Model Architecture

使用Resnet-18作為基本模型架構，他在架構上是使用一連串的殘缺塊形成的殘缺網路去作為模型架構，目的是因為模型愈深，非線性處理到最後可能引發退化現象，也就是模型準確率下降，這是因為我們的模型隨著訓練的最後竟然失去了衡等變化的能力，因此建立快捷連結使其在線性轉換和非線性轉換多了一個平衡，(快捷連結是我們允許某一層跳過一個或多個層到後續層的輸入，目的是不做沒有意義的轉換)。而我會選擇此模型的原因是要將食物辨識11種類別，且食物辨識所需要的特徵值較為複雜，而我為了滿足夠高的準確率又不想因為神經網路加深使模型準確下降。優點是模型能保有一定的準確率且不會下降，也能最為pretrain模型方便後續訓練，缺點是最後得出的準確度雖有0.7到0.8但可能還不夠滿足我們的要求，且跌宕50到150次左右，準確值已到瓶頸。不過整體優點大於缺點，因為可以將其先做50到150次的訓練作為pretrain model，後續再轉換模型架構提高其準確度。

Method

import torch.nn as nn

import torch.nn.functional as F

在 **RestNetBasicBlock** 中，透過 **x + output** 來實現快捷連結。這裡的 **x** 是模塊的輸入，**output** 是第二個卷積層的輸出

class RestNetBasicBlock(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels, stride):

super(RestNetBasicBlock, self).\_\_init\_\_()

# 第一個卷積層

self.conv1 = nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, stride=stride, padding=1)

self.bn1 = nn.BatchNorm2d(out\_channels) # 第一個批標準化層

# 第二個卷積層

self.conv2 = nn.Conv2d(out\_channels, out\_channels, kernel\_size=3, stride=stride, padding=1)

self.bn2 = nn.BatchNorm2d(out\_channels) # 第二個批標準化層

def forward(self, x):

output = self.conv1(x)

output = F.relu(self.bn1(output))

output = self.conv2(output)

output = self.bn2(output)

return F.relu(x + output) # 殘差連接和 ReLU 激活

在 **RestNetDownBlock** 中，額外的 1x1 卷積層被用來調整通道數和輸入的空間尺寸，以使得與卷積層的輸出可以進行相加。最終的輸出是 **extra\_x + out**，其中 **extra\_x** 是額外卷積層的輸出，**out** 是第二個卷積層的輸出

class RestNetDownBlock(nn.Module):

def \_\_init\_\_(self, in\_channels, out\_channels, stride):

super(RestNetDownBlock, self).\_\_init\_\_()

# 第一個卷積層

self.conv1 = nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, stride=stride[0], padding=1)

self.bn1 = nn.BatchNorm2d(out\_channels) # 第一個批標準化層

# 第二個卷積層

self.conv2 = nn.Conv2d(out\_channels, out\_channels, kernel\_size=3, stride=stride[1], padding=1)

self.bn2 = nn.BatchNorm2d(out\_channels) # 第二個批標準化層

# 額外的 1x1 卷積層

self.extra = nn.Sequential(

nn.Conv2d(in\_channels, out\_channels, kernel\_size=1, stride=stride[0], padding=0),

nn.BatchNorm2d(out\_channels) # 批標準化層

)

def forward(self, x):

extra\_x = self.extra(x) # 額外的卷積層

output = self.conv1(x)

out = F.relu(self.bn1(output))

out = self.conv2(out)

out = self.bn2(out)

return F.relu(extra\_x + out) # 殘差連接和 ReLU 激活

class Classifier(nn.Module):

def \_\_init\_\_(self,x):

super(Classifier, self).\_\_init\_\_()

# 第一個卷積層

self.conv1 = nn.Conv2d(3, 64, kernel\_size=7, stride=2, padding=3)

self.bn1 = nn.BatchNorm2d(64) # 第一個批標準化層

self.maxpool = nn.MaxPool2d(kernel\_size=3, stride=2, padding=1)

self.layer1 = nn.Sequential(

RestNetBasicBlock(64, 64, 1),

RestNetBasicBlock(64, 64, 1)

)

self.layer2 = nn.Sequential(

RestNetDownBlock(64, 128, [2, 1]),

RestNetBasicBlock(128, 128, 1)

)

self.layer3 = nn.Sequential(

RestNetDownBlock(128, 256, [2, 1]),

RestNetBasicBlock(256, 256, 1)

)

self.layer4 = nn.Sequential(

RestNetDownBlock(256, 512, [2, 1]),

RestNetBasicBlock(512, 512, 1)

)

self.avgpool = nn.AdaptiveAvgPool2d(output\_size=(1, 1))

self.fc = nn.Linear(512, 11) # 全連接層

def forward(self, x):

out = self.conv1(x)

out = self.bn1(out)

out = F.relu(out)

out = self.maxpool(out)

out = self.layer1(out)

out = self.layer2(out)

out = self.layer3(out)

out = self.layer4(out)

out = self.avgpool(out)

out = out.reshape(x.shape[0], -1)

out = self.fc(out)

return out

Learned with course and homework

在本次的課程中我學到將理論和實作的結合，像我就有遇到其中一個狀況，所有label值等於負一，也就是圖形無法辨識，後來逐行debug才發現在圖片路徑上kaggle是’D/’而傳統python是’\\’所以在kaggle上可以正常識別，但在spyder中無法識別，我原本還以為是模型架構問題所以將捲基層的通道數全部算過一遍，最後用迴圈去看label值才發現問題所在。不過也讓我對每一層有更深入的了解，像是激活層對應的是數學的非線性轉換等。

Conclusion

總結來說，我了解到模型的訓練很花時間，且有可能最後訓練出一堆垃圾，因此我的想法是如果以後想要訓練的話可能可以先挑有一定準確度且適合自身入物的模型跌宕50-150次作為prettain模型，之後再一需其轉換模型以提高實際準確度並降低損失值。而且我這次的程式碼並沒有將-1這個標籤過濾掉，其實這會有一定程度的隱憂，訓練可能會因為部分毀損sample而直接停擺，不過這次老師給的檔案樣本都沒有問題，所以可以不用擔心。但以實作來說還是要考慮部分可能無法識別的標籤做過濾。

結果:

一張含有 文字, 螢幕擷取畫面, 行, 字型 的圖片

自動產生的描述

學生: 數四乙 蔡德誠 s0922123