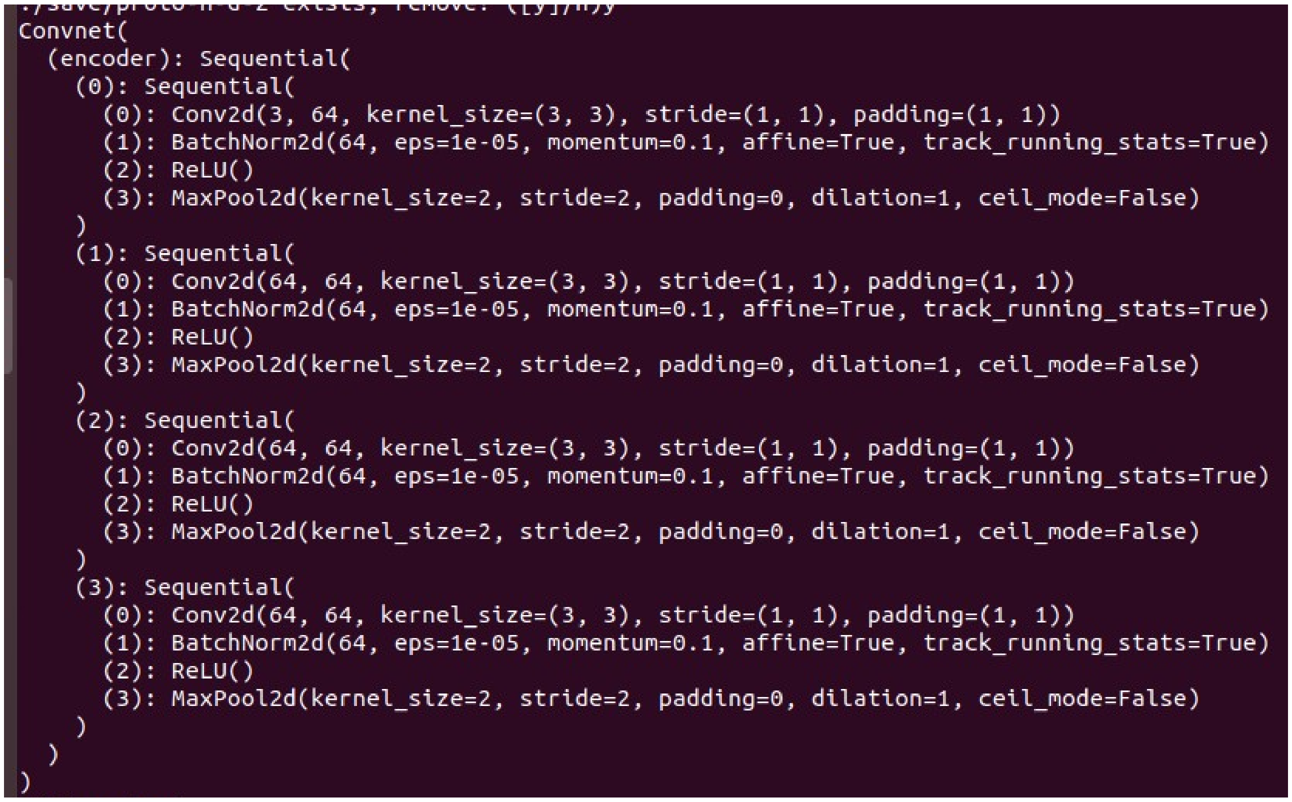
StudentID:B06502158 Name:陳柏元

Problem 1:

1. (11%) Describe the architecture & implementation details of your model. (Include but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot setting for meta-train and meta-test phase)
For Problem 1-2 and 1-3, you will do some experiments about different distance function and different K shot settings.



```
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)
    )
  )
)
```

