Assignment #4 Overfitting Problem

F64126147 胡瑀真

注意事項 1. 書面報告可參考網路資料，但須理解與整理，自己理解與整理後撰寫報告，使用整段網路內容所獲的評分不高

注意事項 2. 勿分享報告給課程同學，相似內容的報告所獲評分不高

1. 說明是否參考或使用網路上程式，或是全部程式自行撰寫?

(允許參考網路程式資源，在書面報告中須說明參考來源與參考程式的範圍。如程式與模型全部自做，亦在報告中說明，有額外分數)

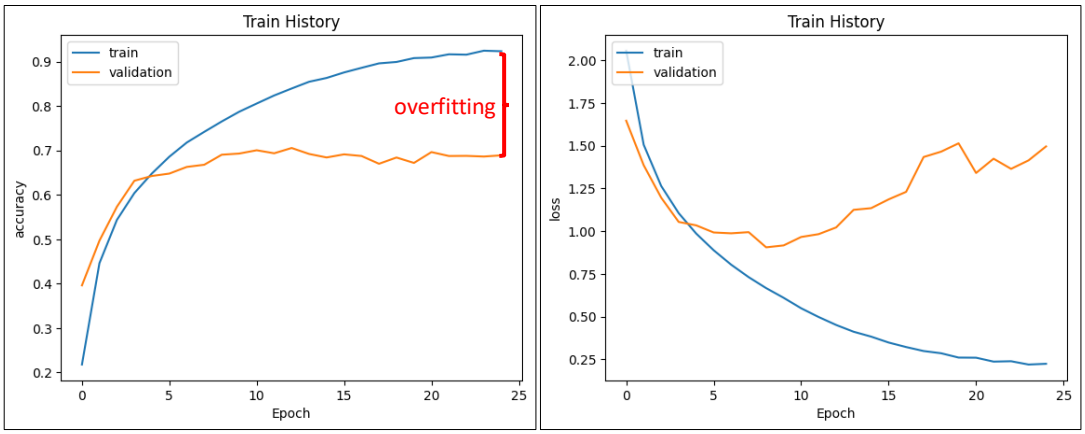
程式主要以第 8 章講義為基礎，並搭配網路資料（網址附在最後）做理解後，對原本的程式做修改，並在多次執行中嘗試可以最佳化 accuracy 的參數（如：Data Augmentation 的影像隨機旋轉角度、Dropout 的值和 Dropout 出現的次數）。

2. 這作業降低模型過渡擬合的過程與最後結果

2.1 降低模型過渡擬合過程

(從課程給定的樣板模型到所獲得最佳模型的過程，須報告至少三個過渡擬合降低的過程，包含最佳模型、不包含最初的樣板模型)

初始訓練結果



Accuracy of testing data = 68.7%

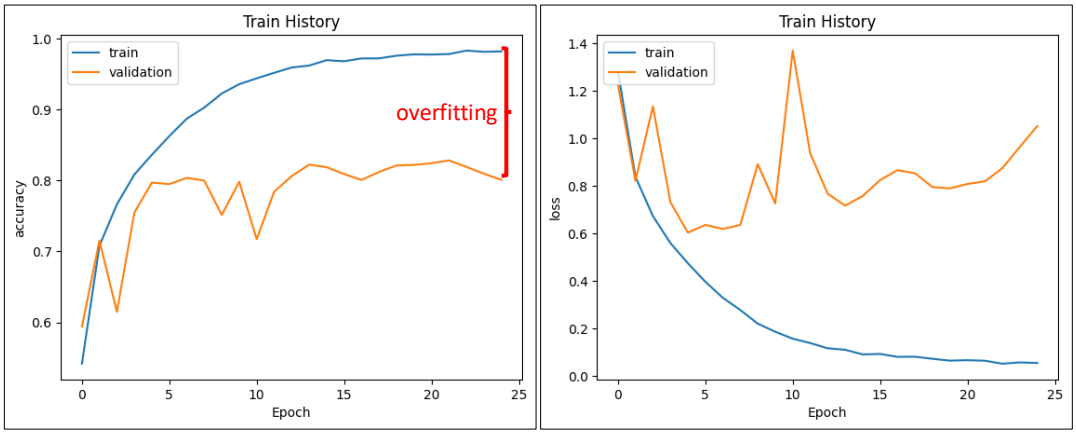
Batch Normalization

從樣板模型的 train history 可知模型有過渡擬合問題（如上圖中紅色記號），表示模型和 training data 表現過好，與 validation data 的表現不佳，失去泛化能力。故參考第 8 章講義和作業 4 講義中的提示，利用 Batch Normalization 可以穩定模型訓練的學習、想像力，並加快收斂速度的特性，首先使用 Batch Normalization 改善過擬合。

```python
model = Sequential()
# 第一組卷積
model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same', input_shape=(32,32,3), use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same', use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same', use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

