DOI: 10.6653/MoCICHE.201810_45(5).0008

人工智慧模型之建置與應用

楊明德／國立中興大學土木工程學系 特聘教授
蔡慧萍／國立中興大學土木工程學系 助理教授
許鈺群／國立中興大學土木工程學系 博士生
曾信鴻／國立中興大學土木工程學系 碩士生

隨著物聯網的時代來臨，各類資料快速產生並經由網路傳遞，將巨量資料處理成有價值的資訊成為當下重要課題，人工智慧（Artificial Intelligence, AI）藉由深層類神經網路的發展與電腦計算效能提升，造就了這一波 AI 的浪潮。AI 的組成元素可分三大項，第一為巨量資料，包括標記及未標記資料；第二是深度學習方法，包括演算法及軟硬體；第三是應用情境，包括各種 AI 的應用與各領域專業知識，應用的範圍涵蓋教育、服務、農業、製造、金融、醫療等人類可能觸及之各種生活情境。本文針對此三大 AI 組成元素進行簡要介紹，並展示一些簡單的 AI 模型，並提出對 AI 發展趨勢的觀察，希冀能迎上這波 AI 產業化、產業 AI 化的全球浪潮。

隨著物聯網的時代來臨，各類資料快速產生並經由網路傳遞，如何處理巨量資料變成當下重要課題，在 1956 年就被提出的人工智慧（Artificial Intelligence, AI）也再度成為矚目焦點。廣義的 AI 可包含任何機器學習的方法，狹義的 AI 則是指基於深層的類神經網路架構應用。類神經網路技術已被發展數十年，但先前因常有過度擬和（overfitting）的問題，所以其發展沉寂了一段時間，近年來有了軟硬體快速進展與巨量資料的產生，尤其在近兩年間，AI 變成火紅的議題，主要原因為二方面的重要突破，分別是演算法—深層類神經網路的發展、與硬體—電腦計算效能提升，所以造就了這一波 AI 的浪潮。

AI 演化的進程可分為弱 AI（Weak AI — 特別型）、強 AI（Strong AI — 通用型）、及超級 AI（Super intelligence AI — 超級型）（圖 1），目前仍在弱 AI 的發展階段。弱 AI 時代，只能針對特別型的應用開發 AI 模型，如只會下圍棋的 Alpha Go，通常以監督式分類方式來訓練模型，而且只能做特殊任務的工作。接著，AI 可能進化到通用型的強 AI，可透過無監督式分類方式自我學習，不須經由人類訓練即可學習執行一項任務，此時 AI 已具多元學習、推理、規劃和表達等人類的邏輯思考能力。最終，AI 會進化到一個具有超級智能的機器 — 超級 AI（Super intelligence AI），會自己思考並解決問題，也具有發展出優於人類的智慧的可能性。

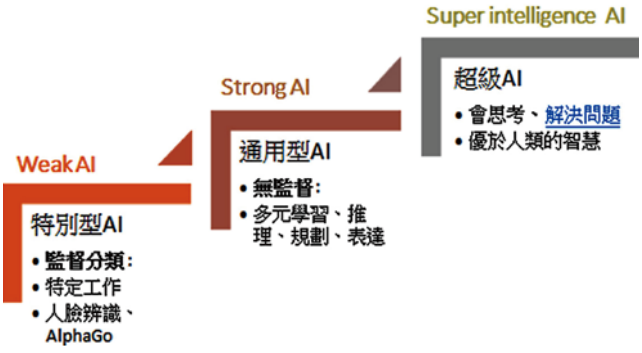

圖 1　AI 的世代演進

AI 的組成元素可粗分三大項，如圖 2，第一是巨量資料，包括標記及未標記資料；第二是深度學習，包括演算法及軟硬體；第三是應用情境，包括各種 AI 的應用與各領域專業知識，應用的範圍涵蓋教育、服務、農業、製造、金融、醫療等人類可能觸及之各種生活情境。

