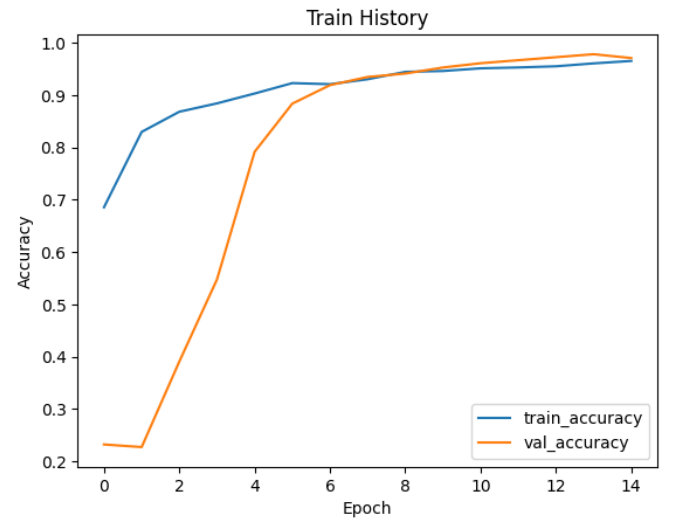
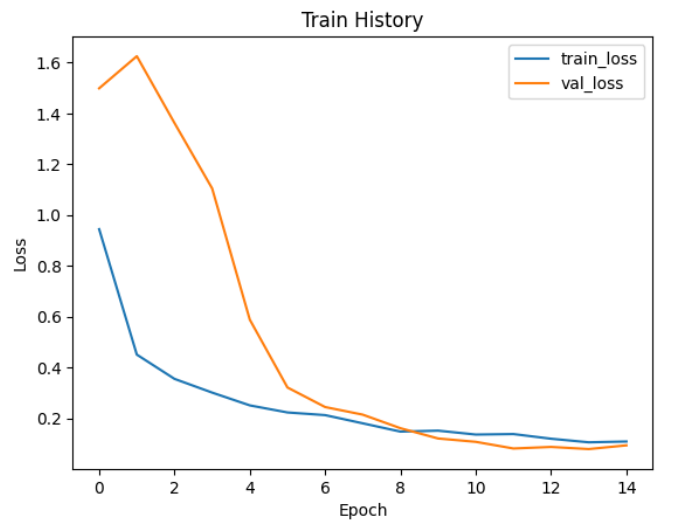
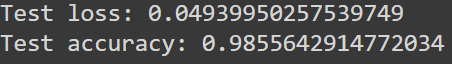
1. 工件辨識準確率及損失率
2. Train accuracy history圖



1. Train loss history圖



1. Test loss ,Test accuracy截圖:

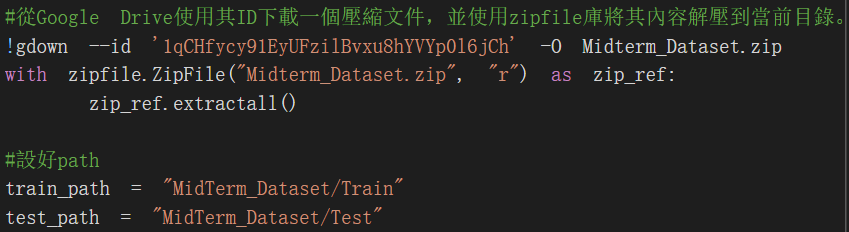


1. 請詳述你如何實作期末專題，包括資料前處理、選擇模型建立、調整參數…等等

* 資料前處理 & Data Augmentation:

根據簡報提供的google drive ID，運用zipfile.extractall()進行解壓縮，之後指定train\_path和test\_path到各自資料夾路徑。

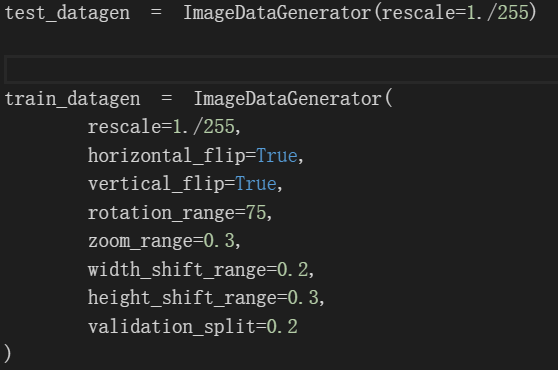
如下圖:



