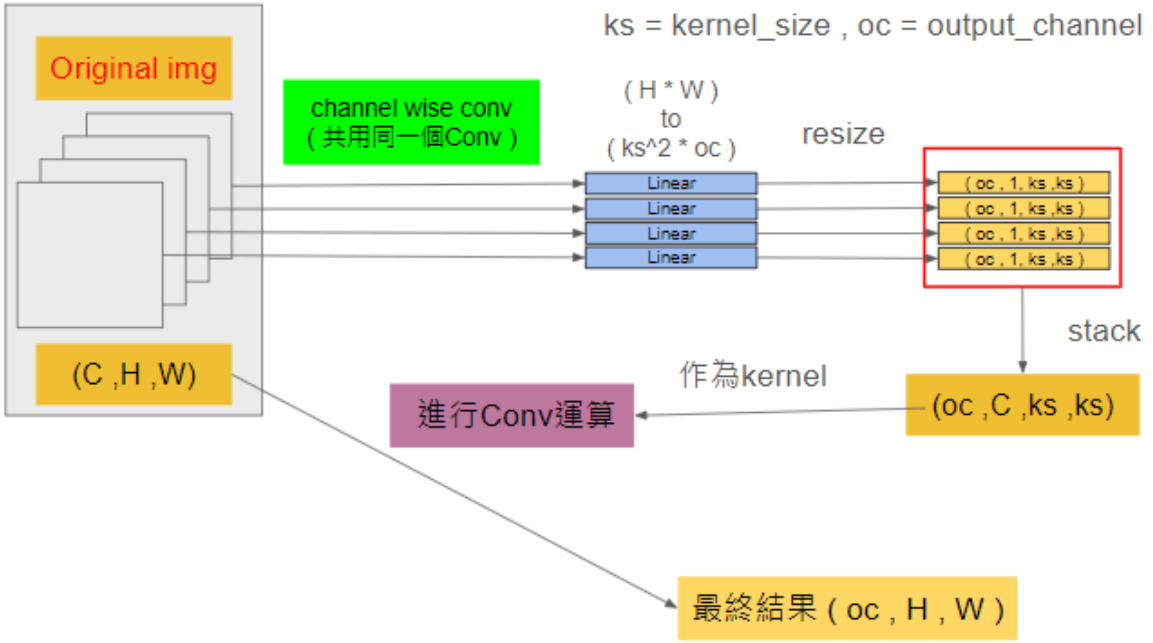
問題**1**;

設計一個Convolution Module，且該Convolution Module可以處理多通道的輸入

方法:

以下是架構圖:



其主要的設計理念是:

