式的差異而有不同結果。其中，類神經網路的模型架構會涉及神經網路層級的數量、每一層級中神經元的數量、各個層級中神經元與鄰近層級中神經元連接的方式及激活函式種類的設定等；這些設定參數必須於類神經網路使用之前以人為方式完成，而類神經網路參數設定的適當與否會大大地影響其運作效能，簡而言之，類神經網路即藉由多次的訓練與學習過程，嘗試找出最佳的參數設定組合，進而能發揮類神經網路的最佳功效。

2.2.2 深度類神經網路

深度類神經網路（DNN）係指至少具有一個隱藏層以上的類神經網路；深度類神經網路架構與淺層類神經網路架構相似，深度類神經網路因其較多且具深度網路層級所以能夠針對較為複雜問題，建構非線性的系統模型，透過其所具備多且具深度的網路層級為所建模型提供較高的抽象表達方式，進而能夠提升類神經網路模型的處理能力。相較

於一般淺層類神經網路（NN），深度類神經網路具備較佳的特徵表達和複雜映射建模的能力。通常，深度類神經網路結構中會包含數量龐大的神經元，在深度神經網路中的每個層級中的神經元會與其他層級中的數量眾多的神經元相互連接，在進行學習過程中，連接神經元之間的權值會經由修訂的方式，進而修訂類神經網路的功能；通常，經過深度學習之後，深度類神經網路將會得到符合類神經網路特徵的深度網路結構，此一深度網路結構就是深層類神經網路，即所謂深度類神經網路（DNN）。通常，深度類神經網路係利用許多個單一層級的非線性網路堆疊建構而成的，而深度類神經網路分為以下三種類型（孫志軍等學者，2012）。

1. 前饋式深度類神經網路（Feed-Forward Deep Neural Networks, FFDNN）由多個層級的神經元網路層堆疊而成，常見前饋式深度網路種類包括：多層級感知機（Multi-Layer Perceptrons, MLP）及卷積類神經網路（Convolutional Neural Networks, CNN）等。
2. 反饋式深度類神經網路（Feedback Deep Neural Networks, FBDNN）由多個層級的神經元網路層堆疊而成，常見反饋式深度類神經網路的種類包括：反卷積網路（Deconvolutional Networks, DN）及階層式稀疏編碼網路（Hierarchical Sparse Coding, HSC）等。
3. 雙向式深度類神經網路（Bi-Directional Deep Neural Networks, BDDNN），透過多個層級的神經元網路層及神經元堆疊而成，其中每層神經元網路可以是獨立的編碼或解碼過程，常見的雙向式深度類神經網路類型包括：深度波爾茲曼機（Deep Boltzmann Machines, DBM）、深度信念網路（Deep Belief Networks, DBN）及推疊式自動編碼器（Stacked Auto-Encoders, SAE）等。

一般而言，深度類神經網路多應用前饋式類神經網路，但在與語言相關的建模應用研究，則將其應用於遞迴類神經網路（Mikolov, et al., 2010），另外卷積類神經網路（CNN）於電腦影像與視訊領域獲得成功的應用（Lecun, et al., 1998）；由上所述對於深度類神經網路的應用範疇，可以發現深度類神經網路（DNN） 已成為機器學習領域中一種重要的方法，廣泛應用於許多的領域。

2.2.3 卷積類神經網路

卷積類神經網路多以英文縮寫 CNNs 或 ConvNets 加以表示，它是目前深度類神經網路領域的發展與應用主流，卷積類神經網路主要是由用於資料樣本特徵提取的卷積層和用於資料樣本特徵處理的池化層組成的多層神經網路，典型卷積類神經網路的主要組成份子包括：1.輸入層、2.卷積層、3.池化層（下採樣層）、4.完全連接層及 5.輸出層等五個部分（李彥冬等學者，2016），如圖 2 所示。

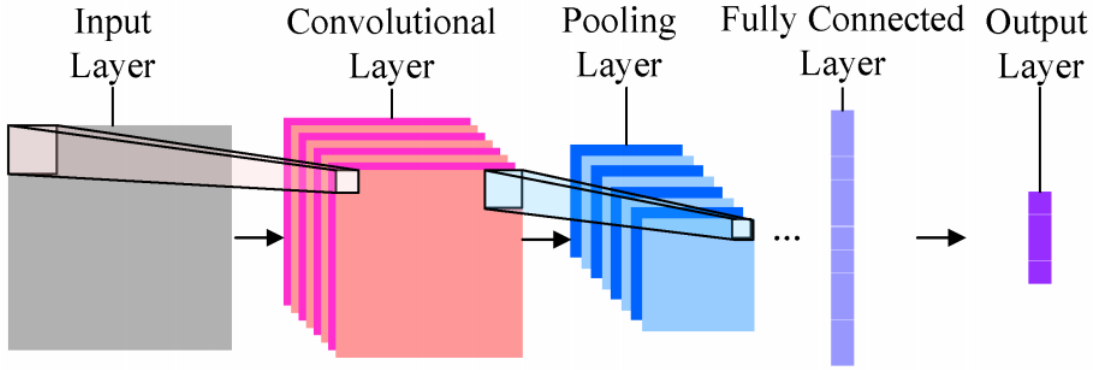


圖 2 卷積類神經網路架構示意圖

資料來源：JYRoy (2020)

影像特徵的萃取與分類一直是人工智慧領域一個基礎且重要的研究方向，卷積類神

經網路提供了端點到端點的機器學習模型，卷積類神經網路透過傳統梯度下降的方式進行模型訓練與相關參數調整；經由相當次數訓練過的卷積類神經網路將能夠學習影像中的特徵，並能加以萃取與分類，以利後續的影像的辨識與分類。卷積類神經網路的特點是網路中每一個層級的網路特徵係經由上一層局部區域的共享權值的卷積核激活（Activate）而得到，此一特點使得卷積類神經網路比其他類神經網路更加適用於影像特徵辨識等相關的應用（李彥冬等學者，2016）。

卷積類神經網路中的卷積層（Convolutional layer）係由若干個特徵影像組合而成，它將藉由將不同的卷積核在輸入影像上滑動並進行一定的運算處理組合而成。通常，每個特徵影像上的全部神經元將共用同一個卷積核所設定參數數值，該卷積核參數設定係經由前一層輸入影像進行卷積運算處理而獲得。卷積核中每一個神經元都有權值參數，與輸入影像相對應的區塊影像元素值進行相乘運算，然後將各項乘積進行加總處理，並透過啟動函數運算得到輸出影像元素值；此一運算處理模式實際等同於在一個神經元上將多個影像輸入信號予以加權處理，然後進行加總運算處理後並啟動輸出的過程（李彥冬等學者，2016）。

卷積類神經網路裡面的池化層（Pooling layer）亦稱為「採樣」層，其運作方式是基於局部相關性原理進行池化採樣處理，期望能藉由此一方式減少資料量的處理運算同時能夠保留有用資訊，池化層實際是一種降階採樣處理，其多透過非線性形式進行處理。池化處理過程係利用採樣函式運算進行處理，而採樣函式則為多種各式各樣的非線性函式，常用的採樣函數包括最大值採樣和平均值採樣。其中，最大值採樣函數輸出值係採用影像區塊中元素的最大值，做為提取局部影像區塊特徵的回應值，通常多用於低層影像區塊特徵的提取；因此，對輸入影像而言，會選取最顯著的值做為特徵值。而均值採樣函數則是計算影像區塊元素的算術平均值做為函數的輸出值，做為提取局部影像區塊特徵的回應值。池化採樣的處理過程與卷積處理過程類似，它們都使用不帶加權參數的採樣函數，從輸入特徵影像的左上角開始按一定順序向右（或向下）滑動，對視窗相應區塊的影像元素進行採樣處理然後輸出其特徵值（李彥冬等學者，2016）。透過池化層持續的運作，卷積類神經網路會持續地降低其所處理資料空間的大小，進而使得參數處理數量及資料處理運算量亦隨之下降，此一處理模式在某種狀況之下也避免因過度的機器學習，造成過度擬合問題的產生。池化層的主要功能是將一張影像或一定數量的影像透過池化處理將其轉換成相對較小的影像，進而能夠得到一樣數量的影像，此一池化處理有助於改善大量資料處理運算耗時的問題。

卷積類神經網路裡面的完全連接層（Fully Connected Layer, FCL），其連接位置通常是位於卷積層與池化層的後面，其連接數量可以為一個或是多個全連接層；卷積類神經網路的完全連接層的結構與一般常見完全連接型態類神經網路的隱藏層構造相同，完全連接層中的所有神經元都會與下一個層級的完全連接層中的全部神經元連接（李彥冬等學者，2016）。

卷積類神經網路結構中的最後層級是輸出層，如果卷積類神經網路是邏輯回歸層時，其輸出層的每個節點將會顯示出輸入影像是屬於某一種類別的機率；一般卷積類神經網路多會應用誤差反向傳播（Error back propagation）演算法進行學習訓練，在學習訓練過程中，卷積層中每個特徵影像的所有神經元都是使用相同的連接權值，透過這樣的連接權值的設定可以大幅減少卷積類神經網路訓練所需要的參數數量（張順等學者，2019）。