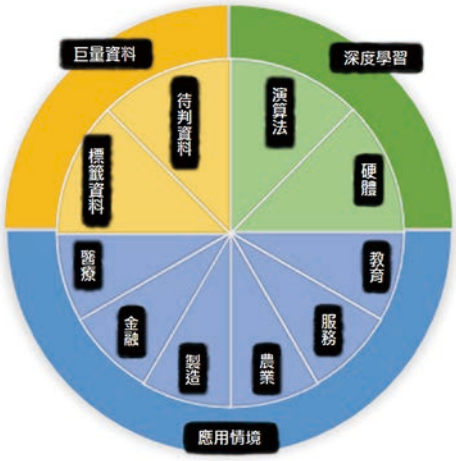

圖 2　AI 的組成元素

資料標記

機器學習是實現人工智慧的一種方法，透過大量的資料訓練讓電腦掌握資料特徵，透過類神經網路建立預測模型，而後以訓練過的模型進行推論，AI 模型建立流程如圖 3 所示。建立模型的流程中，有個相當重要的步驟，就是需要「教」電腦看懂真實世界的現象，這個步驟稱為標記（Label）。人腦有錯綜複雜的神經系統，對事物有極高的學習能力，縱使非過目不忘，兩三次的訓練就能足以初步認知一件東西。對於電腦而言，機器學習則需透過千遍萬遍的學習，才能使電腦認識一種物件，這種訓練過程建立在「餵」給電腦正確資料的基礎上，即所謂經過人類標記的資料，故言「人工智慧建立在工人智慧之上」。標記這工作既基礎，但也費時又無趣，有時還需要許多專家協做才能蒐集到足夠的樣本，在中國大陸號稱有百萬人員從事標記的工作，有趣的是這些人在準備教會電腦識物以便取代自己的工作。

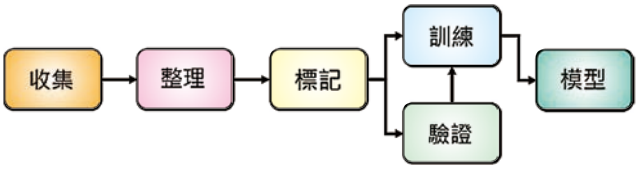

圖 3　AI 模型建立流程

演算法

狹義的 AI 是指基於深層的類神經網路架構應用。早期於 1960 至 1970 年代，由於受到生物神經系統的啟發，資訊科學家提出類神經網路的概念，希望能將電腦設計並訓練成具有人類的智能，但因為受限於電腦的計算能力和資料取得不易，類神經網路並沒有廣受推崇。近 30 年來，資訊科學家陸續開發出多種類神經網路類型，持續發展多樣網路架構、階層及初始化的方式。2010 年以後，類神經網路與深度學習幾乎畫上等號，隨著許多著名應用案例的成功，例如手寫郵遞區號辨識、AlphaGo 的棋弈競賽，類神經網路和深度學習的發展日漸蓬勃。總體而言，類神經網路是深度學習的支柱，根據其網路的架構與連結方式的不同，每種類神經網路皆有其特點、侷限與適用的領域。以下針對目前三個常用的深層的類神經網路學習架構進行簡要介紹，並比較其適用情境。

深度類神經網路（Deep Neural Networks, DNN）

類神經網路（Neural Networks）為模仿生物神經系統的數學模型，藉由神經元（neuron）的連結與連結方式的設定，機器能夠自己以訓練資料（training data）找出每個神經元適合的權重（weight）。類神經網路中的每個神經元都是一個簡單的函數，其構成是將每個輸入數值和其對應權重相乘後加總，再加上閾值（bias）成為激活函數（activation function）的輸入。線性整流函數（Rectified linear unit, ReLU）為其中常用之一種激活函數，其內涵生物學原理[1]，通常比其他激活函數（如：邏輯函數）有更好的效果。類神經網路的結構是由神經元構成一排一排的層，再將各層連接，每層神經元的輸出為下層神經元的輸入，稱為完全連接前饋式網絡（fully connected feedforward network），最後一層為輸出層，其他層為隱藏層（hidden layer）。

一般而言，類神經網路透過運算，只要神經元夠多便可以描述任何函數，但是由於參數也會隨著神經元的數量增加而變多，所以效率較差。因此，「深度」類神經網路（Deep Neural Networks DNN）即指有多個隱藏層（如圖 4），且此較多的層數便可以較少的神經元數量，透過模組化的過程，有效使用參數

描述任何函數。此較少的有效參數也可避免過度擬合（overfitting）的狀況發生，訓練資料的數量需求也較少，增加資料的使用效率。DNN 中單層神經元數量的設定、各層之間神經元連接方式、層的數量及激活函數的類型需在訓練前已先決定，若這些參數定義的不好就無法找出最佳的函數。

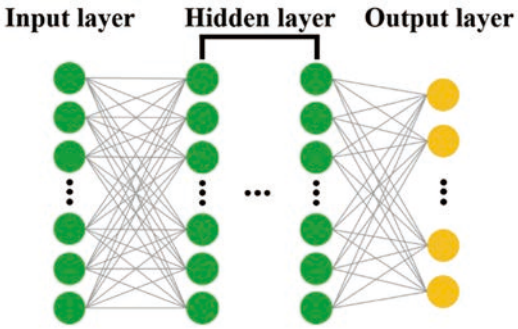

圖 4　深度類神經網路示意圖

DNN 在影像和口語辨識（speech recognition）上的分析表現特別好。例如在口語辨識上，微軟與蘋果等商業龍頭公司接續宣告其口語辨識的詞錯誤率（word error rate, WER）已降低到 10% 以下。在影像辨識上，微軟應用深達 152 層的深度殘留網絡（deep residual network）展現了僅有 3.5% 的低影像辨識錯誤率。

