Homework #4

資工所碩二 R08922122 林念澤

Problem1

1.

model architecture:

```
Convnet(
  (encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (2): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (3): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
  )
)
```

implementation detail:

Number of training episodes	200
Distance function	Euclidean distance
Learning rate schedule	0.001
Data augmentation	X
Optimizer	Adam
Meta-train phase	32-ways, 1-shot
Meta-test phase	5-ways, 1-shot

Accuracy: 49.12 +- 0.86 %

2.

[一] parametric function設計方法：

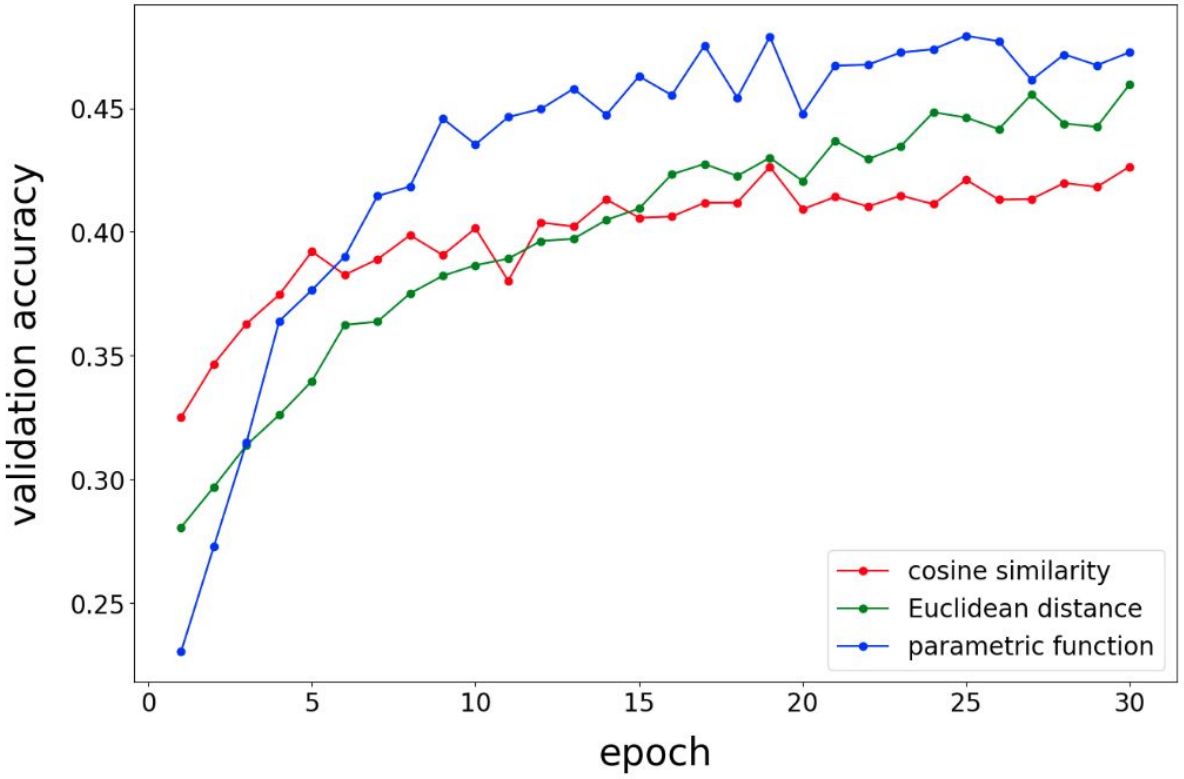
假設A, B∈R^{1600}, 則A,B距離之計算方式為

1. diff <- A - B　　# A, B向量element wise subtraction
2. distance <- Parametric_model(diff)　　# 兩向量之差經Parametric_model計算距離

