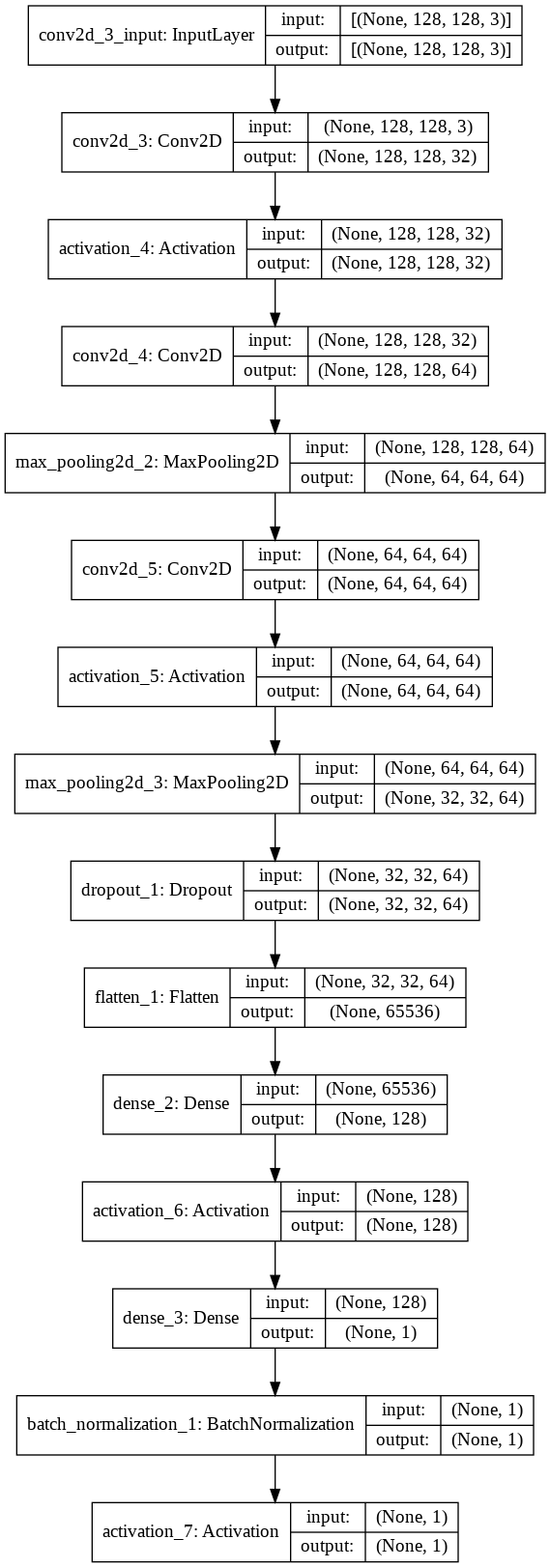
深度學習Tensorflow 期末專題報告

1072B0006 謝可安

1. 摘要：訓練CNN神經網路辨識貓和狗的圖像。
2. 方法與流程圖說明



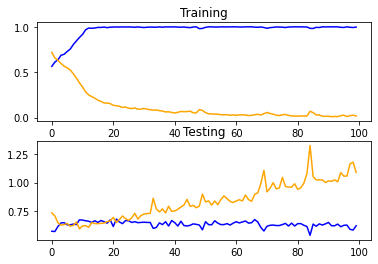
流程圖：CNN神經網路

先使用ImageGenerator()做圖片的前處理（包括rescale、resize），資料增強（包括rotate、shift、zoom、flip等），再利用CNN神經網路架構訓練模型，最後用混淆矩陣驗證模型效果。

1. 實驗結果

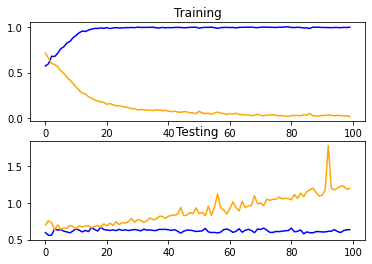
初始訓練資料：貓狗各1001筆，測試資料：貓狗各400筆，共2802筆。

藍色數線為accuracy，橘色數線為loss。可以看出已有overfitting現象：



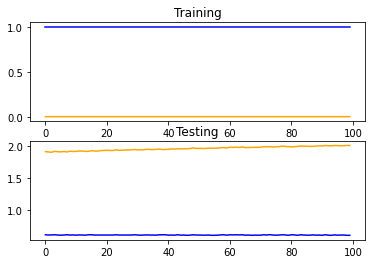
CNN網路

因應overfitting的現象，在Dense層前加了Dropout層卻仍無太大改善：



CNN網路+Dropout

將模型改成VGG16這種較複雜的網路時甚至會出現training accuracy=1，testing accuracy=0的現象。



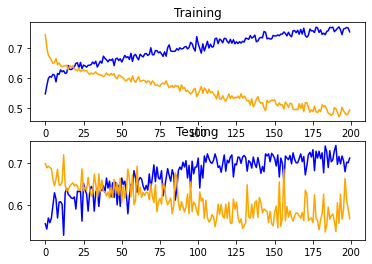
VGG16網路

因為調了一些參數或加Dropout層都無法解決overfitting的問題，所以決定做資料擴增，讓1張原始圖片經過水平翻轉、傾斜、上下左右移動等方式擴增為多張新資料。（第二版）新資料筆數為steps\_per\_epoch\*batch\_size=63\*32=2016張，故總訓練資料數為2002+2016=4018張。