考慮到dataset的 train data不夠大，和擔心有時序上的問題，讓模型可能在訓練過程時，會把時序考慮進去。但此模型只是做辨識物件而已，因此會加上shuffle()。

而且不應該對test data做所謂的augmentation，需要讓test data 被認為是從未接觸過的data，如果做augmentation，可能會造成data leakage，導致產生過於樂觀的實驗結果。

以下在ImageDataGenerator()進行data augmentation:



* + 可能會有點好奇，我之前說不能對test data做data augmentation，為何我在這裡對test data做rescale呢?

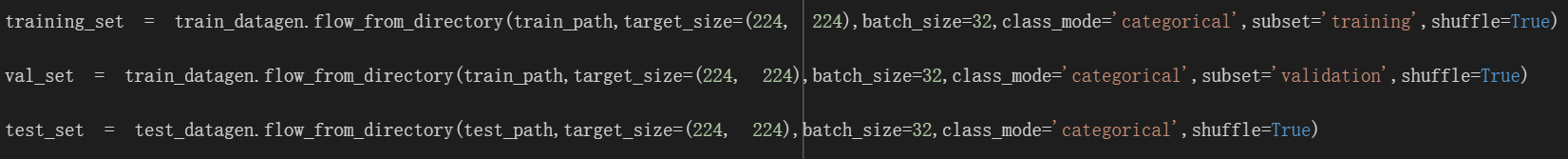
因為這沒有構成任何的數據增加，沒有創建新的樣本和圖像，只是把原始數據以一種標準且一致的方法進行轉化。(將像素值歸一到0到1的範圍)

這時我們來看train data 的 data augmentation吧~

1. Horizontal\_flip : 生成與原始圖像相似但水平翻轉的新圖像
2. Vertical\_flip : 生成與原始圖像相似但水平垂直翻轉的新圖像
3. Rotation\_range = 75 : 在-75到75度之間，隨機將圖片旋轉指定度數
4. Zoom\_range = 0.3 : 在0.3範圍內隨機放大圖像。
5. Width\_shift\_range = 0.2 : 在0.2範圍內隨機地將圖像水平移動的新圖像
6. Height\_shift\_range = 0.3 : 在0.3範圍內隨機地將圖像垂直移動的新圖像
7. Validation\_split = 0.2 : 80%給訓練資料，20%給驗證資料。
   * 為甚麼我們要做augmentation呢?

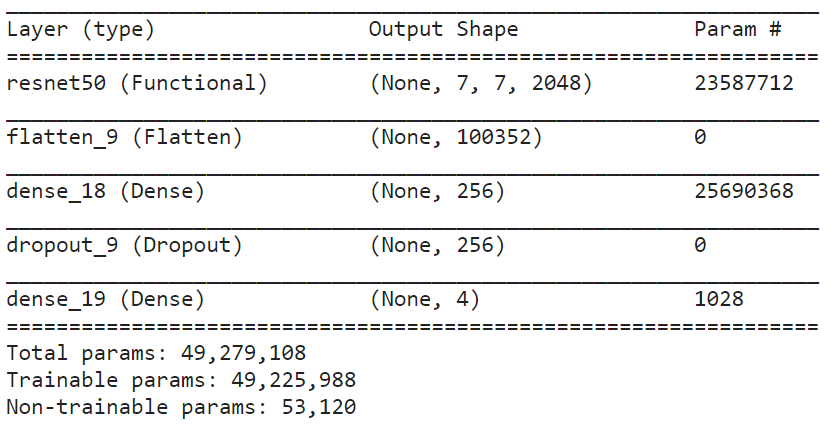
除了可以避免overfitting，避免模型只記住train data 而已，也可以幫助 model 在預測未見過的 data 可能得到更通用的特徵。