卷積類神經網路（Convolutional Neural Networks, CNN）

卷積類神經網路（CNN）在影像和聲音等訊號類型方面的辨識具有極佳成效，其演算結構如圖 5 所示。以影像為例，其架構設計依據輸入的影像，透過卷積層（convolutional layer）及池化層（pooling layer）來強化模式辨識（pattern recognition）及相鄰資料間的關係。CNN 具有局部連接（local connection）和權值共享（shared weight）兩大特點，因此其參數量和複雜度都大大降低，過度擬合的狀況較不易發生。卷積層是 CNN 的核心，通常由數十到數百個 n × n 的濾鏡（filter）組成，每個濾鏡具有一項特徵萃取的功能，會對不同的影像模式（pattern）進行強化。池化層（pooling layer）通常接在卷積層之後，進行類似訊號處理中的降維採樣（down sampling），常用的有最大值池化層（max pooling）和平均值池化層（average pooling）兩種。一般應用於影像識別的 CNN，在處理輸入資料時，有一到三次的卷積層加池化層的處理，之後再接兩層以上的完全連接層（fully connected feedforward network），才輸出預測結果。

CNN 是一種具有先驗知識（prior knowledge）的類神經網路，需要的參數也比 DNN 簡化。CNN 應用卷積層的濾鏡找出影像的模式（pattern），產生特徵圖層（feature map），適用於要分析的影像具有三大特點時，第一、要偵測的特徵（pattern）比整張影像小；第二、同樣的特徵可能出現在影像不同位置；第三、影像進行二次抽樣（subsampling）後不會有太大改變。CNN 在卷積層時進行第一和第二特點的運算，而在池化層時則為第三特點的運算。CNN 的訓練過程以反向傳播（backpropagation）的機器學習技巧建立模型，再用其他非訓練影像測試辨識效果，由精確率（Precision）和召回率（Recall）來進行評估。

CNN 最廣為人知的例子就是 Google 所開發的 AlphaGo 人工智慧電腦，其在棋局對弈的表現優於人類棋王。另外，CNN 還常被應用於口語（speech recognition）、自然語言處理（natural language processing）中的文字資料等等。百度 deep speech、IBM、微軟、Google 等著名公司皆相繼推出其在語音識別方面的 CNN 模型。

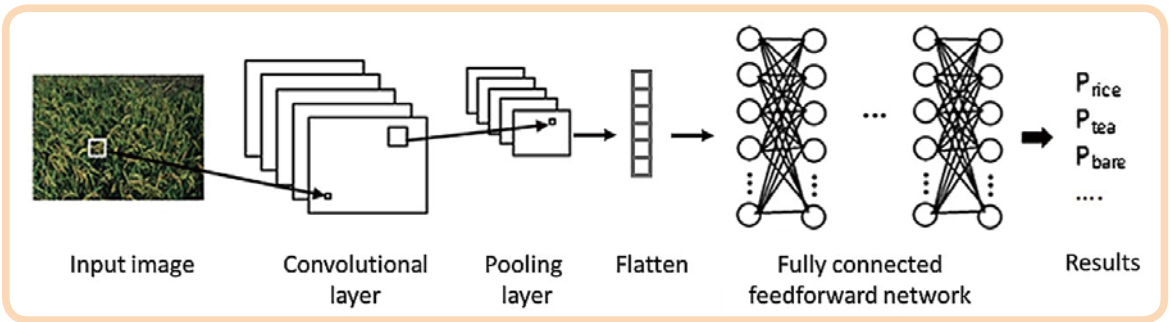

圖 5　卷積類神經網路運算示意圖

遞迴式類神經網絡（Recurrent Neural Network, RNN）

遞迴式類神經網路（RNN）基礎的運算架構如圖 6 所示，每個神經元（以 s 表示）獲得的輸入（X）來自於上一個時間點（下圖左邊的黑色方框，時間以 t 表示），U、V、W 為各時間點的參數，輸出以 O 表示。由於 RNN 具有上一個時間點的記憶（memory），同時也考量輸入的資料序列（data sequence），因此非常適合用於分析成串有序列性的輸入資料，例如文字（text）、基因組（genomes）、手寫文字（handwriting）、口語（spoken word）或是來自感測器（sensors）或股票市場（stock markets）數值化的時間序列資料。常見的應用領域包括時間序列（time series）、自然語言處理（natural language processing）、手寫辨識（handwriting recognition）、口語辨識（speech recognition）等。

