HW4 Report　B05502159　陳慶瑾

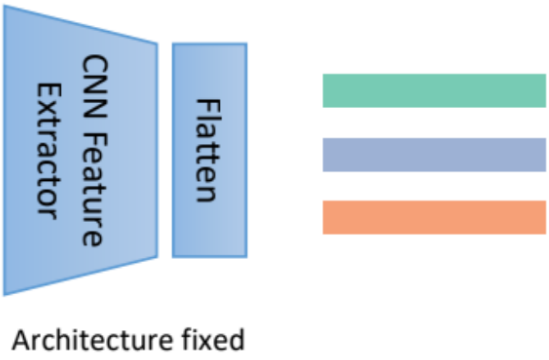
- **Problem1**

1. Architecture and Model

   Acc = 46.98 +- 0.85 %

   架構上我直接將經過 Encoder(Conv4)的 feature 取平均來當作 class 的 prototype。

   1) Training epoch = 200
   2) Optimizer = Adam
   3) Learning rate = 0.001
   4) Dist func = Euclidean
   5) Train way = 30
   6) Test way = 5
   7) # of shot = 1
   8) Episode = 100



Architecture fixed

2. Exp: Different Function (50 epoch/5-way)

   1) Euclidean Distance:　39.11 +- 0.78%
   2) Cosine Similarity:　37.21 +- 0.74%
   3) Sqrt Cosine Similarity:　37.84 +- 0.81%

$$\mathrm{SqrtCos}(x,y) = \frac{\sum_{i=1}^{m}\sqrt{x_i y_i}}{\left(\sum_{i=1}^{m} x_i\right)\left(\sum_{i=1}^{m} y_i\right)}.$$

$$H(x,y) = \left[\sum_{i}^{m}\left(\sqrt{x}_i - \sqrt{y}_i\right)^2\right]^{\frac{1}{2}} = \left[2 - 2\sum_{i}^{m}\sqrt{x_i y_i}\right]^{\frac{1}{2}},$$

   Reference:
   https://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0083-6

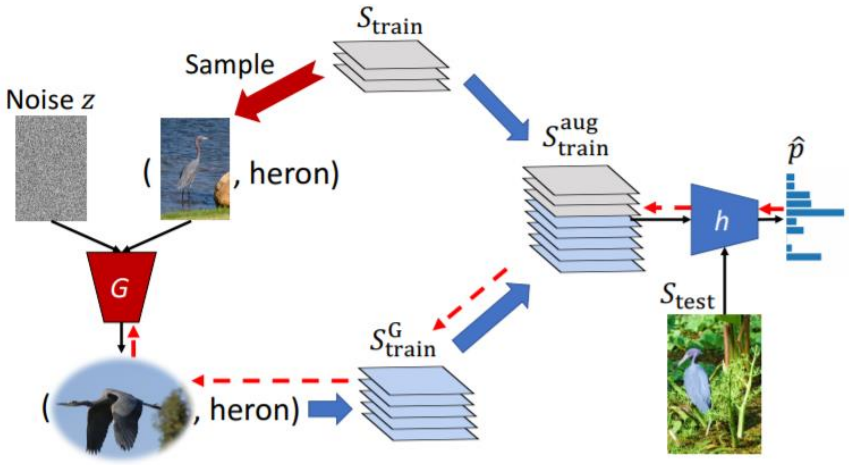
   結論：在各種條件固定的情況下，使用 Euclidean 會有最好的 Accuracy

3. Exp: Different K-shot (k = 1, 5, 10) (50 epoch/5-way)

   1) K = 1:　39.11 +- 0.78%
   2) K = 5:　40.35 +- 0.79%
   3) K = 10:　36.88 +- 0.71%

   結論：從直觀角度來思考，每個種類的資料數越多，理當要學出越好的學習方式。但發現當 K = 10 的時候，accuracy 反而下降。我認為最合理的解釋是 epoch 數不夠。

- **Problem2**
  1. Architecture and Model



Reference: Low-Shot Learning from Imaginary Data
https://arxiv.org/pdf/1801.05401.pdf
在第一題的前提之下，加入一個 generator，generator 的 input 為 size = 128 的 1D random normalized vector 以及從 Encoder output 的 feature，generator 的 output 為 M(1<= M <= 200)根 feature。詳細的 implement 如下：