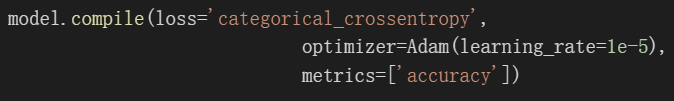
之後再把training\_set 和 val\_test 和 test\_set 接到上面所加的特徵變化，在加一些數值，如下圖:



在訓練過程中，將每個batch設為32，而categorical 指定模型為預測的標籤類型，幾個類別中的其中一個。

* 模型建立:





我使用ResNet50架構作為預訓練基礎的圖像分類的模型。這個模型加載了在ImageNet數據集上訓練的預訓練權重。

首先，預訓練的ResNet50模型被添加為第一層。然後，添加一個Flatten層，將ResNet50模型的輸出轉換為一維向量。之後加入兩個完全連接的Dense層，第一層有256個單元，使用ReLU激活函數。在第一個密集層之後添加了一個Dropout層，以規範化模型並減少overfitting。最後一個密集層有4個單元，相當於label的數量(bolt、locatingpin、nut、washer)，並使用softmax激活函數來輸出是不是符合此label的機率。

接下來把所有層的權重給freeze，讓他們在訓練中不會被更新。這是因為預訓練的權重已經針對ImageNet數據集進行了優化，有更高機率在辨識工件時，有良好的表現。但是我之後unfreeze了模型中的最後10個層，以允許它們的權重在訓練中被更新。這樣做是為了在使用工件dataset的時候對模型進行調整，避免這些工件dataset可能與原始ImageNet數據集的特徵不同。

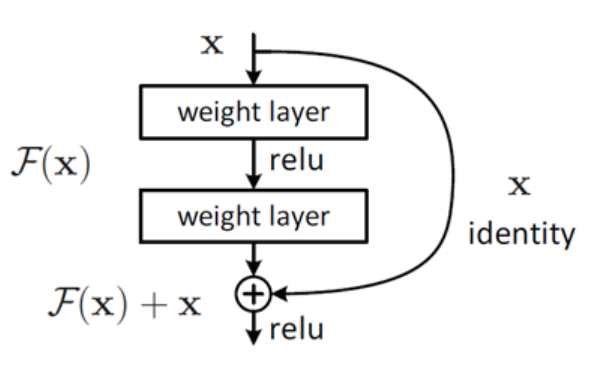
epoch設為15。

* 模型選擇原因:

一開始知道要做圖片辨識的時候，第一個想法跳出來的就是CNN，但是我想說試試看類似CNN但是可以避免掉一些問題的模型，也就是ResNet。

1. ResNet有跨越layer的機制，可以讓model學習到殘差值，避免當訓練到layer太深的時候，產生gradient vanishing的問題，所以ResNet比CNN可以學習到更深層的特徵。

如下圖:



From : <https://neurohive.io/en/popular-networks/resnet/>

* Accuracy & Loss 結果分析:

雖然說最後的Test Accuracy有達到98.55%，但是我們可以看到在epoch為1的時候，validation loss 突然增高，從1.5到1.6，然後到epoch 2之後，又以遞減方式呈現。

以下為可能原因:

1. overfitting，模型不能很好的泛化到新的數據，導致記住train data，而面對新的data時候，沒有很好的分類結果。
2. Noise，由於validation data 的一些過於集中，而產生一些異常樣本，而難以預測，導致loss增加。
3. Learning rate 預設不佳。(1e-5)

* 改善方法:

1. 可以在模型中加入L1 & L2 regularization，避免overfitting。
2. 更改learning rate。
3. Dataset size 不夠大。

* 心得:

在這次工件辨識的midterm project之前，有做過mnist，聲音辨識等等相關dataset。所以這次的經驗可以讓我運用我沒用過的模型，用在我沒試過的dataset，令我感到有挑戰性!最後的結果其實還是不錯的，但還是有蠻多地方可以進行修改，讓結果可以表現的更好。

最一開始對ResNet也是不太熟，但是爬了很多文，看了很多paper才知道，他的辨識準確度比CNN還要突出，這才讓我想要讓此dataset做ResNet model，去試試看他的具體架構是甚麼。

尤其在撰寫報告之後，可以讓我更深入理解之前模糊的地方，藉由這次的工件辨識，知道一些小技巧，希望可以在我之後的專題上，有著更好的表現!