2.3 TensorFlow 與物件影像辨識

2.3.1 Tensorflow

TensorFlow 是由 Google 開發的開源機器學習工具。TensorFlow 運算係植基於張量

（Tensor）圖，藉由形象化的方式來表示類神經網路的模型，然後進行相關的計算處理，進而獲得運算成果。TensorFlow 此一命名係源自於其運算方式，張量（Tensor）係指多個維度的數據組合，而 Flow 則是指所要運算處理的數據將依據所律定的順序及相關法則進行的運算流程，此一運算流程可以藉由圖形（Graph）的方式呈現，意即可以利用圖形中的元素－節點（Nodes）與邊（Edges），將資料運算處理流程以一個具方向性的圖形予以表達，清楚地顯示資料張量將從圖形的起點流向終點的情形；其中，圖形中的節點將用來表示處理運算資料所使用的計算函式（包括處理資料的讀寫操作）、資料輸入（Input）的起點及運算結果輸出（Output）的終點，而圖形中的邊則用來表示處理資料的流動方向，說明各個節點之間的輸入／輸出關係，透過邊的設定可以用於動態調整傳送大小不一的多維度資料。故當使用者應用 TensorFlow 建構完成深度學習的模型後，即可透過模型的起點輸入所需處理的資料張量，同時模型中的各個節點將被分配至相關的運算裝置以非同步分散平行處理的方式，執行節點運算功能，資料張量藉由模型中的各個節點的處理後，將可獲得模型的輸出結果。

Google 為 TensorFlow 建構一套完整深度學習的開發環境，可以提供使用者進行深度學習相關演算法與應用的開發研究，TensorFlow 具備提供使用者簡易的使用環境與交互性設計的特性，使得學術研究與產業應用的各類使用者均可藉由其容易使用與交互性設計的開發特性，進行各式各樣有關深度學習的學術研究與產業應用，進而使得使用者能夠獲得其在深度學習方面所需的成果。

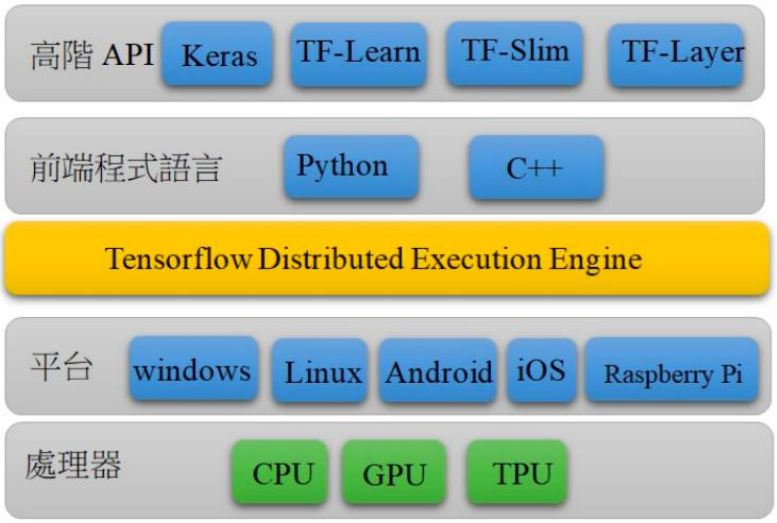


圖 3 TensorFlow 架構圖

資料來源：林大貴(2017)

TensorFlow 吸收先前許多類似處理平台的相關作法與應用優點，透過層級式架構建構其所需的運作環境，如圖 3 所示；TensorFlow 能夠透過多種的硬體裝置執行其運算作業，支援 TensorFlow 運算的硬體裝置包括：中央處理器（CPU）、圖形加速處理器（GPU）及 TensorFlow 運算處理器（TPU），TensorFlow 能夠運用這些硬體裝置的運算功能特性，藉由分散式處理的方式進行深度學習，使其能夠具備深度學習模型建構與運用的能力。TensorFlow 可以透過相關的應用程式介面（API）能夠與其他平台的模型進行構聯，支援其所需的運算處理；此外，TensorFlow 可以支援常用機器學習開發程式語言（如 C++、Python），建構進行深度學習所需的模型。因此本研究採用 TensorFlow 建構可以提升辨識率的機器學習模型，做為深度學習框架。

TensorFlow 雖然可使用多種前端程式語言，但對 Python 的支援是最佳的，且 Python

是一種物件導向且具有豐富函式庫支援的程式語言，其具有程式碼精簡、容易學習及生產力高等特性，應用非常廣泛。在深度學習應用程式介面（API）方面，TensorFlow 是屬於較低階（Zaccone, et al.,2017），所以在設計模型時，必須自行設計：張量乘積、卷積等底層操作，其優點是可自行設計深度學習模型；缺點為開發需要更多程式碼及時間，所以開源社群開發了許多進階的深度學習應用程式介面（API），如： TF-Slim、TF-Layer、TF-Learn 及 Keras 等，可以讓開發者使用更易讀、簡潔的程式碼。而 Google 在 2017 年 6 月首度釋出了 TensorFlow Object Detection API 的開放原始碼，以 TensorFlow 為基礎所開發的物件偵測程式開發架構，讓所有想要開發以深度學習自動辨識物件程式的人，都可以利用這套架構發展自己的系統。故本研究以 TensorFlow Object Detection API 所提供的物件影像辨識模型為基礎，建構物件影像辨識模型。

2.3.2 物件影像辨識

透過卷積類神經網路（CNN）模型，判斷標的圖片是歸屬於何種類別，此一辨識過程稱為分類（Classification）（Krizhevsky, et al., 2012）。但要在一張標的圖片中進行所有出現物件的辨識，同時能夠清楚地標示出辨識物件的位置（Object localization），就要框出影像中每個物件、辨識出物件的類型、偵測出物件移動的方向與距離，最簡單的方式就是應用滑動視窗（Sliding windows）的作法：可以使用一個視窗，其大小必須固定，對標的圖片進行全面性的掃描，每次應用滑動視窗所框列的圖像內容將利用卷積類神經網路進行類別的判斷。此種處理方式屬於暴力運算作法，會耗用龐大的運算處理時間與資源，且其處理速度相對慢。

為了達成快速又有效率地目的，應用深度學習方式的許多物件辨識（Object detection）演算法因應而生且持續精進，為深度學習領域中不斷成長的一支；其中，最知名的演算法是 R-CNN 系列、YOLO 家族及 SSD（Liu, et al., 2017），在這些知名的物件辨識演算法中，以 Faster R-CNN、YOLOv3（Redmon & Farhadi, 2018）及 SSD 為三個最常使用的物件辨識演算法，Faster R-CNN 對小目標檢測效果最佳，但速度最慢；SSD 辨識速度雖然是最快的，但對小目標檢測的效果較差；YOLOv3 吸取了前二個演算法的特點，加以改進。本研究之影像辨識模型主要應用於中共軍機的影像辨識，由於軍用飛機的基本構型相對於其他物件的差異較小，因此能夠提供中共軍機辨識的精準度，確保中共軍機辨識的正確性，故使本研究採用 R-CNN 系列演算法中的 Faster R-CNN 演算法建構物件影像辨識模型。

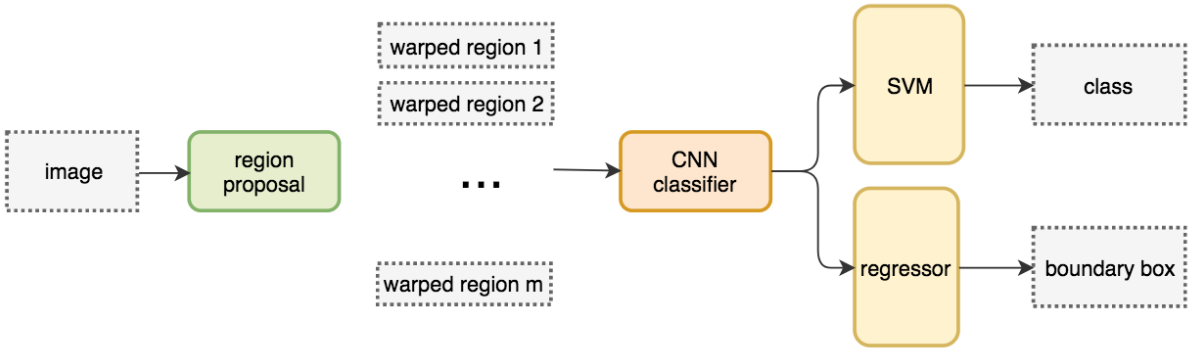


圖 4 R-CNN 運算處理流程

資料來源：Jonathan (2018)

R-CNN 係由 Zeiler 等學者於西元 2014 年提出，R-CNN 演算法的處理流程是：1. 就標的物件選定建議區域（Region proposals）、2.利用 CNN 針對建議區域萃取特徵、3.

