機器學習與深度學習
以鳥類影像辨識為例

臺南一中．姚政彣

驗證與實作

摘要

本研究旨在探討 VGG16 卷積神經網路架構在鳥類影像辨識中的應用，並探究不同參數設定對模型辨識率的影響。研究利用 Google Colab 雲端開發環境，使用 Kaggle 平台上的 bird 400 資料集進行模型訓練，並針對 Class count（類別數量）， Batch Size（批次大小）以及資料增強技術進行參數調整與分析。

研究結果顯示，最佳模型設定為：Batch Size 512，Class count 200，並開啟旋轉範圍 60 度，水平翻轉和圖像寬度 20% 的左右橫移等資料增強技術。在此設定下，模型在訓練集和驗證集上均表現良好，達到 75.6% 的辨識率。

研究結論指出，Class count 與模型辨識率呈負相關，即隨著類別數量增加，模型辨識率下降。而適當增加 Batch Size 則有助於提升模型辨識率，但需權衡記憶體需求和訓練速度。此外，資料增強技術如隨機旋轉、水平翻轉及左右橫移，可有效提升模型的泛化能力和辨識率。

未來研究方向包括：探討更優越的網路架構，例如 ResNet、Inception 或 EfficientNet 等，並考慮輕量化模型以降低計算成本。在訓練算法方面，可以嘗試使用更先進的優化器、設計更具針對性的損失函數，並採用更有效的資料增強技術。此外，開源模型和部署至模型 Hub 平台，以及探索雲端部署、邊緣計算和特定領域應用等，都是值得進一步研究的方向。

本研究證明了 VGG16 架構在鳥類影像辨識中的可行性，並探討了不同參數設定對模型性能的影響。未來透過模型架構改良、訓練算法精進以及模型部署與應用的研究，VGG16 模型有望在圖像分類以及其他領域發揮更大的作用。

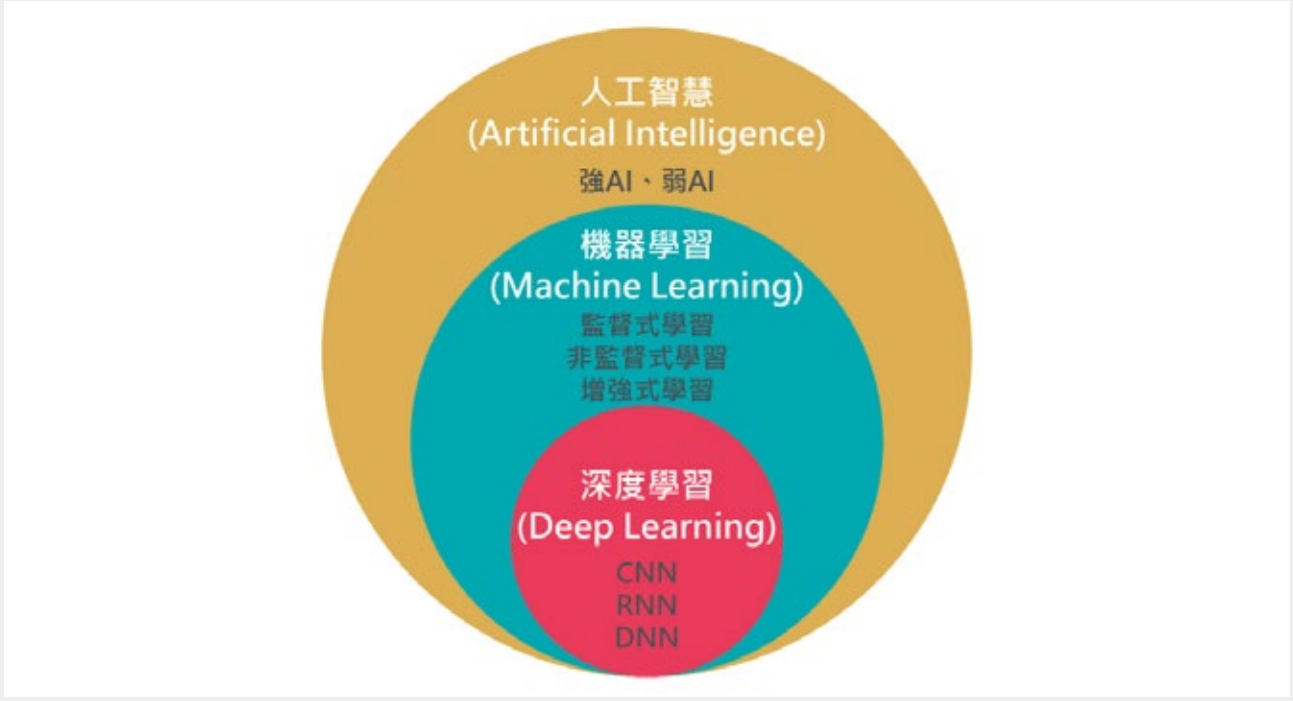
目錄

第一章：研究背景與動機 01

科技的浪潮席捲而來，人工智慧正在悄悄地改變世界。其中，機器學習作為人工智慧的核心技術，正逐漸滲透到我們生活的各個角落，並對未來產生深遠的影響。

機器學習的核心在於賦予機器學習和進化的能力。它不再依賴於人類預先編寫的程式碼，而是通過分析大量數據，找出其中的規律和模式，進而建立預測模型。

深度學習作為機器學習領域的一顆新星，它模仿人腦神經網路的架構，訓練多層神經網路模型，處理具有多個抽象級別的數據。深度學習技術在語音識別、視覺物件識別、物體檢測以及基因組學等領域取得了突破性成果。其核心原理是利用反向傳播算法，指示機器如何調整內部參數，進而發現大數據集中的複雜架構。



▲人工智慧、機器學習與深度學習的關係

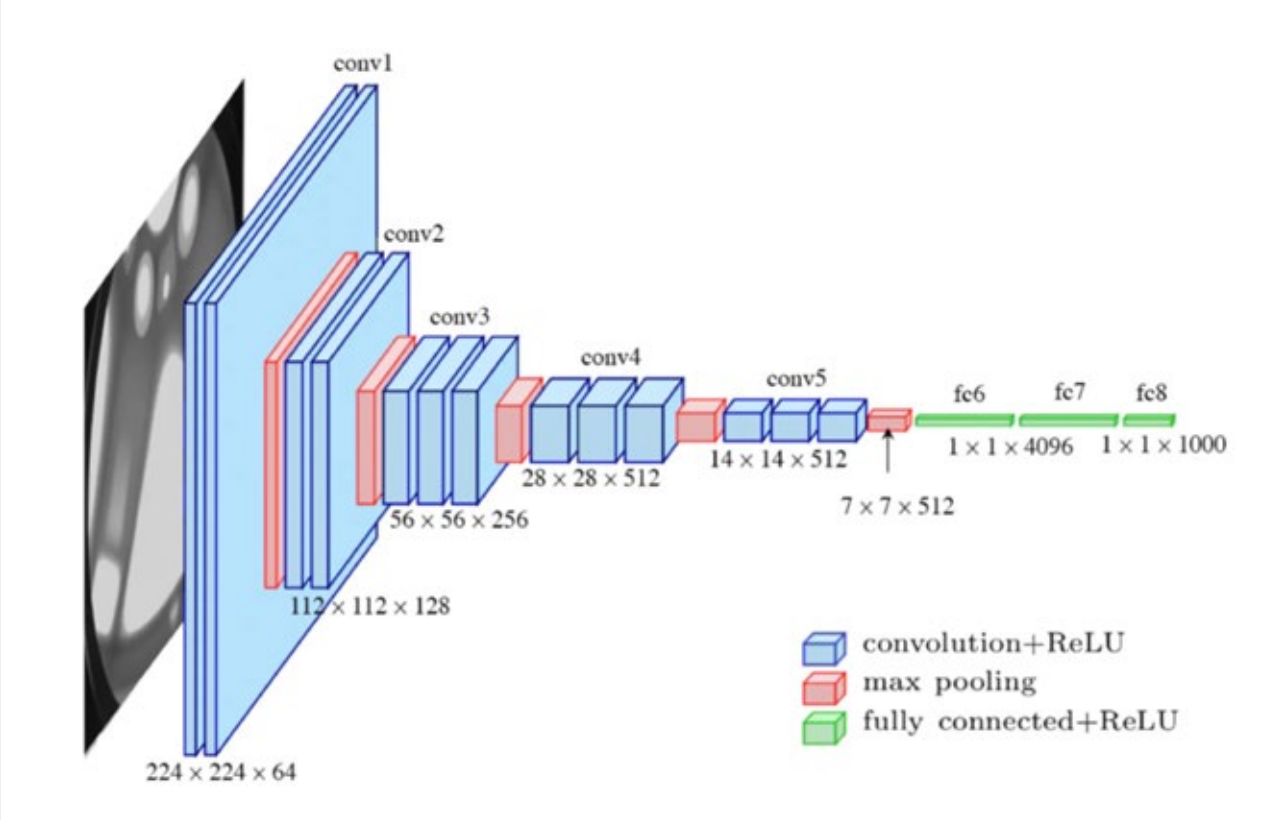
過去數十年間，研究者們開發了種類繁多的神經網路，如卷積神經網路 (CNN) 和遞歸神經網路 (RNN) 等，為解決各種複雜問題提供了強大的工具。隨著網際網路數據量的爆炸式增長和硬體計算能力的提升，神經網路技術的應用也變得更加容易，進一步推動了其發展。

VGG16 便是其中一種表現出色的卷積神經網路 (CNN) 架構，以其簡單而有效的架構聞名，在圖像分類和目標偵測等電腦視覺任務中表現出色。其簡單性、高效能和可擴展性使其成為電腦視覺應用的熱門選擇。

圖像識別和分類作為影像科學和工程領域中最活躍的領域之一，其主要目標是根據感應器或偵測器收集的數據，對圖像場景進行分析和分類。這項技術可以應用在人臉辨識、醫療影像診斷、自動駕駛等領域，為人們帶來更便利、安全的生活體驗，也可以應用於產品質檢、智慧安防、零售分析等產業，提升效率並創造新的商業價值。

人工智慧的發展日新月異，機器學習作為其核心技術，必將在未來扮演更加重要的角色。我們有理由相信，隨著技術的不斷進步，機器學習將會替人類社會帶來更多的便利。

第二章：研究目的 02



▲ VGG16 的模型

VGG16 是一種卷積神經網路 (CNN) 架構，由牛津大學視覺幾何組 (VGG) 的 Karen Simonyan 和 Andrew Zisserman 於 2014 年開發。它以其簡單而有效的架構而聞名，在圖像分類和目標偵測等各種電腦視覺任務中表現出色，VGG16 簡化了深度學習模型的開發過程，並提供了一個更易於使用和理解的介面，使其更易於初學者和經驗豐富的開發人員使用。

本研究主要探討以 VGG16 架構為基礎，研究模型在不同參數下圖像分類中的正確率，並建構可以提升正確率的機器學習模型。

主要研究目的如下：

1. **探究 VGG16 架構對圖像分類模型應用於鳥類圖像分類的可行性：**

   本研究將評估 VGG16 架構中提供的模型在圖像分類上的表現，分析其優缺點，並探討如何改進模型以提升辨識率。

2. **透過相關物件影像辨識模型建構鳥類辨識的機器學習模型：**

   本研究將基於 VGG16 架構，設計和訓練一個專門用於鳥類圖像分類的模型，並對其性能進行評估。

3. **深入探究不同參數對於鳥類圖像分類機器學習模型分類效能的差異：**

   本研究將本研究將基於 VGG16 架構，分析不同參數對於辨識結果的差異性。

第三章、文獻探討 03

本研究主要探討以 VGG16 架構為基礎，研究模型在不同參數下圖像分類的正確率，並建構可以提升正確率的機器學習模型。本章節將針對本研究應用到的相關技術及發展進行介紹，以便為後續實驗探討奠定基礎。

一、人工智慧

「人工智慧」的概念由美國科學家 John McCarthy 於 1955 年提出，其目標是讓電腦具備類似人類的學習和解決複雜問題的能力，包括抽象思考、展現創意、推理、規劃、學習、交流、感知和操作物體等。換言之，人工智慧目的在創造能夠像人類一樣思考和行動的機器。

然而由於當時電腦體積龐大、性能有限，人工智慧發展很快遭遇瓶頸，進入一段被稱為「人工智慧寒冬」的時期。沉寂一段時間後，約二、三十年前，隨著電腦儲存空間和運算性能的突破，人工智慧重新成為主流技術發展重點。結果使得人工智慧的重要分支機器學習取得了突破性的成果，例如支持向量機 (SVM) 模型的發展，有效提升了各種數據的分類處理能力，為人工智慧的應用奠定了基礎。

近年來，隨著技術和演算法的進步，深度學習因爲以新的演算法透過模仿人腦神經網路的架構，訓練多層神經網路模

型，在語音識別、視覺物件識別、自然語言處理等領域有了突破性的進展，推動了人工智慧應用的蓬勃發展。

人工智慧的應用領域非常廣泛，從日常生活的個人語音助理如蘋果 Siri、微軟 Cortana，到工業生產中的機器人控制、自動化生產線，再到醫療領域的疾病診斷、藥物研發，人工智慧技術正在改變著各行各業。例如，分別擊敗人類西洋棋和圍棋高手的 IBM Deep Blue 和 Google DeepMind AlphaGo 都是人工智慧研究的成果，展現了人工智慧在複雜決策和策略規劃方面的強大能力。

可以預見，隨著人工智慧技術的持續發展，其應用範圍將會更加廣泛，並對人類社會產生更加深遠的影響。

二、機器學習

機器學習（Machine Learning）通常被定義為：「透過從過往的資料和經驗中學習並找到其運行規則，最後達到人工智慧的方法。」機器學習的核心在於透過樣本訓練機器辨識出運作模式，而不是用特定的規則來編程。這些樣本可以在資料中找到。換句話說，機器學習是一種弱人工智慧（Narrow AI），它從資料中提取複雜的函數或樣本，來學習和創造算法或規則，並利用這些規則進行預測。

機器學習主要分為三大類：

1. 監督式學習 (Supervised learning)：

在機器學習過程中，由人工的方式將資料貼上標籤 (Label)，告訴機器相對應的值，像是提供標準答案，讓機器學習在輸出時判斷誤差。這種方法對電腦來說最簡單，對人類來說最辛苦，但是正確性比較高。例如，要訓練機器區分貓和狗，則任意選出 100 張貓和狗的照片，並且標註哪些是貓哪些是狗，輸入電腦後讓電腦學習認識貓與狗的外觀特徵，依照特徵就能辨識出貓和狗並進行預測。

2. 非監督式學習 (Unsupervised learning)：

在機器學習過程中，不以人工方式將資料貼上標籤，所有資料都沒有標準答案，機器必須自己尋找答案，預測時比較不準。這種方法不必人工分類，對人類來說最簡單，但是對電腦來說最辛苦，而且判斷誤差比較大。例如，任意選出 100 張照片但是沒有標註，輸入電腦後讓電腦學習認識貓與狗的外觀，因為照片沒有標註，因此電腦必須自己嘗試把照片內的特徵取出來，同時自己進行分類。

3. 半監督式學習 (Semi-supervised learning)：

指少部分資料有標準答案，可提供機器學習輸出判斷誤差使用；大部分資料沒有標準答案，機器必須自己尋找答案，等於是結合監督式與非監督式學習的優點。例如，任意選出 100 張照片，其中 10 張標註哪些是貓哪些是狗，輸入電

