Homework #4

December 29, 2020

Problem 1: Prototypical Network

1. Describe the architecture & implementation details of your model.

首先先將每組 input image 透過 CNN feature extractor (Conv-4) 再將其 flatten 成的 1600 維；接著將其透過 MLP 輸出一組 400 維的 tensor 做平均為個別 prototype。利用 query feature 與每個 prototype 計算距離，在通過 Softmax 找出機率最大的選項。

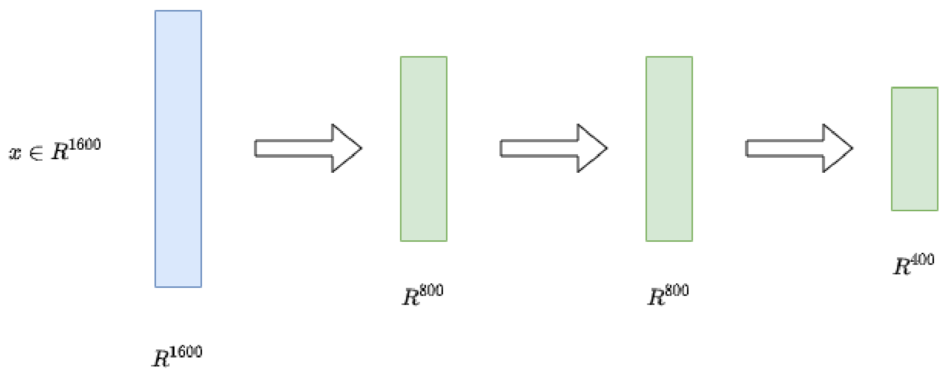

Figure 1: MLP architecture

```
MLP(
  (mlp): Sequential(
    (0): Linear(in_features=1600, out_features=800, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=800, out_features=800, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=800, out_features=400, bias=True)
  )
)
```

MLP architecture 如上圖。

best model：Training epoch 為 200，每個 epoch 有 600 個 episodes (10 way 1 shot with 75 query data for meta training, 5 way 1 shot with 75 query data for meta testing)；optimizer 使用 Adam(lr=1e-4, betas=(0.9, 0.999), weight_decay=1e-6)；另外有使用 lr schedule，每 40 個 epoch 會將 lr 乘 0.9；distance function 為 Euclidean distance。

Accuracy: **46.51 +- 0.94 %**

2. When meta-train and meta-test under the same 5-way 1-shot setting, please report and discuss the accuracy of the prototypical network using 3 different distance function (i.e., Euclidean distance, cosine similarity and parametric function).

Euclidean distance	Cosine similarity	Parametric function
46.38 +- 0.89 %	43.84 +- 0.85 %	45.08 +- 0.88 %

Table 1: The accuracy with 3 different distance function

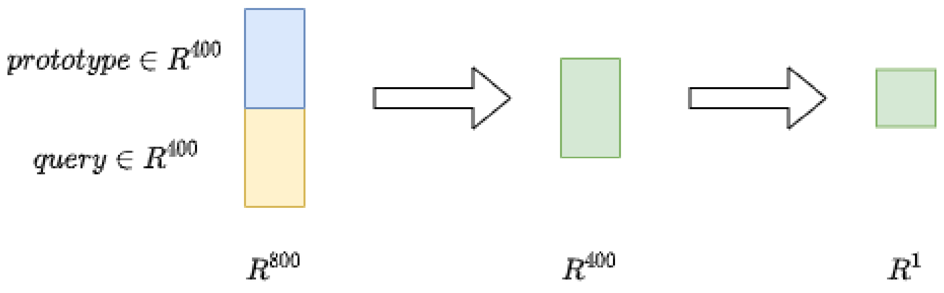

Figure 2: parametric function architecture

```
Sequential(
  (0): Linear(in_features=800, out_features=400, bias=True)
  (1): ReLU()
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=400, out_features=1, bias=True)
)
```

使用的參數都與 1-1 相同。

cosine similarity 可能是因為把距離限制在 -1 到 1，造成 Softmax 計算 $e^{distance}$ 差距不會那麼大；parametric function 則可能會因為 overfitting 使得在 val data 表現沒有很好。

3. When meta-train and meta-test under the same 5-way K-shot setting, please report and compare the accuracy with different shots. (K=1, 5, 10))

K=1	K=5	K=10
46.38 +- 0.89 %	62.39 +- 0.77 %	66.89 +- 0.65 %

Table 2: The accuracy with different shots

使用的參數都與 1-1 相同。

可以看出因為看的 image 變多了所以 accuracy 提高，而從 5-way-1-shot 到 5-way-5-shot

比起從 5-way-5-shot 到 5-way-10-shot，進步較多，可能是因為只看單一照片有機會選到離機率分佈中心點較遠的點，造成成果較差，只要多新增幾張平均或許可以比較趨近於分佈的中心點。