再應用 SVM 分類器進行分類處理、4.最後再進行邊框迴歸（Bounding box regression）運算處理；R-CNN 演算法的處理流程如圖 4 所示。它利用標的物件圖像中的邊緣、紋理及顏色等資訊加以辨識，預先判斷出圖像中辨識目標可能出現的位置，此一預判的位置即為建議區域，然後再將這些區域個別去進行分類（Zeiler & Fergus, 2014），如圖 5 所示。

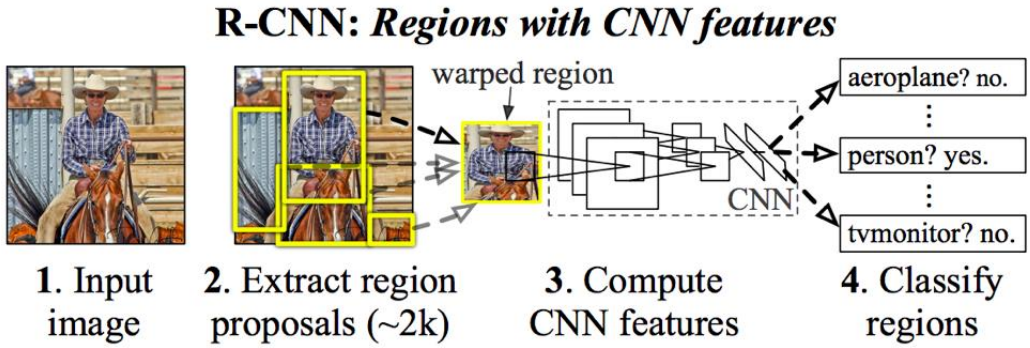


圖 5 R-CNN 模型運作示意圖

資料來源：Girshick, et al.（2014）

Girshick 等學者（2014）提出 Fast R-CNN 演算法，它共享卷積層，全部只進行一次卷積層運算即可，透過卷積層提取出來的特徵即可達到如 R-CNN 演算法所需約 2000 多個萃取區域運算的結果，並在最後一層的卷積層後面增加一個 ROI(Region of Interest) 池化層，然後將經池化處理後的資料輸入完全連結網路層（FCL）進行分類（Classification）與區域化（Localization）運算處理，處理流程如圖 6 所示。Fast R-CNN 演算法不是將所有建議區域都當做卷積層的輸入資料，而是它會輸入一張完整的影像內容，然後於第五層卷積層處理運算時才針對每個建議區域進行特徵的萃取，所以 Fast R-CNN 演算法只做一次卷積層運算處理，如圖 7 所示；ROI 池化是池化層的一種，而且是針對候選區域在特徵圖上的位置（ROI）的池化，它可以輸入大小不一的特徵圖，但可以輸出固定大小的特徵圖， 因此執行完卷積層最後一層的運算處理時，會獲得一張 L * W 長寬的特徵圖，同時也將建議區域對應至此一 L * W 長寬的特徵圖上，之後將在特徵圖上取得各自區域的最大池化(Max pooling)結果，因此每個區域會得到一個相同大小的矩陣。

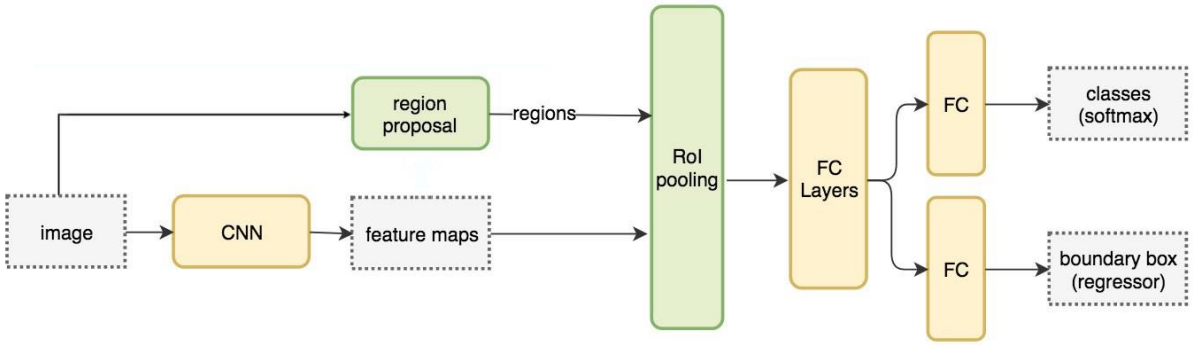


圖 6 Fast R-CNN 運算處理流程

資料來源：Jonathan (2018)

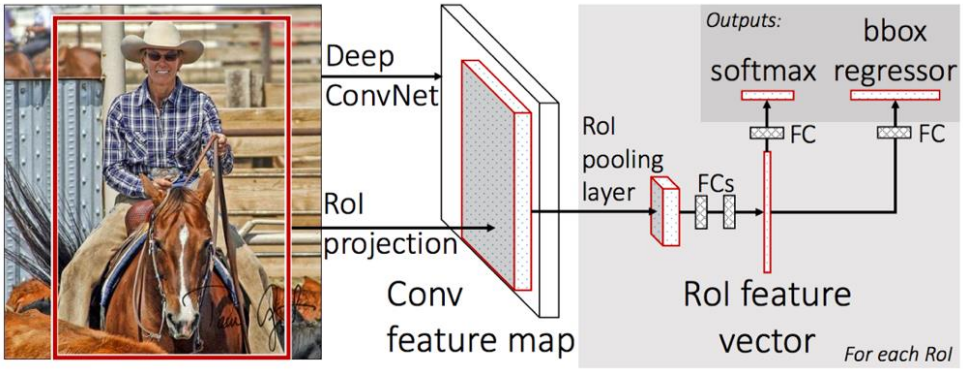


圖 7 Fast R-CNN 架構

資料來源：Girshick（2015）

相較於 R-CNN 演算法，Fast R-CNN 演算法能夠提升運算處理速度的主要原因是：它不如同 R-CNN 演算法一樣，對所有建議區域都透過深度網路進行特徵萃取運算處理，而是將整張影像圖進行一次特徵萃取運算處理，透過空間金字塔池化（Spatial Pyramid Pooling, SPP）處理，只需要進行一次特徵萃取運算處理，然後再將候選框映射到第五個卷積層上，剩下的運算處理只需要在第五個卷積層上進行即可。Fast R-CNN 演算法運算處理效能讓使用者看到了應用建議區域＋卷積類神經網路（Region proposal＋CNN）這一模型框架實際應用於物件辨識的效能，確認了可以針對多種類型的物件進行辨識，同時亦能確保辨識的準確率又能夠提升物件辨識處理運算速度，這是 Fast R-CNN 演算

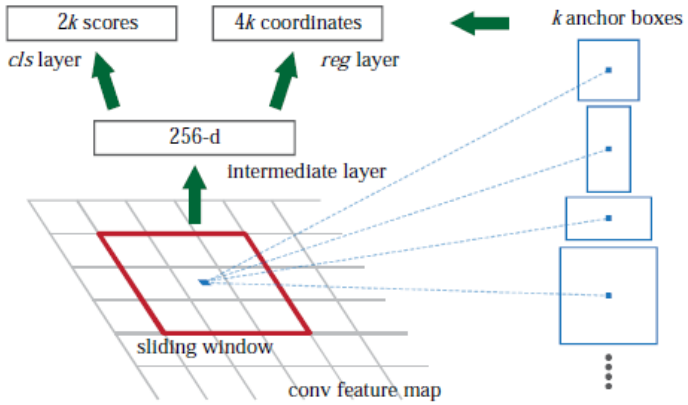


圖 8 Region proposal network

資料來源：Ren et al.（2015）

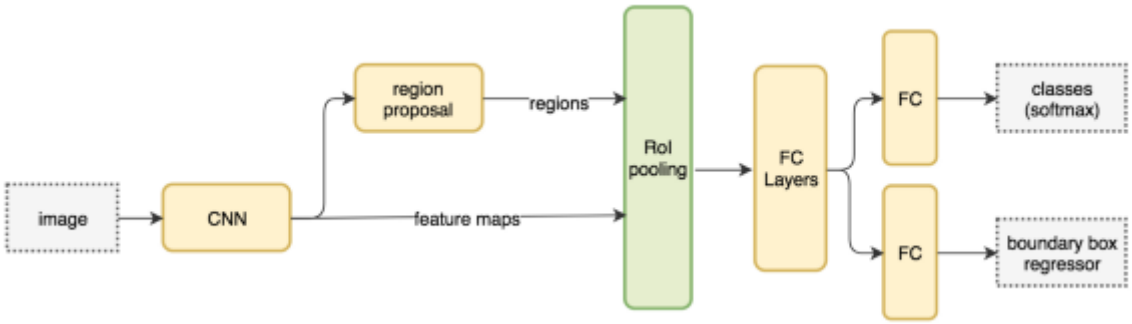


圖 9 Faster R-CNN 運算處理流程

資料來源：Jonathan (2018)

法相當重要的貢獻，也為之後一系列的 Faster R-CNN 演算法奠定深厚的基礎；但不管是 R-CNN 演算法還是 Fast R-CNN 演算法都需要先藉由預先選擇性搜尋（Selective search）的處理，此一處理需要耗費大量的運算處理時間，這是造成此二類演算法運算處理緩慢的主要原因。

Ren et al.（2015）又提出一個植基於 R-CNN 演算法的 Faster R-CNN 演算法，其運算處理速度比其他 R-CNN 演算法更快，二者主要的不同在於建議區域的生成方法，Fast-RCNN 使用的是選擇性搜索，而 Faster R-CNN 演算法用的是 RPN（Region Proposal Network）（如圖 8 所示）替代選擇性搜索，直接從卷積層網路的特徵圖（Feature map）上選取候選區域。並透過 Fast R-CNN 演算法的 ROI pooling（Region of interest pooling），提升了辨識的速度及準確率，Faster R-CNN 演算法運算處理流程如圖 9 所示，其架構如圖 10 所示。Faster R-CNN 演算法因其設計了萃取建議區域的網路 RPN，取代需要費時大量運算處理的選擇性搜索，使得物件辨識的時間大幅降低，此即為 Faster R-CNN 演算法的主要貢獻，Faster R-CNN 演算法。在本研究中亦將植基於 Faster R-CNN 演算法，運用 TensorFlow 深度學習平台建構中共軍機影像辨識訓練模型，藉由訓練過後的中共軍機影像辨識模型針對中共戰機影像進行機型辨識。