腦後讓電腦學習認識貓與狗的外觀，電腦只要把照片內的特徵取出來，再自己嘗試把另外 90 張照片內的特徵取出來，同時自己進行分類。這種方法只需要少量的人工分類，又可以讓預測時比較精準，是目前最常使用的一種方式。

本研究採用監督式學習的方式，將公開可用的鳥類影像資料集，放入機器學習模型中，運用 VGG16 架構訓練，進而發展出提升辨識率的模型。

三、深度學習

深度學習是機器學習的一個分支，其目標是透過多個處理層 (Layer) 中的線性或非線性轉換 (Linear or Non-linear Transform)，自動提取足以代表資料特性的特徵 (Feature)。

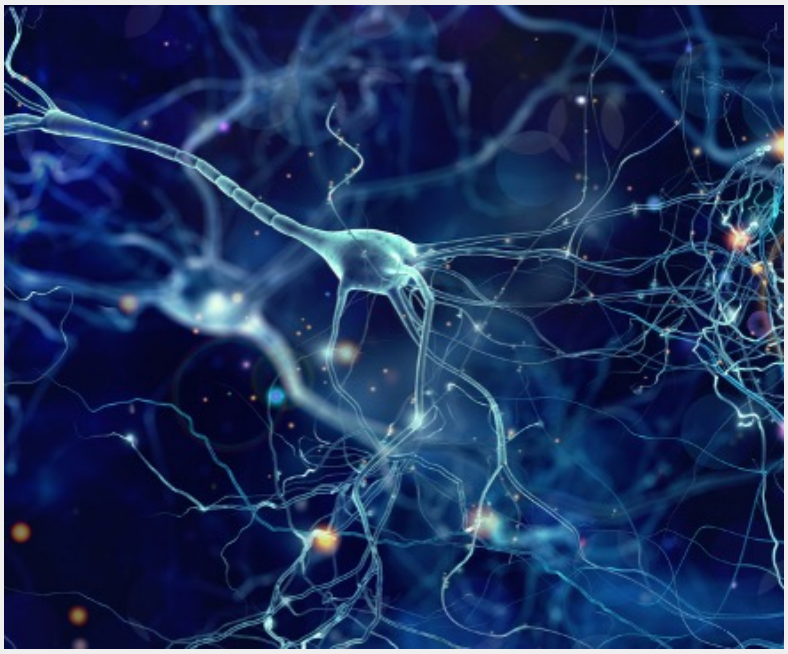
在傳統的機器學習中，特徵通常由人工設計的演算法產生，需要領域專家對資料進行大量的分析和研究，才能產生有用且效果良好的特徵。這個過程稱為特徵工程 (Feature Engineering)。深度學習則具有自動提取特徵的能力，也被視為是一種特徵學習 (Feature Learning, Representation Learning)，可以取代專家進行特徵工程所花費的時間，大幅提升了機器學習模型的開發效率。

憑藉强大的自動特徵提取能力，深度學習在以往機器學習難以突破的應用領域取，例如圖像識別、語音識別、自然語言

處理等，有了新的突破。

簡單來說，深度學習可以被形容為一種「比較深」的類神經網路，並搭配了各式各樣特殊的類神經網路階層，如卷積神經網路、遞歸神經網路等等。這些特殊的階層架構賦予了深度學習模型强大的學習能力，使其能夠從複雜的數據中提取有效的特徵。因此，一些深度學習架構也被稱為深度神經網路 (Deep Neural Network, DNN)。

四、類神經網路



▲人類的神經網路示意圖

類神經網路 (Neural Network, NN) 是一種模仿生物神經系統的數學模型。在類神經網路中，通常會有數個階層，每個階層中會有數十到數百個神經元 (Neuron)。神經元會將上一層神經元的輸入加總後，進行活化函數 (Activation Function) 的

轉換，作為神經元的輸出。每兩個節點之間都需要一定的加權值，用來模擬記憶強度，稱為權重 (Weight)。

經過一定「深度」的節點計算之後，得到一系列的輸出值，這個輸出與期望值進行匹配之後可以用來調整函數和權重，從而更加逼近預期結果。將這個學習得來的神經網路模型應用於新的輸入（即進行預測），往往也能得到類似的結果，這樣就達到了機器學習的目的。

經過一定「深度」的節點計算之後，得到一系列的輸出值，這個輸出與期望值進行匹配之後可以用來調整函數和權重，從而更加逼近預期結果。將這個學習得來的神經網路模型應用於新的輸入（即進行預測），往往也能得到類似的結果，這樣就達到了機器學習的目的，神經網路的輸出結果依網路的連接方式、權重值和激活函數的不同而有所差異。類神經網路的架構指的就是階層數量、每層中的神經元數量、各層之間神經元的連接方式、及活化函數的類型等設定。這些參數設定都是在使用類神經網路前需要由人力設定好的，參數設定的好壞也是影響到類神經網路的效能表現。

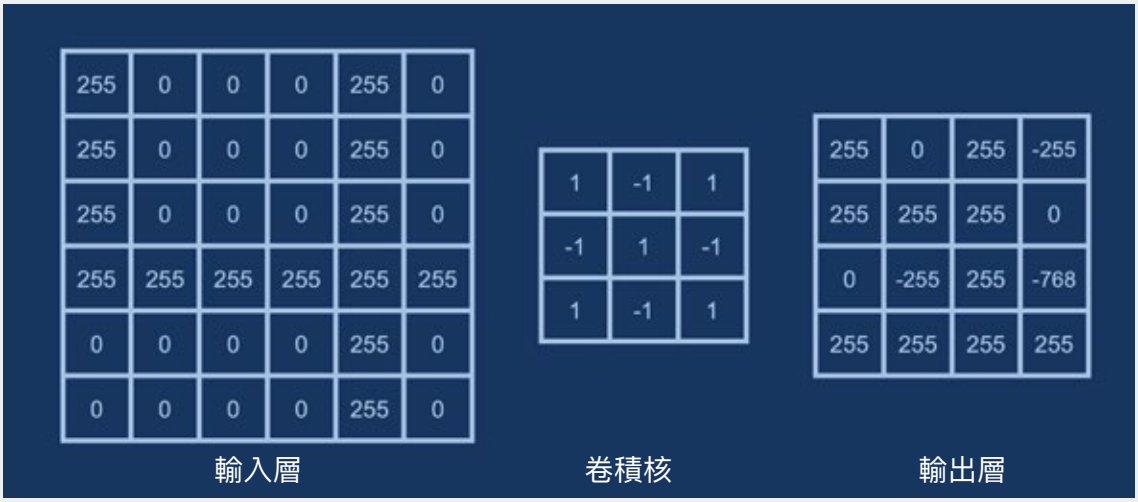
類神經網路的學習和訓練過程就是試著找到最佳的權重設定。透過不斷調整權重，使得神經網路的輸出結果更加接近預期結果，最終訓練出一個能夠完成特定任務的模型。深度學習正是利用了多層類神經網路的架構，透過大量的訓練數據和複雜的算法，自動學習有效的特徵，並完成各種複雜的任務。

五、卷積神經網路

卷積神經網路 (Convolutional Neural Network, CNN) 又被稱為 CNNs 或 ConvNets，它是目前深度神經網路領域的發展主力，在圖片辨別上甚至可以做到比人類還精準的程度。

卷積網路流程如下：

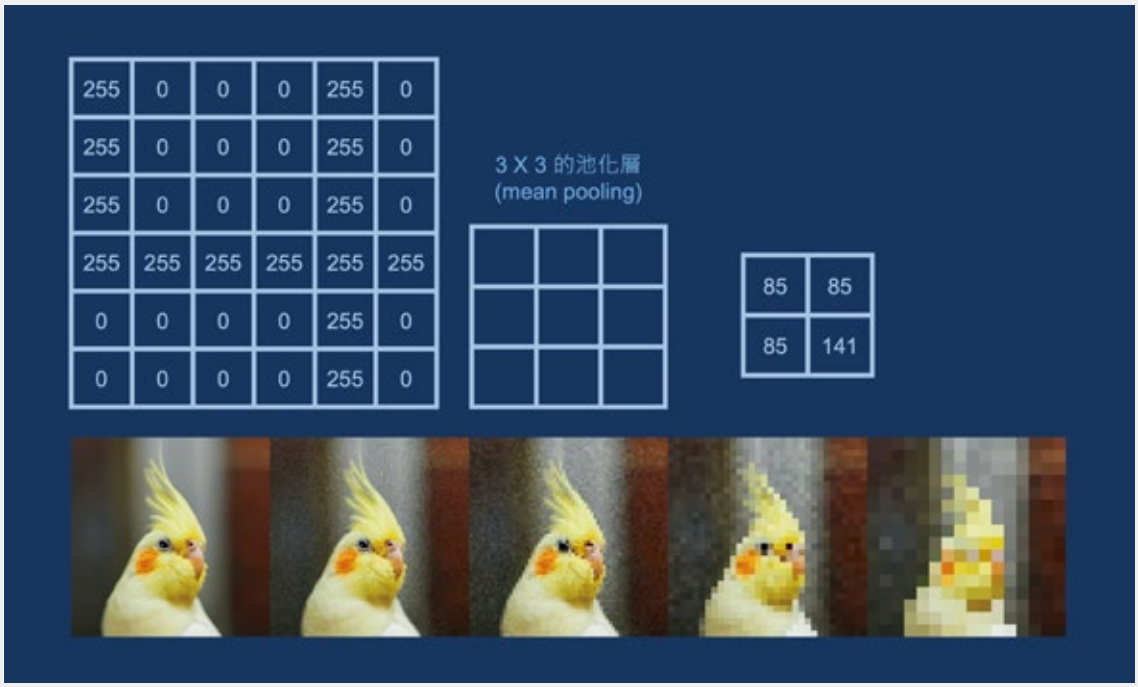
1. 卷積層 (Convolutional Layer)：



▲卷積層的輸出

卷積運算將原始圖片與特定的卷積核 (Kernel)，也稱為 Filter 做運算。每次矩陣移動 1 個位置（左到右和上到下都移動 1)，而這移動稱為「Stride」，最後結果為卷積層的輸出。卷積層的作用是提取圖像中的局部特徵，例如邊緣、紋理等。

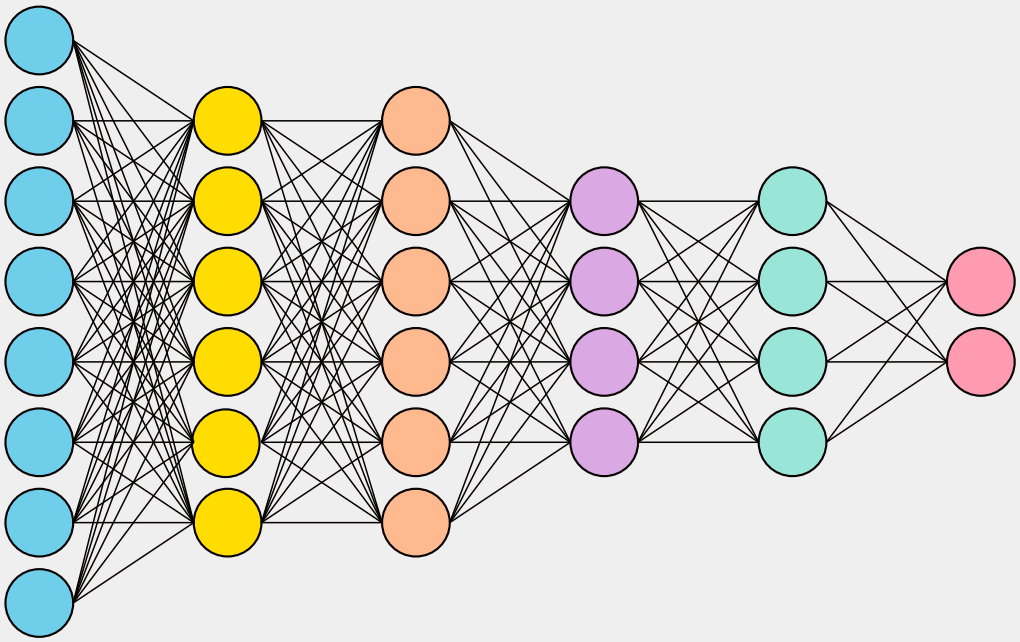
2. 池化層 (Pooling Layer)：



▲池化

池化層跟卷積層一樣，有個 Filter（大多為 2x2)，對卷積層出來的特徵圖 (Feature Map) 做運算，也就是對特徵圖做降維，並且保留重要的特徵。池化常用的方式有兩種，Max Pooling 和 Mean Pooling。以 Max Pooling 為例，利用 2x2 的 Filter 在特徵圖上用 Stride 為 2 進行 Pooling，在 Filter 經過的地方，取出最大值。Max Pooling 主要的好處是若原始資料有些微旋轉，在區塊內選取到的還是會是同一個值，這也就是他的特性之一。池化層的作用是降低特徵圖的維度，並保留重要的特徵，同時提高模型的魯棒性。