Parametric_model架構如下：

```
(fc): Sequential(
  (0): Linear(in_features=1600, out_features=400, bias=True)
  (1): ReLU()
  (2): Linear(in_features=400, out_features=100, bias=True)
  (3): ReLU()
  (4): Linear(in_features=100, out_features=1, bias=True)
  (5): ReLU()
)
```

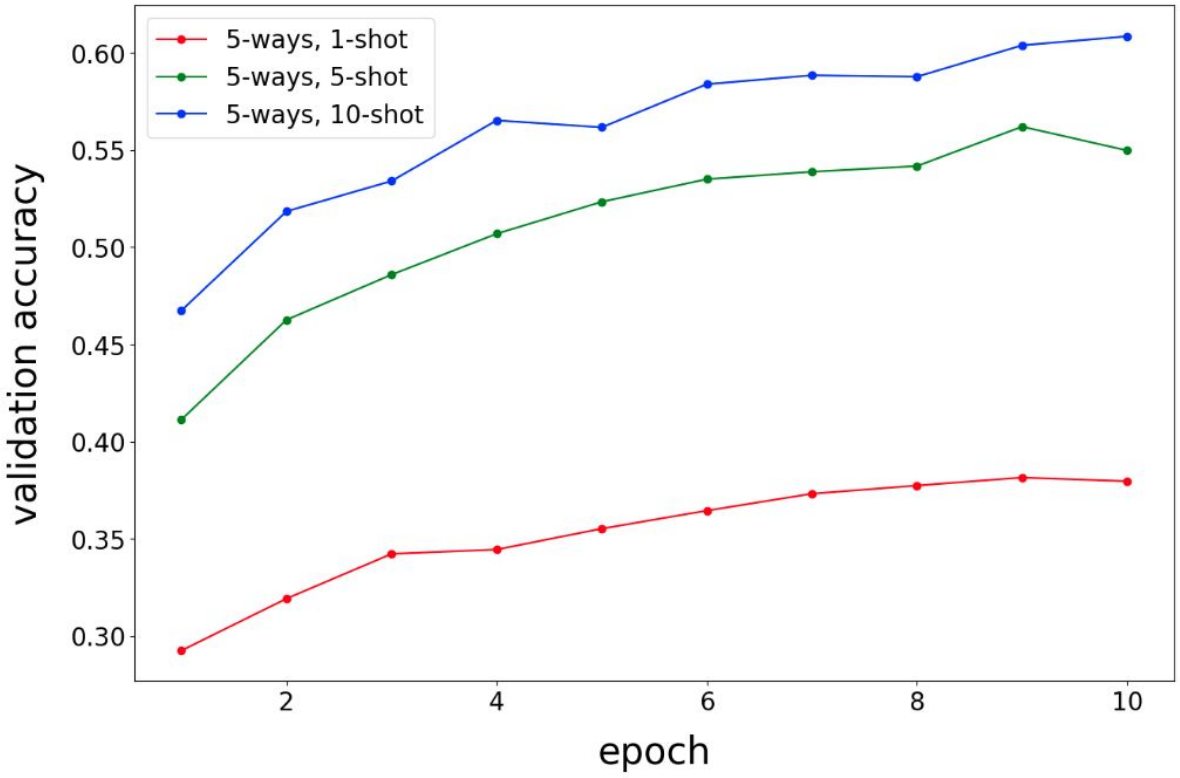
[二] 實驗結果(僅訓練30個epoch)



[三] discussion

從結果發現cosine similarity在一開始有最高的準確度，但後面的成長幅度不大，相較之下，Euclidean distance與parametric function在後期明顯比cosine similarity的準確度來的高

3.