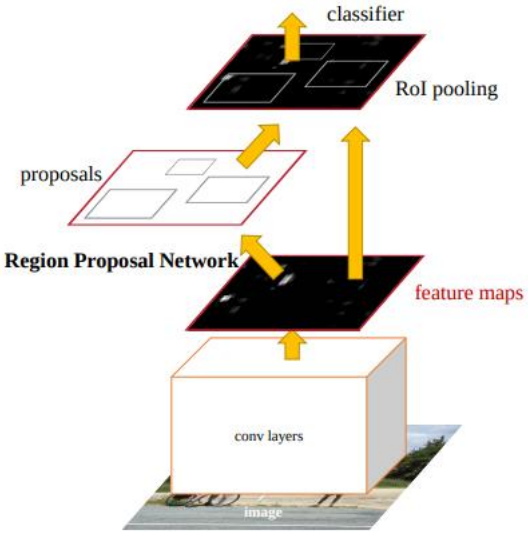


圖 10 Faster R-CNN 架構

資料來源：Ivan (2019)

三、植基於faster R-CNN建構中共戰機影像辨識機器學習訓練模型

本研究以TensorFlow Object Detection API所提供Faster R-CNN物件影像辨識模型為基礎，藉由中共戰機影像資料之整理分類並運用TensorFlow張量運算，建構中共戰機影像辨識機器學習訓練模型，藉以針對中共戰機影像辨識中共戰機機型；中共戰機影像辨識機器學習訓練模型建構詳如下列個小節說明。

3.1 中共戰機影像辨識機器學習訓練模型建構流程

中共戰機影像辨識機器學習訓練模型建構流程主要分為五個階段，分別為資料準備、建立模型、建立TensorFlow計算圖、訓練模型、測試模型，並說明每個階段所做的處理方式，運用TensorFlow框架進行張量運算，建構出影像辨識模型，如圖11所示。

3.2 資料準備

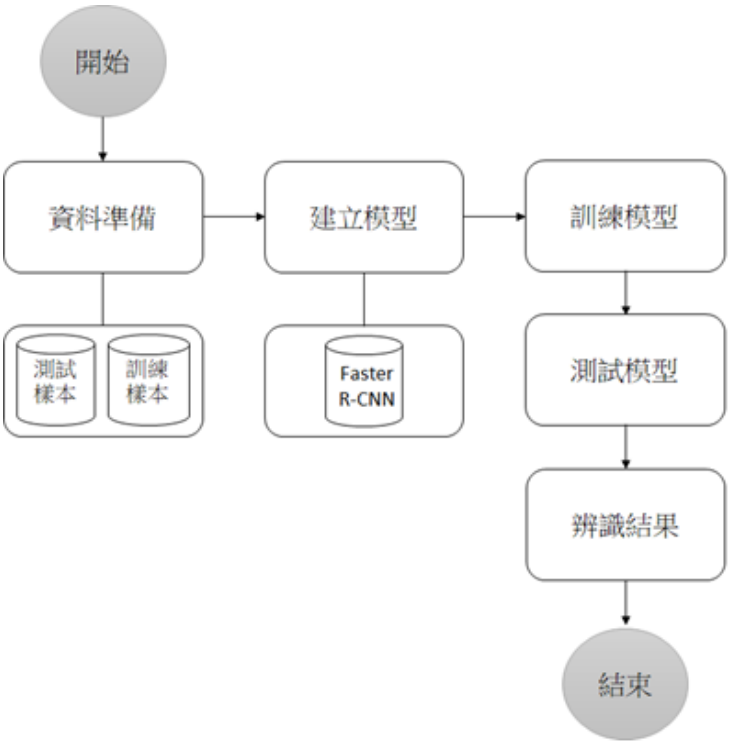


圖11 中共戰機影像辨識機器學習訓練模型建構流程

本研究使用Google Chrome擴充功能中的圖片助手（ImageAssistant）批量圖片下載器，經由Google網頁搜尋並下載建構中共戰機影像辨識機器學習訓練模型所需之圖片，做為建構研究所提影像辨識機器學習訓練模型所需的資料集。透過Google網頁搜尋並下載，計有700張中共各式戰機JPG圖檔，其中包含J-10A/B/C/S、J-11A/B、J-15、J-16/Su-30MKK、J-20、J-31、Su-35等七種機型的戰鬥機（詳如圖12所示）各100張，並將其80%（每個機型各80張）放入訓練資料夾，20%（每個機型各20張）放入測試資料夾，每個資料集計有560個訓練樣本，140個測試樣本。

3.3 建立模型

一般而言，訓練一個的植基於多層次類神經網路的高效能深度學習模型，縱使有多張具高運算效能的圖形處理顯示卡（GPU）通常也需要非常久的時間，而使用預訓練模型（Pre-trained model）為初始值來進行訓練，可以加速深度學習模型訓練的過程，降低模型訓練所需的時間。有鑑於此，本研究將應用TensorFlow Object Detection 模型三種機器學習模型：faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco，本研究應用這三種模型做為預先訓練的模型，透過相同的中共戰機圖片jpg檔案做為訓練樣本與測試樣本，藉以建構三種中共戰機影像辨識機器學習訓練模型，並比較此三種中共戰機影像辨識機器學習訓練模型對於中共戰機影像辨識的效能。

本研究所採用的三種機器學習模型均植基於Faster R-CNN演算法，其中，faster_rcnn_resnet101_coco與faster_rcnn_resnet50_coco模型係以深度殘差網路（Deep Residual Network, Deep ResNet）做為Faster R-CNN演算法中各個卷積層的網路結構基礎，faster_rcnn_resnet101_coco模型使用101層的深度殘差網路，而faster_rcnn_resnet50_coco模型則使用了50層的深度殘差網路。而faster_rcnn_inception_v2_coco模型係以Inception模組做為Faster R-CNN演算法中各個卷積層與池化層的網路結構基礎。

殲10 （J-10）

殲11 （J-11）

殲15 （J-15）

殲16 （J-16）

殲20 （J-20）

殲31（J-31）

Su-35

圖12 中共七種戰鬥機機型

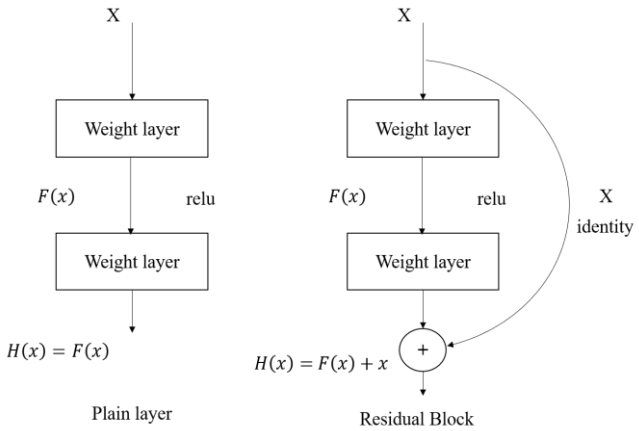


圖13 殘差網路與一般網路輸出處理函式示意圖

資料來源 ：He, et al.（2016）

經研究發現非常深度網路會出現退化問題（Degradation problem），通常深度學習模

型中網路層數的增加能夠有效地提升物件辨識的準確率，但是當深度學習模型中網路層數到達一定層數之後，其準確率物件辨識的出現飽和現象，甚至出現下降情形，因此，提出深度殘差網路（Deep Residual Network, Deep ResNet）。對一般的網路而言，H（x）是期望輸出，所以最後運算的結果H（x）就相當於是F（x）；而殘差網路則加入變數x此一處理捷徑，故當F（x）加上先前輸入的 x 之後，因此H（x）=F（x）+x，此時如果F（x）= 0時，H（x）會等於 x；由此可知殘差網路的功能就是要讓F（x）盡量接近於零，如此方能使得網路持續深化，其準確率才不會飽和或是下降。殘差網路與一般網路輸出處理函式示意圖如圖13所示。

Inception模組主要處理思維是將不同層級的卷積層經由並聯處理加以結合，透過不同層級卷積層的處理結果，藉以產生矩陣，並在深層維度矩陣中加以串接，進而產生一個更深層維度的矩陣；Inception模組能夠透過此一方式反覆地串接處理方式，進而能以高效能的處理方式，將網路的深度與寬度進行擴充，藉以建構更大的網路，同時亦能避免深層網路學習所可能造成過度擬合的現象。透過Inception模組可以先將較大的矩陣進行降低維度的運算處理，然後再將不同大小的矩陣進行聚合運算處理，如此可經由對不同大小的矩陣進行特徵值的萃取，方便後續的運算處理，此為Inception模組的主要優點。Inception-v1模組採用3種不同大小的卷積核，包括：1*1、3*3、5*5等三種卷積核和一個3*3大小的最大池化核，提升了網路對不同大小尺度的適應能力（osc_zc54ocuh, 2021）。Inception模組架構示意圖如圖14所示。