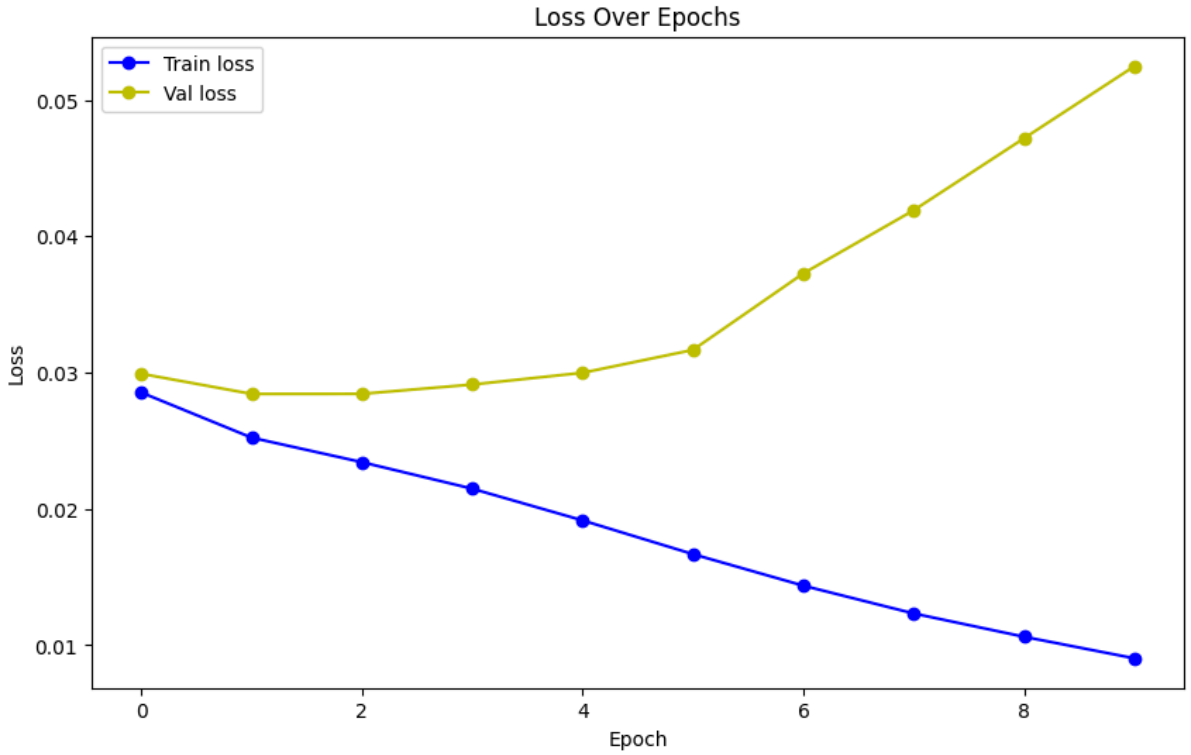
給定一個**channel**的**feature_map**，就使用**linear**來產生該**feature_map**的**kernel**，再使用該**kernel**對原圖進行捲積，即可達成處理多個Feature map 的資訊

Result

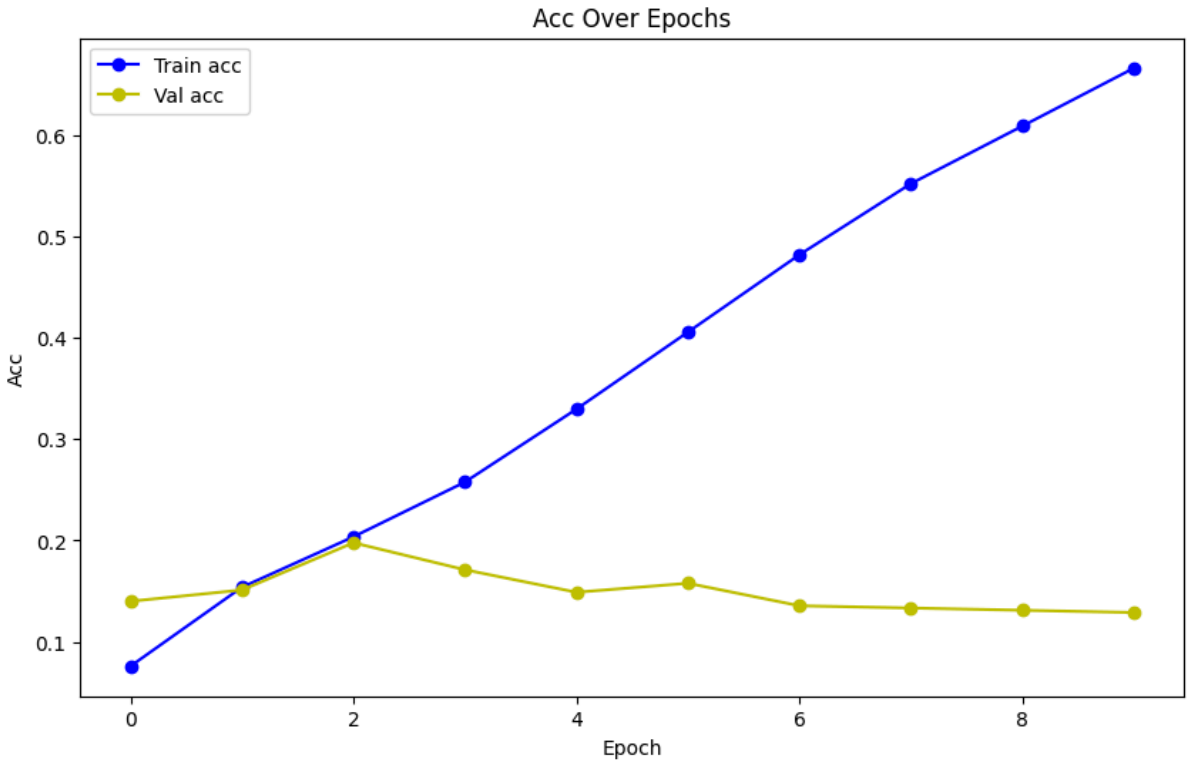
在3_channel下的實驗:

Flex_channel_CNN

Loss Curve :

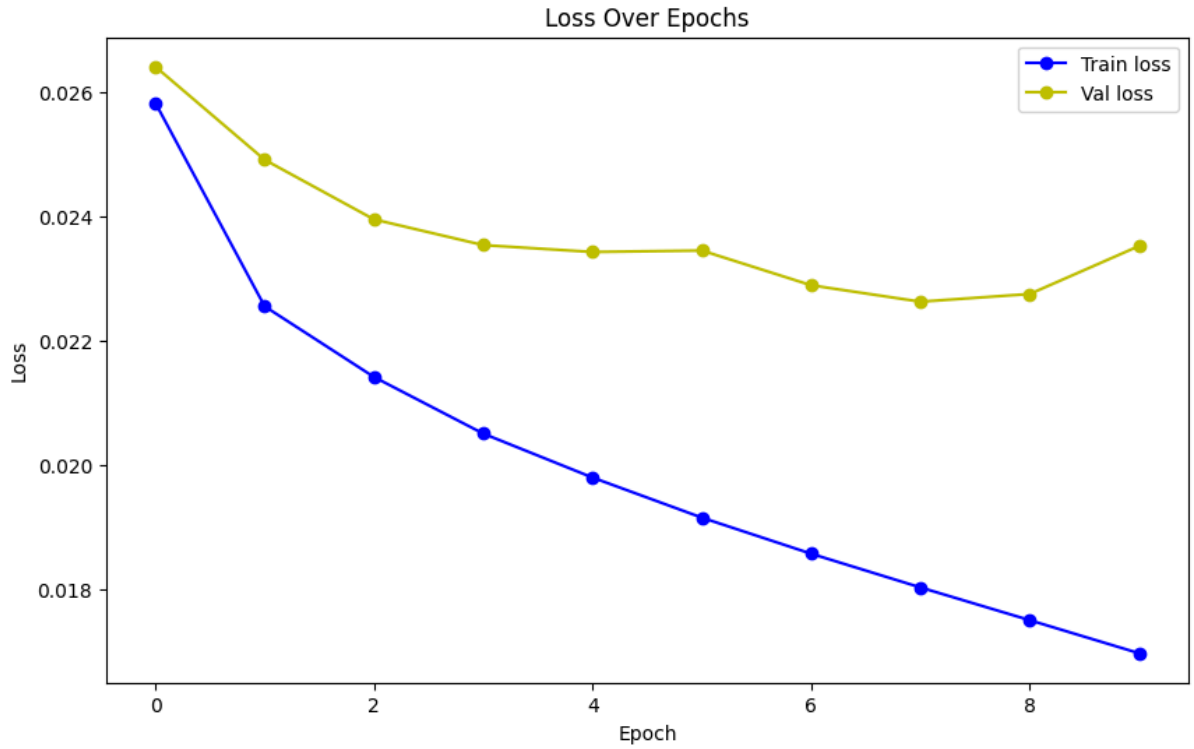


Accuracy Curve : val acc最高到0.19

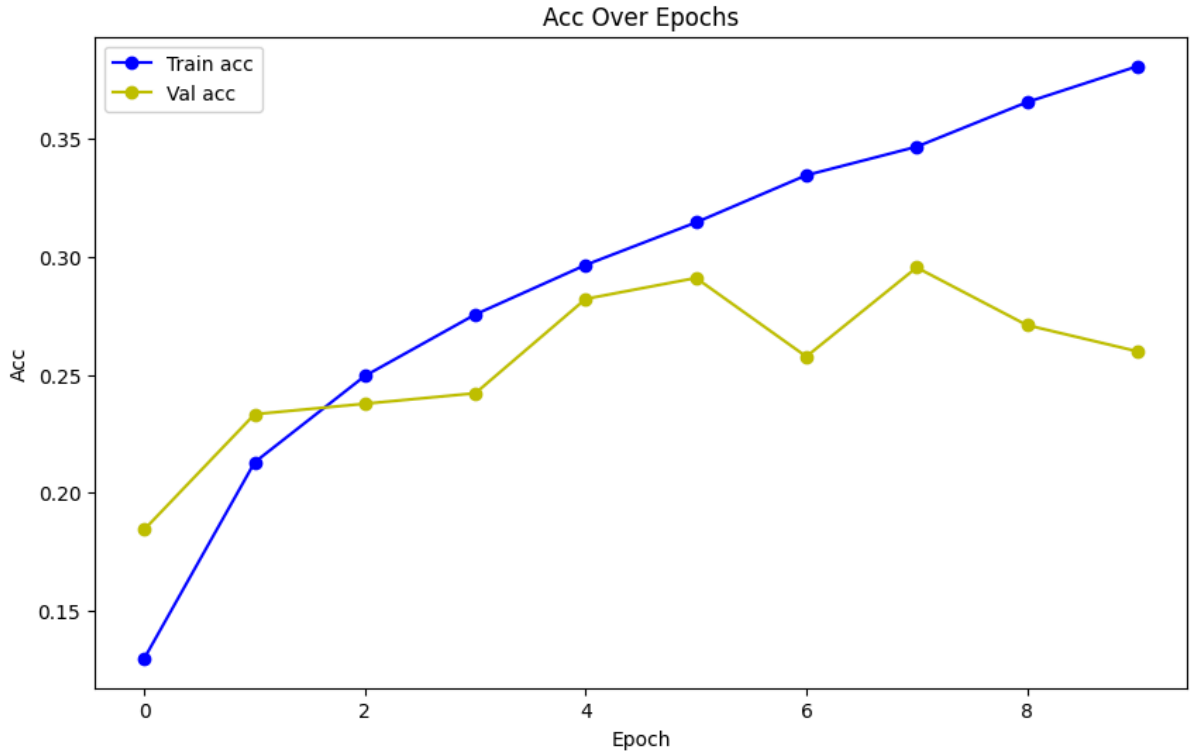


3_layer_CNN

Loss Curve :



Accuracy Curve : val acc最高到0.29



參數量比較:

在輸入3_channel , 圖片大小為(128 , 128 , 3)的狀況下
輸出channel數量為32, kernel_size = 3*3 , padding = 'same'

使用一般Conv方法

Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding = 'same')
即可達成
所使用的參數量為
kernel部分 = 3 (輸入通道數)* 3 * 3(kernel大小) * 32(kernel數)
= 864
Bias 部分 = 32(kernel數)
總共：864 + 32 = 896

若使用自創的方法

則是需要用到
Conv2d(in_channels=1, out_channels=1, kernel_size=3,padding='same')
和 nn.Linear(width*height,ks * ks * output_channel)
Conv部分需要
kernel部分 = 1 (輸入通道數)* 3 * 3(kernel大小) * 32(kernel數)
= 288
Bias 部分 = 32(kernel數)
總共：288 + 32 = 320
Linear部分需要
Linear部分
=128*128(feature_map大小)*3*3(kernel大小)*32(輸出channel數)
= 4718592
Bias 部分 = 32(輸出channel數)
總共： 4718592 + 32 = 4718624

