中原大學 資訊工程學系

110 學年度專題實驗期末報告

Jigsaw puzzle with self-supervised learning

指導教授：余執彰 老師

組員：資訊四甲 10727130 黃意勛

資訊四甲 10727136 賴文裕

資訊四甲 10727156 曾奕豪

目錄

第一章 前言

1-1 動機：

資訊科技日新月異，越來越多的產業為了獲得更高的產能，機及嘗試與與科技搭上線，但還是有些傳統藝能的工作需要靠專業人士全心投入才能使產業運行，比方說化石復原、古物修復、圖像復原，於是我們便思考如何能將這些領域與我們所學的知識結合，做到能夠輔助專業人士做判斷，甚至是直接代替他們作業，使他們能專心於研究上，從這個方向開始我們的研究主題。

1-2 研究過程：

我們想到拼圖問題能作為實現我們目標的橋樑，於是我們查閱與之相關的論文並發現原來拼圖問題並比想像中複雜得多，單單三乘三就有 36 萬種組合，對於模型的負荷量相當大，且論文中準確度也尚有進步空間，於是我們將目標定在如何提高模型架構對亂序拼圖的準確度。

經過對相關文獻的閱讀，我們最終採用 Self-supervisor learning 搭配孿生網路來建立模型，Self-supervisor learning 這種作法不用對每張圖作標記，而是利用模型產生結果再與正解作比對，得到 loss 並更新模型。孿生網路能用一樣的標準對每塊拼圖作分析。

第二章 訓練環境與工具

2-1 Tensorflow 介紹：

TensorFlow 是一個開源軟體庫，用於各種感知和語言理解任務的機器學習。用於研究和生產許多 Google 商業產品，如語音辨識、Gmail、Google 相簿和搜尋。主要分 TensorFlow 1 代版和 TensorFlow 2 代版。我們的模組便是使用 TensorFlow 2.5.0。

2-2 Karas 介紹：

Keras 是一個用 Python 編寫的開源神經網路庫，能夠在 TensorFlow、Microsoft Cognitive Toolkit、Theano 或 PlaidML 之上執行。Keras 旨在快速實現深度神經網路，專注於使用者友好、模組化和可延伸性，是 ONEIROS（開放式神經電子智慧機器人作業系統）專案研究工作的部分產物。我們的模型是使用 Keras 2.4.3 版本。

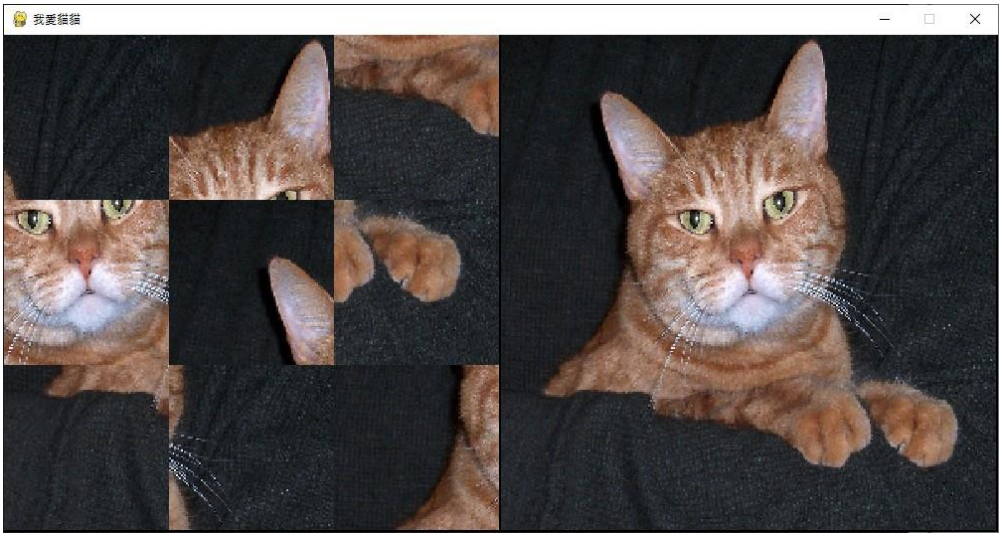
Tensorflow logo

Karas logo

2-3 Pygame 介紹：

Pygame 是將 SDL 多媒體函式庫幫裝起來的模組，能提供許多已經撰寫完畢的函式，一般用於撰寫一些簡易遊戲，但 Pygame 提供的圖形介面使它不只可以用來做遊戲，與多媒體有關的事物也可以做，我們將其用在與模組相連，使我們能夠更直觀得找出那些圖片組合對模型來說相對吃力。