我 train 了 200 個 epoch，使用了 euclidean distance 作為 prototypical network 的 distance function,learning rate schedule 則使用了 lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)，data argumentation 則只是做了 normalize(mean=[0.485, 0.456, 0.406,std=[0.229, 0.224, 0.225])，optimizer 則使用了 Adam with lr=0.001, 30-way 1-shot for training。
training 的過程則是將 support 跟 query 送進 model

The accuracy of the validation is **46.88** by eval.py。

2. (12%) 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). You should also describe how you design your parametric function.
分別把來自 query（450*1600）的 data，和來自 support（30*1600）的 data，作 unsqueeze(1)和 unsqueeze(1)，並 expand 兩者成（450*30*1600），再用這兩個 matrix 來做三種 prototypical network 的 distance function。
分別做了 10 個 epoch，結果如下。

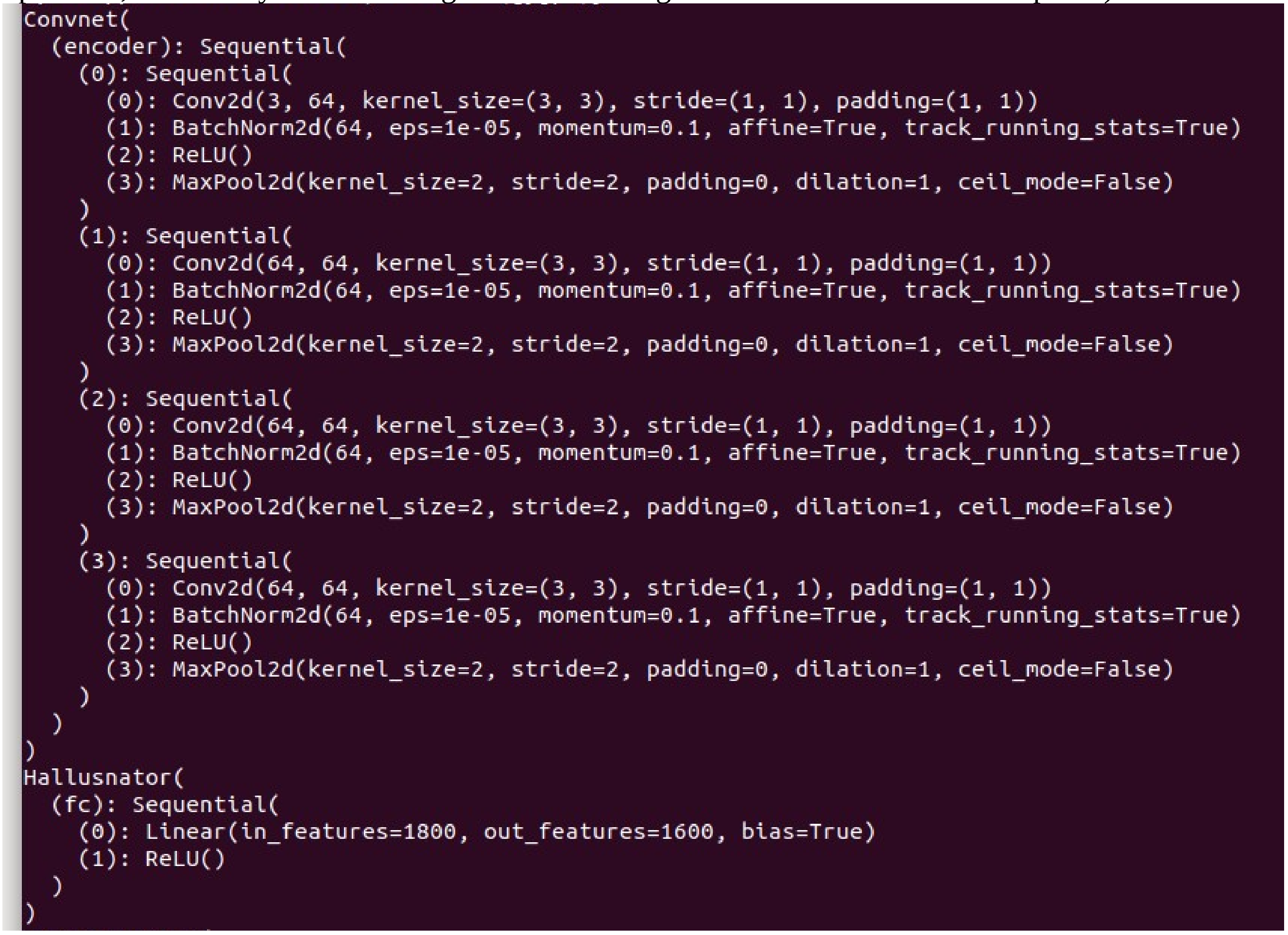
| | Euclidean distance | cosine similarity | absolute difference |
|---|---|---|---|
| val accuracy | 0.3546 | 0.3528 | 0.3402 |