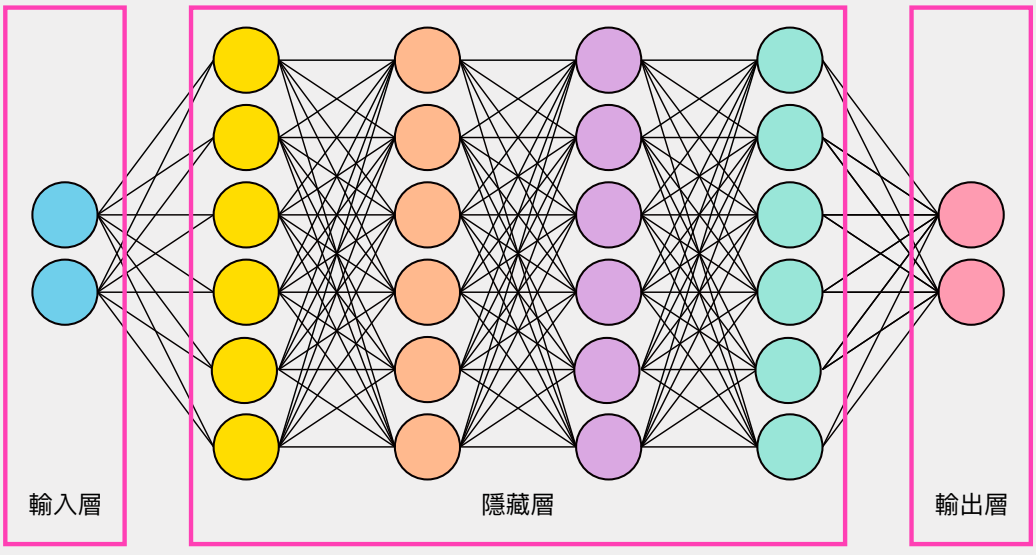
3. 全連接神經網路層 (Fully Connected Layer)：



▲全連結神經網路層

將池化層的輸出展平成一維向量，並輸入到全連接神經網路層進行分類。全連接層的作用是整合提取到的特徵，並進行最終的分類。

4. 輸出層 (Output Layer)：

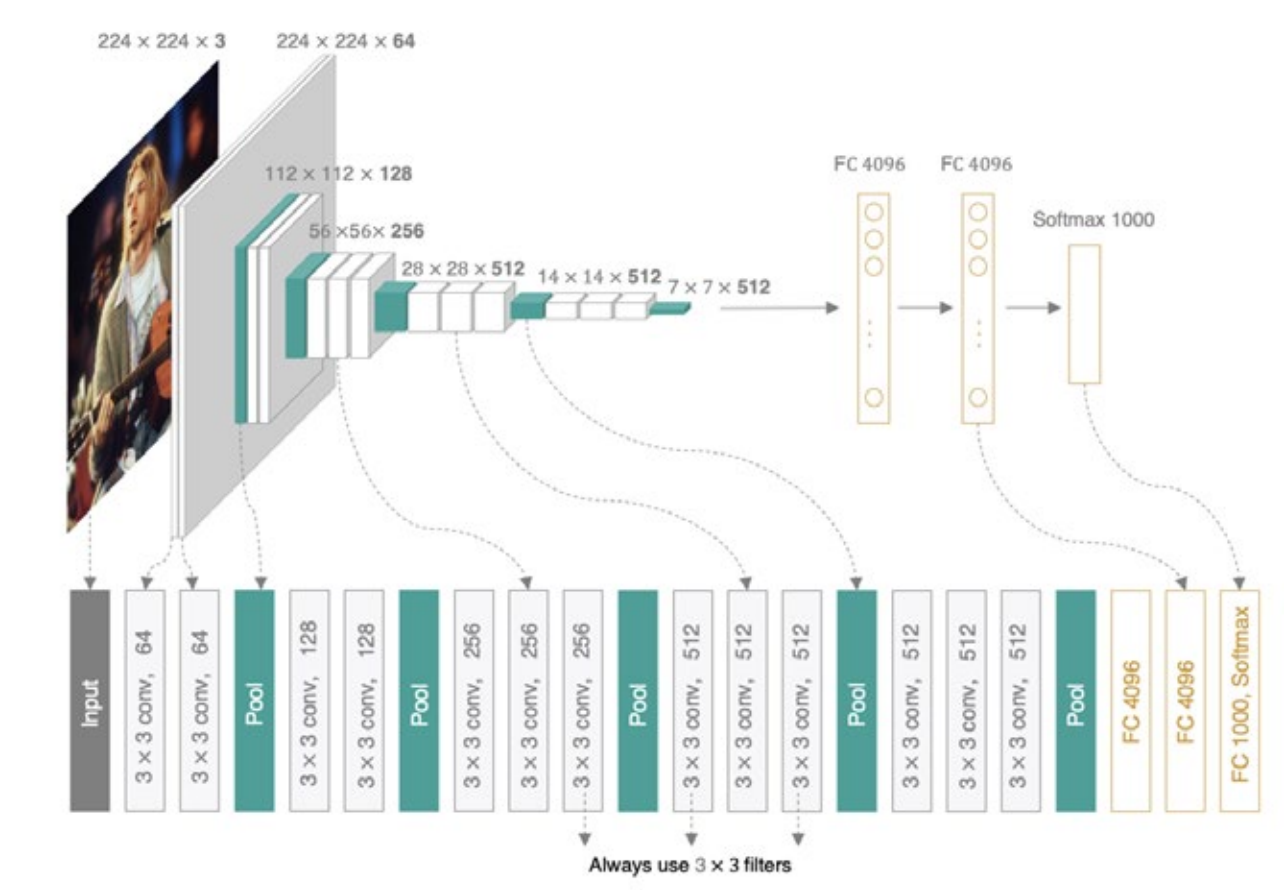


▲輸出層

輸出層的節點數量與分類的類別數量相同，每個節點代表一個類別的預測概率。

卷積神經網路在圖像辨識領域是目前最常用的深度學習模型之一。其强大的特徵提取能力和分類能力，使其在各種圖像相關的任務中都表現出色，例如物體檢測、人臉識別、圖像分割等。

六、VGG16 架構



▲ VGG16 架構

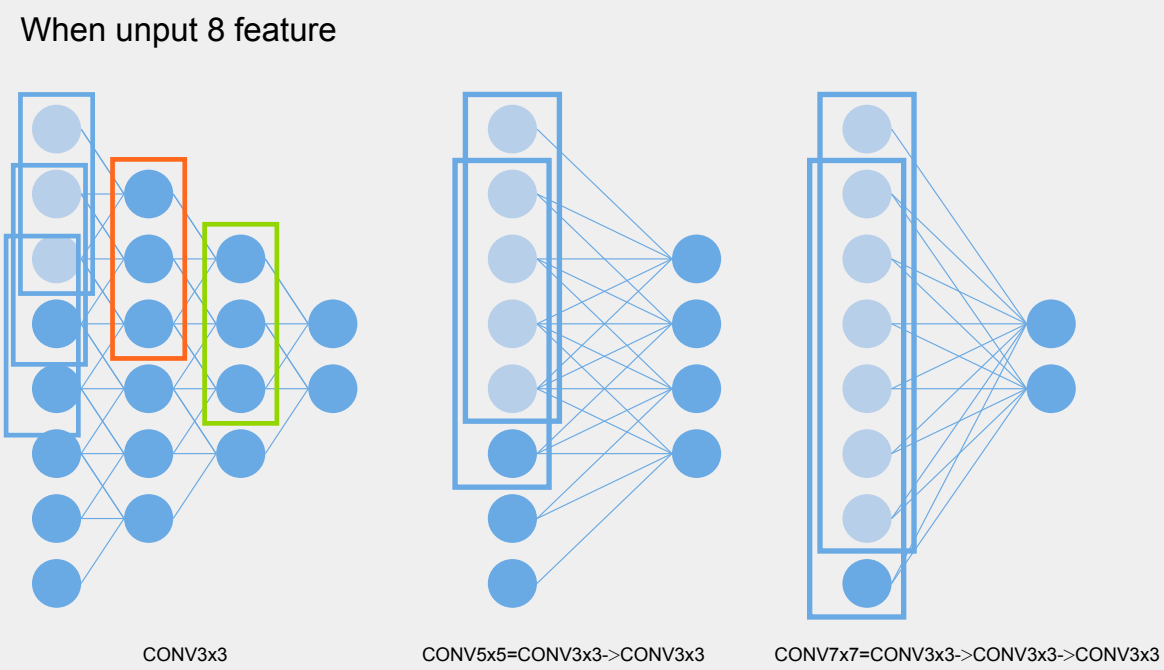
VGG16 是一種卷積神經網路（CNN）架構，由牛津大學視覺幾何組（VGG）的 Karen Simonyan 和 Andrew Zisserman

於 2014 年開發。它以其簡單而有效的架構而聞名，在圖像分類和目標偵測等各種電腦視覺任務中表現出色。

VGG16 的架構設計非常簡潔，主要由一系列卷積層、池化層和全連接層組成，它僅使用 3x3 卷積核和 2x2 最大池化層，並以順序方式堆疊這些層。這種簡單性使得 VGG16 易於理解和實現。

1. 卷積層：

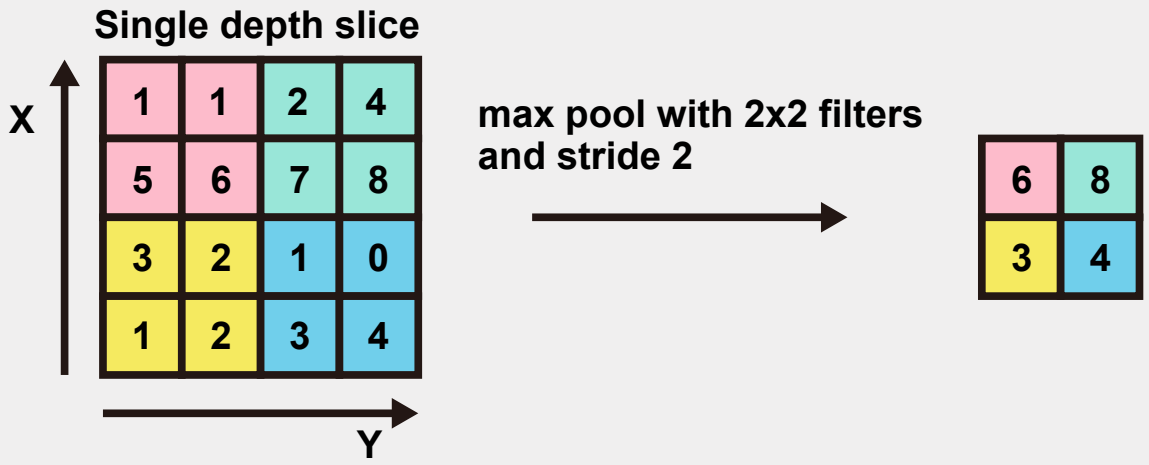
VGG16 統一使用 3x3 的卷積核，並堆疊多個卷積層以增加接受區 (Receptive field)。



▲ 3x3 卷積核

2. 池化層：

池化層用於降低特徵圖的空間解析度，並引入非線性。VGG16 使用最大池化層統一為 2x2。



▲ 2x2 池化層

3. 全連接層：

全連接層用於將特徵圖轉換為最終的類別預測，全連結層就是最後的分類器，將前面所擷取出來的特徵，經過權重的計算之後，來辨識出這個所輸入的圖像到底屬於哪一個分類，VGG 架構中有 3 個全連接層。

VGG16 在各種電腦視覺任務中表現出色，因此常應用於：

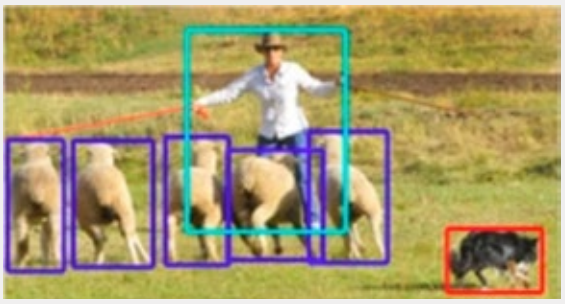
1. 圖像分類：

VGG16 可以用於對圖像進行分類，例如識別圖像中的物體類別、場景類別等。

▲圖像分類

2. 目標偵測：

VGG16 可以用於檢測圖像中的目標物體，例如人臉、車輛、動物等。



▲目標偵測

3. 圖像分割：

VGG16 可以用於將圖像分割成不同的區域，例如前景和背景、不同類型的物體等。

▲圖像分割

Vgg16 憑借其簡潔的架構、強大的特徵提取能力、良好的遷移學習性能、開源性和易於獲取等優點，成為深度學習領域

中一個重要的模型，並為後續 CNN 模型的發展奠定了基礎。

需要注意的是，VGG16 也存在一些缺點，例如參數量較大、計算成本較高、容易過擬合等。但是，這些缺點可以通過模型剪枝、量化、正則化等一些技術手段來緩解。

VGG16 是一種功能強大的 CNN 架構，在圖像分類領域中表現出色。其簡單性、高效能和可擴展性使其成為電腦視覺應用的流行選擇。

第四章、研究方法與研究架構 04

本研究主要探討應用 VGG16 架構於物件影像辨識中的正確率。

一、研究架構

本研究架構主要分為六個階段，分別為：

1. 準備資料集：

收集一個包含要辨識的鳥類類型的影像資料集。資料集應分為訓練集、驗證集和測試集。

2. 資料預處理：

將資料集內的圖片，調整成特定規格，以利之後訓練使用。

3. 使用 VGG16 架構建立模型：

使用 VGG16 架構作為模型的基礎，從頭開始訓練模型。

4. 測試模型：

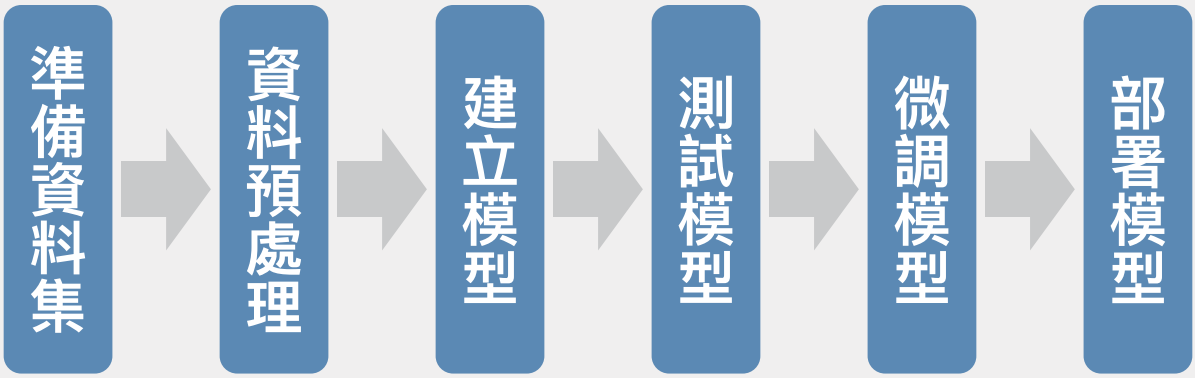
使用測試資料對模型進行測試，評估模型的辨識性能。

5. 微調模型：

調整模型中的參數，以提高其在最終的正確率。

6. 部署模型：

將訓練好的模型部署到應用程式中。



▲研究架構

二、研究軟體設備

本研究使用的軟體環境如下：

1. Google Colab：

Colab 全名是 Google Colaboratory，是一個基於雲端的 Jupyter 筆記本環境，讓使用者可以在瀏覽器上寫程式碼，並且可以利用 Google 的雲端運算資源執行程式碼，Google Colab 的主要優勢在於其易用性、訪問性和免費的資源，成為一個理想的選擇。

三、準備資料集

針對鳥類辨識資料準備過程包括：

1. 資料收集：

從公開資料集或自行拍攝收集鳥類影像資料。

2. 資料標記：

使用標記工具對鳥類影像資料進行標記，標出鳥類的種類。

3. 使用鳥類影像資料集：

使用公開可用的鳥類影像資料集，本研究使用 kaggle 上的 bird 400 資料集。

四、資料預處理

對鳥類影像資料進行預處理，例如調整大小、裁剪、增强等，以便訓練模型。

1. 調整大小 :

將所有圖片調整成為 112x112 的大小。

2. 調整顏色格式 :

將 BGR 格式調整為 RGB 格式。

五、建立模型

本研究使用 VGG16 架構作為模型。

六、評估模型

使用驗證集評估模型的辨識率，以了解模型在未見過資料上的表現。

七、測試模型

使用測試資料對模型進行測試，評估模型的辨識正確率，並分析模型的優缺點。

八、調整參數

1. 使用資料增強：

資料增強是一種有效的方法，通過對訓練數據進行旋轉、翻轉、平移等操作，來增加數據的多樣性，有助於提高模型的泛化能力和準確率。

2. 減少種類數目：

在相同的資料集下減少需辨識的種類數，可以降低模型的複雜度，提高訓練效率和預測準確率。然而，需要注意的是，減少種類數目也可能會減少模型的應用範圍。

3. 變更批次大小：

批次大小是指每次訓練時用來更新模型參數的樣本數量。調整批次大小可以影響模型的訓練速度和性能。通常情況下，較大的批次大小可以提高訓練速度，但可能會導致模型收斂到局部極值；較小的批次大小則可能更有助於模型避免陷入局部極值，但訓練速度較慢。因此，根據具體情況和資源限制，可以調整批次大小以獲得更好的性能。