註：SDL 為 Simple DirectMedia Layer，以 C 語言撰寫



Pygame 提供圖形介面

Pygame logo

第三章 實作方法

3-1 資料蒐集：

本程式的訓練成果會受到資料集的格式影響，此次研究項目非常需要龐大的數據，加上為了使模型能有更高的準確度，我們資料選用相同主題的影像——貓咪，於是訓練用的資料集皆來自 Kaggle 上已統整過的貓咪影像，由於經典的拼圖問題是以九宮格的形式來呈現，所以影像的解析不僅要夠清楚，長寬以比例也要越接近正方形越好，避免在資料預處理的時候產生偏頗的訓練樣本，使模組訓練出非預期的結果。

3-2 資料預處理：

除了找尋新的圖像擴充資料集外，我們還使用資料增強以產生足夠大的訓練樣本，主要透過對圖片的旋轉、翻轉去增加訓練的資料集。

資料增強方法：

1. **旋轉轉換**(Rotation)：隨機旋轉影象一定角度。
2. **翻轉轉換**(flip): 沿著水平或者垂直方向翻轉影象。
3. **縮放轉換**(zoom): 按照一定的比例放大或者縮小影象。
4. **平移轉換**(shift): 在平面上對影像以一定方式進行平移。

5. **尺度轉換(scale)**：影像按照指定的尺度，進行放大縮小。

6. **噪點擾動(noise)**：對影象的每個畫素 RGB 進行隨機擾動，常用的噪點模式是高斯噪點

```
datagen = ImageDataGenerator( #實體化
    rotation_range = 90,  #圖片隨機轉動的角度
    width_shift_range = 0.2, #圖片水平偏移量
    height_shift_range = 0.2, #圖片垂直偏移量
    zoom_range = 0.3) #隨機放大或縮小
```

影像增強

接著我們會對圖片作調整大小，統一是 192*192 的規格，由於我們的圖經過塞選後大多接近正方形，所以調整後不至於變形得太嚴重，再來會透過程式將圖片切割成九塊 64*64 大小的小圖並分散到九個資料夾，分散到各個資料夾的主要原因是這能使我們在訓練過程中取得資料能更直觀。

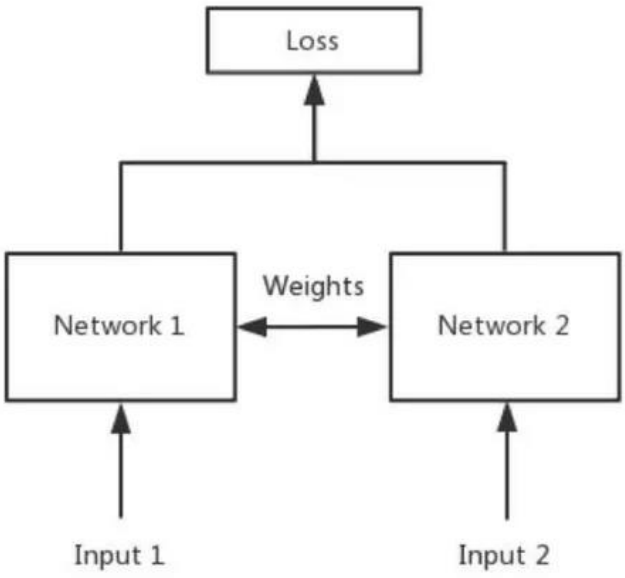
```
def inputimage(imgpath, imaname):##輸入照片
    img = cv2.imread(imgpath)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    size = img.shape
    if size[0] >= size[1]:
        img = cv2.resize(img, (size[0], size[0]), interpolation=cv2.INTER_CUBIC)
        length = int(size[0])
    else:
        img = cv2.resize(img, (size[1], size[1]), interpolation=cv2.INTER_CUBIC)
        length = int(size[1])

    img_list = []
    unit = int(length/3)

    for i in range(3):
      for j in range(3):
        temp = img[unit*i:unit*(i+1), unit*j:unit*(j+1)]
        img_list.append(temp)

    output(img_list, imaname)
```

影像切割

以上的預處理原先是跟模型一起進行的，由於這會使我們訓練起來較沒效率，所以我們決定將其分開進行，伴隨的是失去模型整體的彈性，但我們認為綜合起來這是較好的作法。

3-3 神經網路架構：

我們的模行主要分為兩個階段，前半段為孿生網路，後半段為全連接層。孿生網路(Siamese Network)是特殊一種網路架構，簡單描述就是連體的神經網路，神經網路會通過共享權值來實現連體的效果，常用於辨別兩張圖片是否相同，下圖為孿生網路的簡單示意圖。