3. (12%) 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)
分別做了 5 個 epoch，結果如下。

| K | 1 | 5 | 1 |
|---|---|---|---|
| val accuracy | 0.3205 | 0.5484 | 0.6158 |

可觀察到隨 K 值上升，accuracy 有明顯跟著上升。

Problem 2:
1. (10%) Describe the architecture & implementation details of your model. (Include but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot M-augmentation setting for meta-train and meta-test phase)

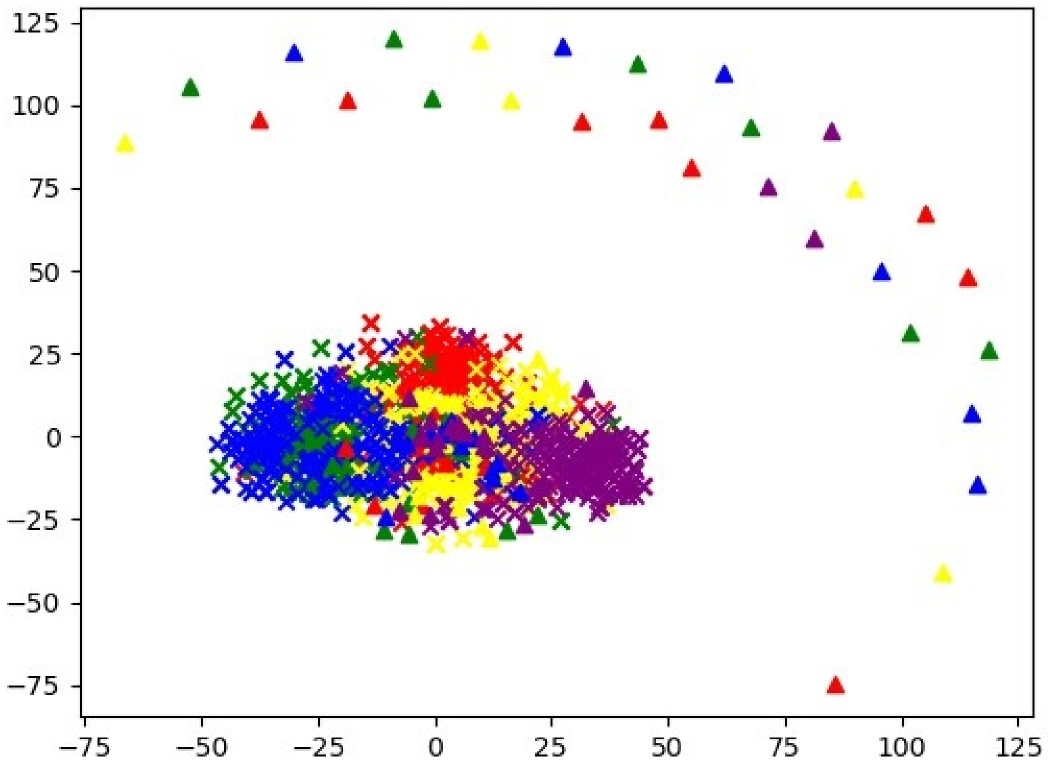


```
Convnet(
  (encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, 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)
    )
  )
)
Hallusnator(
  (fc): Sequential(
    (0): Linear(in_features=1800, out_features=1600, bias=True)
    (1): ReLU()
  )
)
```