九、部署模型

將訓練好的模型部署到應用程式中。

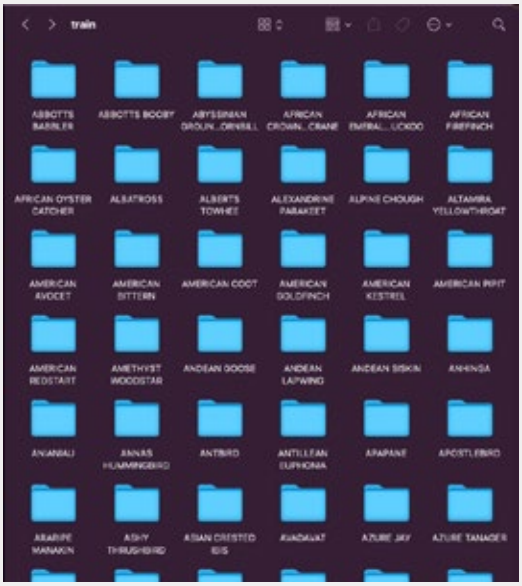
第五章、訓練與實作 05

一、準備資料集

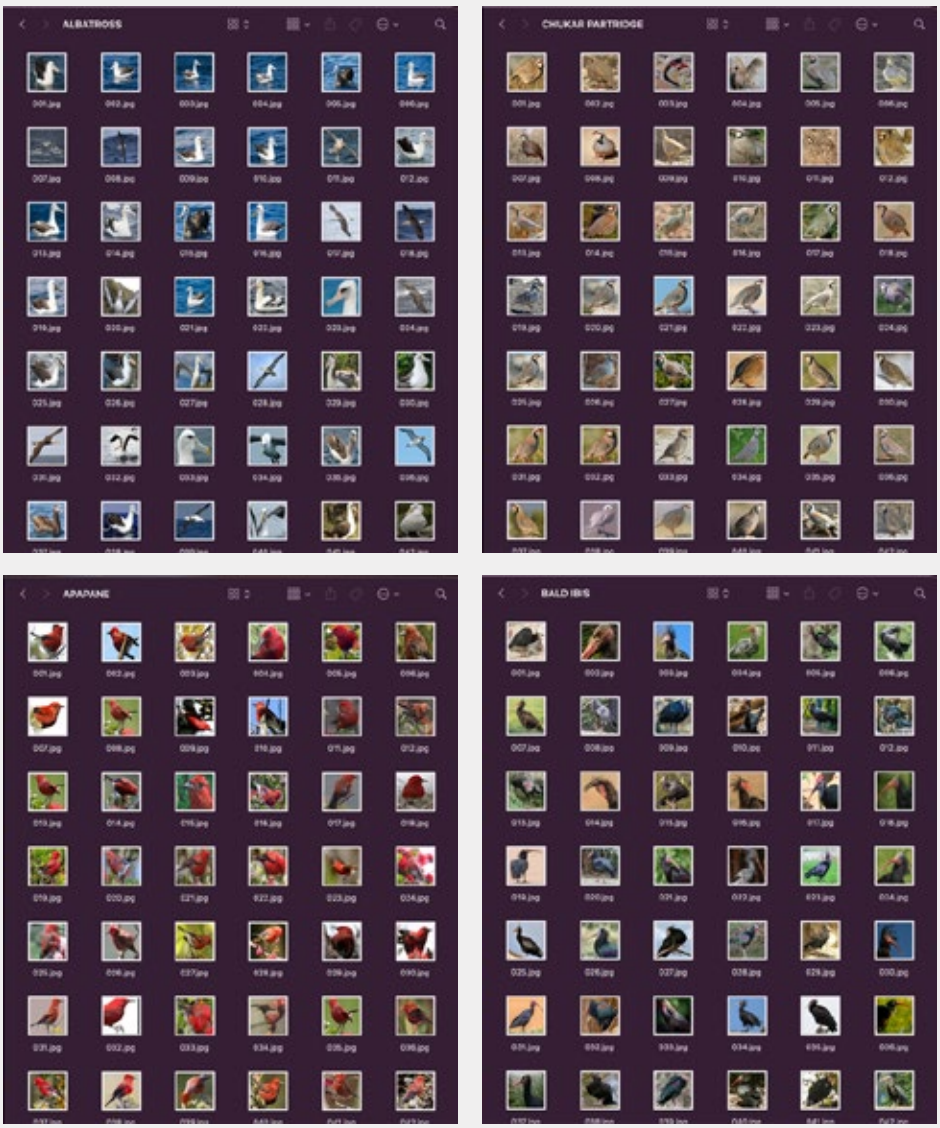
步驟包含資料收集、資料標記、使用鳥類影像資料集。本研究使用 kaggle 上的 bird 400 資料集。

class_index	class	SCIENTIFIC NAME
0	ABBOTTS BABBLER	Malacocincla abbotti
1	ABBOTTS BOOBY	Papasula abbotti
2	ABYSSINIAN GROUND HORNBILL	Bucorvus abyssinicus
3	AFRICAN CROWNED CRANE	Balearica regulorum
4	AFRICAN EMERALD CUCKOO	Chrysococcyx cupreus
5	AFRICAN FIREFINCH	Lagonosticta rubricata
6	AFRICAN OYSTER CATCHER	Haematopus moquini
7	ALBATROSS	Diomedeidae
8	ALBERTS TOWHEE	Pipilo aberti
9	ALEXANDRINE PARAKEET	Psittacula eupatria
10	ALPINE CHOUGH	Pyrrhocorax graculus
11	ALTAMIRA YELLOWTHROAT	Geothlypis flavovelata
12	AMERICAN AVOCET	Psittacula eupatria
13	AMERICAN BITTERN	Botaurus lentiginosus
14	AMERICAN COOT	Fulica americana
15	AMERICAN GOLDFINCH	Spinus tristis
16	AMERICAN KESTREL	Falco sparverius
17	AMERICAN PIPIT	Anthus rubescens
18	AMERICAN REDSTART	Setophaga ruticilla
19	AMETHYST WOODSTAR	Calliphlox amethystina
20	ANDEAN GOOSE	Chloephaga melanoptera
21	ANDEAN LAPWING	Vanellus resplendens
22	ANDEAN SISKIN	Spinus spinescens
23	ANHINGA	Anhinga anhinga
24	ANIANIAU	Magumma parva
25	ANNAS HUMMINGBIRD	Calypte anna
26	ANTBIRD	Thamnophilidae
27	ANTILLEAN EUPHONIA	Euphonia musica
28	APAPANE	Himatione sanguinea
29	APOSTLEBIRD	Struthidea cinerea
30	ARARIPE MANAKIN	Antilophia bokermanni

▲ birds latin names.csv 包含本次使用資料集內的鳥類種類



▲本次使用資料集所包含的 400 種鳥類圖片資料夾（節錄）



▲資料夾內部分鳥類圖片

二、建置 Google Colab 雲端開發環境

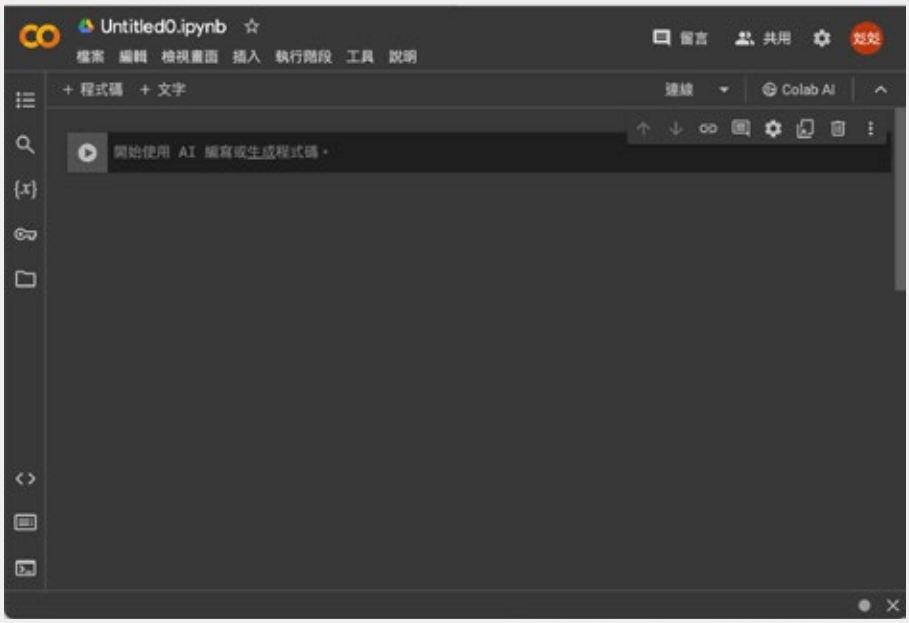
本研究使用 Google Colab 雲端開發環境來執行訓練模型。

1. 開啟網頁瀏覽器：

■前往 https://colab.research.google.com/

■使用 Google 帳戶登入。

■點擊 " 新增筆記本 " 或開啟現有的筆記本。



▲ Google Colab 雲端開發環境

2. 檢查運行環境 (Runtime)：

■點擊 " 執行階段 " 選單。

■選擇 " 變更執行階段類型 "。

■選擇 " 硬體加速器 "："GPU" 適合深度學習訓練和 GPU 密集型任務。

■點擊 " 儲存 "。

三、資料上傳 Google 雲端硬碟

首先需將鳥類圖片資料夾資上傳至 Google 雲端硬碟，再將 Google 雲端硬碟的鳥類圖片資料導入到 Google Colab 內。

```
from google.colab import drive
drive.mount('/content/drive')
import os
os.chdir('/content/drive/MyDrive/Bird_Data/archive')
os.listdir()
```

▲雲端硬碟的資料導入到 Google Colab 內

四、載入所需函式庫

在 Python 中，載入函式庫可以節省時間和精力、提高代碼的簡潔性、充分利用現有的函式庫。

```
import pandas as pd
import cv2
from tensorflow import keras
from glob import glob
import os
import numpy as np
```

▲載入函式庫

■ pandas

用於數據操作和分析的函式庫。

■ cv2

OpenCV 函式庫，是一個用於圖像處理和計算機視覺的函式庫。

■ tensorflow

Google 開發的深度學習框架，用於訓練深度學習模型。

■ keras

提供了簡單易用的 API，使得訓練深度學習模型變得更加容易。

■ glob

用於文件模式匹配的 Python 模板，方便對文件進行批量處理。

■ **os**

Python 的內置模塊，包括文件操作、路徑操作、環境變量等。

■ **numpy**

用於數值計算的 Python 函式庫，提供了多維數組操作功能。

五、資料預處理

將 bird 400 資料集內的鳥類圖片轉為指定大小、格式，並加上標記後儲存。

```
birds_latin_names=pd.read_csv('birds latin names.csv')
idx_to_class=birds_latin_names.iloc[:n,1].values

class_count=len(idx_to_class)

tmp=[(idx_to_class[i],i) for i in range(class_count)]
class_to_idx=dict(tmp)
```

▲讀取名為 birds latin names.csv 的 CSV 檔案，
然後創建一個索引到類別名稱的字典

■ **birds_latin_names=pd.read_csv('birds latin names.csv')**

從 CSV 檔案中讀取資料，並將其存儲在名為 birds_latin_names 內。

■ **idx_to_class=birds_latin_names.iloc[:200,1].values**

DataFrame 的前 n 列和第 2 列，並將其轉換為 NumPy 陣列，存儲在 idx_to_class 中。

■ **class_count=len(idx_to_class)**

計算 idx_to_class 陣列中的元素個數，即類別的數量，並將其存儲在 class_count 中。

■ **tmp=[(idx_to_class[i],i) for i in range(class_count)]**

創建一個名為 tmp 的列表，其中每個元素都是一個元組，包含類別名稱和相應的索引值。

■ **class_to_idx=dict(tmp)**

使用 dict() 函式將 tmp 列表轉換為一個字典，最後將生成的字典存儲在 class_to_idx 中。

```
def load_data_set(path:str,data_set:str,class_list:list):
    x=[]
    y=[]
    for class_name in class_list:
        path_list=glob(os.path.join(path,data_set,class_name,'*'))
        for tmp in path_list:
            img=cv2.imread(tmp)
            img=cv2.resize(img,(112,112))
            img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
            x.append(img)
            y.append(class_to_idx[class_name])
    y=np.array(y)
    y=keras.utils.to_categorical(y,class_count)
    return np.array(x),y
```

▲此函式 load_data_set 的目的是從指定路徑中載入圖像數據集，並將其準備成用於訓練的格式

■ **def load_data_set(path:str,data_set:str,class_list:list):**

定義了一個函式，包含了三個參數，分別是 path（數據集所在的路徑）data_set（數據集的名稱或子目錄）和 class_list（包含數據集中所有類別名稱的列表）。

■ **x=[]**

y=[]

創建了兩個空列表 x 和 y，用於存儲輸入圖像和對應的標籤。

■ **for class_name in class_list:**

遍歷 class_list 中的每個類別名稱。

■ **path_list=glob(os.path.join(path,data_set,class_name,'*'))**

使用 glob 函式找到指定類別下所有圖像的路徑列表，且使用 os.path.join() 於拼接路徑。

■ **for tmp in path_list:**

遍歷 path_list 中的每個圖像路徑。

■ **img=cv2.imread(tmp)**

使用 OpenCV 讀取圖像。

■ **img=cv2.resize(img,(112,112))**

調整圖像大小為 (112, 112)。

■ **img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)**

將圖像通道從 BGR 轉換為 RGB。

■ **x.append(img)**

y.append(class_to_idx[class_name])

將調整大小後的圖像添加到 x 中，並將相應的類別索引添加到 y 中。

■ **y=np.array(y)**

將 y 轉換為 NumPy 陣列。

■ **y=keras.utils.to_categorical(y,class_count)**

將類別索引轉換為 one-hot 編碼形式，使用 keras.utils.to_categorical 函式實現這個轉換。

■ **return np.array(x),y**

回傳調整大小後的圖像陣列 x 和 one-hot 編碼的標籤 y。

```
x_train,y_train=load_data_set('','train',idx_to_class)
x_test,y_test=load_data_set('','test',idx_to_class)
x_valid,y_valid=load_data_set('','valid',idx_to_class)
```

▲用於載入訓練、測試和驗證數據集，並將它們準備成訓練模型所需的格式

■ **x_train,y_train=load_data_set('','train',idx_to_class)**

載入訓練數據集。'' 是數據集所在的路徑，在這裡是空字串，意味著數據集應該位於當前工作目錄下，'train' 是指定的數據集名稱，idx_to_class 是類別名稱的索引到類別名稱的映射。函式將載入數據集並將其準備成訓練模型所需的格式，最後將輸入和目標值分別存儲在 x_train 和 y_train 中。

■ **x_test,y_test=load_data_set('','test',idx_to_class)**

類似於第一行，但是載入的是測試數據集。

■ **x_valid,y_valid=load_data_set('','valid',idx_to_class)**

類似於前兩行，但是載入的是驗證數據集。

```
np.save('x_train200',x_train)
np.save('y_train200',y_train)

np.save('x_valid200',x_valid)
np.save('y_valid200',y_valid)

np.save('x_test200',x_test)
np.save('y_test200',y_test)
```

▲將準備好的訓練、驗證和測試數據集保存為 NumPy 陣列的檔案

■ **print(x_train.shape)**

print(y_train.shape)

將訓練集的輸入數據 x_train 保存為名為 x_train200.npy 的檔案。np.save() 函式用於保存 NumPy 陣列。

■ **np.save('x_train200',x_train)**

np.save('y_train200',y_train)

np.save('x_valid200',x_valid)

np.save('y_valid200',y_valid)

np.save('x_test200',x_test)

np.save('y_test200',y_test)