在深度學習中常用梯度下降法（gradient descent）來找出最佳函數（最佳參數組合），但此方法有隨機性，僅能提供局部的最佳解，也會產生梯度消失與爆炸的問題（gradient vanishing and exploding）。為了解決此問題，90 年代中期，德國學者 Hochreiter 和 Juergen Schmidhuber（1997）提出了長短期記憶神經單元（Long Short-Tern Memory units, LSTMs）[3]。LSTMs 在傳統 RNN 神經元中加入了一個記憶細胞（memory cell），以三個閘門，分別為輸入閘（input gate）、遺忘閘（forget gate）、輸出閘（output gates）來控制每個時間點的記憶是否留存。LSTMs 在每個時間點都具有很多運算層時，效果甚佳，另外 LSTMs 也在語音辨識和機器翻譯方面應用甚廣 [4,5]。學者也陸續發展多種不同形式的記憶細胞，例如門閘遞迴單元（gated recurrent unit, GRU）[6] 是 LSTM 的簡化版本，將輸入閘與遺忘閘進行連動 [7]，運算速度更快且被廣泛使用。RNN 也發展出多種變形結構，例如雙向的 Elman Networks 和 Jordan Networks 等。

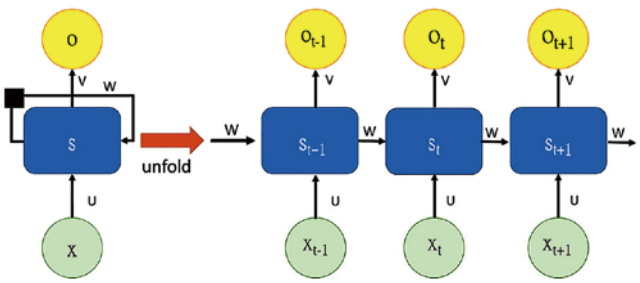

圖 6　遞迴式類神經網路運算示意圖（重繪自 LeCun *et al.*, 2015）[2]

深層類神經網路框架

機器學習在建構一套演算方式時，會使用多種且繁雜的演算法進行組合，故需先建立一套可以直接調用的模組，以避免運算程式碼過於冗長，以利機器學習修改與優化。若已預先建立一套專門為機器學習編程的函式庫，將各演算法與資料操作功能模組化，在建構神經網路模型時，就只需依照各框架指定方式，將各模組一一堆疊形成網路，在調整與優化時，只需要抽換、新增模組，或是更改各模組的輸入參數即可。如此一來，神經網路模型的建置就如堆疊積木一樣簡單，大幅降低了機器學習的入門門檻。

以下介紹七個以 Python 編寫的主流機器學習框架，分別是 Apache MXNet [8]、BVLC Caffe [9]、Google TensorFlow [10]、Microsoft CNTK [11] 及 Facebook PyTorch [12] 等五個獨立的框架，以及兩個整合多種框架的工具 - Keras team 釋出的 Keras [13] 及 NVIDIA 釋出的 DIGITS [14]。各種框架都有其優勢，也有其弱勢，使用者應依需求來挑選適用框架，以下簡單敘述各框架之優缺點，並將各框架之比較簡列於表 1。

Apache MXNet: A flexible and efficient library for deep learning

MXNet 為一靈活且高效能的深度學習套件，MXNet 支援多種高階語言，如 Python、R、Matlab、C++、Perl、Julia、Scala 等，也支援多種平台，如 Linux、MacOS、Windows，及 Raspberry Pi、NVIDIA Jetson TX2 與 AWS。硬體方面支援 CPU 與 GPU。高效能方面，MXNet 提供對高階語言的多 GPU、分散式系統計算優化。

BVLC Caffe: Convolutional Architecture for Fast Feature Embedding

Caffe 是目前最為廣泛使用的機器學習框架之一，主要支援 CNN 相關之影像辨識演算，但在 RNN 類型的網路支援度不佳。Caffe 支援 Python、Matlab 等介面，核心語言以 C++ 實踐，並支援 NVIDIA cuDNN 之深度類神經網路加速套件。Caffe 的特點在於其網路之建立都基於配置文件形式，將需要使用之函式寫入堆

表 1 主流機器學習框架簡述

框架	維護單位	支持語言	特點
MXNet	Apache	Python / R / Matlab / C++…	支援多種高階語言與多平台，亦能進行分散式運算，AWS 官方支援的框架。
Caffe	BVLC	Python / Matlab / C++…	廣泛使用的框架之一，發展較久、社群資源多，運行相對穩定，但是架構較舊，自訂義網路層不易。
TensorFlow	Google	Python / Java / Go / C++…	現今使用比率最高的機器學習框架，由 Google 維護，更新頻率快速、開發自由度高、社群龐大，適合新穎研究或原型開發。
CNTK	Microsoft	Python / C++ / BrainScript	支援大部分神經網路演算法，運算硬體優化佳，使其效率極高。網路配置與執行檔案分開，易於模型跨平台部署。
PyTorch	Facebook	Python	用法簡潔、直觀，對於熟悉 NumPy 的使用者，容易上手，模型設計快速。
Keras	Keras team	Python	整合多種框架使其用法更為彈性，模組化的功能設計，使模型設計僅需簡單幾行文字堆疊就能完成，適合初學者。
DIGITS	NVIDIA	Protobuf / Lua / Python	將資料集建立、模型訓練、模型推論與部署，整合於圖形介面的系統，流水線式的生產模型原型，可快速調整與優化模型參數。