我 train 了 200 個 epoch，使用了 euclidean distance 作為 prototypical network 的 distance function,learning rate schedule CNN 和 Hallucinator 則使用了 lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)，data argumentation 則做了 normalize(mean=[0.485, 0.456, 0.406,std=[0.229, 0.224, 0.225])，RandomHorizontalFlip()，RandomRotation(10)，optimizerCNN 和 Hallucinator 則使用了 Adam with lr=0.001， 30-way 1-shot for training。

The accuracy of the validation is **47.67** by eval.py。

2. (10%) 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). Briefly explain your result of t-SNE visualization. Example for fig2-3.png is shown below.



可以觀察到 hallucinated data 為隨機取值，可能使 real data 分群狀況變佳，但仍較為紊亂。

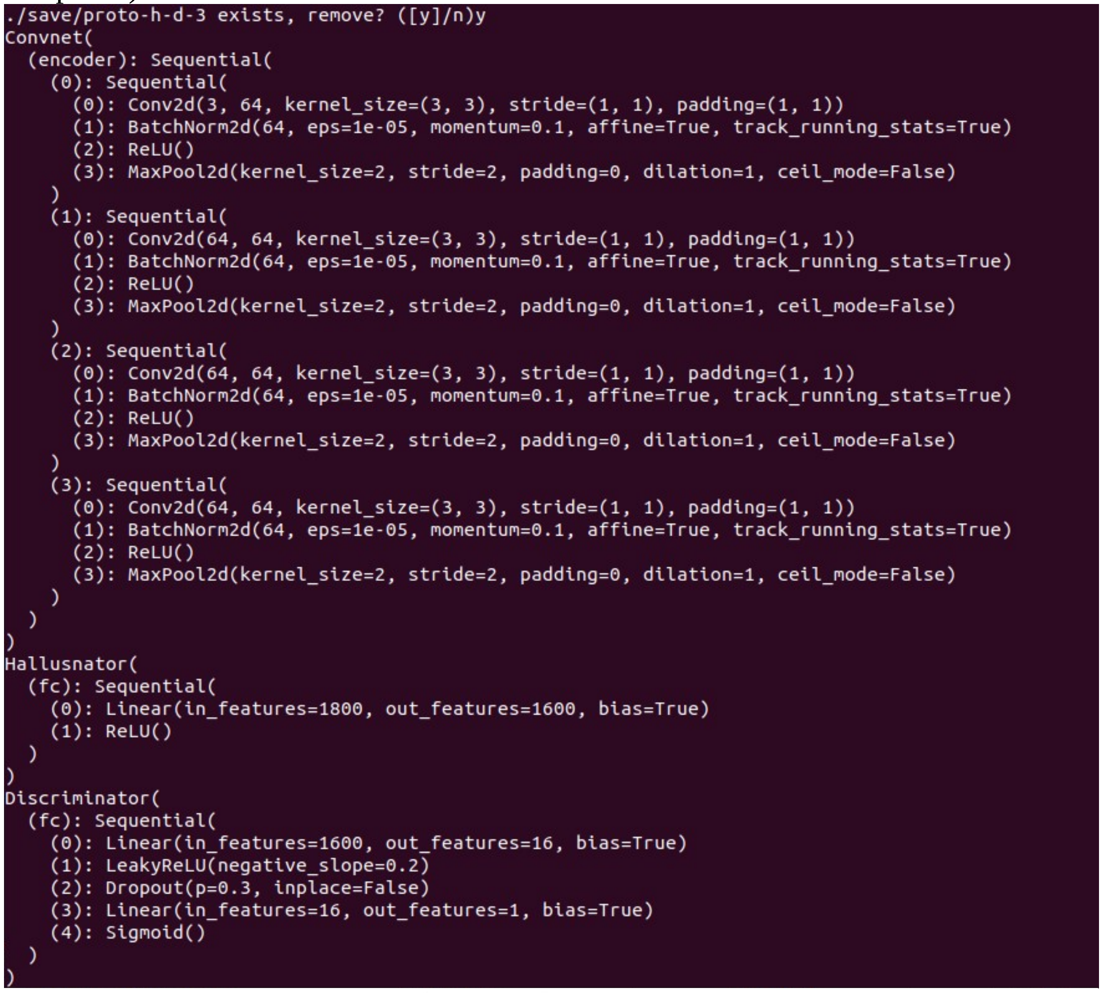
3. (10%) When meta-train and meta-test under the same 5-way 1-shot M-augmentation setting, please report and compare the accuracy with different number of hallucinated data. (M=10, 50, 100)
分別做了 5 個 epoch，結果如下。