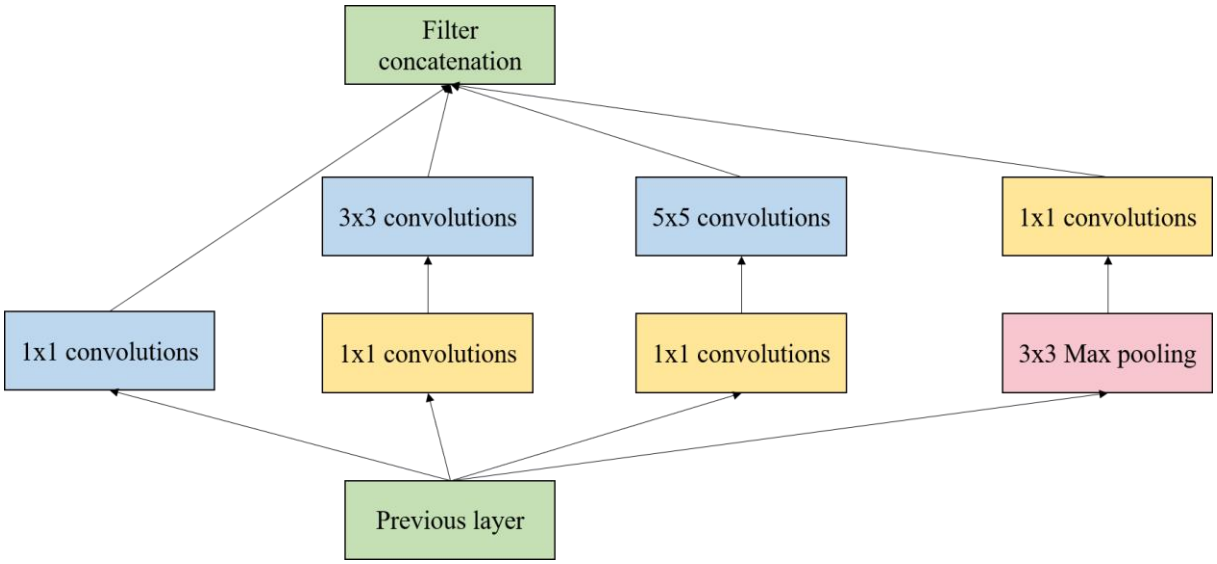


圖14 Inception模組架構示意圖

資料來源 : Szegedy et al. (2015)

由於，在深度學習類神經網路的訓練過程往往相當地繁雜，通常當前面幾個層級的網路產生微小的差異，這些差異經過後面數個層級的網路處理與傳遞之後，這些微小差異會產生放大現象。通常，類神經網路在進行訓練時所使用的參數會持續地變更，前面網路層所使用訓練參數的變化將會造成之後網路層資料輸入的分佈會產生變化，此一現象如發生在深度學習類神經網路中間網路層的訓練過程，此一現象被稱為內協變數移位（Internal covariate shift）。Inception-v2應用批量標準化（Batch Normalization, BN）方法解決深度學習類神經網路中間網路層的內協變數移位此一問題，批量標準化演算法的主要原理係在每一層網路的前面，插入一個標準化網路層，意即將前一層網路輸出的資料進行標準化處理後，再將經標準化處理的資料做為下一層網路的輸入資料。批量標準化

演算法對於每個使用函式所產生的數值都可以進行學習，它利用γ、β這二個參數進行縮放與平移的處理，使得經標準化處理後的輸入資料均能透過此二參數進行縮放與平移處理（osc_zc54ocuh, 2021）。

3.4模型訓練與測試

本研究將應用TensorFlow Object Detection 模型中三種機器學習模型：faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco做為預先訓練的模型來加速訓練的過程，訓練模型使用研究從網際網路所蒐集的七種中共戰機影像圖資料集，包括：J-11A/B、J-15、J-16/Su-30MKK、Su-35、J-10A/B/C/S、J-20、J-31等七種中共戰機，共計700張JPG圖檔，其中560張JPG圖檔做為訓練樣本，140張JPG圖檔做為測試樣本。本研究將執行至少200,000次訓練使誤差降低，並提高準確率，流程圖如圖15所示。

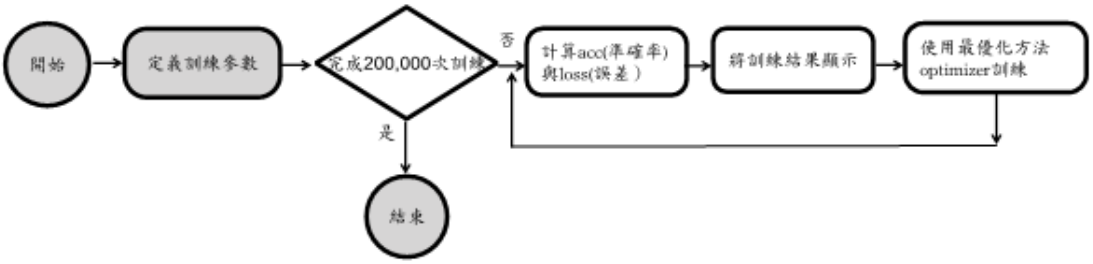


圖15 模型訓練流程圖

當研究所提中共戰機影像辨識機器學習模型完成模型訓練之後，將匯入測試樣本，進行辨識率分析。本研究共準備兩個測試資料集，一個測試樣本共21筆，包含J-11A/B、J-15、J-16/Su-30MKK、Su-35、J-10A/B/C/S、J-20、J-31七種機型的戰鬥機各3張；另一個測試樣本共10筆，每張圖片含一架飛機以上，流程圖如圖16所示。

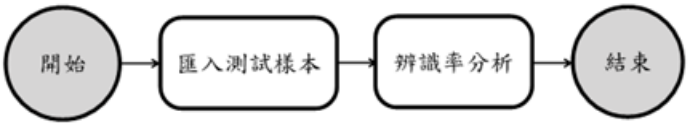


圖16 模型測試流程圖

四、模型建置、訓練與測試

本節將依據建構中共戰機影像辨識機器學習訓練模型建置、訓練與測試之所需，首先進行實作環境建構，完成實作環境建構後即進行準備模型訓練所需的資料集，並應用faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco等預先訓練模型做為中共戰機影像辨識機器學習訓練模型的基礎，完成相關的模型訓練設定，然後再進行中共戰機影像辨識機器學習訓練模型的訓練工作，最後，訓練完成的中共戰機影像辨識機器學習訓練模型進行中共戰機影像識別測試，藉以瞭解植基於三種不同預訓練模型的中共戰機影像辨識機器學習訓練模型對於中共戰機辨識準確率的高低。下面小節將針對前述相關實作工作進行詳實的說明。

4.1 實作環境建構

中共戰機影像辨識機器學習訓練模型建構與測試將藉由TensorFlow運算平台進行，因此，首先將建置TensorFlow運算平台，簡化TensorFlow運算平台建置作業，本研究藉由安裝Anaconda套裝載台的方式完成TensorFlow運算平台的建置。首先，本研究透過瀏覽器連網至Anaconda官網下載64位元版本的Anaconda套裝載台安裝程式，並進行Anaconda套裝載台安裝工作，如圖17所示。

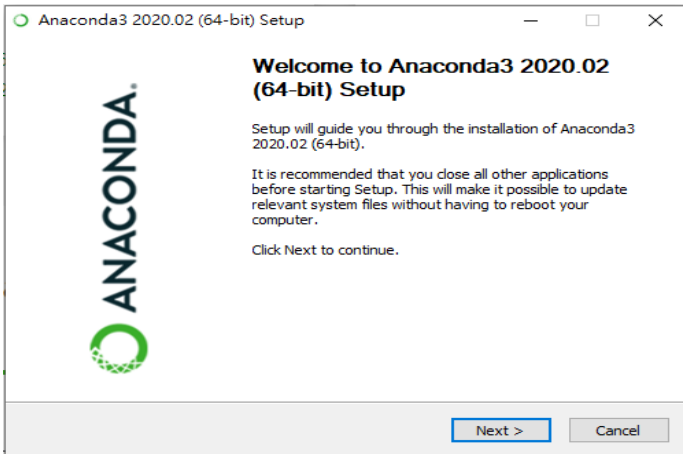
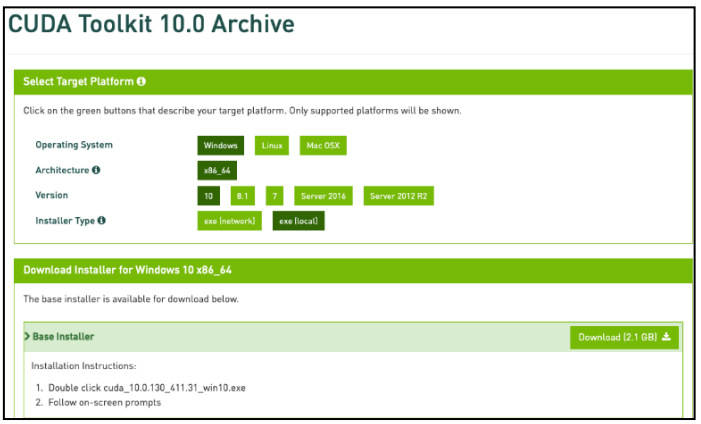