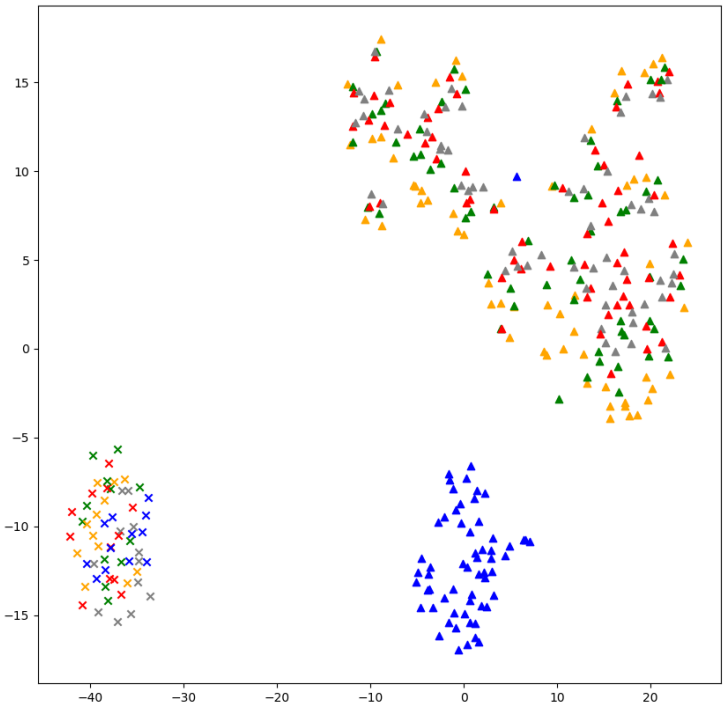
```
class Generator(nn.Module):
    def __init__(self, total_dim):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(total_dim, 4096),
            nn.LeakyReLU(),
            nn.Linear(4096, 2048),
            nn.ReLU(),
            nn.Linear(2048, 1600),
            nn.ReLU(),
        )

    def forward(self, z):
        return self.model(z)
```

Accuracy = 48.70 +- 0.88%

1) Training epoch = 200
2) Optimizer = Adam
3) Learning rate = 0.001
4) Dist func = Euclidean
5) Train way = 30
6) Test way = 5
7) # of shot = 1
8) Episode = 100
9) Learning rate(G) = 0.0004
10) Optimizer(G) = Adam

2. T-SNE



從 real data 跟 hallucinated data 的兩個 pool 來看，generator 對於 accuracy 最大的貢獻在於 class0(藍色)的 classification。其餘的幾個 class 和 real data 的 feature 一樣，沒有太多的群聚現象。

3. Exp: Different M (M = 10, 50, 100) (50epoch/5way-1shot)
   1) M = 10: 48.49 +- 0.87%
   2) M = 50: 47.92 +- 0.83%
   3) M = 100: 48.45 +- 0.89%

4. Discussion
   1) 就 accuracy 以及 tSNE 來說，可以推斷 generator 還是需要有一個 discriminator 來對抗學習，才有機會訓練出一個能輸出有效 feature 的 generator。另外就 epoch 來說，有觀察到大概訓練到 40epoch 的時候，generator 就會趨近於一個穩態，也推測是因為沒有 discriminator 回傳的 loss 來引導 generator 的學習。

   2) 從實驗結果會發現，M = 5, 10, 50, 100 的 accuracy 都在誤差範圍內，沒有顯著的差異，我推測是因為 generator 的能力不夠強，所以生成的 feature 並沒有太多的差異性，又 prototype feature 是所有 feature 平均的結果，導致不管增加幾倍的資料量，prototype feature 都不會有太多的變化性。

Reference:

https://arxiv.org/pdf/1801.05401.pdf

https://github.com/Abhipanda4/Feature-Generating-Networks

https://github.com/yinboc/prototypical-network-pytorch

https://arxiv.org/pdf/1703.05175.pdf

no collaborators