疊成層，再由層層連接形成網路，相較其他網路框架易於上手，但也因此 Caffe 的網路定義檔案較為冗長，且新的功能擴充不易（須以 C++ 實踐）。對於深度學習的新手來說，Caffe 是個好的入門框架，因其擁有龐大的預訓練模型、社群支持，相對易於找到資源，其程序運行也較穩定，適合高穩定需求的工業生產環境。

Google TensorFlow: An open source machine learning framework for everyone

TensorFlow 為 Google Brain Team 於 2015 年釋出的機器學習套件，是目前最為熱門且使用率最高的機器學習套件，主要語言為 Python，另支援 Java、Go、C++ 等高階語言，核心以 C++ 及 CUDA 實踐。此架構主要應用於 CNN 與 RNN 的演算，亦支援強化學習（Reinforcement Learning），而 TensorFlow 不只侷限於神經網路運算，只要將計算表示為支援的計算圖形式，就可以利用 TensorFlow 計算，例如偏微分方程式（Partial Differential Equations, PDE）求解。

Microsoft CNTK: A free, easy-to-use, open-source, commercial-grade toolkit that trains deep learning algorithms to learn like the human brain

CNTK（Computational Network Toolkit, 現為 Cognitive Toolkit）為微軟研究院於 2016 年釋出的開放原始碼深度學習套件，其包含 MLP、CNN、RNN 及 LSTM 等演算模型，且可詳細設定網路模型模組，不需要自行實踐底層 C++ 或 CUDA，就能利用這些模組設計新的網路層。

CNTK 設計為性能導向的框架，在大規模並行訓練上擁有非常高的效率。因為網路結構是由獨立配置文件定義，再透過指令執行程序，因此跨平台部署相對容易、快速。

Facebook PyTorch: A deep learning framework for fast, flexible experimentation

PyTorch 為 Facebook 於 2017 年釋出之深度學習套件，參考原以 Lua 編寫的 Torch，後以 Python 實踐之故名 PyTorch（但實際上與 Torch 差異甚多），其語法簡潔、直觀、易上手，甚至於張量（Tensor）的操作與 NumPy 近乎相同。PyTorch 另一特點是神經網路模型拓樸排序（或是稱為有向圖）為動態的，在每一次迭代過程後，能對應輸入值來建立新的模型，對 RNN 演算方式特別有效，但是如 TensorFlow 這種靜態建立有向圖的框架，就必須把每種可能都先寫入神經網路模型的源碼中，在初次執行時先編譯程執行碼。

Keras: Focus on enabling fast experimentation

Keras 是款建立於 TensorFlow、Theano 及 CNTK 之上的高層神經網路 API，以 Python 實踐，相較其他單一框架套件，Keras 更適合想快速體驗機器學習的入門新手，因其用法直觀，網路建立只需依照說明文件形式編寫，能快速建立多層之深度神經網路，利於建立神經網路模型原型。然而 Keras 是高層神經網路 API，較不適於底層更動，也較難以理解神經網路的原理，若時間充裕，還是建議深入瞭解 TensorFlow。

NVIDIA DIGITS: Interactive Deep Learning GPU Training System

DIGITS 全名為 Deep Learning GPU Training

System，為 NVIDIA 開發的圖形介面多 GPU 深度學習系統，目前將 NVCaffe、Torch 及 TensorFlow 整合在統一平台上，此系統特點是同時涵蓋資料集建立、網路訓練、網路推論及網路部署的功能，能非常簡單地將模型訓練後，隨即測試與部署，將神經網路模型建立變成流水線生產的模式。然而此系統為 NVIDIA 所整合的環境，各深度神經網路套件的功能並非完全支援，且要新增功能較為不易，此環境適合對指令介面操作不熟悉的使用者，以及需要快速調整與優化模型參數的用戶。

應用情境

AI 的應用的範圍涵蓋教育、服務、農業、製造、金融、醫療等人類可能觸及之各種生活情。日新月異的應用情境發展在世界各國隨處可見，例如美國的 Amazon 的虛擬助理 Alexa、倉庫機器人 Kiva 可以執行指揮將包裝和運送過程更優化、還可檢測仿冒品。中國的阿里巴巴也廣泛使用 AI，應用於物流、線上支付、螞蟻金融等方面，並正在嘗試對審批交易進行中之臉部識別。其他應用的例子，例如保險公司藉由 AI 技術，分析借貸者的近 50 種臉部表情，偵測其回答有關收入、償還計畫等問題的真實程度，決定是否核定其借貸申請。一般公司也可藉此 AI 分析找到最佳的申請者給予試用，減少新聘人員的成本與風險。營運管理方面則可利用 AI 從過去的銷售紀錄與其可能影響因子（如天氣），藉由天氣預報預測銷售量，調整進貨量及降低庫存。目前最常應用的簡單 AI 模型多屬於影像分類與物件偵測，以下以實際案例簡介此兩類型應用。