```python
model.add(Flatten())
model.add(Dense(128, use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dense(128, use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dense(10, activation='softmax'))
model.summary()
```

將每個 Conv2D 和 Dense 層中的的 Activation function 取出，並在其後新增 Batch Normalization()，最後把取出的 Activation function 放回。因重複性過高，故僅截上圖中第一組卷積，和 Flatten 之後為例。



Accuracy of testing data = 79.1%

從上圖的 train history 可知，train accuracy 隨訓練持續上升，最後將近 100%，而 validation accuracy 雖在前期有較大波動，但最終整體維持於 80%上下。因此，相較樣板模型 validation 和 training accuracy 分別約 90%和 70%，經過 Batch Normalization 處理後的模型有顯著的 validation accuracy 提升，表示過渡擬合的情況有所改善，模型泛化能力提升。

此外，測試資料的 accuracy 從 68.7%上升至 79.1%，增加超過 10%，表示 Batch Normalization 的效果相當顯著、良好。

Data Augmentation

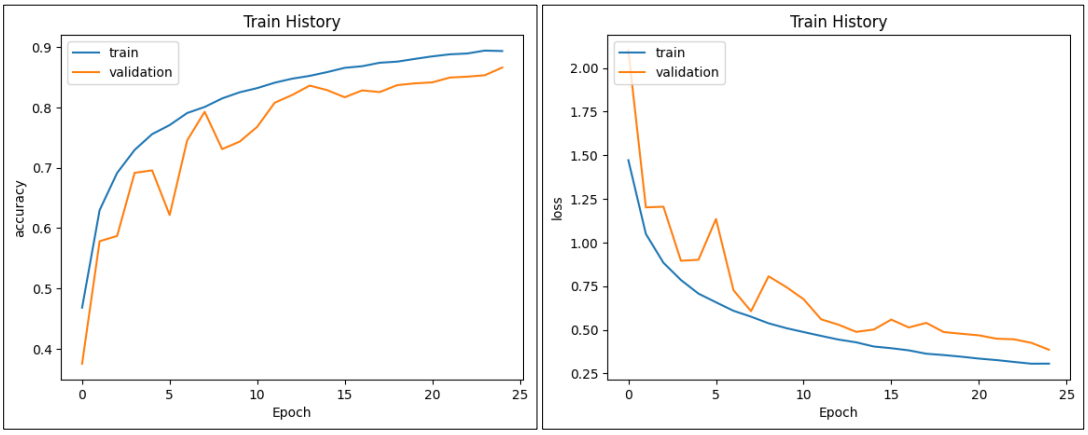
從 Batch Normalization 處理後的模型 train history 可發現，雖然其過渡擬合相較樣板模型已有明顯緩解，但訓練與資料集的準確度仍有一定差距，表示過渡擬合的情

況仍存在(如上圖中紅色記號)。因此，推測過渡擬合是因為訓練資料的多樣性太低，而不是源於模型過於複雜，故運用 Data Augmentation，藉增加資料多樣性，達到進一步減輕過渡擬合現象，並增加模型的泛化、推演能力。