[實驗結果]： 與預期結果相符合， 隨著shot的數量上升， 在同epoch下， shot數較大者具有較高的準確度

Problem2

1.

model architecture:

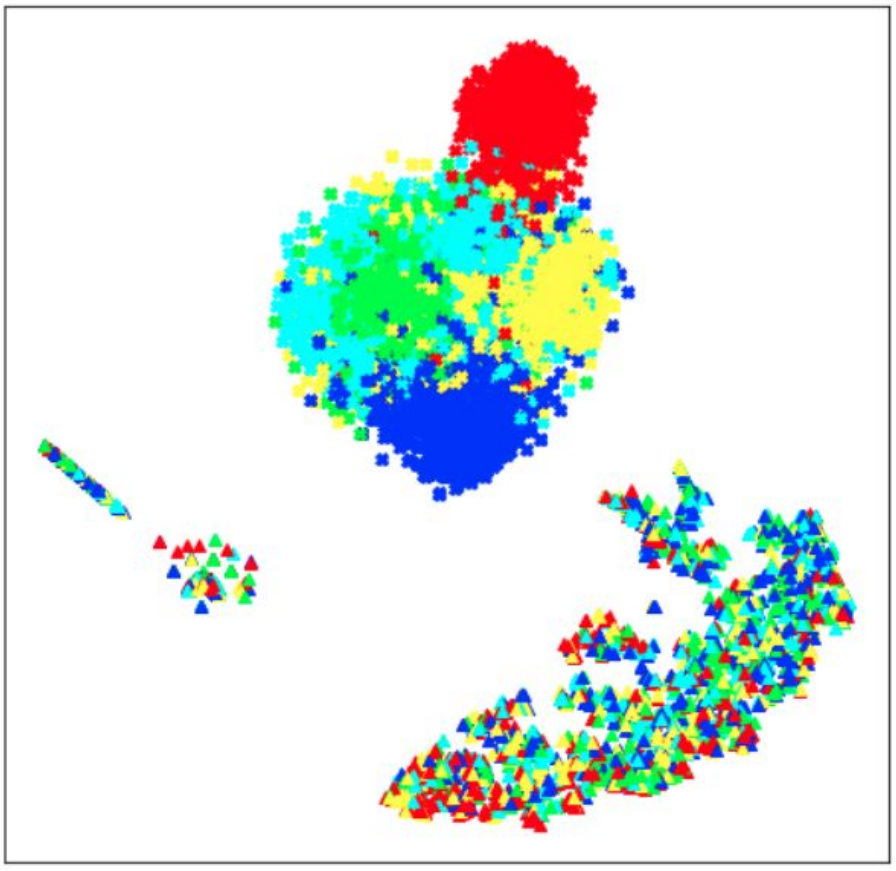
```
PrototypicalHallucination(
  (feature_extractor): Convnet(
    (encoder): Sequential(
      (0): Sequential(
        (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
        (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
      (1): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
        (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
      (2): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
        (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
      (3): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU()
        (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      )
    )
  )
  (hallucinator): Sequential(
    (0): Linear(in_features=1600, out_features=1600, bias=True)
    (1): BatchNorm1d(1600, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=1600, out_features=1600, bias=True)
    (4): BatchNorm1d(1600, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=1600, out_features=1600, bias=True)
    (7): BatchNorm1d(1600, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
  )
)
```

implementation detail:

Number of training episodes	200
Distance function	Euclidean distance
Learning rate schedule	0.001
Data augmentation	X
Optimizer	Adam
Meta-train phase	32-ways, 1-shot
Meta-test phase	5-ways, 1-shot
M	1