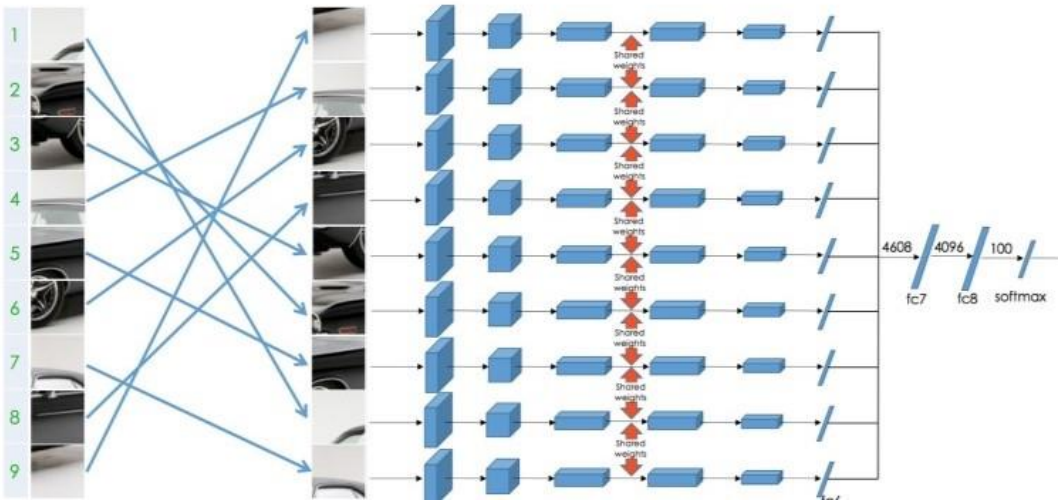


我們模型前端半部是由九個孿生網路組成，採用孿生網路這種架構主要原因是由於每塊拼圖都可能因為組合順序不同，導致再次進入模型時可能會進到不同條孿生網路中，為了確保每

塊拼圖在每次進入模型時能產生相同的特徵點，孿生網路能達到我們想要的效果。

而構成這九條孿生網路的是以預先續練完的 Resnet50V2，相較於論文中使用 Alexnet，我們認為它能在擷取每塊拼圖的特徵值時達到更好的效果。

接著 Concatenate layer 會將孿生網路所擷取到的特徵值攤平並與後半部的全連接層相連，全連接層是我們主要在調整的 layer，它會透過前面網路所得到的特徵值去判別出拼圖錯亂的順序是四十種的哪一種。



模型架構圖

產生的結果會透過 Loss 函數(categorical crossentropy)與正解計算，此 Loss 函數會對每一個 class 產生的值取 log，正確的值乘上一，其餘乘零再相加，這使模型在預測正確時會得到較小的值，錯誤則反之，由此種方式更新模型。

$$loss = -\sum_{i=1}^{n} \hat{y}_{i1} logy_{i1} + \hat{y}_{i2} logy_{i2} + \cdots + \hat{y}_{im} logy_{im}$$

3-4 訓練流程：

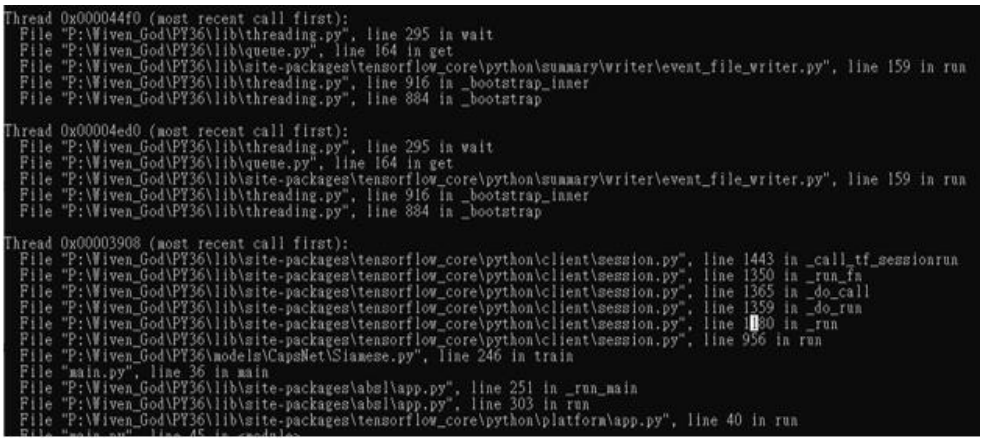
訓練開始前會先將檔案名稱取出，以便我們在每次 batch 拿資料時使用，再來將資料分為 train data 與 validation data，由於我們遇過將圖片一次全部載入模型導致記憶體不足的問題，所以訓練時使用的是 test_on_batch()而非 fit()，兩者功能事實上是相同的，只是實作上 test_on_batch()較適合我們每次 batch 都換資料的作法。