影像分類

機器學習一個模型的流程包括資料收集與整理、標記、訓練與驗證、最後才能餵進模型訓練模型的參數。舉例來說，任兩張貓咪的照片是完全不同的物體，需要收集大量的貓咪照片，經過整理，涵蓋各種情況之貓咪影像進行訓練，標記上貓的標籤，並以部分標記影像進行資料驗證，電腦才有機會學會看懂貓咪的照片。常用的影像類別已有整理好的資料集可提供進行訓練，例如圖 7 為機器學習常用的 CIFAR-10 資料集，左列簡單的幾個類別，包括飛機、汽車、貓、狗等 10 個常見類別，此資料集擁有大約 60,000 張已標記的影像。

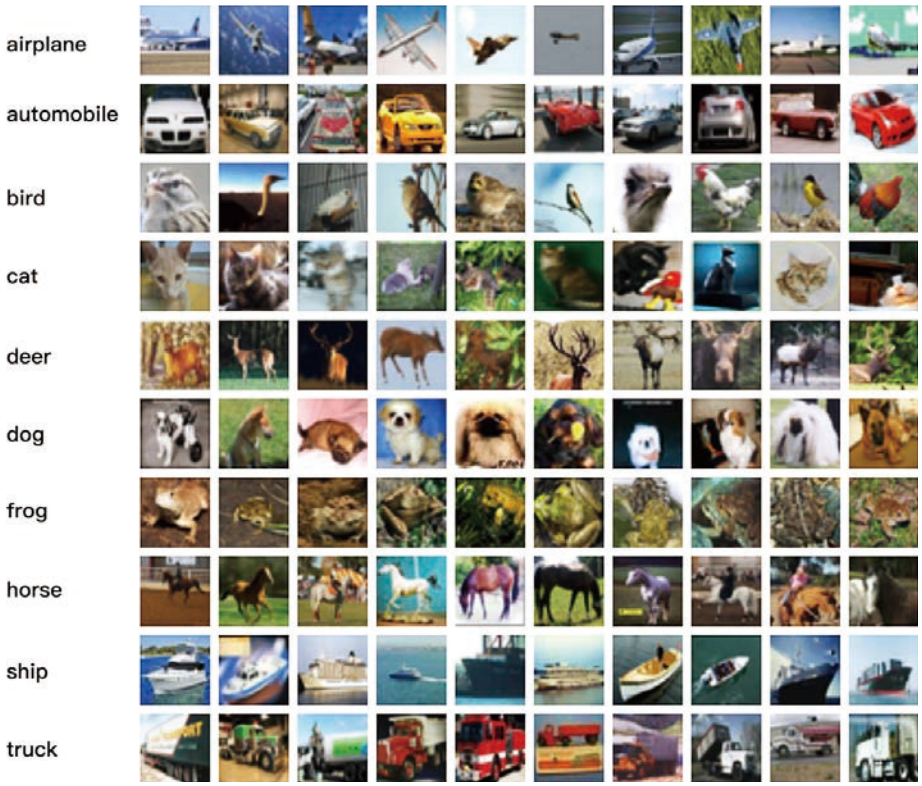

圖 7　CIFAR-10 資料集（資料來源：CIFAR 資料集官網）

若想自行建立資料集則需大量人力收集影像、進行整理與標記，於實務上相當困難。舉例而言，在農業方面有許多害蟲會影響作物產量，造成許多困擾，例如最近幾年討論熱烈的荔枝椿象，牠會以刺吸方式危害荔枝、龍眼等無患子科植物，導致落花、落果與嫩枝幼果枯萎，大幅影響產量。且荔枝椿象在受驚動後就會噴射有害酸液，造成農民困擾。目前各界皆積極研發荔枝椿象的防治方式，本研究團隊也積極發展 AI 影像辨識技術，希望能透過影像，進行荔枝椿象的判釋。因此需要投資大量人力收集荔枝椿象各生長階段的影像，經過整理與預處理並進行標記，再進行 AI 模型訓練，以便發展荔枝椿象的自動化分類系統（如表 2）。

表 2　荔枝椿象影像辨識技術成果

			
卵：100% 若蟲（一齡）：0% 若蟲：0% 成蟲：0%	若蟲：62.66% 成蟲：36.78% 若蟲（一齡）：2.56% 卵：0%	若蟲：100% 卵：0% 若蟲（一齡）：0% 成蟲：0%	成蟲：99.93% 若蟲（一齡）：0.07% 卵：0% 若蟲：0%