圖17 Anaconda套裝載台安裝作業圖

(a) Nvidia GTX GeForce 1660驅動程式下載



(b) Nvidia CUDA工具軟體下載

(c) Nvidia CuDNN 深度學習程式庫下載

圖18 Nvidia圖形加速處理軟體下載

為了能夠加快TensorFlow運算平台的處理速度，本研究將安裝支援GPU運算能力的TensorFlow運算平台，而具支援GPU運算能力的TensorFlow運算平台需要針對電腦的圖形加速處理器安裝相關的驅動程式與支援高速圖形加速處理運算所需的工具軟體；而本研究所使用電腦的圖形加速處理器的型號為Nvidia GTX GeForce 1660，因此，需至Nvidia官網下載Nvidia GTX GeForce 1660的驅動程式，並下載Nvidia支援支援高速圖形加速處理運算所需的工具軟體CUDA，且為搭配TensorFlow運算平台的版本，本研究選擇CUDA 10.0版工具軟體下載，然後進行安裝CUDA及Nvidia GTX GeForce 1660驅動程式，如圖18（a）（b）所示。由於，TensorFlow運算平台的運算涉及深度學習，為了能夠支援工具軟體CUDA在深度學習方面的運算，Nvidia提供了cuDNN深度學習程式庫，本研究亦選擇支援CUDA 10.0工具軟體深度學習運算所需的cuDNN 7.5版深度學習程式庫下載並執行安裝動作，如圖18（c）所示。

完成Nvidia圖形加速處理相關軟體安裝之後，即可依據下列步驟完成支援建構中共戰機影像辨識機器學習訓練模型所需的軟體載台安裝；圖19為安裝支援建構中共戰機影像辨識機器學習訓練模型所需軟體的操作指令。

```
步驟一 ： 建立TensorFlow-GPU虛擬環境
#conda create -n tensorflow_gpu pip python=3.6

步驟二 ： 安裝支援GPU運算的TensorFlow運算平台及Keras API
#activate tensorflow_gpu   //啟用TensorFlow-GPU虛擬環境
#pip install tensorflow_gpu   //在TensorFlow-GPU虛擬環境中安裝TensorFlow
#pip install keras   //在TensorFlow-GPU虛擬環境中安裝Keras

步驟三：從GitHub下載TensorFlow Object Detection API
#git clone https://github.com/tensorflow/models.git

步驟四：編譯 Protobuf 函式庫
#cd models/research
#protoc object_detection/protos/*.proto --python_out=.
```

圖19 安裝支援建構中共戰機影像辨識機器學習訓練模型所需軟體的操作指令

- 步驟一：建立TensorFlow-GPU虛擬環境。
  在先前安裝的Anaconda套裝載台中建立TensorFlow-GPU虛擬環境。
- 步驟二：安裝支援GPU運算的TensorFlow運算平台及Keras API。
  在Anaconda套裝載台啟用TensorFlow-GPU虛擬環境，完成TensorFlow-GPU虛擬環境開啟之後，即可在此一開啟的虛擬環境中安裝支援GPU運算的TensorFlow運算平台及Keras API。
- 步驟三：下載TensorFlow Object Detection API的原始碼，複製於特定的目錄之中。
  進入官網http://git-scm.com/安裝Git後，從GitHub上面下載TensorFlow Object Detection API的原始碼並複製於特定的目錄之中。
- 步驟四：編譯 Protobuf 函式庫。
  使用Tensorflow Object Detection API 之前，需先編譯 Protobuf 函式庫；ProtoBuf是由 Google 所推出跨平台的語言的「可擴展的序列化資料結構」，它具有資料體積小及傳輸快等優點，提供各種編譯器讓使用者可以把檔案編譯成所需要的語言格式可用於不同的語言上，例如：Python、Go、Java、C++ 等其他語言。

4.2 資料集準備

為了能夠進行中共戰機影像辨識機器學習訓練模型的訓練工作，本研究準備了中共

各式戰機資料集共有700張JPG圖檔，並確保每架機型的每個角度都包含在裡面，其中包含J-11A/B、J-15、J-16/Su-30MKK、Su-35、J-10A/B/C/S、J-20、J-31、七種機型的戰鬥機各100張，並將其80%（每個機型各80張）放入訓練資料夾，20%（每個機型各20張）放入測試資料夾，每個資料集計有560個訓練樣本，140個測試樣本。

訓練資料的準備需經過三道處理程序，第一道處理程序，首先，需要標註所有中共戰機影像於圖片檔案中相對應的位置及其標籤（此一標籤為中共戰機機型），並產生模型訓練時所需的XML格式檔案，為能順利執行此一處理程序，研究使用LabelImg此一軟體工具，經由使用LabelImg工具軟體標註每個影像檔案中中共戰機影像所在的位置，同時產生中共戰機影像於整張影像圖片中所在位置的XML的標註檔案，LabelImg工具軟體的使用如圖20所示。

第二道處理程序需把所有已經LabelImg工具軟體標註過的中共戰機影像圖片檔XML檔案，依照訓練集與測試集分別放置到兩個目錄下，利用Tensorflow Object Detection API提供的xml_to_csv.py指令碼，把對應的xml格式轉換成csv格式，同時在存放中共戰機影像圖片檔案的目錄下（如：D:\images）分別建立train_labels.csv和test_labels.csv二個文件檔案。接下來，開啟Tensorflow Object Detection API提供的generate_tfrecord.py文件，將本研究所需辨識七種中共戰機機型的代碼（中共戰機機型代碼需與先前經LabelImg工具軟體標註的中共戰機代碼相同）寫入其中，如圖21所示。然後，執行generate_tfrecord.py

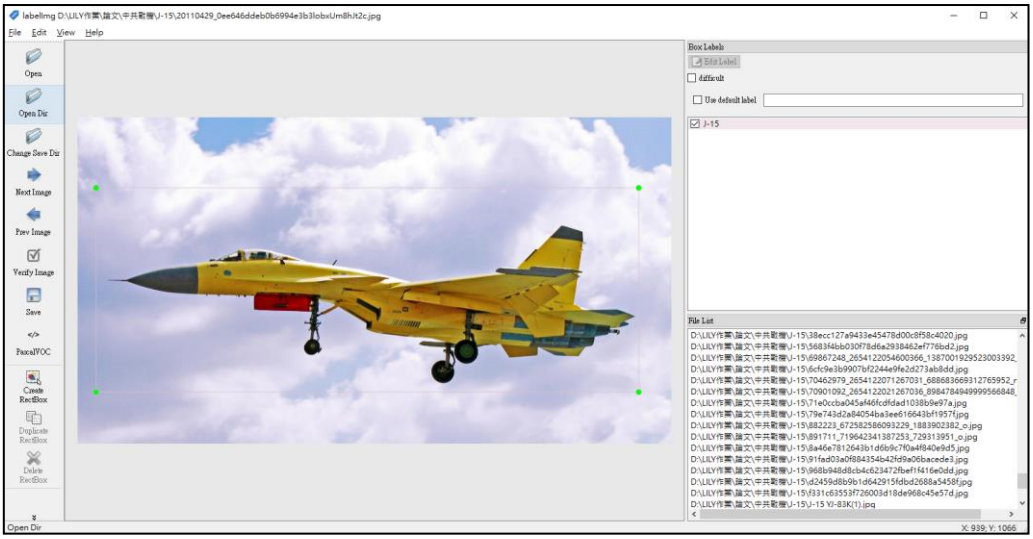


圖20 LabelImg工具軟體使用畫面

```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'J-10':
        return 1
    elif row_label == 'J-11':
        return 2
    elif row_label == 'J-15':
        return 3
    elif row_label == 'J-16':
        return 4
    elif row_label == 'J-20':
        return 5
    elif row_label == 'J-31':
        return 6
    elif row_label == 'su-35':
        return 7
    else:
        None
```

圖21 generate_tfrecord.py代碼編輯

```
(tensorflow_gpu) D:\>python generate_tfrecord.py --
csv_input=data/train_labels.csv --output_path=data/train.record

(tensorflow_gpu) D:>python generate_tfrecord.py --
csv_input=data/test_labels.csv  --output_path=data/test.record
```

圖22 generate_tfrecord.py執行指令

文件（執行指令如圖22所示），以產生訓練用途及測試用途所需的二個檔案：train.record與test.record。

第三道處理程序則是建立標籤地圖（Label map），研究利用文本編輯器建立一個新的本文文件檔案，並將此一檔案命名為fighter_labelmap.pbtxt，其中，此一檔案名稱可由使用者命名設定，但檔案類型（即附檔名）須為.pbtxt，fighter_labelmap.pbtxt檔案內容詳如圖23所示。

```
item {
  id: 1
  name: 'J-10'
}
item {
  id: 2
  name: 'J-11'
}
item {
  id: 3
  name: 'J-15'
}
item {
  id: 4
  name: 'J-16'
}
item {
  id: 5
  name: 'J-20'
}
item {
  id: 6
  name: 'J-31'
}
item {
  id: 7
  name: 'su-35'
}
```