討論：

原本的conv方法並不會受圖片大小的影響而增加參數量, 但會受輸入的圖片的channel數影響, 我的方法則相反, 不受輸入的圖片channel數影響參數量, 但會受到圖片大小影響, 故我的方法較適合用在經過多層捲積或者Maxpooling之後, 其圖片大小變得很小, 但其channel數很大的時候, 可以有效的節省參數。

問題**2:**

使用 2~4層的CNN layer, Transformer 或 RNN ,使其performance到達Resnet34的90%

方法**:**

以下是print(model)的結果

```
DenseNet(
  (conv1): Conv2d(3, 512, kernel_size=(7, 7), stride=(2, 2), padding=(2, 2), bias=False)
  (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (features): ModuleList(
    (0): Sequential(
      (0): Bottleneck(
        (conv): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Transition(
      (conv): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (2): Sequential(
      (0): Bottleneck(
        (conv): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc): Linear(in_features=1024, out_features=50, bias=True)
)
```

總共使用四層Conv2D的module

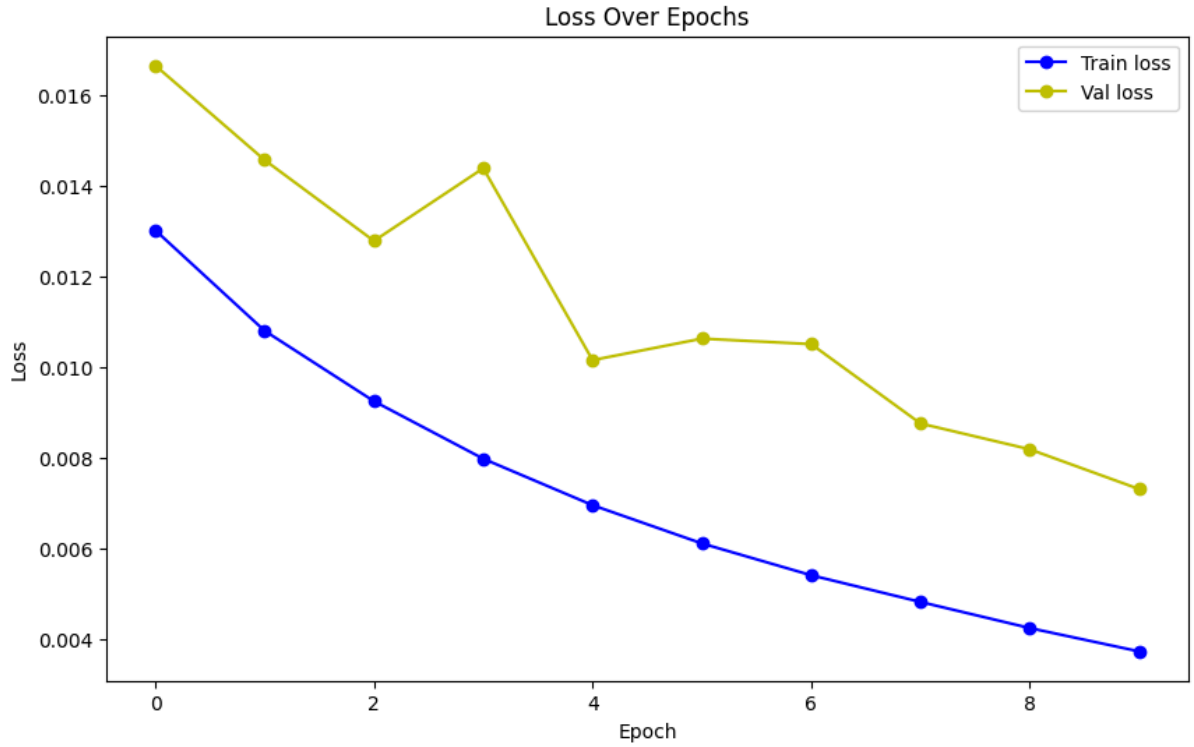
其設計方式為:

根據DenseNet的概念進行修改, 實際做法為

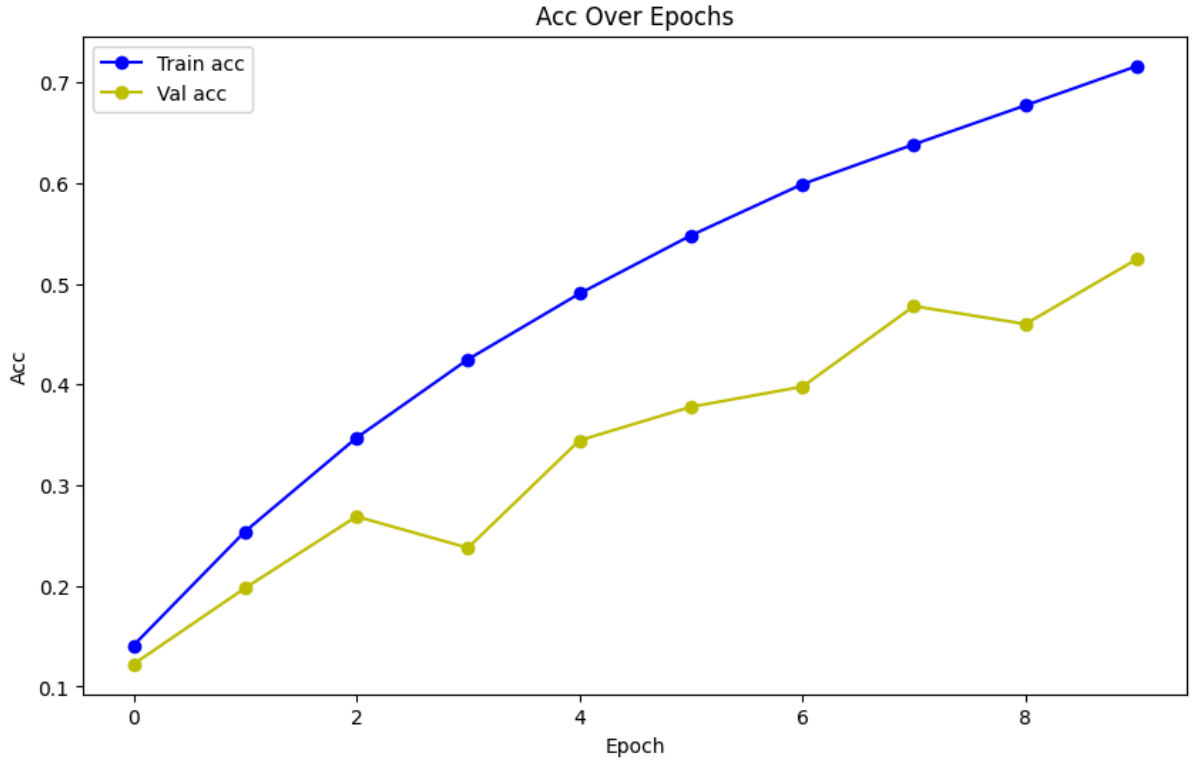
1. 經過第一層Conv2D使其從channel數從3->512
2. 經過BatchNorm標準化和MaxPool2D濃縮資訊
3. 經過第二層Conv2D產生channel 512的feature_map
4. concat(第一層Conv2D , 第二層Conv2D)channel為1024
5. 將concat後的結果過第三層Conv2D輸出為512channel
6. 經過BatchNorm標準化和MaxPool2D濃縮資訊
7. 將第三層Conv2D經過第四層Conv2D輸出為512channel
8. 經過BatchNorm標準化
9. concat(第三層Conv2D , 第四層Conv2D)channel為1024
10. 經過BatchNorm標準化和Adaptive_avg_pooling
11. 最後經過Linear輸出結果

Result:
以下是**2~4CNN_model**在10epochs內的結果

Loss Curve :