同理將目標值保存於 y_train，x_valid，y_valid，x_test，y_test。

六、建立模型

本研究使用 VGG16 架構作為模型。

```
def VGG16(data, classes):

    inputs = Input(shape=data)

    base_model = keras.applications.VGG16(weights='imagenet', include_top=False, input_tensor=inputs)


    base_model = keras.Model(inputs, base_model.layers[-1].output)

    base_model.trainable = False
    x = base_model(inputs)

    fc = Flatten()(x)
    fc = Dense(2048, activation='relu')(fc)
    fc = Dense(2048, activation='relu')(fc)

    output = Dense(classes, activation='softmax')(fc)
    model = keras.Model(inputs, output)

    return model

model = VGG16((112,112,3),n)
```

▲建立 VGG16 架構的模型

■ inputs = Input(shape=data)

創建模型的輸入層，其中 data 是輸入數據的形狀。

■ base_model = keras.applications.VGG16(weights='imagenet', include_top=False, input_tensor=inputs)

使用 keras.applications.VGG16 函式建立一個預訓練的 VGG16 模型。

■ base_model = keras.Model(inputs, base_model.layers[-1].output)

創建一個新模型，將輸入層連接到 VGG16 模型的最後一層卷積層。

■ **base_model.trainable = False**

將 VGG16 模型的所有層設置為不可訓練，這樣在訓練時不會更新這些層的權重。

■ **x = base_model(inputs)**

將輸入張量通過 VGG16 模型的特徵提取部分，得到特徵並儲存於 x。

■ **fc = Flatten()(x)**

添加一個展平層，將卷積層的輸出展平為一維向量。

■ **fc = Dense(2048, activation='relu')(fc)**

添加一個全連接層，包含 2048 個神經元，使用 ReLU 激活函數。

■ **fc = Dense(2048, activation='relu')(fc)**

再添加一個全連接層，包含 2048 個神經元，使用 ReLU 激活函數。

■ **output = Dense(classes, activation='softmax')(fc)**

添加一個全連接層作為輸出層，使用 softmax 激活函數，用於進行多類別分類。

■ **model = keras.Model(inputs, output)**

創建最終的模型，將輸入層和輸出層連接起來，形成一個完整的深度學習模型。

■ **return model**

回傳建立好的深度學習模型。

七、模型訓練

利用預先處理的資料訓練鳥類辨識模型。

```
my_callbacks=[
                keras.callbacks.EarlyStopping(
                    patience=10,
                    monitor='val_accuracy',
                    restore_best_weights=True
                )
]
```

▲列表包含了一個回調函式，以防止模型過度擬合。

■ my_callbacks

這是一個回調函式，用於在訓練過程中提前停止訓練，以防止模型過度擬合。

■ keras.callbacks.EarlyStopping

patience=10,

指定在性能不再改善時要等待的時期數。

■ monitor='val_accuracy',

指定要監控的性能指標。我們監控驗證集的準確率 val_accuracy。

■ restore_best_weights=True

在訓練過程中得到的最佳權重。提前停止訓練後，模型將恢復到具有最佳性能的那個時期的權重。

```
result=model.fit(
    datagen.flow(x_train,y_train,batch_size=batch_size),
    validation_data=(x_valid,y_valid),
    batch_size=batch_size,
    epochs=epochs,
    callbacks=my_callbacks,
    shuffle=True
)
```

▲訓練深度學習模型

■ **result=model.fit**

datagen.flow(x_train,y_train,batch_size=batch_size),

使用 datagen 對訓練數據進行批次生成。x_train 是輸入圖像的數據，y_train 是相應的目標值，batch_size 是批次大小。

■ **validation_data=(x_valid,y_valid),**

指定驗證數據集。x_valid 是驗證集的輸入數據，y_valid 是相應的目標值。

■ **batch_size=batch_size,**

指定訓練過程中的批次大小。

■ **epochs=epochs,**

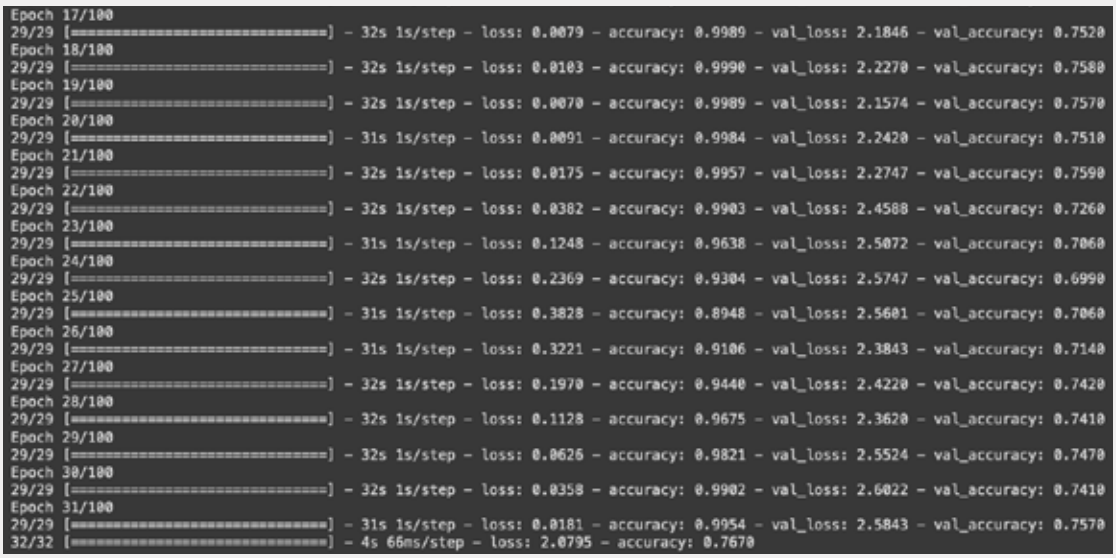
指定整個訓練數據集遍歷的次數。

■ **callbacks=my_callbacks,**

指定一個回調函式，用於在訓練過程中執行額外的操作。

■ **shuffle=True**

在每個時期開始時對訓練數據進行洗牌，增加模型的泛化性。



▲監看模型訓練情況

■ **Epoch 1/100**

表示第一個訓練時期。

■ **57/57 [==========]**

表示在每個訓練時期中，共進行了 57 個批次的訓練。

■ **32s 1s/step**

表示每個批次的平均訓練時間約為 32 秒。

■ **loss: 15.8655**

表示在訓練集上的平均損失值為 15.8655。

■ **accuracy: 0.1075**

表示在訓練集上的平均準確率為 0.1075。

■ **val_loss: 3.2461**

表示在驗證集上的平均損失值為 3.2461。

■ **val_accuracy: 0.3000**

表示在驗證集上的平均準確率為 0.3000。

八、參數調整

對模型內部分參數進行調整，以提高模型對辨識的正確率。

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移
2048	400	TRUE	TRUE	TRUE
1024	200	FALSE	FALSE	FALSE
512	100			
256				
128				
64				
32				
16				

▲本研究所調整之參數

```
batch_size = 512
epochs = 100

datagen=ImageDataGenerator(
    rotation_range=0,
    horizontal_flip=False,
    width_shift_range=0
)
```

▲調整參數最佳化模型的訓練過程，以達到更好的性能。

■ batch_size = 512

指定了在每個訓練步驟中使用的樣本數量。

■ epochs = 100

指定了模型將對整個訓練數據集進行多少次遍歷。

■ datagen=ImageDataGenerator

一個影像數據生成器，在訓練過程中對影像進行數據增強。

■ **rotation_range=0,**

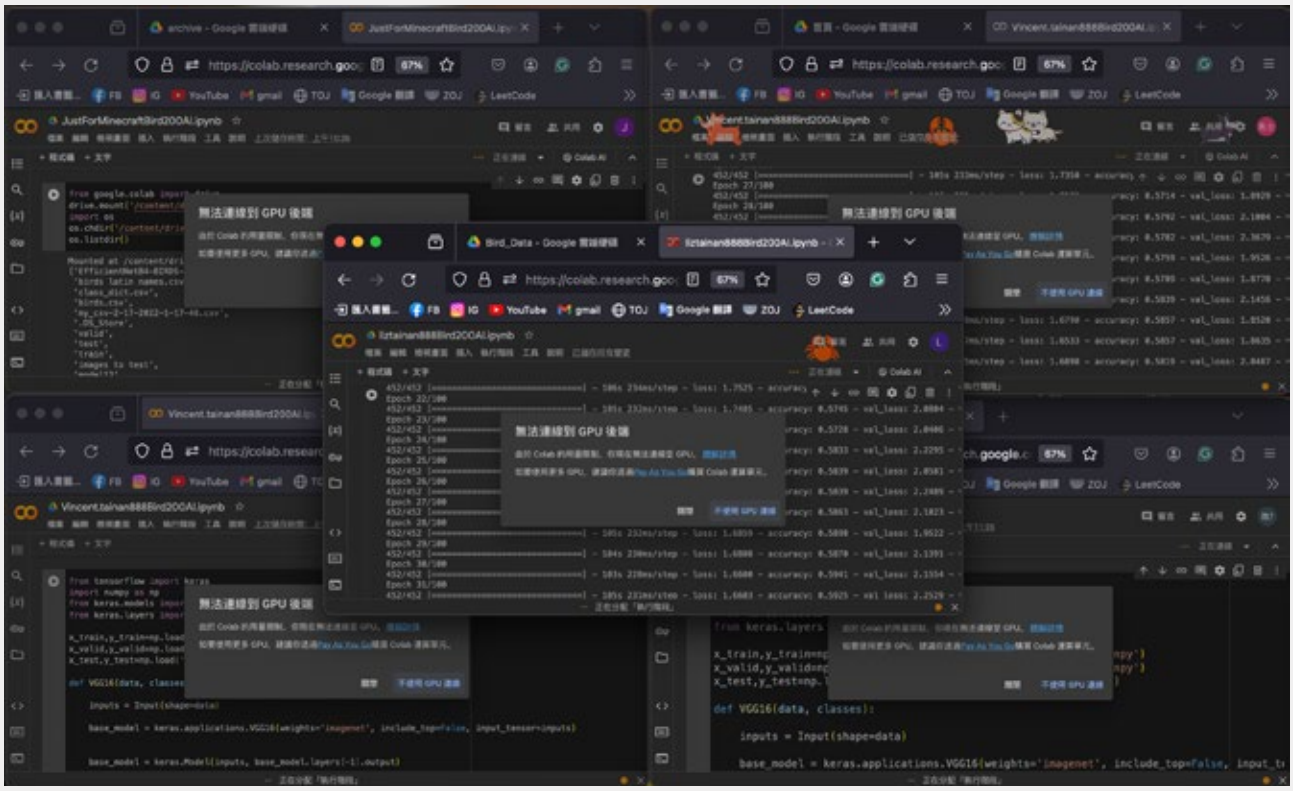
旋轉角度為 0 度。

■ **horizontal_flip=False,**

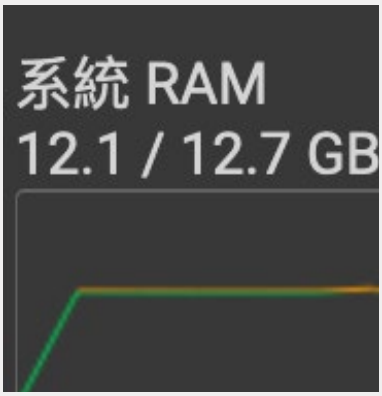
關閉水平翻轉。

■ **width_shift_range=0**

水平平移的範圍為 0。



▲模型的訓練過程中，常會因為雲端 Colab 用量限制無法運算。



▲模型的訓練，需要大量的算力。

第六章、數據收集 06

一、收集訓練後的數據

收集在不同參數設定下的 loss，accuracy，val_loss，val_accuracy 數值。

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
2048	200	TRUE	TRUE	TRUE	10	第一組	ram is dead	ram is dead	ram is dead	ram is dead
						第二組	ram is dead	ram is dead	ram is dead	ram is dead
						第三組	ram is dead	ram is dead	ram is dead	ram is dead
						第四組	ram is dead	ram is dead	ram is dead	ram is dead
						平均	ram is dead	ram is dead	ram is dead	ram is dead
						標準差	ram is dead	ram is dead	ram is dead	ram is dead
						最大值	ram is dead	ram is dead	ram is dead	ram is dead
						最小值	ram is dead	ram is dead	ram is dead	ram is dead
1024	200	TRUE	TRUE	TRUE	10	第一組	0.682	0.807	1.655	0.756
						第二組	0.827	0.767	1.560	0.748
						第三組	0.591	0.829	1.465	0.759
						第四組	0.828	0.767	1.582	0.746
						平均	0.732	0.793	1.565	0.752
						標準差	0.117	0.031	0.079	0.006
						最大值	0.828	0.829	1.655	0.759
						最小值	0.591	0.767	1.465	0.746

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
512	200	TRUE	TRUE	TRUE	10	第一組	0.961	0.738	1.621	0.760
						第二組	1.039	0.733	2.054	0.741
						第三組	0.912	0.749	1.540	0.766
						第四組	0.914	0.749	1.609	0.755
						平均	0.956	0.742	1.706	0.756
						標準差	0.059	0.008	0.235	0.011
						最大值	1.039	0.749	2.054	0.766
						最小值	0.912	0.733	1.540	0.741
256	200	TRUE	TRUE	TRUE	10	第一組	1.116	0.709	1.671	0.745
						第二組	1.167	0.698	1.518	0.747
						第三組	1.089	0.716	1.637	0.747
						第四組	1.175	0.694	1.781	0.732
						平均	1.137	0.704	1.652	0.743
						標準差	0.041	0.010	0.108	0.007
						最大值	1.175	0.716	1.781	0.747
						最小值	1.089	0.694	1.518	0.732

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
128	200	TRUE	TRUE	TRUE	10	第一組	1.457	0.633	1.848	0.700
						第二組	1.375	0.656	1.825	0.725
						第三組	1.350	0.670	2.288	0.735
						第四組	1.388	0.649	1.836	0.729
						平均	1.392	0.652	1.949	0.722
						標準差	0.046	0.016	0.226	0.015
						最大值	1.457	0.670	2.288	0.735
						最小值	1.350	0.633	1.825	0.700
64	200	TRUE	TRUE	TRUE	10	第一組	1.713	0.587	2.215	0.680
						第二組	1.642	0.600	2.348	0.692
						第三組	1.671	0.590	2.110	0.693
						第四組	1.622	0.604	2.235	0.688
						平均	1.662	0.595	2.227	0.688
						標準差	0.039	0.008	0.097	0.006
						最大值	1.713	0.604	2.348	0.693
						最小值	1.622	0.587	2.110	0.680

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
32	200	TRUE	TRUE	TRUE	10	第一組	2.017	0.511	3.252	0.611
						第二組	1.969	0.526	3.299	0.618
						第三組	1.862	0.543	2.478	0.646
						第四組	1.912	0.534	2.577	0.629
						平均	1.940	0.528	2.901	0.626
						標準差	0.068	0.014	0.434	0.015
						最大值	2.017	0.543	3.299	0.646
						最小值	1.862	0.511	2.478	0.611
16	200	TRUE	TRUE	TRUE	10	第一組	2.562	0.436	4.155	0.511
						第二組	2.371	0.444	4.314	0.492
						第三組	2.612	0.491	3.991	0.593
						第四組	2.672	0.412	3.983	0.493
						平均	2.554	0.446	4.111	0.522
						標準差	0.130	0.033	0.157	0.048
						最大值	2.672	0.491	4.314	0.593
						最小值	2.371	0.412	3.983	0.492