每次 batch 我們會從 train data 中隨機取一部份的圖片，並將每張圖片都做四十種不同的亂序，比方說我們一次 batch 取十張圖片，每張都會產生四十種不同的排序，這樣我們一次 batch 總共就會產生四百種不同的訓練資料，直到每張圖片都被餵給 model 後完成一次 epoch，再用 validation data 計算每次的準確度與 loss，大約跑三十個 epoch 會結束訓練，最後儲存訓練完的權重並將結果繪製成圖。

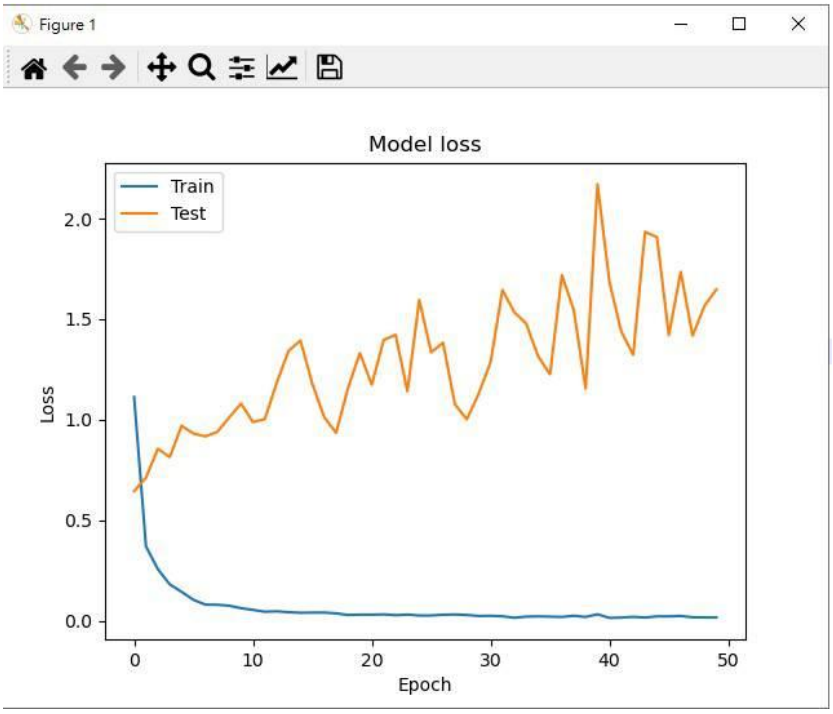
第四章 問題與解決方法

4-1 版本不合：

我們在一開始使用的作業系統是 Windows 配合 Anaconda 來運行 python，但是 Anaconda 對於 Windows 的支援並不是想像中的好。我們在一開始執行一支網路上找到的開源碼，不料該程式會時不時爆出執行緒的錯誤。在不得已的情況下我們只能再找尋一個範例程式碼改寫成我們要的樣子。但是在範例程式碼中有些套件太新了，Anaconda 對於 Windows 的支援版本沒有達到我們需要的版本。我們只能使用相對較低階的版本。後來發現一項重大的問題，我們在訓練模型時每當在進行驗證時準度上不來，loss 更是一直增加。在偶然使用 Google 提供的 kaggle 時發現，原來版本一直存在著衝突的問題。不得已的情況下我們將作業系統換成 Linux 才終於得到改善。最後我們是使用 Docker Container 配合著 tensorflow-gpu 2.5.0、Keras 2.4.3 以及 Python3.6.9 等套件完成訓練。



執行緒錯誤問題



Loss 不斷增加問題

4-2 記憶體不足：

在我們剛開始訓練模型時，我們是使用函式庫提供的 fit()函式進行訓練，但由於一次將十萬張圖片載入模型會導致記憶體空間不足，於是我們將資料預處理與模型訓練分開，train_on_batch()取代 fit()作為模型訓練的函式，這麼做雖使程式整體的彈性下降，這裡的彈性指的是模型不能再更換資料集的當下就開始訓練，但由於資料預先處理完畢，我們在訓練上的效率是有提升的。