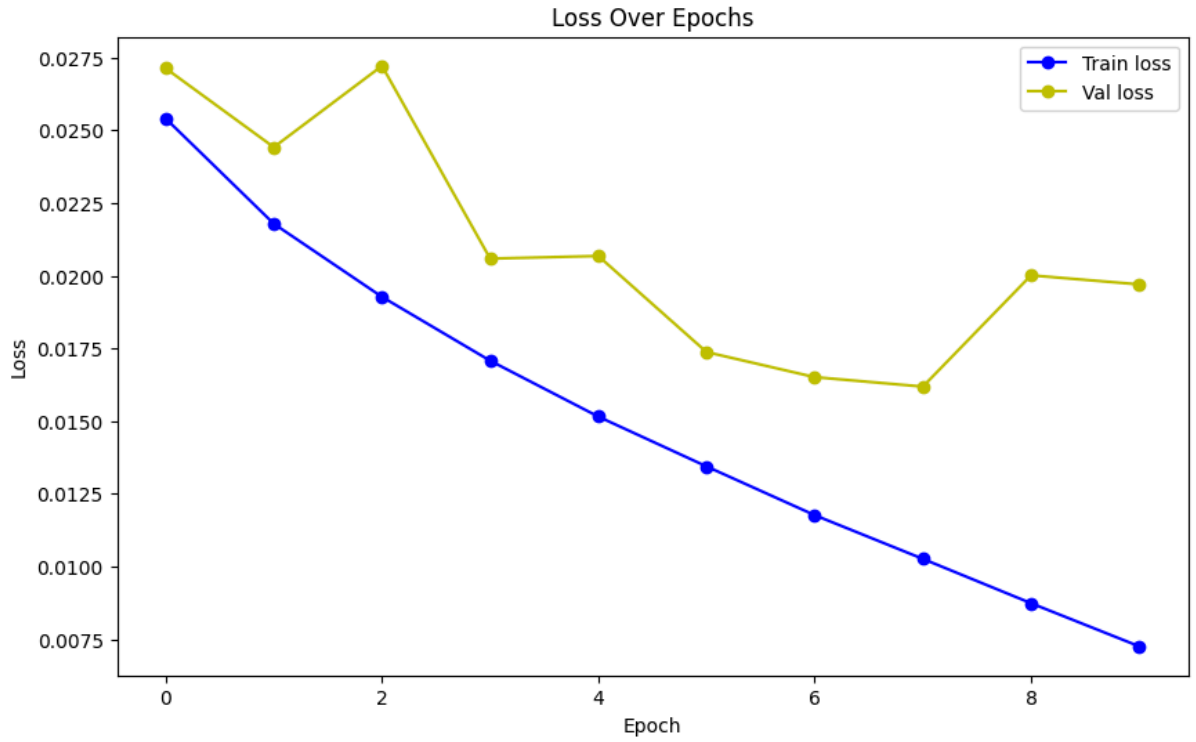
Accuracy Curve : 最高performance到達0.524 , testing_acc為0.53