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
2048	200	FALSE	TRUE	TRUE	10	第一組	ram is dead	ram is dead	ram is dead	ram is dead
						第二組	ram is dead	ram is dead	ram is dead	ram is dead
						第三組	ram is dead	ram is dead	ram is dead	ram is dead
						第四組	ram is dead	ram is dead	ram is dead	ram is dead
						平均	ram is dead	ram is dead	ram is dead	ram is dead
						標準差	ram is dead	ram is dead	ram is dead	ram is dead
						最大值	ram is dead	ram is dead	ram is dead	ram is dead
						最小值	ram is dead	ram is dead	ram is dead	ram is dead
1024	200	FALSE	TRUE	TRUE	10	第一組	0.279	0.917	1.786	0.669
						第二組	0.195	0.943	1.792	0.692
						第三組	0.284	0.916	1.917	0.672
						第四組	0.169	0.951	2.408	0.671
						平均	0.231	0.932	1.976	0.676
						標準差	0.058	0.018	0.295	0.011
						最大值	0.284	0.951	2.408	0.692
						最小值	0.169	0.916	1.786	0.669

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
512	200	FALSE	TRUE	TRUE	10	第一組	0.340	0.905	1.777	0.659
						第二組	0.357	0.901	1.952	0.692
						第三組	0.314	0.915	1.849	0.684
						第四組	0.309	0.925	2.192	0.698
						平均	0.330	0.911	1.942	0.683
						標準差	0.023	0.011	0.181	0.017
						最大值	0.357	0.925	2.192	0.698
						最小值	0.309	0.901	1.777	0.659
256	200	FALSE	TRUE	TRUE	10	第一組	0.567	0.857	1.623	0.691
						第二組	0.519	0.867	1.997	0.670
						第三組	0.546	0.858	2.303	0.658
						第四組	0.546	0.859	2.460	0.674
						平均	0.544	0.860	2.096	0.673
						標準差	0.020	0.005	0.369	0.014
						最大值	0.567	0.867	2.460	0.691
						最小值	0.519	0.857	1.623	0.658

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
128	200	FALSE	TRUE	TRUE	10	第一組	0.788	0.813	2.061	0.641
						第二組	0.795	0.824	2.891	0.651
						第三組	0.763	0.827	2.137	0.643
						第四組	0.701	0.851	3.100	0.698
						平均	0.762	0.829	2.547	0.658
						標準差	0.043	0.016	0.526	0.027
						最大值	0.795	0.851	3.100	0.698
						最小值	0.701	0.813	2.061	0.641
64	200	FALSE	TRUE	TRUE	10	第一組	1.059	0.752	2.586	0.628
						第二組	1.060	0.759	2.993	0.641
						第三組	1.079	0.757	2.727	0.652
						第四組	1.010	0.776	3.345	0.623
						平均	1.052	0.761	2.913	0.636
						標準差	0.029	0.011	0.334	0.013
						最大值	1.079	0.776	3.345	0.652
						最小值	1.010	0.752	2.586	0.623

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
32	200	FALSE	TRUE	TRUE	10	第一組	1.452	0.665	4.731	0.600
						第二組	1.445	0.668	2.848	0.590
						第三組	1.341	0.687	4.809	0.600
						第四組	1.358	0.695	4.917	0.574
						平均	1.399	0.679	4.326	0.591
						標準差	0.058	0.015	0.989	0.012
						最大值	1.452	0.695	4.917	0.600
						最小值	1.341	0.665	2.848	0.574
16	200	FALSE	TRUE	TRUE	10	第一組	1.753	0.623	5.712	0.481
						第二組	2.341	0.502	4.846	0.432
						第三組	2.195	0.563	4.963	0.495
						第四組	1.946	0.524	5.461	0.412
						平均	2.059	0.558	5.246	0.455
						標準差	0.261	0.056	0.410	0.040
						最大值	2.341	0.623	5.712	0.495
						最小值	1.753	0.502	4.846	0.412

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
2048	200	FALSE	FALSE	TRUE	10	第一組	ram is dead	ram is dead	ram is dead	ram is dead
						第二組	ram is dead	ram is dead	ram is dead	ram is dead
						第三組	ram is dead	ram is dead	ram is dead	ram is dead
						第四組	ram is dead	ram is dead	ram is dead	ram is dead
						平均	ram is dead	ram is dead	ram is dead	ram is dead
						標準差	ram is dead	ram is dead	ram is dead	ram is dead
						最大值	ram is dead	ram is dead	ram is dead	ram is dead
						最小值	ram is dead	ram is dead	ram is dead	ram is dead
1024	200	FALSE	FALSE	TRUE	10	第一組	0.157	0.955	2.347	0.615
						第二組	0.142	0.957	2.424	0.604
						第三組	0.157	0.953	2.111	0.592
						第四組	0.136	0.961	2.398	0.610
						平均	0.148	0.956	2.320	0.605
						標準差	0.011	0.004	0.143	0.010
						最大值	0.157	0.961	2.424	0.615
						最小值	0.136	0.953	2.111	0.592

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
512	200	FALSE	FALSE	TRUE	10	第一組	0.226	0.937	2.332	0.624
						第二組	0.248	0.933	2.686	0.596
						第三組	0.240	0.933	2.125	0.619
						第四組	0.239	0.936	2.491	0.611
						平均	0.238	0.935	2.409	0.613
						標準差	0.009	0.002	0.238	0.012
						最大值	0.248	0.937	2.686	0.624
						最小值	0.226	0.933	2.125	0.596
256	200	FALSE	FALSE	TRUE	10	第一組	0.437	0.892	2.453	0.583
						第二組	0.366	0.918	2.877	0.585
						第三組	0.398	0.908	2.662	0.603
						第四組	0.393	0.934	4.062	0.608
						平均	0.398	0.913	3.013	0.595
						標準差	0.029	0.018	0.720	0.013
						最大值	0.437	0.934	4.062	0.608
						最小值	0.366	0.892	2.453	0.583

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
128	200	FALSE	FALSE	TRUE	10	第一組	0.604	0.854	2.678	0.595
						第二組	0.621	0.859	2.520	0.589
						第三組	0.653	0.847	2.453	0.589
						第四組	0.633	0.861	2.518	0.601
						平均	0.627	0.855	2.542	0.594
						標準差	0.021	0.006	0.095	0.006
						最大值	0.653	0.861	2.678	0.601
						最小值	0.604	0.847	2.453	0.589
64	200	FALSE	FALSE	TRUE	10	第一組	0.869	0.815	3.407	0.582
						第二組	0.861	0.838	3.261	0.576
						第三組	0.818	0.852	4.321	0.582
						第四組	0.851	0.841	3.672	0.585
						平均	0.850	0.836	3.665	0.581
						標準差	0.022	0.015	0.469	0.004
						最大值	0.869	0.852	4.321	0.585
						最小值	0.818	0.815	3.261	0.576

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
32	200	FALSE	FALSE	TRUE	10	第一組	1.119	0.769	5.886	0.545
						第二組	1.152	0.742	3.831	0.506
						第三組	1.176	0.749	4.402	0.533
						第四組	1.221	0.745	3.536	0.518
						平均	1.167	0.751	4.414	0.526
						標準差	0.043	0.012	1.045	0.017
						最大值	1.221	0.769	5.886	0.545
						最小值	1.119	0.742	3.536	0.506
16	200	FALSE	FALSE	TRUE	10	第一組	1.326	0.695	7.513	0.481
						第二組	1.768	0.631	5.582	0.434
						第三組	1.326	0.678	5.281	0.460
						第四組	1.725	0.557	5.516	0.362
						平均	1.536	0.640	5.973	0.434
						標準差	0.243	0.062	1.035	0.052
						最大值	1.768	0.695	7.513	0.481
						最小值	1.326	0.557	5.281	0.362

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
2048	200	FALSE	FALSE	FALSE	10	第一組	ram is dead	ram is dead	ram is dead	ram is dead
						第二組	ram is dead	ram is dead	ram is dead	ram is dead
						第三組	ram is dead	ram is dead	ram is dead	ram is dead
						第四組	ram is dead	ram is dead	ram is dead	ram is dead
						平均	ram is dead	ram is dead	ram is dead	ram is dead
						標準差	ram is dead	ram is dead	ram is dead	ram is dead
						最大值	ram is dead	ram is dead	ram is dead	ram is dead
						最小值	ram is dead	ram is dead	ram is dead	ram is dead
1024	200	FALSE	FALSE	FALSE	10	第一組	0.001	1.000	2.096	0.539
						第二組	0.205	0.940	2.761	0.534
						第三組	0.018	0.995	2.584	0.537
						第四組	0.091	0.975	2.690	0.534
						平均	0.079	0.977	2.583	0.536
						標準差	0.093	0.028	0.348	0.003
						最大值	0.205	1.000	2.690	0.539
						最小值	0.001	0.940	2.096	0.534

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
512	200	FALSE	FALSE	FALSE	10	第一組	0.090	0.976	2.516	0.549
						第二組	0.111	0.970	2.801	0.535
						第三組	0.131	0.969	3.021	0.546
						第四組	0.097	0.975	2.780	0.545
						平均	0.107	0.972	2.779	0.544
						標準差	0.018	0.003	0.207	0.006
						最大值	0.131	0.976	3.021	0.549
						最小值	0.090	0.969	2.516	0.535
256	200	FALSE	FALSE	FALSE	10	第一組	0.239	0.950	4.612	0.537
						第二組	0.203	0.973	5.891	0.543
						第三組	0.197	0.966	3.834	0.551
						第四組	0.231	0.965	5.346	0.523
						平均	0.217	0.963	4.921	0.539
						標準差	0.021	0.010	0.894	0.012
						最大值	0.239	0.973	5.891	0.551
						最小值	0.197	0.950	3.834	0.523

batch size	class count	隨機旋轉	隨機水平翻轉	隨機左右橫移	patience		loss	accuracy	val_loss	val_accuracy
128	200	FALSE	FALSE	FALSE	10	第一組	0.350	0.951	4.817	0.544
						第二組	0.305	0.910	5.642	0.564
						第三組	0.289	0.936	5.235	0.523
						第四組	0.342	0.942	5.612	0.541
						平均	0.321	0.935	5.326	0.543
						標準差	0.029	0.017	0.387	0.017
						最大值	0.350	0.951	5.642	0.564
						最小值	0.289	0.910	4.817	0.523
64	200	FALSE	FALSE	FALSE	10	第一組	0.426	0.963	7.125	0.533
						第二組	0.461	0.934	6.321	0.512
						第三組	0.412	0.971	6.942	0.552
						第四組	0.479	0.985	6.125	0.550
						平均	0.444	0.963	6.628	0.537
						標準差	0.031	0.021	0.481	0.018
						最大值	0.479	0.985	7.125	0.552
						最小值	0.412	0.934	6.125	0.512