```
 97 img_gen = ImageDataGenerator(rotation_range=15, width_shift_range=0.1, height_shift_range=0.1,
 98                              horizontal_flip=True, zoom_range=0.1, validation_split=0.2)
 99
100 batch_size = 32
101 train_generator = img_gen.flow(x_train_norm, y_TrainOneHot, batch_size=batch_size, subset='training')
102 val_generator = img_gen.flow(x_train_norm, y_TrainOneHot, batch_size=batch_size, subset='validation')
103
104 steps = train_generator.n // train_generator.batch_size
105 validation_steps = val_generator.n // val_generator.batch_size
106
107 train_history = model.fit(train_generator, steps_per_epoch=steps, validation_data=val_generator,
108                           validation_steps=validation_steps, epochs=50, shuffle=True)
```

改變原本的 fit()，從 tensorflow.keras.preprocessing.image 匯入 ImageDataGenerator，隨機變換原本影像的角度、縮放等，再用改變過的影像做模型訓練，使影像有更多變化性。

(97) 使用 ImageDataGenerator()語法，將影像隨機旋轉±15 度、隨機水平移動±10%、隨機垂直移動±10%、隨機水平翻轉圖片、隨機放大或縮小影像，最後分割 20%的資料作為驗證資料。
(100) 設定 batch_size 為 32，表示 32 筆資料為一組。
(101~102) 運用 flow()語法，以動態取樣的方式分別設定訓練和驗證資料的 generator，計算後續 fit()語法會需要的 train_generator 和 validation_data。
(104~105) 計算後續 fit()語法會需要的 steps_per_epoch、validation_steps，分別表示 32 筆資料為一組後，每一輪 epoch 要從 generator 中取幾次 batch 才能完整訓練。
(107) 使用 fit()語法訓練模型。



Accuracy of testing data = 87.0%

從 Data Augmentation 後的 train history 可發現，訓練資料的 accuracy 隨訓練穩定上升至 90%，驗證資料的 accuracy 雖然在前期（ 第 5 至 10 次 epoch ）有較大起伏，但之後趨於穩定，並接近 87%，由於訓練與驗證的 accuracy 差距小，顯示過渡擬合的問題被解決、模型的訓練過程穩定，又測試資料的 accuracy 從 79.1%升至 87.0%，上升近 8%，表示 Data Augmentation 的效果非常好。

Early Stopping + Dropout

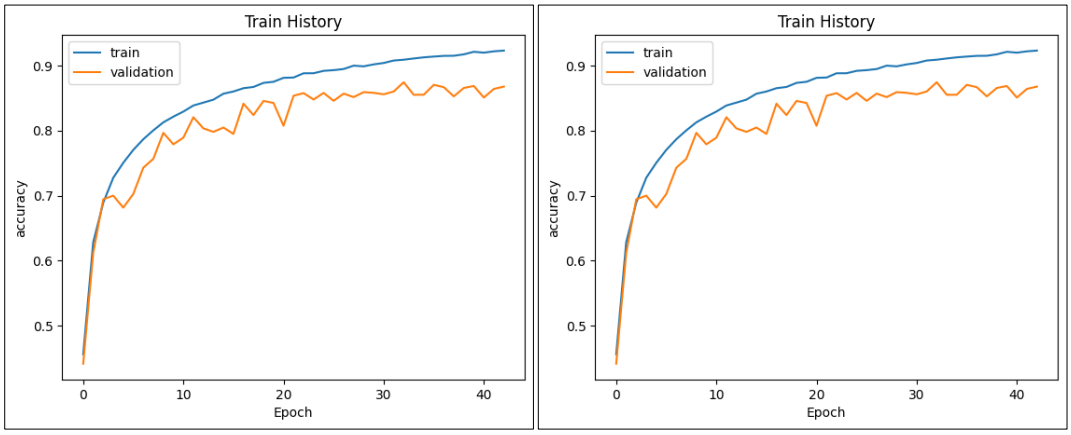
藉 Data Augmentation 後的 train history 可知，過渡擬合問題已不存在。因此，為增加 accuracy，我增加 epoch 從 25 至 50，並使用 Early stopping 搭配 Dropout，讓模型在發生過渡擬合前便停止訓練，並在訓練過程中隨機地忽略一些神經元，使模型對神經元的特定權重敏感度下降，提升其泛化能力也不易過渡擬合。

```
88 model.add(Dense(128, use_bias=False))
89 model.add(BatchNormalization())
90 model.add(Activation('relu'))
91 model.add(Dropout(0.2))
92 model.add(Dense(10, activation='softmax'))
93 model.summary()
```

由於 Dropout 只針對大且帶有 Dense layer 的 NN 有效果，故增加 Dropout(0.2)於輸出層前。原先曾嘗試只用 Dropout，但反而使測試資料 accuracy 下降。