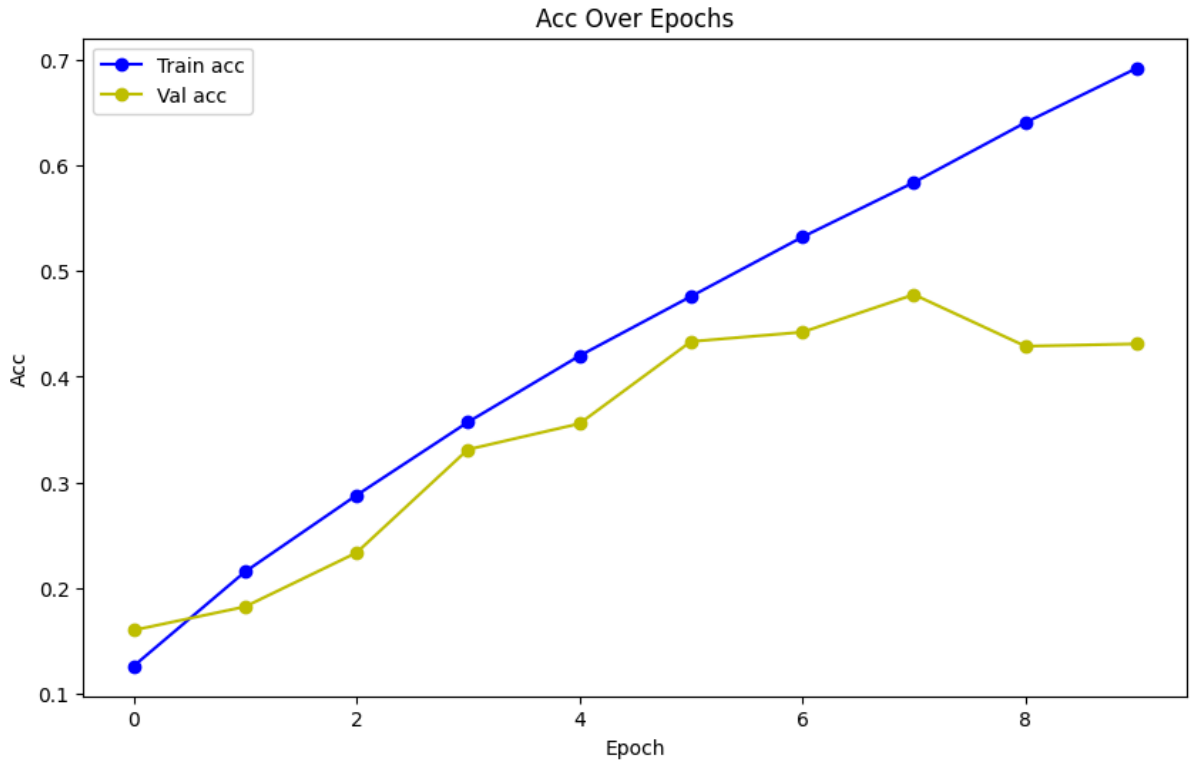
Result:

以下是**Resnet34**在10epochs內的結果

Loss Curve :



Accuracy Curve : 最高performance到達0.477 , testing_acc為0.45



參數量
以下是**2~4CNN_model**的參數量

```
==========================================================================================
Total params: 5,375,538
Trainable params: 5,375,538
Non-trainable params: 0
Total mult-adds (G): 3.86
==========================================================================================
```

以下是**Resnet34**的參數量

```
==========================================================================================
Total params: 21,847,722
Trainable params: 21,847,722
Non-trainable params: 0
Total mult-adds (G): 1.20
==========================================================================================
```

可以發現其自行設計模型的參數量約為Resnet34的**2.5**倍左右
才能達到和Resnet34相似的performance
這兩個模型本質上都是使用Conv. layer處理圖片的資訊
但這兩個模型在深度上有相當大的差距
可以推測說，在相同的參數下，將模型加深會比單純增加**kernel**數
獲得更高的performance

解決問題的流程:

由於在限制下，只能使用2~4層的Conv2D，模型深度也沒有辦法太深，故只能從增加參數量下手，而當中有嘗試過CBAM，SENET這些增加channel之間關係，但這兩個方法一方面會用掉不少Conv2D(CBAM的channel attention和spatial attention就要用掉3層，SENet就要用掉2層)，一方面他們的輸入輸出的Chanel數都相同，沒有辦法有效的擴大參數量，偶然之下發現一般疊四層一般的Conv2D經過BN和Maxpooling效果不錯，但仍未到達ResNet34的90%，後來嘗試看看DenseNet的方法，借用其**Concat**前面**layer**輸出的想法，可以更有效的利用前面layer的output，且也能解省Conv layer，也可以在硬體資源有限的情況下，擴大output的channel數量。

Github連結:
https://github.com/IIRchen/Deep_Learning_HW2.git