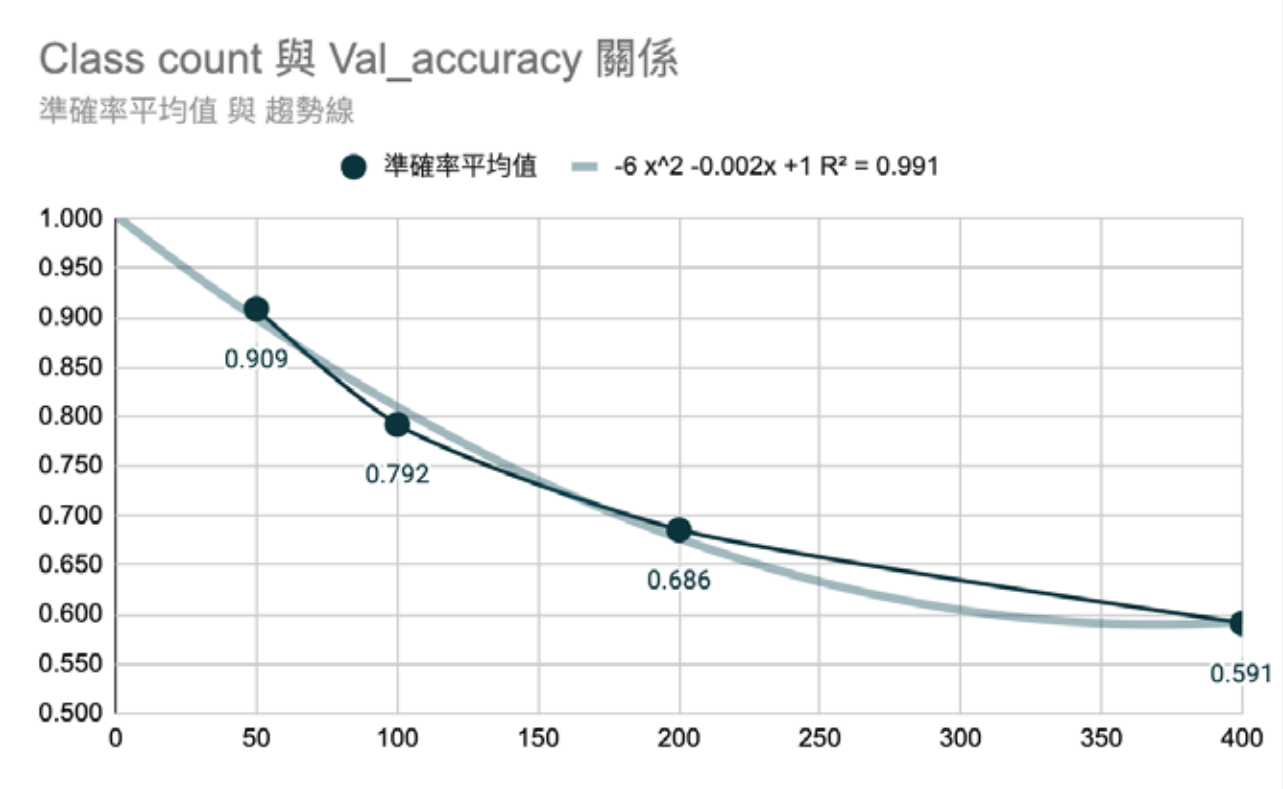
第七章、數據分析 07

本研究涵蓋了不同 Batch Size 和 Class count 的設定下，加上不同的數據增強方式（隨機旋轉、隨機水平翻轉、隨機左右橫移）進行訓練的結果。

一、不同 Class count 之影響

Class count 指的是數據集中不同類別的數量。調整 Class count 可以影響模型的訓練速度和效果。

在這份圖表中，可以觀察到不同大小的 Class count 對模型性能的影響。



1. 當類別數量為 50 時，模型準確率最高，為 0.909。

2. 隨著類別數量的增加，模型準確率逐漸下降。

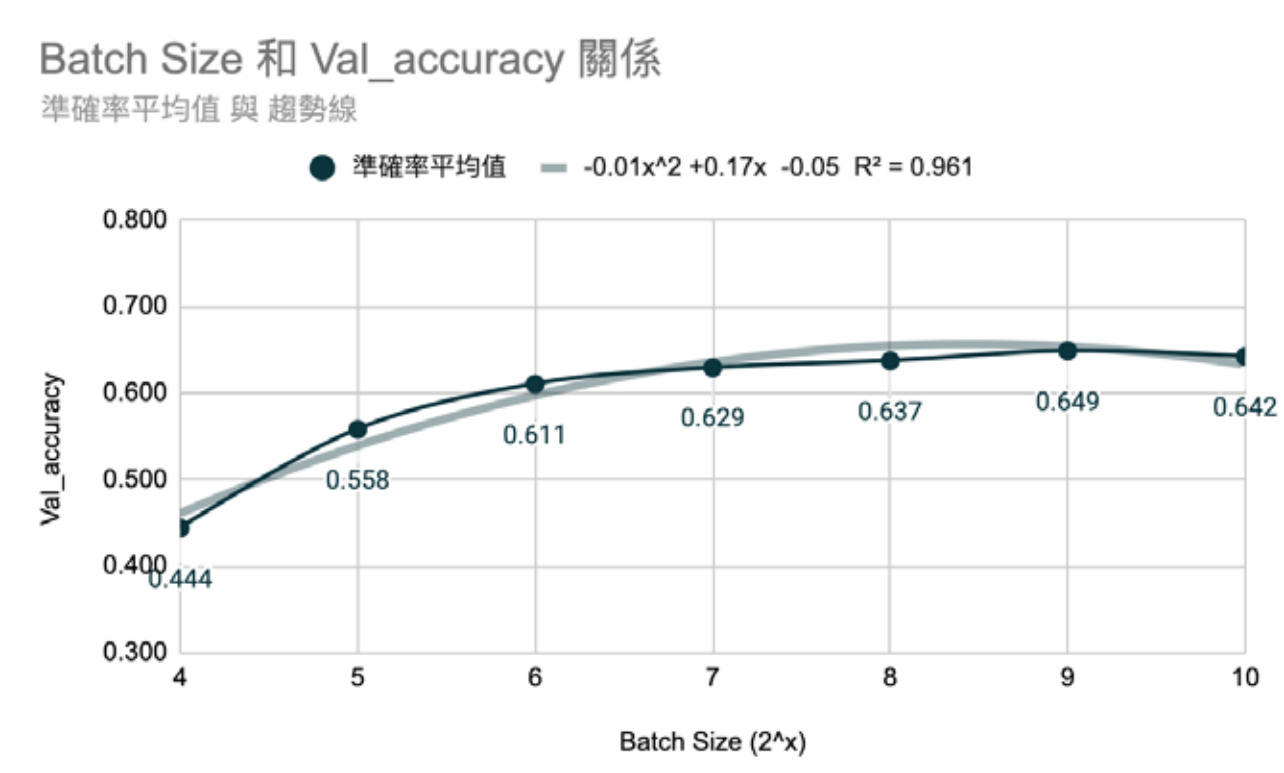
3. 由趨勢線 $-6x^2-0.002x+1$ 可以了解，二次擬合的曲線向下彎曲，這意味著隨著類別數量的增加，模型準確率呈下降趨勢。

總而言之，在類別數量越小的情況下，模型準確率越大。然而，當類別數量過小時，此模型便沒有訓練的意義，因此，以後實驗便會將 Class count 調整為 200。

二、不同 Batch Size 之影響

Batch Size 是指在模型訓練過程中一次性送入模型的樣本數量。調整 Batch Size 可以影響模型的訓練速度和效果。較大的 Batch Size 可加快訓練速度，但會增加記憶體的需求。

在這份圖表中，可以觀察到不同的 Batch Size 對模型性能的影響。

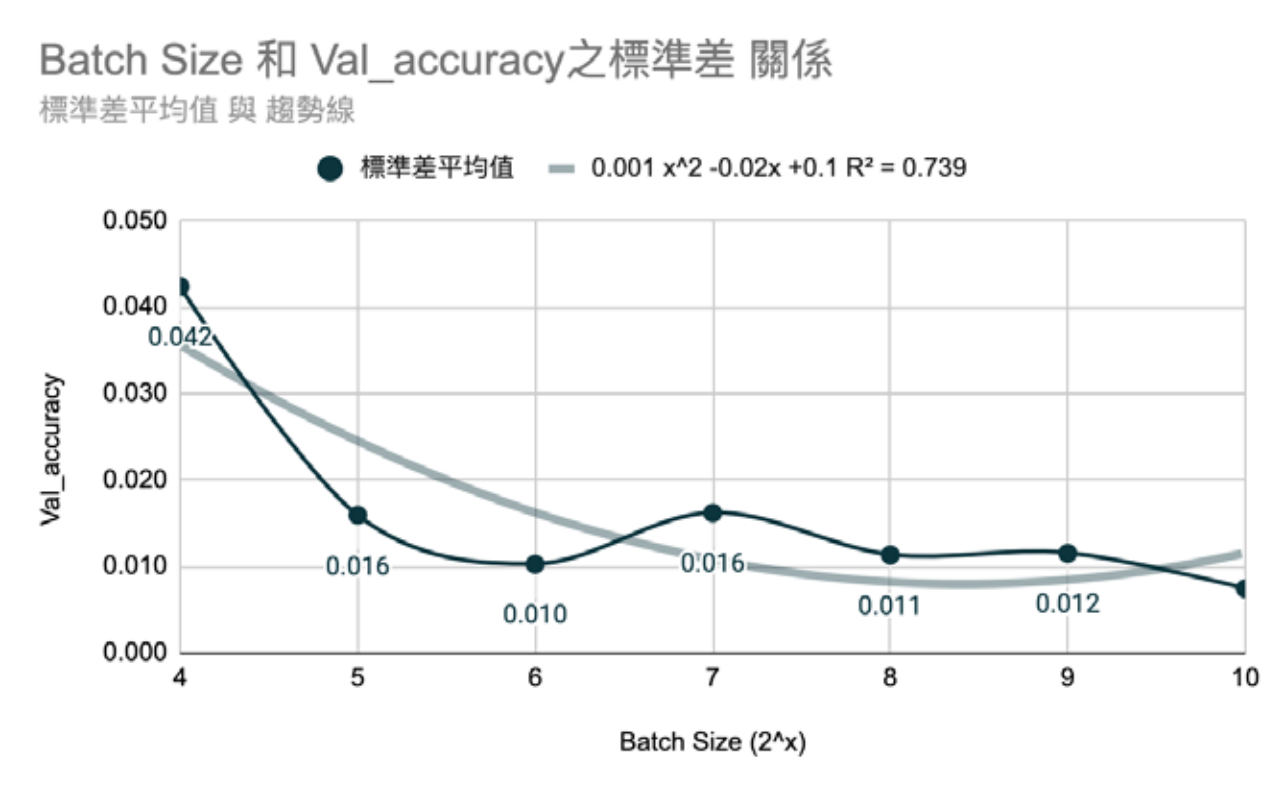


1. 當 Batch Size 從 16 增加到 1024 時，模型的準確率呈現出逐漸提高的趨勢，這表明增加 Batch Size 可能有助於提高模型的準確率。

2. 當 Batch Size 增加到 512 時，模型的準確率達到了最高值，為 0.649。而當 Batch Size 增加或減少時，模型的準確率相對減少，這表明存在一個最佳的 Batch Size。

3. 從趨勢線可以了解，在 Batch Size 為 256 至 512 之間，可以獲得最佳的模型準確率，這可能是一個合理的 Batch Size 範圍。

4. 這表示在一定程度上，增加 Batch Size 有助於提高模型的準確率，但需要權衡內存需求和訓練速度。



觀察其值之標準差與 Batch Size 的關係圖可以發現。

1. 當 Batch Size 為 16 時，標準差最大值為 0.042，這表示在 Batch Size 為 16 時，Val_accuracy 的分布相對不均勻，可能存在一定程度的波動性。

2. 在 Batch Size 為 1024 時，有最小值 0.007，這意味著在這個批次大小下，模型的性能變異性非常小，即模型的表現相對穩定。

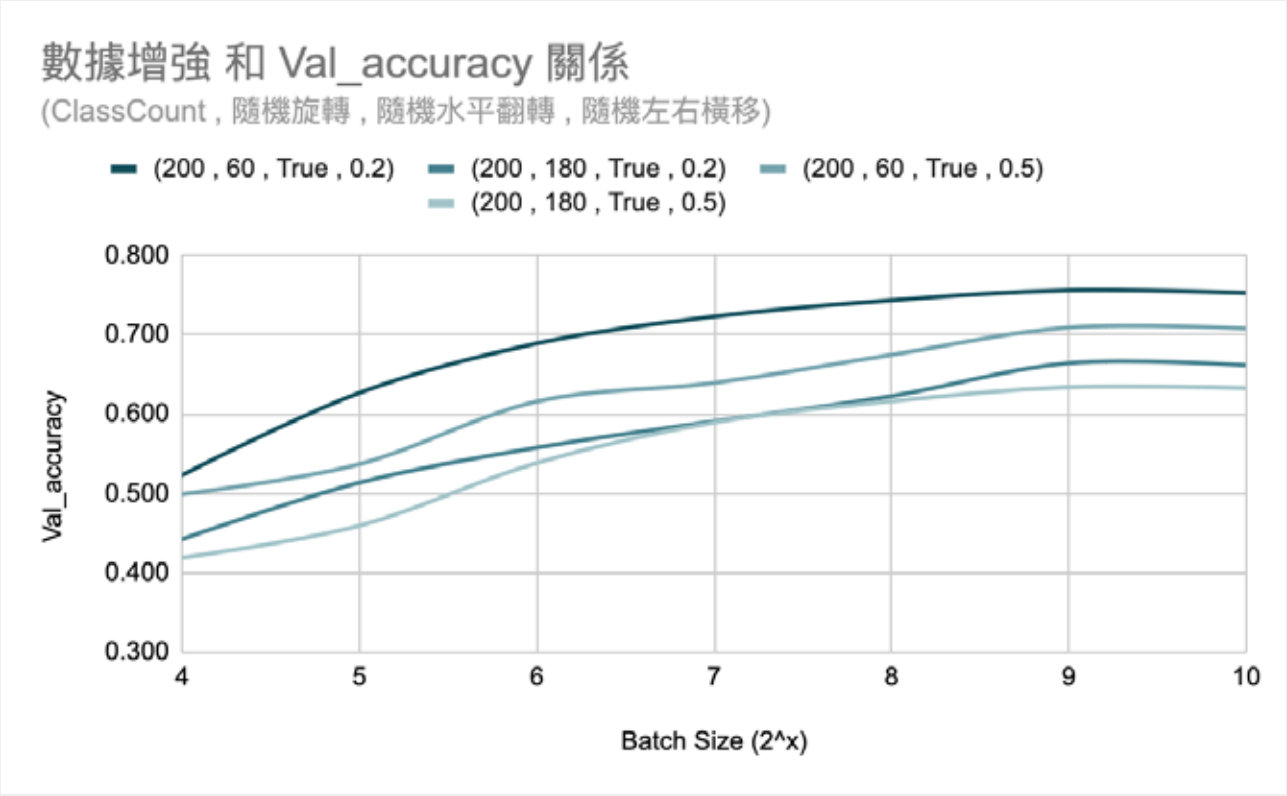
3. 若觀察其趨勢線，可以發現，在 Batch Size 為 256 至 512 之間時，Val_accuracy 之標準差相對最小，故可推測此範圍內包含一個訓練時相對最穩定的 Batch Size，因此之後實驗 Batch Size 應調整為 256 或 512。

三、數據增強之影響

數據增強用於擴充訓練數據集，而改善模型的性能和泛化能力。本研究中，數據增強包括隨機旋轉、隨機水平翻轉和隨機左右橫移等方式。

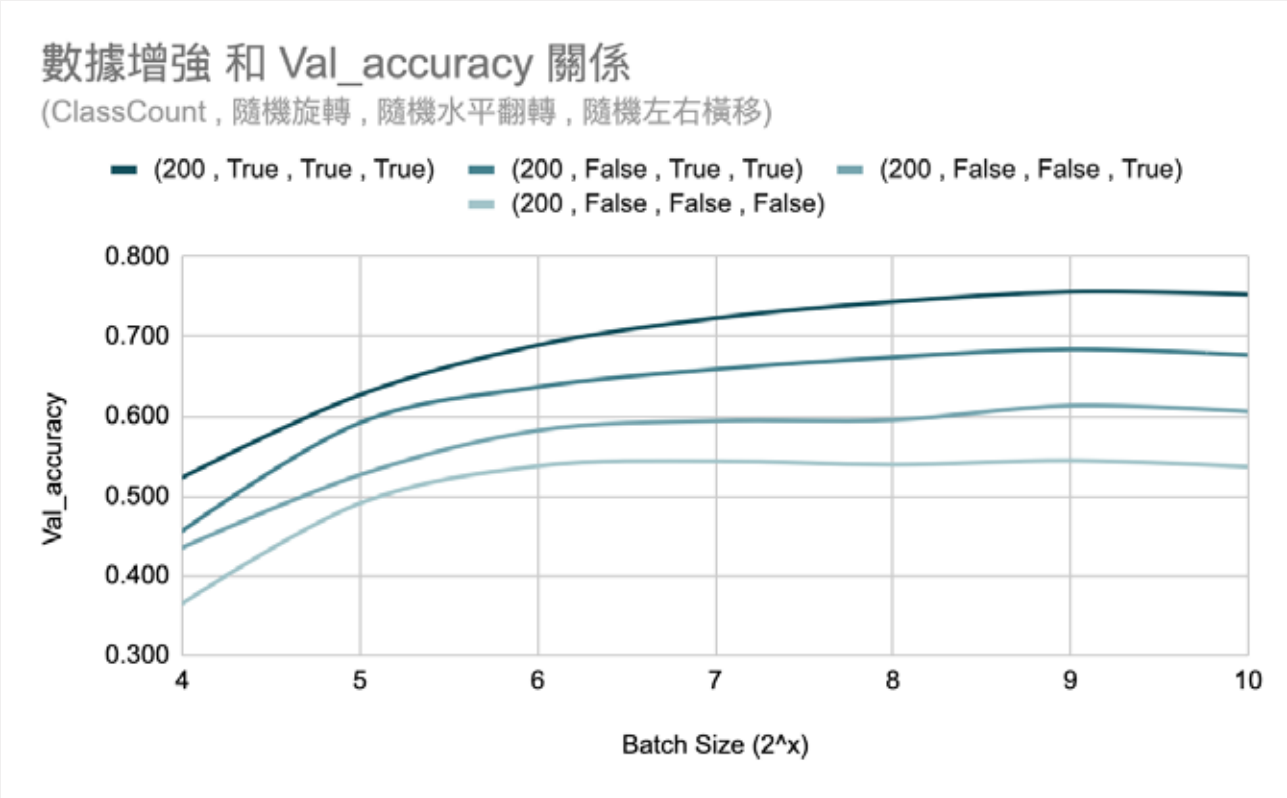
此部分將調整隨機旋轉的角度與隨機左右橫移的大小，用以尋找可得到最佳模型準確率的參數。

隨機旋轉	隨機左右橫移
60	0.2
180	0.5



由圖表可得知，在 (200 , 60 , True , 0.2) 的參數下，可得到最佳模型準確率。而若將隨機旋轉角度調為 180 度，模型準確率相對下降；同時，若將隨機左右橫移大小調整為 0.5，模型準確率亦相對下降。同理，當隨機旋轉角度調為 180 度且隨機左右橫移大小調整為 0.5，模型準確率變明顯下降。這可能是因為同時增加旋轉角度過大，或橫移大小導致了圖像變形和信息損失，進而影響了模型的訓練和準確性。

此部分將隨機旋轉設為 rotation_range=60，代表隨機旋轉度數介於 –60 度至 60 度之間；隨機水平翻轉設為 horizontal_flip=True，代表會隨機水平映像；隨機左右橫移設為 width_shift_range=0.2，代表會隨機左右橫移圖像寬度之 20%，及為 112 像素 20%，22 像素。