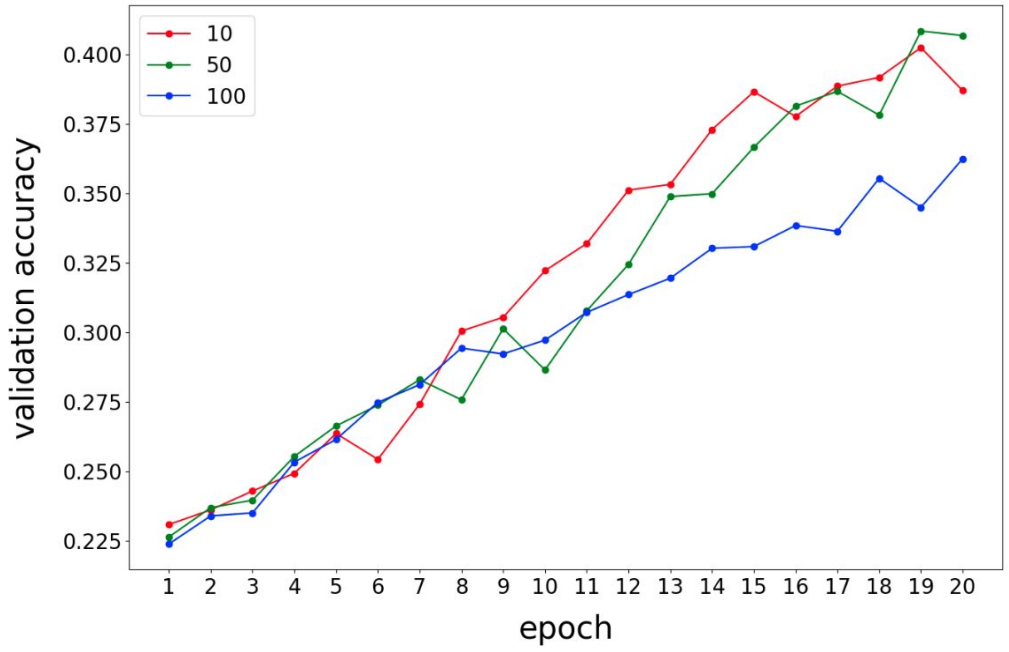
Accuracy: 0.4969 +- 0.89

2.



從結果來看model將real data處理的不錯，很明顯的分成5群，然而hallucinated data另外形成一群且毫無規律可言

3.



當hallucinated data數量太多時，可能因為部份hallucinated data品質不佳導致最終準確度惡化

4.
雖然從結果來看使用data hallucination確實些微的提昇了模型的準確度，但似乎有許多hallucinated data品質是不太好的，這點可由t-sne結果與隨著hallucianation的數量增加，準確度卻沒提昇甚至下降看出，挑選好的hallucainated data或使用更強的hallucinator可能是一個可行的解決方法

Problem3

1.

由於在Problem1-2發現paramatric function能夠加速模型收斂，Problem2-2發現許多hallucineted data品質不佳，因此Problem3我採用parametric function作為衡量距離的標準，並將real data與hallucinated data(長度為1600的向量)送入一層全連階層後決定其權重後作加權平均。模型架構在feature extractor及hallucinator部份不變，僅額外增加parametric function及weight evaluator。其架構如下：

parametric function:

```
(fc): Sequential(
  (0): Linear(in_features=1600, out_features=400, bias=True)
  (1): ReLU()
  (2): Linear(in_features=400, out_features=100, bias=True)
  (3): ReLU()
  (4): Linear(in_features=100, out_features=1, bias=True)
  (5): ReLU()
)
```

weight evaluator:

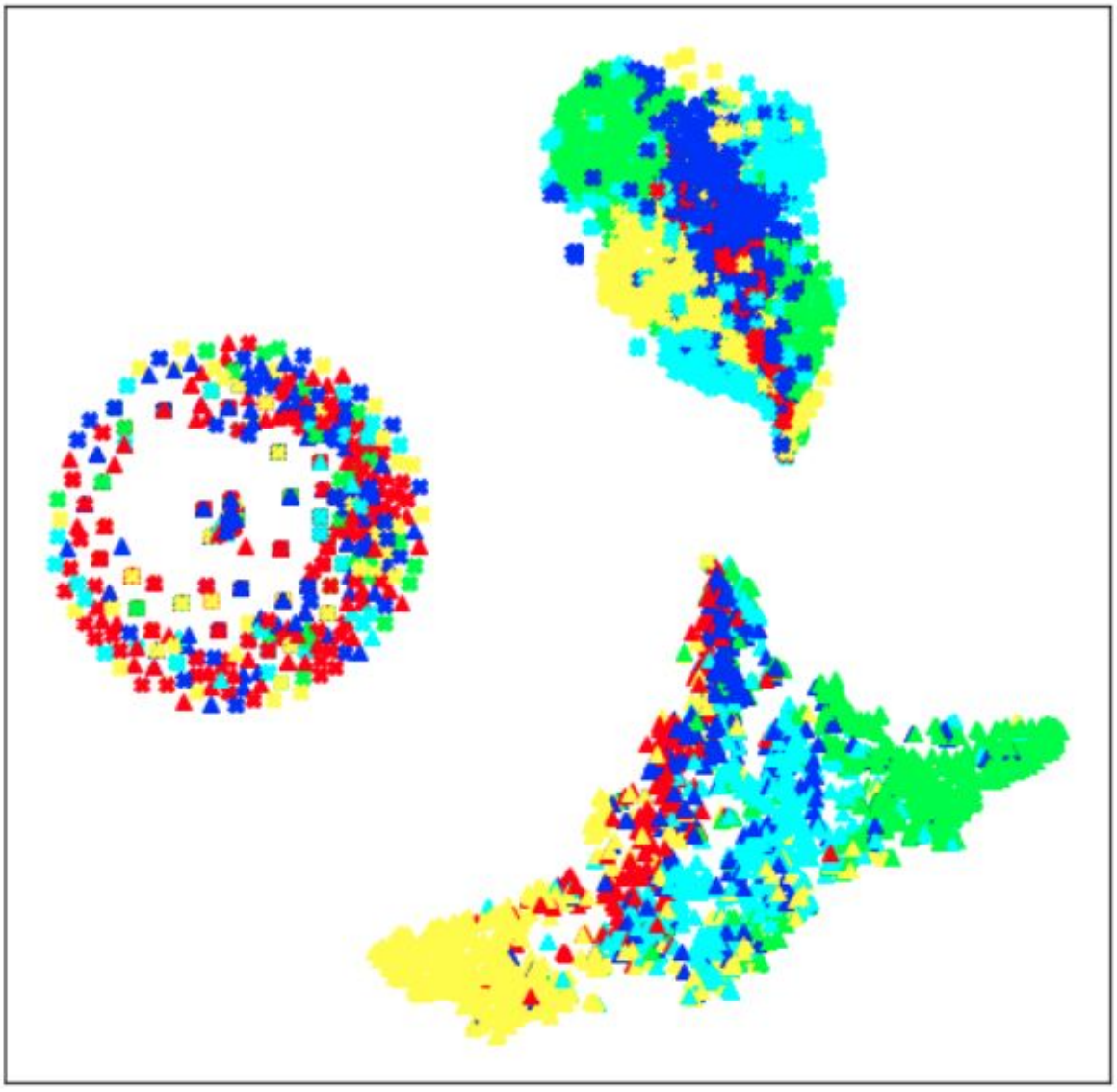
```
(0): Linear(in_features=1600, out_features=1, bias=True)
(1): ReLU()
```

implementation detail:

Number of training episodes	200
Distance function	Parametric function
Learning rate schedule	0.001
Data augmentation	X
Optimizer	Adam
Meta-train phase	32-ways, 1-shot
Meta-test phase	5-ways, 1-shot
M	1

Accuracy: 0.4988 +- 0.87

2.



從結果來看hallucinated data在投影後的空間中同類別的sample變得比較會聚在一起，然而有一群data(包含real及hallucinated)，獨立形成一群(圓圈部份)且各類別混成一團

3.

我認為hallucinated data品質的提昇是準確度能夠增加的主因，然而在訓練過程中也發現以parametric function作為衡量距離的方法很容易overfitting，也因此雖然從t-sne的結果來看效果好很多，但準確度上升幅度卻很有限。

Reference:

Promblem1: https://github.com/yinboc/prototypical-network-pytorch

Collaborators: NO