學號：b07901014　系級：電機二　姓名：陳希格

1. 請描述如何將助教的程式碼 (包含 classification 與 regression) 從二階的 MAML 改成一階的 MAML (作答請盡可能詳盡，以免助教誤會)，並且比較其最後在 classification 上的 accuracy (5-way-1-shot)。因此你的 GitHub 上會有 p1_classification.py 和 p1_regression.py 兩個檔案，分別是 classification 和 regression 的一階版本。

   配分: classification 修改 (1) regression 修改 (1) report 一階做法在 classification 上的 accuracy (0.5)

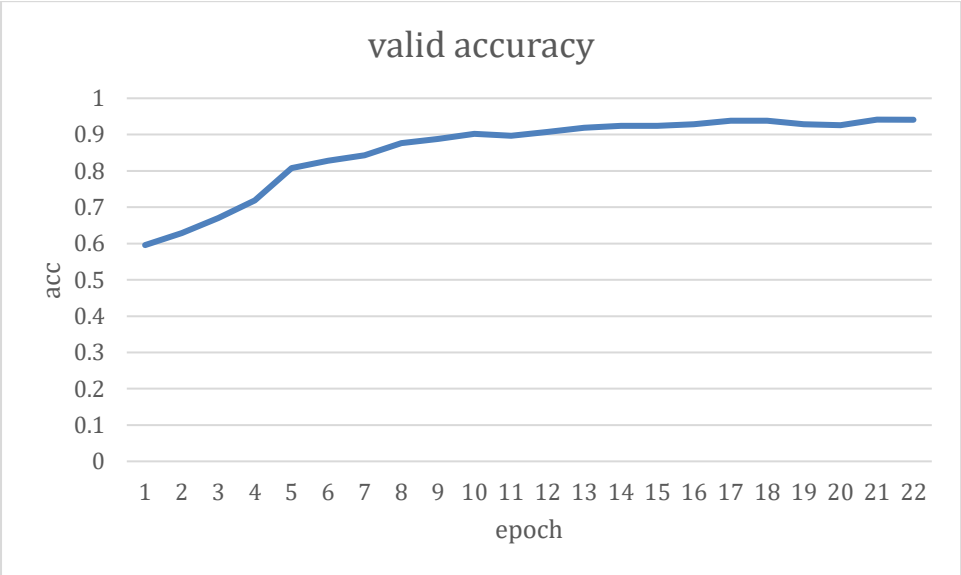
   regression 將 little_l.backward 的 create_graph 改成 False
   classification : 將 grads = torch.autograd.grad 的 create_graph 改成 False
   一階 accuracy = 0.86 (epoch =160)
   二階 accuracy = 0.91 (epoch = 40)
   accuracy 下降了一些，但其實不多，如果把 epoch 大一點的話 accuracy 應該還可以更高

2. 請將 classification 的程式碼改成 inner loop 更新 5 次，並且使用 adagrad 與二階的 MAML，寫出其 pseudo code 與回報 accuracy (5-way-1-shot omniglot 分類任務)。並且以 outer loop 的更新次數為橫軸，分類的準確率為縱軸作圖，比較其差異。因此你的 GitHub 上要有 p2.py，對應本題的程式碼。配分:pseudo code (1) 作圖(1) report accuracy (0.5)

```
for inner_step in range(inner_train_step): # 這個 for loop 是 Algorithm2 的 line 7~
                                           # 實際上我們 inner loop 只有 update 一次
                                           # 所以我們還是用 for loop 來寫
    train_label = create_label(n_way, k_shot).cuda()
    logits = model.functional_forward(train_set, fast_weights)
    loss = criterion(logits, train_label)
    grads = torch.autograd.grad(loss, fast_weights.values(), create_graph = True) #
    if inner_step == 0:
        adag =  list(grads)
    count = 0
    fast_weightss = OrderedDict()
    for ((name, param), grad) in zip(fast_weights.items(), grads):
        if inner_step == 0:
            adag[count] =  0
        newpa = param - inner_lr * (grad /(adag[count] + ee)**0.5)
        fast_weightss[name] = newpa
        adag[count] +=  grad**2
        count+=1
        # 這裡是用剛剛算出的 ∇loss 來 update θ 變成 θ'
    fast_weights = fast_weightss
```

Adagrad 更新

除了上面的更改外，設定 inner_train_step = 5，還有 ee(ε) = 1 ，不然再 loop 很少次數的狀況下，adagrad 會表現得不好。



Test accuracy : 0.8566666666666666

前期收斂的快了很多，但後期就卡住了上不太去。

3. 實作論文 "How to train your MAML" (https://arxiv.org/abs/1810.09502) 中的一個 tip，解釋你使用的 tip 並且比較其在 5-way-1-shot 的 omniglot 分類任務上的 accuracy。助教其實已經實作了一個，請找出是哪一個 tip 並且不要重複。因此你的 GitHub 上要有 p3.py，對應本題的程式碼。

   配分：report 實作 tip 後的 accuracy (1) 解釋你使用的 tip (1) 找出助教實作的 tip (0.5)

   出助教實作的 tip: BNWB

   自己實作的 tip: DA ，也就是前面的 epoch 用一階的 MAML，後面用二階的 MAML，這樣可以兼顧速度和準確度。

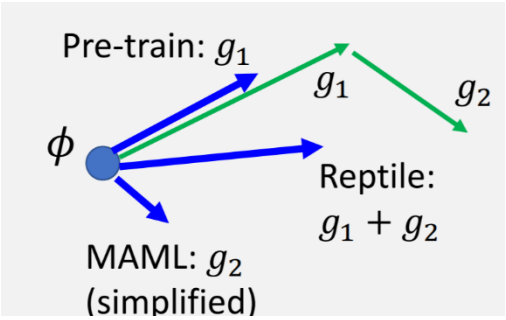
   實作成果:

   一階每個 epcoh 花費 20 秒，二階每個 epcoh 花費 32 秒，有成功加速(共 40 個 epoch)

   Test accuracy: 0.8916666666666669 ，比一階的高出許多(且不用更新那麼多 epoch)，但還是比二階的低一些，算是有兼顧速度和準確度。

4. 請實作 reptile 在 omniglot dataset 上，訓練 5-way-1-shot 的分類任務，並且回報其 accuracy。這題應該在 GitHub 上會有 reptile.sh 的 shell script，執行方式詳見投影片。

   配分：程式碼 (2) 回報 acc (0.5)



   實作方法: g1+g2 (其實也可以 inner loop 很多次，但為了計算速度只實作 g1+g2。)

   Test accuracy: 0. 9066666666666666 (epoch =20)

   可以發現 reptile 的表現極佳，epoch 所需的次數較少，且 accuracy 也很高。