物件偵測

物件偵測（Object Detection）也是常見的簡單 AI 模型。影像分類能夠分出不同物件的類別，但無法得知多目標數量、位置與分類成果等資訊，而這正是物件偵測可提供的資訊。透過目前熱門的即時物件偵測系統 Yolo（You Only Look Once）提供的資料庫與訓練模型，可簡單的進行物件化辨識，一般的物件偵測成果如圖 8 所示，影像為本研究團隊的研究室與研究生之影像，可分出多物件類別與位置，物件偵測可應用的情境相當多元，如人臉辨識、身份辨識、安控等，都是透過物件辨識的方式訓練出的模型應用情境。

雲端運算與邊緣運算

建立資料集並訓練出模型後，便可以輸入未經標記的資料，透過模型進行推論。以分類的案例來說，新的影像透過模型辨識即可推論出可能所屬的類別。單純的情境可以在單一電腦內完成，但複雜的情境則需透過雲端處理，因為雲端伺服器通常有較大的資料庫、多樣的訓練模組、及高性能的電腦運算能力，一般雲端架構可如圖 9 所示。

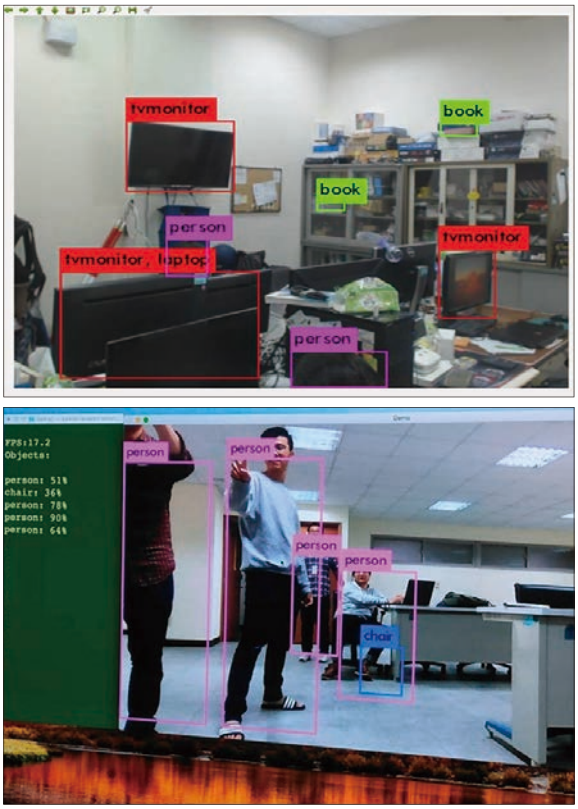

圖 8　物件偵測技術成果

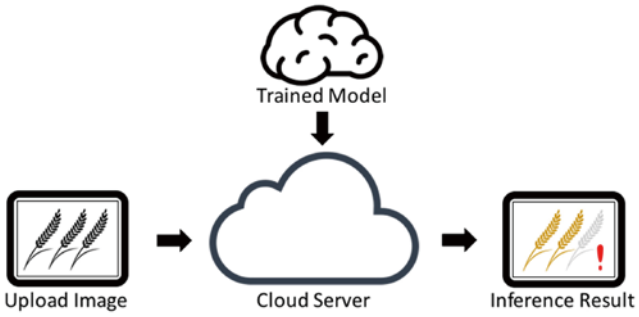

圖 9　AI 雲端化運算架構

將已訓練完成的模型放置於雲端伺服器並針對接口之訊息保持監聽（Listen）狀態，透過網路與應用程式介面（API）的方式，將上傳的影像進行推論，推論完成之結果可回應給使用者端。手機、無人機、移動裝置及 IOT 的邊緣端感測器負責影像資料蒐集，雲端伺服器負責大量運算，此集中運算方式無需在使用端機器上執行複雜、高負載的機器學習程式與訓練模型，為目前最常用的 AI 型態。

透過 AI 雲端運算架構，能夠於雲端伺服器進行集中式運算，但是需透過網路做資料傳輸，因此受限於網路頻寬與延遲。若需做更迅速反應、穩定獲取運算成果、及高度資料安全等要求，則須往分散式運算架構發展，導入邊緣運算（Edge computing）技術與微型化

