Lab4 Image classification deep learning

一、Description of model

```
############ Build the model here ##########
class ConvNet(nn.Module):
    def __init__(self,num_classes=2):

        super(ConvNet, self).__init__()

        #self.resnet_layer = nn.Sequential(*list(model.children())[:-2])
        #
        #self.transion_layer = nn.ConvTranspose2d(2048, 2048, kernel_size=14, stride=3)
        #self.pool_layer = nn.MaxPool2d(32)
        #self.linear_layer = nn.Linear(2048, 8)

        self.layer1 = nn.Sequential(
        nn.Conv2d(3, 8, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(8),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))

        self.layer2 = nn.Sequential(
        nn.Conv2d(8, 16, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(16),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))


        self.layer3 = nn.Sequential(
        nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))

        self.layer4 = nn.Sequential(
        nn.Conv2d(32, 64, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))

        self.layer5 = nn.Sequential(
        nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(128),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))

        self.layer6 = nn.Sequential(
        nn.Conv2d(128, 256, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(256),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))

        self.fc1 = nn.Linear(2304, 1000)
        self.fc2 = nn.Linear(1000, 500)
        self.fc3 = nn.Linear(500, 100)
        self.fc4 = nn.Linear(100, num_classes)


    def forward(self, x):

        #out = self.resnet_layer(x)
        #out = self.transion_layer(out)
        #out = self.pool_layer(out)
        #out = out.view(out.size(0), -1)
        #out = self.linear_layer(out)

        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = self.layer6(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc1(out)
        out = self.fc2(out)
```

6 層 CNN
Conv kernel size:5*5
Maxpool kernel size:2*2

4 層 FC
最後輸出為 two classes

將多維 tensor 展開成一維，方便後面 FC 使用

二、Description of hyper parameter

Batch size:100

```
# you can use different batch size
train_data_loader = data.DataLoader(train_dataset, batch_size=100, shuffle=True,  num_workers=4)
val_data_loader = data.DataLoader(val_dataset, batch_size=100, shuffle=True,  num_workers=4)
```

有時候如果 batch size 設太大會出現 CUDA out of memory 的錯誤訊息，通過調小 batch size 可以解決該問題。

Learning rate:0.0001

```

lr = 0.0001
```

在閱讀相關 paper 後發現，當 batch size 調小時，learning rate 跟著調小有機會得到較好的結果。

Optimizer:Adam

```
optimizer = optim.Adam(model.parameters(), lr=lr)
```

Adam 演算法結合了 momentum 和 RMSprop 的優點，不僅在計算參數更新方向前會考慮前一次的參數更新的方向，可以避免在 gradient descent 的過程中卡在 local minima，也會根據據梯度大小對學習率進行加強或衰減，當梯度越大就降低 learning rate，避免一次對參數做太大程度的更新，梯度越小則提升 learning rate。

Epoch:3

```
# start training
epochs = 3
```

發現如果 epoch 次數太多，validation 雖然提高，但 test 的結果卻變差，推測可能產生 overfitting

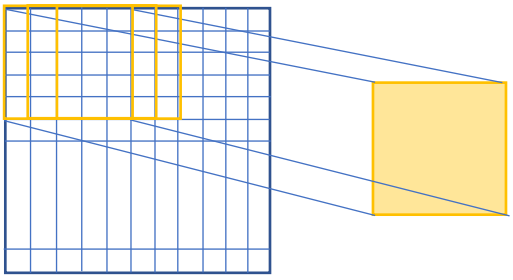
三、Difficulties

1.

剛開始 train model 把結果上傳到 kaggle，結果都只有 70 幾%，或是會跳出 RuntimeError: CUDNN_STATUS_BAD_PARAM 的錯誤訊息後來推測是因為 image 經過多層 convolution layer 以及 maxpooling 後，image size 發生以下改變：

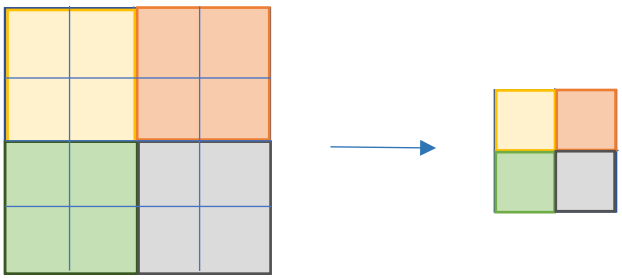
Convolution layer:

我將 kernel size 訂為 5*5，故根據下圖描述，經過處理的 image 邊長將比原本的 image 小 4



Maxpooling:

我將 kernel size 訂為 2*2，故根據下圖描述，經過處理的 image 邊長會變為原本的 1/2



所以如果一開始隨便設一數字做為 image size，則可能在經過處理後出現小數或負值而影響訓練結果。

2.

```
# data transform, you can add different transform methods and resize image to any size
img_size = 444
```

```
        self.layer6 = nn.Sequential(
        nn.Conv2d(128, 256, kernel_size=5, stride=1, padding=0),
        nn.BatchNorm2d(256),
        nn.ReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2))


        self.fc1 = nn.Linear(2304, 1000)
        self.fc2 = nn.Linear(1000, 500)
        self.fc3 = nn.Linear(500, 100)
        self.fc4 = nn.Linear(100, num_classes)
```

一開始我常常在改動參數過後，而顯示 fully connect 的 input 大小輸入錯誤，導致矩陣乘法無法執行，後來才發現這個數字應該是經過多層 CNN 得到的參數數量，以上述為例，image size 經過 6 層 CNN 後大小為 3*3，而我們在最後一層得到 256 個 kernel，所以 3*3*256 就是 fc 的 input 大小 2304。

3.

這次的 training data 跟 testing data 好像有些差距，所以在做 validation 時得到的 accuracy 參考價值較低。

四、Improve accuracy

1.

Pretrained model

```
import torchvision.models as models
```

```
####################  implement your optimizer ###################################
## you can use any training methods if you want (ex:lr decay, weight decay.....)
#model = models.resnet101(pretrained=True)
#for param in model.parameters():
#    param.requires_grad = False

#model.fc = nn.Linear(100352, 2)
#model = model.to(device)

```

使用 resnet101 的 model，然後藉由將參數的 requires_grad 設為 false 防止在做 backpropagation 的時候計算這些梯度，然後將 FC 最後一層改成對應我們需要的分類結果。

結果：正確率並沒有顯著提升

2.

增加 CNN 和 FC 層數

```
    def forward(self, x):

        #out = self.resnet_layer(x)
        #out = self.transion_layer(out)
        #out = self.pool_layer(out)
        #out = out.view(out.size(0), -1)
        #out = self.Linear_layer(out)

        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = self.layer6(out)
        out = out.reshape(out.size(0), -1)
        out = self.fc1(out)
        out = self.fc2(out)
        out = self.fc3(out)
        out = self.fc4(out)

        return out
```

結果：就如同上課時老師提到的 neural network 深度越深，訓練出來的模型表現得越好。

3.

Normalize

```
train_transform = transforms.Compose([
                                transforms.Resize((img_size,img_size)),
                                transforms.ToTensor()
                                #train_transform.Translation(),
                                #transforms.RandomHorizontalFlip()
                                #transforms.ToPILImage()
                                #transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
                                ])
```

結果：正確率沒有提升