在上方圖表中，可以觀察到不同的數據增強對模型性能的影響。

1. 隨機旋轉：

在此實驗中，隨機旋轉的角度範圍為 –60 度至 60 度。可以觀察到，在開啟隨機旋轉的情況下，模型的準確率相對提高。故可推測隨機旋轉有助於增加訓練數據的多樣性，相對的提高模型的準確率。

2. 隨機水平翻轉：

開啟隨機水平翻轉後，觀察到模型的準確率整體高於未開啟隨機水平翻轉的其他組別。故可推測隨機水平翻轉可以幫助模型學習不同角度和方向的特徵，可能有助於提高模型的準確率。

3. 隨機左右橫移：

在設定為 20% 圖像寬度的情況下，觀察到模型的準確率也相對高於未開啟的組別。故推測隨機左右橫移可以引入圖像平移的變化，有助於模型學習對平移不變的特徵，提高模型的準確率。

綜上所述，在訓練深度學習模型時起到了重要作用，能夠增加訓練數據的多樣性，提高模型的準確率。

第八章、結論 08

一、最佳模型設定

在本研究中，通過對模型參數的調整和實驗結果的分析，我們得出了最佳模型設定如下：

1. 最佳參數設定：

batch size : 512

class count : 200

rotation_range : 60

horizontal_flip : True

width_shift_range : 0.2

2. 實驗結果：

loss : 0.956

accuracy : 0.742

val_loss : 1.706

val_accuracy : 0.756

根據這些結果，可以得出這個模型在給定的參數設定下，在訓練集和驗證集上都表現良好。這個模型的訓練過程相對穩定，且具有良好的正確率，可以應用於相應的任務中。

二、Class count 與 Val_accuracy 關係

1. 隨著 Class count 增加，Val_accuracy 下降：

從數據中可以看出，當 Class count 從 50 增加到 400 時，Val_accuracy 逐漸下降，這表明隨著類別數量的增加，模型的準確率也相應下降。

2. 二次擬合曲線向下彎曲：

根據趨勢線 $-6x^2-0.002x+1$，可以看出二次擬合曲線向下彎曲，這意味著隨著類別數量的增加，模型的準確率呈現下降的趨勢，並且下降趨勢呈現加速度的增加。

3. Val_accuracy 最高點出現在 Class count 為 50 時：

在數據中，Val_accuracy 的最高值出現在 Class count 為 50 時，達到 0.909，這可能意味著在這個類別數量下，模型的性能表現最佳。

4. 在 Val_accuracy 與分類種類中抉擇：

雖然分類種類越少，Val_accuracy 的值可以提高，然而，若為了將 Val_accuracy 提高而將分類種類減少，便背離了訓練此模型的初心，同時也會失去模型所要解決問題的豐富性和細節性。

在 Val_accuracy 與分類種類作決折時，需要權衡考慮。如果問題的多樣性和細節性對於解決特定的應用非常重要，那麼即使 Val_accuracy 稍微下降，保持較多的分類種類可能更

為合適。相反，如果模型的主要目標是在簡化情況下提供較高的準確率，那麼減少分類種類可能是一個選擇。

三、Batch Size 與 Val_accuracy 關係

1. 訓練速度：

較大的 Batch Size 可以使模型訓練速度加快，因為每次更新權重時都使用了更多的樣本。

2. 記憶體使用量：

較大的 Batch Size 會增加記憶體的使用量，因為需要一次性載入更多的樣本數據。

3. 模型泛化能力：

Batch Size 的大小也會影響模型的泛化能力。一般來說，較小的 Batch Size 可以增加模型的泛化能力，但也可能導致訓練不穩定。而較大的 Batch Size 則可能使模型過度擬合訓練數據。

因此使用較大的 Batch Size 進行初步訓練，通過使用較大的 Batch Size，可以加快訓練速度並更快地使模型收斂。這有助於了解模型在訓練數據集上的表現情況。在模型達到一定的準確率後，可以使用較小的 Batch Size 進行微調。較小的 Batch Size 通常能夠提高模型對訓練數據的細緻度，有助於模型更好地泛化到新的數據上，提高在未見過數據上的準確率。通過比較使用不同 Batch Size 進行訓練的模型性能，選擇最適

合的 Batch Size。該選擇可能取決於訓練時間、準確率和模型的泛化能力等因素。最後，根據比較的結果和實際需求，選擇最佳的 Batch Size。這可能需要綜合考慮訓練效率、模型性能和泛化能力等因素。

四、資料增強與 Val_accuracy 關係

1. 提高 Val_accuracy：

資料增強可以增加訓練數據的多樣性，使模型更好地學習到數據的特徵和模式，進而提高 Val_accuracy。例如，隨機旋轉、翻轉和平移等增強技術可以生成更多樣化的圖像，有助於模型學習到不同角度和姿勢下的特徵，從而提高準確率。

2. 減少過擬合：

通過引入資料增強，可以減少模型對訓練數據的過度擬合，使其更好地泛化到新的、未見過的數據上。這可以進一步提高在驗證集上的準確率，因為模型對於新數據的泛化能力更強。

3. 影響選擇最佳參數：

資料增強也可能影響到選擇最佳的模型參數，例如批次大小、學習率等。在使用資料增強的情況下，某些參數的最佳值可能會發生變化，因此需要通過實驗來確定最適合的參數配置。

4. 適當的增強方式：

將隨機旋轉角度調整為 180 度或將隨機左右橫移大小調整為

0.5 時，模型的準確率相對下降。這可能是因為過大的旋轉角度或橫移導致了圖像變形和信息損失，進而影響了模型的訓練和準確性。

資料增強是一種有效的技術，可以在許多情況下提高模型的 Val_accuracy。但是，它並不是萬能的，並且需要根據實際的任務和數據來調整和最佳化。

第九章、未來展望 09

VGG16 模型雖然在圖像分類中表現出色，但仍有許多改進和研究的空間，以下列舉幾個未來研究的方向：

一、模型架構改良：

1. 研究更優越的架構：

VGG16 架構固然經典，但近年來深度學習領域發展迅速，出現了許多更強大的架構，如 ResNet、Inception、EfficientNet 等。 這些模型在特徵提取能力和準確度方面都超越了 VGG16，可以作為未來研究的基礎。

2. 輕量化模型：

VGG16 模型參數量龐大，計算成本較高。 研究者可以研究模型剪枝、量化等技術，在盡可能保持模型性能的同時降低其複雜度，使其更適合在資源受限的設備上部署。

3. 混合模型：

結合不同模型的優勢，例如將 VGG16 的特徵提取能力與其他模型的分類能力相結合，可能獲得更佳的性能。

二、訓練算法精進：

1. 優化器選擇：

改用更先進的優化器，如 AdamW、Ranger 等，以提高模型訓練效率和收斂速度。

2. 損失函數設計：

針對特定任務設計更具針對性的損失函數，例如針對類別不平衡問題設計 focal loss 等，以提高模型的泛化能力。

3. 數據增強技術：

採用更有效資料增強技術，如 Mixup、CutMix 等，以提升模型的魯棒性和泛化能力。

4. 自監督學習：

利用大量的無標籤數據進行自監督學習，以提升模型的特徵提取能力和泛化能力。

三、模型打包與共享：

1. 開源模型：

將訓練好的 VGG16 模型及其變體開源，方便其他研究者和開發者使用和改進。

2. 模型 Hub：

將模型上傳至 TensorFlow Hub 或 PyTorch Hub 等平台，方便使用者方便調用和部署模型。

四、模型部署與應用：

1. 雲端部署：

將模型部署在雲端服務器上，提供 API，方便使用者進行圖像分類等任務。

2. 邊緣計算：

將模型部署在移動設備或嵌入式系統上，實現即時圖像分類等功能。

3. 特定領域應用：

將 VGG16 模型應用於醫療影像分析、空拍圖像識別等特殊領域，研究其在不同場景下的應用。

VGG16 模型的研究和應用仍然具有很大實用性。通過改進模型架構、訓練算法、模型打包和部署方式，可以進一步提升 VGG16 模型的性能和應用範圍，使其在圖像分類以及其他領域發揮更大的作用。

參考文獻 10

人工智慧
https://zh.wikipedia.org/zh-tw/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD

機器學習
https://zh.wikipedia.org/wiki/%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0

深度學習
https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0

類神經網路
https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

卷積神經網路
https://zh.wikipedia.org/wiki/%E5%8D%B7%E7%A7%AF%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

VGG16
https://arxiv.org/abs/1409.1556

https://www.cc.ntu.edu.tw/chinese/epaper/0042/20170920_4206.html

https://blog.csdn.net/cz_00001/article/details/131836074

https://danjtchen.medium.com/vgg-%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E5%8E%9F%E7%90%86-d31d0aa13d88

https://blog.csdn.net/light169/article/details/123270587

人工智慧簡介—機器學習
https://omnixri.blogspot.com/p/ntust-edge-ai-ch1-1.html

圖像分類、目標檢測、語義分割、實例分割和全景分割的區別
https://blog.csdn.net/kk123k/article/details/86584216

卷積神經網路的運作原理
https://brohrer.mcknote.com/zh-Hant/how_machine_learning_works/how_convolutional_neural_networks_work.html

殘差神經網絡 ResNet
https://zh.wikipedia.org/zh-tw/%E6%AE%8B%E5%B7%AE%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

Inception
https://medium.com/ching-i/inception-%E7%B3%BB%E5%88%97-inceptionv2-inceptionv3-93cd42054d23

EfficientNet
https://medium.com/ching-i/efficientnet-%E8%AB%96%E6%96%87%E9%96%B1%E8%AE%80-e828ac005ce8

AdamW
https://bigdatafinance.tw/index.php/data-visualization/low-frequency-stock/97-2015-06-24-14-05-24/tech/data-processing/651-adamw

深度學習 – 各種新優化器介紹
https://kilong31442.medium.com/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E5%90%84%E7%A8%AE%E6%96%B0%E5%84%AA%E5%8C%96%E5%99%A8%E4%BB%8B%E7%B4%B9-lookahead-ranger-lars-830ca2250dd5

數據增強之 Mosaic 數據增強的優點、Mixup,Cutout,CutMix 的區別
https://blog.csdn.net/jq_98/article/details/123208912

```
[ ] import pandas as pd
    import cv2
    from tensorflow import keras
    from glob import glob
    import os
    import numpy as np

    birds_latin_names=pd.read_csv('birds latin names.csv')
    idx_to_class=birds_latin_names.iloc[:200,1].values

    class_count=len(idx_to_class)

    tmp=[(idx_to_class[i],i) for i in range(class_count)]
    class_to_idx=dict(tmp)

    def load_data_set(path:str,data_set:str,class_list:list):
        x=[]
        y=[]
        for class_name in class_list:
            path_list=glob(os.path.join(path,data_set,class_name,'*'))
            for tmp in path_list:
                img=cv2.imread(tmp)
                img=cv2.resize(img,(112,112))
                img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
                x.append(img)
                y.append(class_to_idx[class_name])
        y=np.array(y)
        y=keras.utils.to_categorical(y,class_count)
        return np.array(x),y

    x_train,y_train=load_data_set('','train',idx_to_class)
    x_test,y_test=load_data_set('','test',idx_to_class)
    x_valid,y_valid=load_data_set('','valid',idx_to_class)

    print(x_train.shape)
    print(y_train.shape)

    np.save('x_train200',x_train)
    np.save('y_train200',y_train)

    np.save('x_valid200',x_valid)
    np.save('y_valid200',y_valid)

    np.save('x_test200',x_test)
    np.save('y_test200',y_test)
```

▲資料預處理的程式碼

```
[ ] from tensorflow import keras
    import numpy as np
    from keras.models import *
    from keras.layers import *

    x_train,y_train=np.load('x_train200.npy'),np.load('y_train200.npy')
    x_valid,y_valid=np.load('x_valid200.npy'),np.load('y_valid200.npy')
    x_test,y_test=np.load('x_test200.npy'),np.load('y_test200.npy')

    def VGG16(data, classes):

        inputs = Input(shape=data)

        base_model = keras.applications.VGG16(weights='imagenet', include_top=False, input_tensor=inputs)


        base_model = keras.Model(inputs, base_model.layers[-1].output)

        base_model.trainable = False
        x = base_model(inputs)

        fc = Flatten()(x)
        fc = Dense(2048, activation='relu')(fc)
        fc = Dense(2048, activation='relu')(fc)

        output = Dense(classes, activation='softmax')(fc)
        model = keras.Model(inputs, output)

        return model

    model = VGG16((112,112,3),200)

    model.compile(
        loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy']
    )
    batch_size = 512
    epochs = 100
    my_callbacks=[
                  keras.callbacks.EarlyStopping(
                      patience=10,
                      monitor='val_accuracy',
                      restore_best_weights=True
                  )
    ]

    from keras.preprocessing.image import ImageDataGenerator
    datagen=ImageDataGenerator(
        rotation_range=0,
        horizontal_flip=False,
        width_shift_range=0
    )

    result=model.fit(
        datagen.flow(x_train,y_train,batch_size=batch_size),
        validation_data=(x_valid,y_valid),
        batch_size=batch_size,
        epochs=epochs,
        callbacks=my_callbacks,
        shuffle=True
    )

    model.evaluate(x_test,y_test)

    model.save('model17')
```

▲訓練深度學習模型來進行圖像分類的程式碼

```
[ ] import tensorflow as tf
    from tensorflow import keras

    os.chdir('/content/drive/MyDrive')

    model=keras.models.load_model('model10')

    import pandas as pd
    os.chdir('/content/drive/MyDrive/Bird_Data/archive')
    birds_latin_names=pd.read_csv('birds latin names.csv')
    idx_to_class=birds_latin_names.iloc[:200,1].values

    class_count=len(idx_to_class)

    tmp=[(idx_to_class[i],i) for i in range(class_count)]
    class_to_idx=dict(tmp)

    import numpy as np
    x_test,y_test=np.load('x_test200.npy'),np.load('y_test200.npy')

    import matplotlib.pyplot as plt
    def show_img(img_to_show):
        plt.axis('off')
        if len(img_to_show.shape)==3:plt.imshow(img_to_show)
        else: plt.imshow(img_to_show,cmap='gray')
        plt.show()

    print(model.evaluate(x_test,y_test))

    from sklearn.metrics import precision_recall_fscore_support
    y_pred=model.predict(x_test)
    y_pred=np.argmax(y_pred,axis=1)
    y_test_class=np.argmax(y_test,axis=1)
    print('(precision, recall, fscore)')
    print(precision_recall_fscore_support(y_test_class,y_pred,average='macro'))

    import hashlib

    data_num=x_test.shape[0]
    print('total',data_num)

    while True:
        tmp=input('enter a number : ')
        try:tmp=int(tmp)
        except ValueError:
            sha3_256=hashlib.sha3_256()
            sha3_256.update(tmp.encode())
            tmp=int.from_bytes(sha3_256.digest(),byteorder='big')
        n=tmp%data_num
        x=x_test[n,:,:]
        x=np.expand_dims(x,0)
        print(x.shape)
        y_pred=model.predict(x)

        pre_ans=np.argmax(y_pred)
        ans=np.argmax(y_test[n])
        print('idx :',n)
        print('AI  :',idx_to_class[pre_ans],'| class_idx :',pre_ans)
        print('ans :',idx_to_class[ans],'| class idx :',ans)
        show_img(x_test[n])
```

▲使用深度學習模型進行圖像分類預測的程式碼