圖23 fighter_labelmap.pbtxt檔案內容

4.3 模型訓練

進行中共戰機影像辨識機器學習訓練模型的訓練工作前，需在TensorFlow Object Detection API中，將所有使用預先訓練的模型參數、訓練參數與驗證參數都定義在一個.config設定檔中，在object_detection/samples/configs/目錄下有許多範例設定檔；由於，本研究將使用TensorFlow Object Detection API所提供三種預先訓練模型，分別為：faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco，因此將修改faster_rcnn_resnet101_coco.config、faster_rcnn_resnet50_coco.config及faster_rcnn_inception_v2_coco.config這三個範例設定檔，並使用一般的文字編輯器編輯對這三個設定檔進行修改編輯，圖24及圖25為修改faster_rcnn_resnet101_coco.config檔案內容，僅將要修改部分以黃色表示。

完成使用預先訓練模型的訓練參數與驗證參數設定之後，本研究將透過TensorFlow

```
# Faster R-CNN with Resnet-101 (v1), configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
  faster_rcnn {
    num_classes: 7 #將num_classes更改為分類器檢測的不同對象的數量。
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
```

圖24 修改faster_rcnn_resnet101_coco.config檔案設定（一）

```
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "config/faster_rcnn_resnet101_coco_2018_01_28/model.ckpt"
  from_detection_checkpoint: true
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
    input_path: "train.record"
  }
  label_map_path: "fighter_labelmap.pbtxt"
}

eval_config: {
  num_examples: 700 #將num_examples更改為\images\test目錄中的圖像數。
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/fighter_labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}
```

圖24 修改faster_rcnn_resnet101_coco.config檔案設定（二）

Object Detection API所提供的模型訓練程式對中共戰機影像辨識機器學習訓練模型進行訓練的工作，依據先前的設定，本研究將針對faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco三種預先訓練模型分別執行至少200,000次訓練，期使搭配這三種預先訓練模型的中共戰機影像辨識機器學習訓練模型的影像辨識損失（Loss）值降低於0.05以下，以提高中共戰機影像機型辨識的準確率；在執行中共戰機影像辨識機器學習訓練模型訓練時，本研究亦透過TensorFlow Object Detection API所提供的模型訓練監看工具（TensorBoard）瞭解模型訓練情況。圖25為執行模型訓練的指令（以應用faster_rcnn_inception_v2_coco預先訓練模型為例），圖26為TensorBoard監看畫面。

```
python C:\Tensorflow\models\research\object_detection\legacy\train.py
--logtostderr --train_dir=training/
--pipeline_config_path=D:\LILY\thesis\project\config\
faster_rcnn_inception_v2_coco_2018_01_28\training\faster_rcnn_inception_v2_coco.config
```

圖25 應用faster_rcnn_inception_v2_coco預先訓練模型執行模型訓練指令

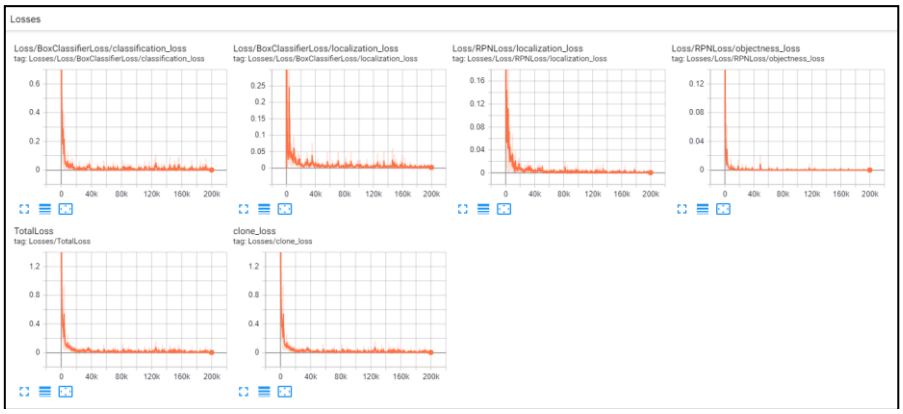


圖26 TensorBoard監看畫面

當訓練模型完成訓練工作後，即可透過TensorFlow Object Detection API所提供的訓練模組匯出程式將完成訓練的模型匯出，匯出的模型將會被TensorFlow Object Detection API儲存於「model」這個特定資料夾中；當訓練模組的完成匯出後，即完成訓練模組的訓練工作。圖27顯示TensorFlow Object Detection API匯出訓練模組程式

```
python C:\Tensorflow\models\research\object_detection\export_inference_graph.py
--input_type image_tensor --pipeline_config_path training/faster_rcnn_inception_v2_coco.config
--trained_checkpoint_prefix training/model.ckpt-200000 --output_directory model
```

圖27 TensorFlow Object Detection API匯出訓練模組程式

4.4 測試結果

完成中共戰機影像辨識機器學習訓練模型的訓練之後，本研究利用TensorFlow Object Detection API所提供的影像物件偵測程式（位於TensorFlow Object Detection API目錄下的object_detection/object_detection_tutorial.ipynb），加以複製並使用一般的文字編輯器進行編輯，修改相關設定，如圖28所示。

本研究共準備兩組中共戰機影像測試資料集，其中一組的測試樣本共計21筆，包含J-11A/B、J-15、J-16/Su-30MKK、Su-35、J-10A/B/C/S、J-20、J-31七種機型的戰鬥機各3張；另一組的測試樣本共10筆，每張測試圖片將有一架以上不同機型的中共戰機。由於，本研究應用三種預先訓練模型（faster_rcnn_resnet101_coco、faster_rcnn_resnet50_coco及faster_rcnn_inception_v2_coco）做為中共戰機影像辨識機器學習訓練模型的預先訓練模型，故將針對這三種預先訓練模型所訓練出的中共戰機影像辨識機器學習訓練模型進行中共戰機影像測試，以瞭解植基於此三種預先訓練模型的中共戰機影像辨識機器學習

```
  },
  "outputs": [],
  "source": [
   "PATH_TO_TEST_IMAGES_DIR = 'D:/LILY/thesis/project/config/faster_rcnn_resnet101_coco_2018_01_28/images/test_single'\n",
   "TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(11, 22) ]\n",
   "\n",
   "# Size, in inches, of the output images.\n",
   "IMAGE_SIZE = (12, 8)"
  ]
 },
 {


  "outputs": [],
  "source": [
   "# What model to download.\n",
   "MODEL_NAME = 'D:/LILY/thesis/project/config/faster_rcnn_resnet101_coco_2018_01_28/model'\n",
   "\n",
   "# Path to frozen detection graph. This is the actual model that is used for the object detection.\n",
   "PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'\n",
   "\n",
   "# List of the strings that is used to add correct label for each box.\n",
   "PATH_TO_LABELS = os.path.join('D:/LILY/thesis/project/config/faster_rcnn_resnet101_coco_2018_01_28/training', 'fighter_labelmap.pbtxt')"
  ]
```

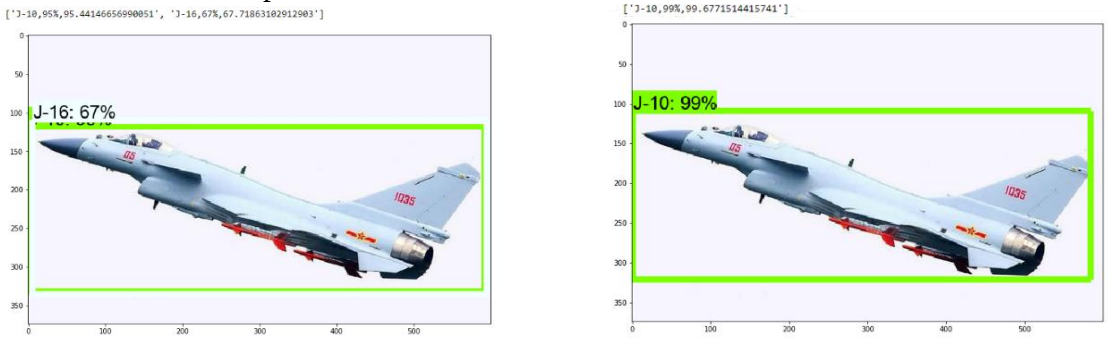
圖28 影像物件偵測程式修改內容

訓練模型對於中共戰機機型辨識的準確度；本研究透過戰機辨識率比較的方式來探究植基於三種不同預先訓練模型的中共戰機影像辨識機器學習訓練模型對於中共戰機影像辨識的效能。