| M | 10 | 50 | 100 |
|---|---|---|---|
| val accuracy | 0.2976 | 0.2941 | 0.3117 |

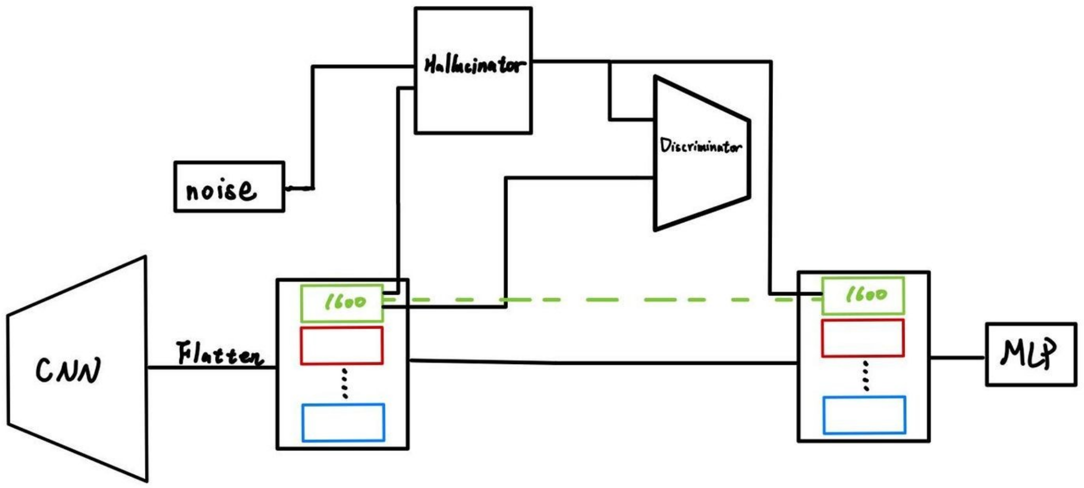
可觀察到隨 K 值上升，accuracy 有跟著上升，經測試自己使用了 200。

4. (5%) Discuss what you’ve observed and learned from implementing the data hallucination model.
從 tsne 結果可觀察到，可能因為加入了相聚甚遠的 halucinated data，讓原本相聚甚近的 data 反而分的比較開了，使得計算 prototypical distance 在區分不同 class 的時候變得比較容易一些了，也因此 data 有了些微的 accuracy 上升。

Problem 3:
1. (9%) Describe the architecture & implementation details of your model. (Include but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot M-augmentation setting for meta-train and meta-test phase)