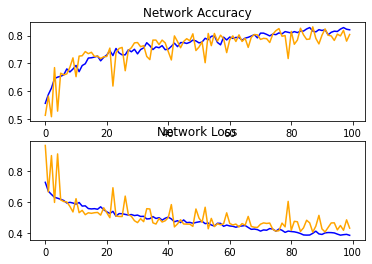


(第一版)原始影像及擴增後的影像

（第一版）原本打算將擴增完的資料放入for loop，經由ImageDataGenerator.flow()存到一個位於主目錄的'/content/'的新增資料夾’aug/’內，待進行resize的處理。但發現擴增後的圖無法被存到’/content/aug/’，而是暫時存在colab的RAM中，導致OOM(out of memory)報錯，訓練時間也較長。於是參考了另一個網站的程式碼（第二版），並不另外把擴增後的資料用for loop存檔，而是使用生成器生成完前處理好的圖片後直接指定給訓練網路的函數network.fit\_generator()，才成功run起來。最後的學習曲線如下：



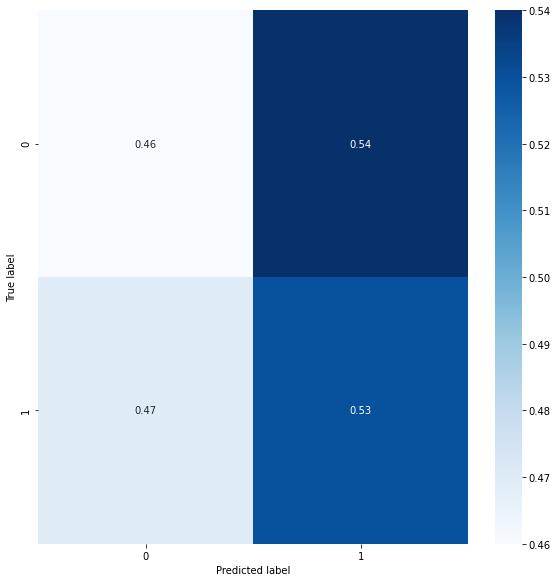
CNN網路+資料增強



CNN網路+資料增強，此圖中藍色數線為training，橘色數線為testing

增加了資料增強後，可以看出overfitting現象已減少，雖然曲線變得較不平滑，但training和testing的準確率都有逐漸上升的趨勢。

為了想驗證模型準確率而增加了混淆矩陣，但不知道是模型訓練不夠好還是混淆矩陣的設定錯誤，顯示出來的效果並不佳，還有待改良。



混淆矩陣

1. 貢獻說明

資料庫來自[此網站](https://www.kaggle.com/biaiscience/dogs-vs-cats)，圖片前處理部分參考[此網站](https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/)，資料擴增部分參考[此網站](https://www.itread01.com/content/1546542668.html)，混淆矩陣部份參考[此網站](https://androidkt.com/keras-confusion-matrix-in-tensorboard/)。貢獻為拼湊各個小段程式、debug、將部分程式改寫成自己比較熟悉的寫法、資料擴增、混淆矩陣。

1. 參考文獻
2. Jason Brownlee (May 17, 2019). *How to Classify Photos of Dogs and Cats (with 97% accuracy)*. Retrieved from <https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/>.
3. ITREAD01（2019年1月4日）。*【深度學習】ImageDataGenerator的使用*。取自 <https://www.itread01.com/content/1546542668.html>。
4. 黃馨平（2019年1月19日）。*小數據的逆襲?*。取自<https://medium.com/@jackycsie/%E5%B0%8F%E6%95%B8%E6%93%9A%E7%9A%84%E9%80%86%E8%A5%B2-c04fee852539>。
5. knowledge Transfer（2019年1月29日）。*TensorFlow Keras Confusion Matrix in TensorBoard*。取自 <https://androidkt.com/keras-confusion-matrix-in-tensorboard/> .
6. RyanAkilos (2019). *A simple example: Confusion Matrix with Keras flow\_from\_directory.py*. Retrieved from <https://gist.github.com/RyanAkilos/3808c17f79e77c4117de35aa68447045> .