- 應用faster_rcnn_resnet101_coco及faster_rcnn_inception_v2_coco訓練模型中共戰機影像辨識測試比對的結果

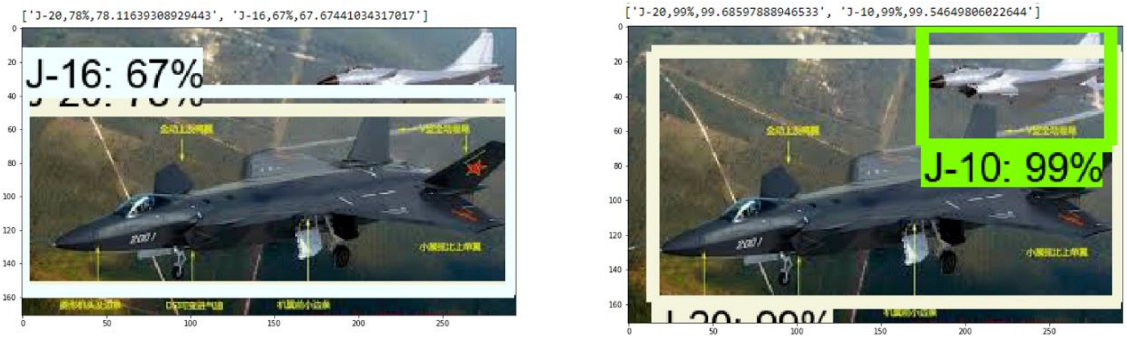
從單一機型辨識率來看，在各別機型的圖片中，應用faster_rcnn_inception_v2_coco的訓練模型會多辨識出J-16中共戰機的錯誤結果，如圖29（a）所示；在多種中共戰機（含一架以上中共戰機影像）的圖片中，應用faster_rcnn_resnet101_coco的訓練模型可以正

faster_rcnn_inception_v2_coco　　faster_rcnn_resnet101_coco



(a) 單一機型辨識

faster_rcnn_inception_v2_coco　　faster_rcnn_resnet101_coco



(b) 多種機型辨識

圖29　應用faster_rcnn_inception_v2_coco及faster_rcnn_resnet101_coco訓練模型中共戰機影像辨識測試結果

確辨識出J-20以及J-10二種中共戰機，應用faster_rcnn_inception_v2_coco的訓練模型則沒有辨識出J-10中共戰機，且會將J-20中共戰機影像辨識為J-20及J-16二種中共戰機，如圖29（b）。

- 應用faster_rcnn_resnet101_coco及faster_rcnn_resnet50_coco訓練模型中共戰機影像辨識測試比對的結果

從單一機型辨識率來看，在各別機型的圖片中，應用faster_rcnn_resnet50_coco的訓練模型會多一個辨識目標框的判斷結果，將部分中共戰機影像判斷為另一架中共戰機影像，如圖30（a）所示；在多種中共戰機（含一架以上中共戰機影像）的圖片中，應用faster_rcnn_resnet101_coco的訓練模型可以正確辨識出J-20以及J-10二種中共戰機，應用faster_rcnn_resnet50_coco的訓練模型則雖正確辨識出J-10中共戰機，但會將J-20中共戰機影像誤判為J-16中共戰機，如圖30（b） 所示。綜合以上兩點，faster_rcnn_resnet101_coco的辨識率較faster_rcnn_resnet50_coco高。

faster_rcnn_resnet50_coco　　faster_rcnn_resnet101_coco

(c) 單一機型辨識

faster_rcnn_resnet50_coco　　faster_rcnn_resnet101_coco

(d) 多種機型辨識

圖30　應用faster_rcnn_resnet50_coco及faster_rcnn_resnet101_coco訓練模型中共戰機影像辨識測試結果

- 應用faster_rcnn_resnet101_coco及faster_rcnn_resnet50_coco訓練模型中共戰機影像辨識測試比對的結果

從單一機型辨識率來看，在各別機型的圖片中，應用faster_rcnn_resnet50_coco的訓練模型會多一個辨識目標框的判斷結果，將部分中共戰機影像判斷為另一架中共戰機影像，如圖30（a）所示；在多種中共戰機（含一架以上中共戰機影像）的圖片中，應用faster_rcnn_resnet101_coco的訓練模型可以正確辨識出J-20以及J-10二種中共戰機，應用faster_rcnn_resnet50_coco的訓練模型則雖正確辨識出J-10中共戰機，但會將J-20中共戰機影像誤判為J-16中共戰機，如圖30（b） 所示。綜合以上兩點，faster_rcnn_resnet101_coco的辨識率較faster_rcnn_resnet50_coco高。

由上面對於中共戰機影像辨識的結果可以發現：應用faster_rcnn_resnet101_coco訓練模型所建構的中共戰機影像辨識機器學習訓練模型對於中共戰機影像辨識可以獲得較為正確且高的辨識結果。

五、結論

隨著資訊技術的進步，人工智慧（AI）應用的範疇逐漸的增加，也日漸受到人們的關注，從民間到政府、從家庭到產業、從民生到科技，各項應用不斷地推陳出新，人工智慧的應用甚至於躍上軍事戰備的領用。從西元1980年到西元2010年這三十年之間，人工智慧的學者提出許多不同階層、架構和初始化方式的類神經網路，而TensorFlow就是當前十分著名的深度學習平台，它能夠支援各類型的類神經網路演算法。

本研究主要探討於TensorFlow平台，發展出一套應用於飛機機型的影像辨識模型，並提升辨識率。以Tensorflow Object Detection API所提供的物件影像辨識模型為基礎，運

用現行Object Detection演算法Faster R-CNN建立深度學習模型，再以TensorFlow設計張量運算流程，調整其張量、參數，並研究所訓練之模型於物件影像辨識中的辨識率，建構可以提升辨識率的機器學習模型。

從實驗結果得知以下幾點：

- 運用Faster R-CNN演算法搭配ResNet-101神經網路層，不論單一機或多種機型辨識率皆較搭配inception V2神經網路層高。
- 運用Faster R-CNN演算法搭配ResNet-101神經網路層，不論單一機或多種機型辨識率皆較搭配ResNet-50神經網路層高。

本研究在實作部分於Anaconda平台運用虛擬環境，以TensorFlow所提供之物件辨識API，並使用自製資料集，建構提升辨識率的機器學習模型，預期所發展之物體辨識技術可應用於軍事領域（軍用武器裝備）、土地利用（都市規劃、商業分佈）、交通傳輸（停車場分析、汽車監控）、醫療領域等。

未來期望能朝以下以幾點做精進與改良：

1. 本研究所自製之資料集僅含有700張圖片，飛機機型的各個角度並不全面，期盼可以搜集更多張不同角度的圖片，以精進辨識率。
2. 本研究所使用之Faster R-CNN演算法僅搭配兩個神經網路層做比較，未來可使用不同神經網路層做為探討。
3. 本研究所建構之物件影像辨識模型僅針對辨識率，盼望後續可對訓練時間效率進行提升。

國防領域相關應用

本研究所提應用Tensorflow 深度學習機制於中共軍機影像辨識，透過研究所建構中共戰機影像辨識機器學習訓練模型，可以協助國軍人員對於中共戰機機型之辨識；且此一深度學習機制環境與訓練模型之建立，將有助於國軍相關專業人員對於人工智慧與深度學習領域的熟悉與相關應用，可逐步提升國軍對於人工智慧與深度學習技術應用能量，強化國軍於人工智慧與機器學習領域的應用與探索。

參考文獻

李彥冬，郝宗波，雷航（2016）。卷積類神經網路研究綜述。*電腦應用*，*36*（9），2508-2515。

林大貴（2017）。*TensorFlow + Keras 深度學習人工智慧實務應用*。博碩出版社。

林敬恆 (2019)。AI 來襲！三分鐘看懂人工智慧。下載於 https://makerpro.cc/2019/05/introduction-to-ai/ （2021 年 7 月 15 日）。

孫志軍，薛磊，許陽明，王正（2012）。深度學習研究綜述。*電腦應用研究*，*29*（8），2806-2810。

郭利敏（2017）。基於卷積類神經網路的文獻自動分類研究。*圖書與情報*，*6*，96-103。

張順，龔怡宏，王進軍（2019）。深度卷積類神經網路的發展及其在電腦視覺領域的應用。*電腦學報*，*42*（3），453-482。

楊宗恩（2017）。*以 LSTM 深度神經網路語言模型建構英文課程重點摘要*。國立屏東大學資訊管理學系碩士班碩士論文，屏東縣。

Ivan (2019)。[物件偵測] S3: Faster R-CNN 簡介。下載於 https://ivan-eng-murmur.medium.com/object-detection-s3-faster-rcnn-%E7%B0%A1%E4%BB%8B-5f37b13ccdd2 （2021 年 7 月 20 日）。

JYRoy (2020)。深度學習三：卷積神經網路。下載於 https://iter01.com/537205.html （2021

年 7 月 20 日）。
Ashley, K. & Gordon, T. (2005). An Introduction to Artificial Intelligence and Law. *Tutorial Handout of Introduction to AI and Law at ICAIL*.
Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, *20*(3), 542-542.
Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... & Shah, H. (2016, September). *Wide & deep learning for recommender systems*. In Proceedings of the 1st workshop on deep learning for recommender systems (pp. 7-10).
Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*.
Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition （pp. 580-587）.
Girshick, R. (2015). *Fast r-cnn*. In Proceedings of the IEEE international conference on computer vision （pp. 1440-1448）.
Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485-585). Springer, New York, NY.
He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
Jonathan Hui. (2018). What do we learn from region based object detectors (Faster R-CNN, R-FCN, FPN)? download from https://jonathan-hui.medium.com/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9 (retrieved on July 18, 2021)
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.
LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, *86*（11）, 2278-2324.
Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11-26.
Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.). (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.
Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). *Recurrent neural network based language model*. In Eleventh annual conference of the international speech communication association.
Nallapati, R., Xiang, B., & Zhou, B. (2016). Sequence-to-sequence rnns for text summarization.
Noble, W. S. (2006). What is a support vector machine?. *Nature biotechnology*, *24*(12), 1565-1567.
Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*, 91-99.
Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, 149-171.
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). *Going deeper with convolutions*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Zaccone, G., Karim, M. R., & Menshawy, A. (2017). *Deep learning with TensorFlow*. Packt Publishing Ltd.
Zeiler, M. D., & Fergus, R. (2014, September). *Visualizing and understanding convolutional networks*. In European conference on computer vision （pp. 818-833）. Springer, Cham.