Problem 2: Data Hallucination for Few-shot Learning

1. Describe the architecture & implementation details of your model.
將 N 組 K 個 CNN Extractor 的 output 取 M 個 sample 和隨機的 noise 串接起來通過 Hallucinator，每組多產生 M 個 features，再將其與原本的 features 串接，通過 MLP 產生 K + M 個 400 維的向量，將其平均為 N 組 prototype。

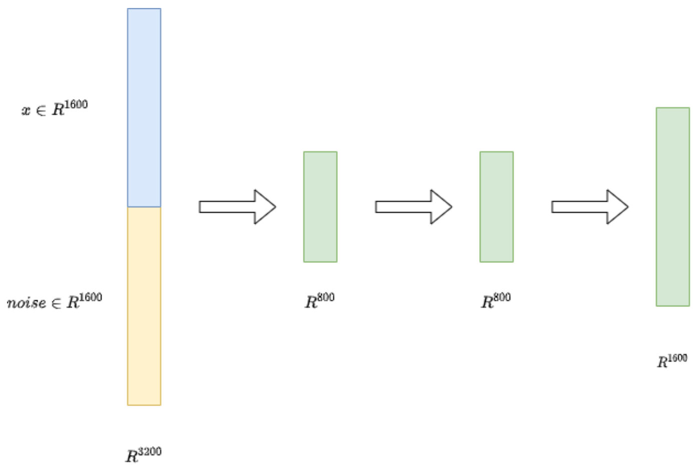

Figure 3: Hallucinator architecture

```
Hallucinator(
  (h): Sequential(
    (0): Linear(in_features=3200, out_features=800, bias=True)
    (1): BatchNorm1d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=800, out_features=800, bias=True)
    (4): BatchNorm1d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=800, out_features=1600, bias=True)
    (7): ReLU()
  )
)
```

Hallucinator architecture 如上圖。
best model：Training epoch 為 200，每個 epoch 有 600 個 episodes (5 way 1 shot 10 augmentation with 75 query data)；optimizer 使用 AdamW(lr=1e-4, betas=(0.9, 0.999), weight_decay=1e-2)；另外有使用 lr schedule，每 40 個 epoch 會將 lr 乘 0.9；distance function 為 Euclidean

distance；使用 ColorJitter(hue=0.05, saturation=0.05)、RandomHorizontalFlip()、Random-Rotation(10, resample=Image.BILINEAR) 做 data augmentation。

Accuracy: **50.36 +- 0.93 %**

2. To analyze the quality of your hallucinated data, please visualize the real and hallucinated (training) data in the latent space (the space where you calculate the prototypes of each class, e.g., the output of the MLP layer) by mapping the features to 2D space (with t-SNE).

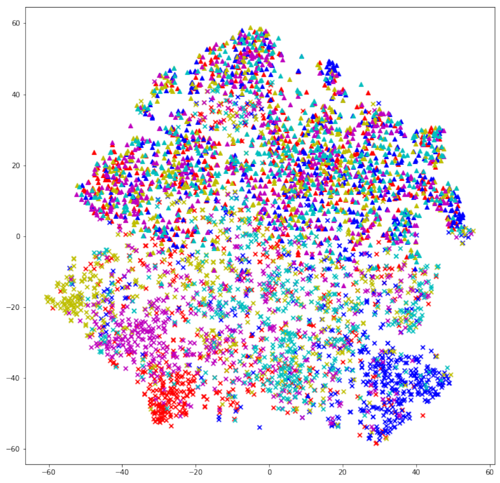

Figure 4: tsne

可以看到 real data (x) 在下方有成功分群，而 hallucinated data (▲) 則是散落在上方，分群效果並不明顯，另外 real 和 hallucinated 兩者間相互混合。

3. When meta-train and meta-test under the same 5-way 1-shot M-augmentation setting, please report and compare the accuracy with different number of hallucinated data. (M=10, 50, 100)

M=10	M=50	K=100
50.36 +- 0.93 %	49.69 +- 0.89 %	50.03 +- 0.90 %

Table 3: The accuracy with different number of hallucinated data

可以看到三者正確率沒差多少，但 M=10 最高，可能是因為 hallucinated data 的 variance

較高且四處散落所以 sample 越多點會造成 real data 的影響越少，因此成果較低。

4. Discuss what you’ve observed and learned from implementing the data hallucination model.

- Hallucinator initial weight 使用 identity 會比較快收斂，因為在剛開始會產生與真實資料相近的資料，最後結果可能會較低
- 使用 AdamW 可能比較不會走到 local minimum，成果有可能好一點
- 在 meta train M 調大，meta test M 調小表現會比較好
- data augmentation 可以結果可以上升 1-2%