的 AI 晶片開發，可將訓練好的模型植入於手機、無人機、移動裝置及 IOT 的設備，達到快速、穩定、安全的目的，並降低對雲端的依賴及負擔。例如，無人機巡航飛行時，若須判定當下鏡頭影像類別時，可有兩種做法：一是透過網路傳回雲端伺服器解算後，將成果回傳回無人機端，並控制無人機做出對應的反應，此法為雲端化概念，需要維持穩定的網路傳輸速率與伺服器通暢。二是於無人機端進行即時解算，直接可將解算成果之對應指令給無人機做出立即反應，此法即為邊緣運算概念。目前受限於邊緣運算能力，目前只能處理較簡單之模型，惟隨著 AI 晶片之發展，邊緣運算之應用將會愈發多元與蓬勃，尤其適合台灣擁有晶片設計與製造能力且缺乏世界級資訊服務公司的小國家。

後語

在土木領域，設計最佳化、工程監造、材料機器搬運、工安維護與預防、要徑管理等，各種生產情境難度大增，雖可透過大量感測器做監測，但是資訊大量增加下，分析的壓力也會隨之遽增，而人類的分析速度很難跟上大量快速的資料供應，AI 卻可找出不易洞見的規律模式，進而避免先前發生的錯誤再發生，甚至做到提前預測與預防，可適時的提出警告或調整排程、減少呆料及廢料的產生、降低發生工安問題的機率。

目前的弱 AI 主要功能是協助操作者及管理者從大量感測器搜集的數據中篩選出重要的部份，以方便操作管理者做出分析與決策判斷，目前的弱 AI 尚未具有能力自動做出具有邏輯性的決策，因此還需要結合專家智慧以作資訊的轉譯。以生產流程為例，首先，要將生產流程標準化系統化，以便達到資訊流自動化，才能以達到智慧製造與生產為目標。進而再利用感測裝置與網路，即時收集大量資訊成為大數據，再導入深度學習做即時辨識與分析，才可建立一個較完整的 AI 生態系。目前全球主要 IT 公司，尤其以美國與中國為首的世界級公司，幾乎在 AI 領域都投入大量研發能力，逐漸造成強者恆強、大者恆大的趨勢。

電影關鍵報告（Minority Report）中，為了降低犯罪率，採取積極的犯罪預防手段，依據諸多的電腦影像判識結果，事先逮捕可能罪犯以便防患未然。電影中，先知腦中可擷取出預測的影像，而在未來現實世界中，很可能 AI 就可以扮演先知的腳色，負責預知這項工作，並結合判識做出決策。當決策問題極度複雜而超過人腦可處理範圍時，決策的工作就得交給電腦處理，但若全都依照電腦的預測做決策，而人類已無能力判斷真偽的時候，是不是就是一個電腦主宰人類社會的來臨？其實，目前大部分的複雜問題都是電腦負責處理，雖然 Tesla 自駕車發生首起致命車禍震驚世界，但是在此同時世界上有多少人駕車正在發生車禍，連酒後不開車都難以杜絕，更遑論要讓所有的人駕車安全優於自駕車，只是人類無法接受車禍是由電腦造成而非是一個可以容許犯錯的凡人，畢竟「人非聖賢，孰能無過」。然而，比起人類的智慧進步速度而言，可預期未來的人工智慧的進步速度將會更為飛快，影響也更為深遠。

誌謝

部分研究成果承蒙科技部人工智慧專案計畫（MOST-107-2634-F-005-003-）經費補助，謹此致謝。

參考文獻

1. Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011), “Deep sparse rectifier neural networks.” Proceedings of the fourteenth international conference on artificial intelligence and statistics.
2. LeCun, Y., Bengio, Y. and Hinton, G. (2015), Deep learning. Nature, 521(7553), pp. 436-444.
3. Hochreiter, S. and Schmidhuber, J. (1997), Long short-term memory. Neural computation, 9(8), pp. 1735-1780.
4. Sutskever, I., Vinyals, O. and Le, Q.V. (2014), Sequence to sequence learning with neural networks. NIPS’14 Proceedings of the 27th International Conference on Neural Information Processing Systems, 2, pp. 3104-3112.
5. Bahdanau, D., Kyunghyun, C. and Yoshua, B. (2014), Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations (ICLR2015), pp.1-15.
6. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference (EMNLP 2014), pp. 1-15.
7. Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014), Empirical evaluation of gated recurrent neural networks on sequence modeling. Presented in NIPS 2014 Deep Learning and Representation Learning Workshop, pp. 1-9.
8. Apache, “MXNet: A Scalable Deep Learning Framework”, https://mxnet.apache.org/
9. BVLC, “Caffe | Deep Learning Framework”, http://caffe.berkeleyvision.org/
10. Google, “TensorFlow: An open source machine learning framework for everyone”, https://www.tensorflow.org/
11. Microsoft, “The Microsoft Cognitive Toolkit”, https://www.microsoft.com/en-us/cognitive-toolkit/
12. Facebook., “PyTorch”, https://pytorch.org/
13. Keras-team., “Keras: Deep Learning for humans”, https://keras.io/
14. NVIDIA., “DIGITS: Interactive Deep Learning GPU Training System”, https://developer.nvidia.com/digits