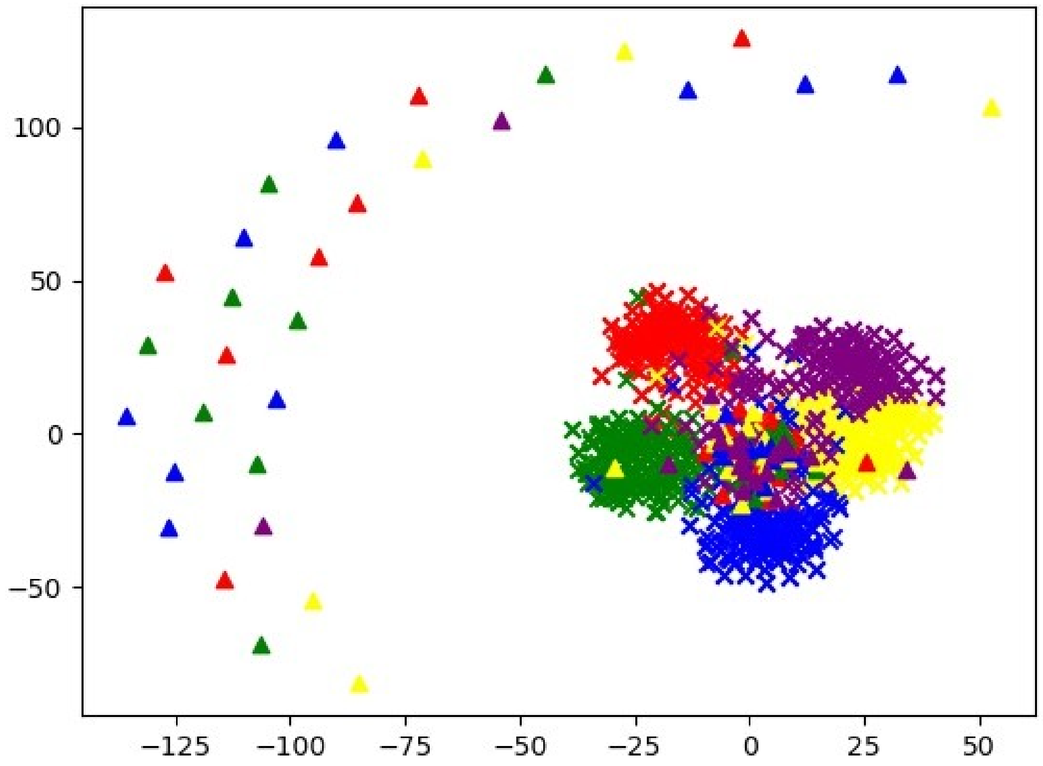
```
./save/proto-h-d-3 exists, remove? ([y]/n)y
Convnet(
  (encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, 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)
    )
  )
)
Hallusnator(
  (fc): Sequential(
    (0): Linear(in_features=1800, out_features=1600, bias=True)
    (1): ReLU()
  )
)
Discriminator(
  (fc): Sequential(
    (0): Linear(in_features=1600, out_features=16, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Dropout(p=0.3, inplace=False)
    (3): Linear(in_features=16, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```



我 train 了 200 個 epoch，使用了 euclidean distance 作為 prototypical network 的 distance function，learning rate schedule CNN，Hallucinator 和 Discriminator 都使用了 lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)，data argumentation 則只是做了 normalize(mean=[0.485, 0.456, 0.406,std=[0.229, 0.224, 0.225])，optimizer CNN，Hallucinator 和 Discriminator 都使用了 Adam with lr=0.001, 30-way 1-shot for training。

The accuracy of the validation is **48.18** by eval.py。

2. (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).



可以觀察到部份 hallucinated data 和 real data 仍存在些許的距離，且因 hallucinated data 仍為隨機取值，並沒有成群。但 real data 分群狀況變佳，可以觀察到明顯的五種類別分開來。

3.(4%) Try to explain why the improved model performs better than the one in Problem 2 (e.g., discuss the difference between your visualization of latent space in Problem 2 and Problem 3).
增加了 discrimimator 以後 accuracy 有明顯的上升，tsne 中的 real data 則分群分的比 problem 2 好上許多。可能因為不再是隨意的 sample  hallucinated data，而是帶有 discriminator loss 生成 hallucinated data，但因為 hallucinator 仍較少層，所以 tsne 中 hallucinated data 仍隨意分佈。但 real data 分群狀況變佳，可以觀察到明顯的五種類別分開來。