Problem 3: Improved Data Hallucination Model

1. Describe the architecture & implementation details of your model.

多增加 Discriminator 以及 adversial loss，每一個 episodes 會先固定其他模型去訓練 discriminator，要把 real data 分類成 1，hallucinated data 分類成 0，接著再去訓練 discriminator 以外的模型，把原本的 few shot loss 加上 adversial loss，要讓 hallucinated data 可以騙過 discriminator。

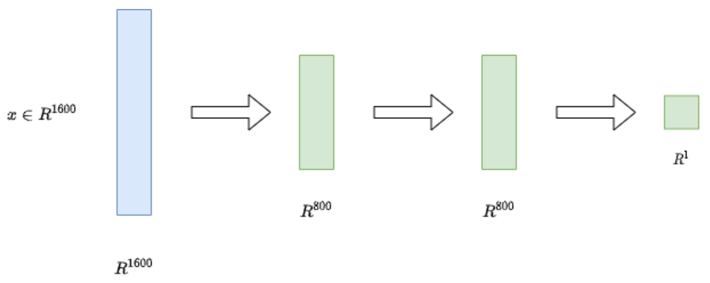

Figure 5: Discriminator architecture

```
Discriminator(
  (d): Sequential(
    (0): Linear(in_features=1600, out_features=800, bias=True)
    (1): BatchNorm1d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
    (3): Linear(in_features=800, out_features=800, bias=True)
    (4): BatchNorm1d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): LeakyReLU(negative_slope=0.2, inplace=True)
    (6): Linear(in_features=800, out_features=1, bias=True)
    (7): Sigmoid()
  )
)
```

Discriminator 架構如上圖。

best model：Training epoch 為 300，每個 epoch 有 600 個 episodes (5 way 1 shot 10 augmentation with 75 query data)；optimizer 使用 AdamW(lr=1e-4, betas=(0.9, 0.999), weight_decay=1e-2)；另外有使用 lr schedule，每 40 個 epoch 會將 lr 乘 0.9；distance function 為 Euclidean distance；使用 ColorJitter(hue=0.05, saturation=0.05)、RandomHorizontalFlip()、RandomRotation(10, resample=Image.BILINEAR) 做 data augmentation；few shot loss 和 adversial loss 比例為 1 : 0.15。

Accuracy: **52.01 +- 0.91 %**

2. To analyze the quality of your hallucinated data, please visualize the real and hallucinated (training) data in the latent space (the space where you calculate the prototypes of each class, e.g., the output of the MLP layer) by mapping the features to 2D space

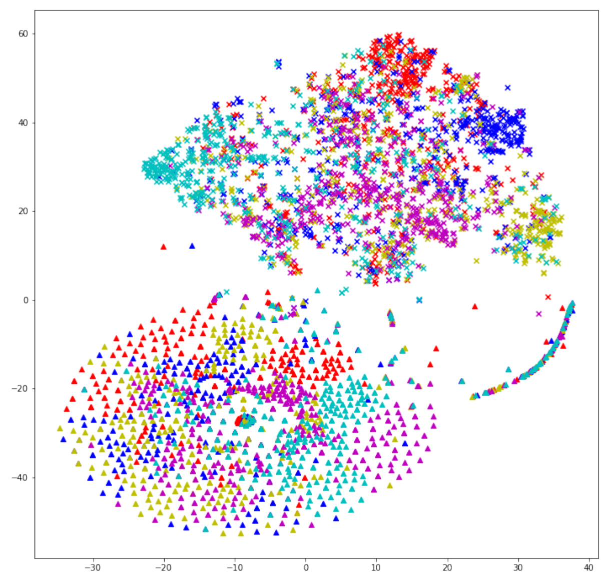

Figure 6: tsne

可以看到 real data (x) 在上方有成功分群，而 hallucinated data (▲) 則是在下方分群，real 和 hallucinated 兩者間因為 discriminator 較強的關係兩者分離，此外兩者可能有互補的功用，在 real data 上可能沒分好的點可以透過 hallucinated data 去做微調。

3. Why the improved model performs better than the one in Problem 2

加 adversial loss 可能可以讓 hallucinated data 所在範圍縮小，使得 variance 減少，而從 tsne 圖看，可以證實這點，adversial loss 會把 hallucinated data 集中，不會散落在各處，此外也可以看到，hallucinated data 分群效果比起原本的好，形成 real data 與 hallucinated data 互補的功用。

Reference

1. Prototypical Network (`https://github.com/oscarknagg/few-shot`)