```
107 early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, min_delta=0.0001, restore_best_weights=True)
108 train_history = model.fit(train_generator, steps_per_epoch=steps, validation_data=val_generator,
109                           validation_steps=validation_steps, epochs=50, shuffle=True, callbacks=[early_stopping] )
```

使用 EarlyStopping()語法，偵測 val_accuracy 的值，當該數值超過 10 個週期都沒有改善便停止訓練，並判斷最小變化量為 0.0001，且訓練結束時權重會回復到訓練過程中表現最佳的數值。



Accuracy of testing data = 87.9%

從 Early stopping 搭配 Dropout 後的 train history 可知，訓練資料的 accuracy 隨著訓練增加而上升，最後達到約 95%，而驗證資料的 accuracy 雖有微小起伏，但整題而言逐漸上升，並趨近於 88%。因兩曲線無明顯落差，故無過渡擬合情況。另外，測試資料的 accuracy 增加微幅，僅從 87.0%升至 87.9%。

2.2 所完成的最佳模型

(模型結構、所使用的超參數、模型訓練與驗證、測試資料的分類準確度)

模型結構：

模型具備三組濾波器數量從 32、64 增至 128 的卷積層，且每組內含三層 Conv2D，又每個 Conv2D 皆搭配一個 Batch Normalization 與激活函數 ReLU，最後接 MaxPooling2D 做降維。

在 Flatten 攤平成一維向量後，加入兩層 Dense(128)，並在第二層 Dense 後使用 Dropout(0.2)以提升泛化能力、抑制過擬和，最後用 Dense(10)和 softmax 做輸出。

模型建立後，訓練前，使用 Data Augmentation 增加訓練資料多樣性；訓練時，使用 Early Stopping 使模型在過渡擬合前停止訓練，避免過渡擬合。

使用超參數：

Optimizer：Adam；Batch Size：32；Epochs：50；Dropout rate：0.2；Validation Split：0.2；EarlyStopping：monitor=val_accuracy、patience=10、restore_best_weights=True

模型訓練與驗證：

模型前後運用 Batch Normalization 和 Dropout 穩定模型訓練過程、減緩過渡擬合，再使用 Data Augmentation 藉由將影像旋轉、平移、水平翻轉和縮放，提高訓練資料多樣性，顯著提升模型資料泛化能力，解決過渡擬合問題。

訓練過程中，利用 Early Stopping 在驗證資料準確度無明顯提升時，即停止訓練，並保留準確度最佳時的模型權重。

測試資料的分類準確度：

原始分類準確度為：68.7%；使用 Batch Normalization 後，分類準確度為 79.1%；接著，加入 Data Augmentation 後，分類準確度為 87.0%；最終，加入 Early stopping 搭配 Dropout 後，分類準確度為 87.9%。

根據最後的 train history 可知，兩曲線差距小，模型無過渡擬合問題，學習效果良好。

3. a) 敘述對模型過渡擬合(model overfitting)的理解
   b) 這作業所提供的樣板模型(template model)，其訓練是否發生過渡擬合?

a) 當過渡擬合發生時，通常是因為模型和訓練資料的擬合過於完美，使模型甚至學習到訓練資料的例外或極端情況，導致模型失去泛化、舉一反三的能力，不過，有時過渡擬合是因訓練資料的多樣性過低，而不是因模型訓練的過於複雜。過渡擬合時，training accuracy 相當高，但 validation accuracy 相較之下低許多。

b) 從樣板模型的 train history 可以發現，training accuracy 隨著 epoch 增加而逐漸上升，最終超過 90%；然而，validation accuracy 在約第 5 個 epoch 後便停滯在 70%左右波動，表示模型擬合訓練資料的效果好，卻無法有效的應用到驗證資料上。此外，training loss 會隨訓練而持續下降，而 validation loss 一開始雖下降，但在 10 epoch 後逐漸上升，表示模型訓練到後期，其泛化能力下降。

綜上所述，訓練和驗證資料的 accuracy 和 loss 變化趨勢，皆符合過渡擬合的特徵，故合理推測樣板模型訓練時出現過渡擬合的情況。

4. a) 敘述對梯度消失(gradient vanishing)的理解

   b) 這作業所提供樣板模型，其訓練是否出現梯度消失問題?

   a) 梯度消失(gradient vanishing)是模型深度愈深愈容易出現的問題。在進行反向傳播，梯度(gradient) $\frac{\partial l}{\partial w}$ 多次偏微計算、愈往 input layer 傳遞時，若梯度變得很小，出現後面 layer 已經收斂完畢，但前面 layer 已經沒有有效的梯度可以訓練權重，導致維持初始亂數的情況，則會令模型訓練得到的答案等同於亂猜，訓練成果不理想。

   綜上所述，當梯度消失發生，靠近 input layer 的權重無法更新，則模型的訓練會停滯。即使初期 training accuracy 增加、loss 減少，但隨訓練繼續進行、梯度消失，accuracy 和 loss 曲線仍會趨近平坦。

   b) 觀察樣板模型的 train history 中 accuracy 和 loss 曲線可發現，其 training accuracy 隨訓練穩定遞增，training loss 也隨訓練遞減。由於兩曲線形狀並沒有梯度消失會出現的 accuracy 和 loss 停滯不前、趨於平坦，故推測樣板模型的訓練沒有梯度消失問題。

參考資料

1. 上課講義.....
2. http://www.deeplearning.com
3. Day 17～AI 從入門到放棄 - 資料增強
   https://ithelp.ithome.com.tw/articles/10247445
4. [DAY11] NN model 的訓練設定－訓練週期(epoch)與批次(batch)
   https://ithelp.ithome.com.tw/articles/10298302
5. 【 第 11 天 】訓練模型-Keras Application 重要函數
   https://ithelp.ithome.com.tw/m/articles/10272770
6. [Day 18] 回呼模組 (2)：EarlyStopping
   https://ithelp.ithome.com.tw/m/articles/10361486