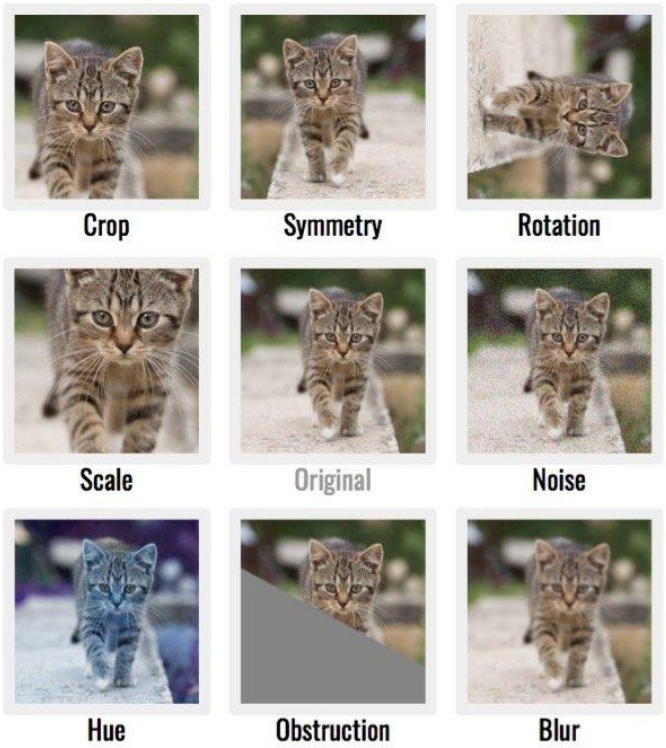
4-3 組合過於多種：

三乘三拼圖問題的組合數高達九階，無法將所有組合都列舉出來，不然參數會過於龐大，沒有足夠的資料與優秀的硬體，會導致訓練結果無法達到理想且訓練時數十分攏長。

我們將模型分為四十類，並盡量將在順序不同但相差不多的組合選出代表放入這四十類中，將問題化簡使其不會對模型產生過多負擔，這能使模型準確度，但並不能將問題從根本上的解決。

4-4 資料量不足：

在我們參考的論文中有提到他們的訓練資料單單圖片就有一百四十萬張，相較於他們的資料數，我們的十萬張真的無法相比，而這連帶的使我們不敢將模型的分類數增加，理由是在增加分類數會導致問題的複雜度再次提升，我們現有的資源不足以應對更複雜的網路，我們能做的只有想辦法增加圖片的數量與進行圖像增強，透過對圖片進行翻轉、調色、切割等等手法來增加有效資料。



圖像增強的幾種方式

第五章 總結

5-1 未來展望：

在與指導教授多次的討論後，得出了或許當前的模型架構有些地方尚可改進，比方說試著不使用現有的 Loss 函數，而是自行嘗試去定義更適合模型的 Loss 函數，又或是自定義 layer 等等方法，跳脫原本的框架，嘗試更多可能的辦法。

去收集更多的資料集與訓練組合也是我們該更積極去做的，資料集還有許多能精進的地方，那些資料對模型的影響較多，能使模型更加擁有更高的準確度是我們該改進的地方。

5-2 結論：

從結果來看，我們離當初的目標還是有一段距離，不論是模型的完成度，訓練的結果，還是對於整個深度學習我們都還有好多路要走，而且我們對於問題的應變能力還十分不足，常常會拐了一個大彎才解決事情，尚有許多能夠進步的空間。

但相比大三剛上開始接觸深度學習，從一開始什麼都不懂，到漸漸能聽懂老師在說什麼，到能自己找出該用什麼辦法解決問題，也算是有進步，真的要感謝指導教授不厭其煩的指出我們問題出在哪，能怎麼解決，未來不管會不會繼續走機器學習這條路，這都會是難得可貴的經驗。

參考文獻

- https://link.springer.com/chapter/10.1007/978-3-319-46466-4_5
- https://keras.io/api/models/model_training_apis/
- Keras Loss Functions: Everything You Need To Know - neptune.ai
- Introduction to gradients and automatic differentiation (tensorflow.org)
- https://zhuanlan.zhihu.com/p/35040994
- https://towardsdatascience.com/siamese-networks-line-by-line-explanation-for-beginners-55b8be1d2fc6
- https://allen108108.github.io/blog/2019/10/22/L1%20,%20L2%20Regularization%20%E5%88%B0%E5%BA%95%E6%AD%A3%E5%89%87%E5%8C%96%E4%BA%86%E4%BB%80%E9%BA%BC%20_/
- https://r23456999.medium.com/%E4%BD%95%E8%AC%82-cross-entropy-%E4%BA%A4%E5%8F%89%E7%86%B5-b6d4cef9189d
- https://www.tensorflow.org/tutorials/keras/classification