Homework #4 Report

Deep Learning for Computer Vision

資工碩一　張凱庭　R10922178

Problem 1

1. Model architecture and implementation details

5-way 1-shot	
Accuracy	48.61 ± 0.84 %

Meta-Train Setting	
Batch Size	100
Number of way	30
Number of shot	1
Number of query	15

Meta-Test Setting	
Number of way	5
Number of shot	1
Number of query	15

Hyperparameters	
Total Epochs	150
ConvergenceEpoch	127
Optimizer	ADAM
Learning rate	0.001

2. Accuracy of the prototypical network using 3 different distance function

For the comparison of the distance metric, these 3 models are all trained under the same setting except for the part that calculates the pairwise distance. The design of the learnable parametric function is shown below, it takes in the **subtraction of the output of the query set and the proto**. During 50 epochs, the learnable parametric function has the best performance.

Distance	Euclidean	Cosine	Parametric

Function	distance	similarity	function
Accuracy	0.4526	0.4324	0.4698

Parametric Function Architecture

```
ParametricDist(
  (dist): Sequential(
    (0): Linear(in_features=1600, out_features=64, bias=True)
    (1): ReLU()
    (2): Linear(in_features=64, out_features=32, bias=True)
    (3): ReLU()
    (4): Linear(in_features=32, out_features=1, bias=True)
    (5): ReLU()
  )
)
```

3. **Accuracy with different shots. (K=1, 5, 10)**

Shots	K=1	K=5	K=10
Accuracy	0.4856	0.7568	0.8037

For the comparison of the different shots, these 3 models are run under the same setting as problem 1.1, except that only run on 50 epochs. For the model evaluation except using the test case provided by TA, I use my own sample test case. As the result shows, when the shot increases the model's performance has huge improvement.

Problem 2

1. **Describe the implementation details of your SSL method for pre-training the ResNet50 backbone.**
   For the self-supervised learning, I used the BYOL implementation from this github repository: [https://github.com/lucidrains/byol-pytorch].

Hyperparameters	
Total Epochs	100
Optimizer	Adam
Learning rate	0.0003
Batch size	64
Data augmentation	default

BYOL Setting

```
net = BYOL(
    resnet,
    image_size=128,
    hidden_layer='avgpool',
    projection_size=256,
    projection_hidden_size=4096,
    moving_average_decay=0.99
)
```

After trained 100 epochs, the loss result in 0.2409.

2. **Following Problem 2-1, please conduct the Image classification on the Office-Home dataset as the downstream task for your SSL method. Also, please complete the following Table, which contains different image classification settings, and compare the results.**

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Accuracy on valid set
A	-	Train full model	0.2414
B	w/ label (TA's backbone)	Train full model	0.3571
C	w/o label (ssl backbone)	Train full model	0.4236
D	w/ label (TA's backbone)	Train classifier only	0.3399
E	w/o label (ssl backbone)	Train classifier only	0.3966

3. **Discuss or analyze the results in Problem 2-2**
   From the result we can tell that, during the fine-tuning phase training the full model has better performance than training the classifier layer only. It's probably because that when at the fine-tuning phase, the CNN kernel is trying to learn the different patterns of the new dataset.
   For the pre-training phase, training without the label gives better performance to the fine-tuning phase. It's probably because when labels are given the model would be biased on labels during learning to extract features. All we want is the model to learn to extract useful and good features, we don't need to confuse it with the label information, the classifier will be handled in downstream tasks.

Reference

1. Prototypical network: https://github.com/yinboc/prototypical-network-pytorch
2. BYOL: https://github.com/lucidrains/byol-pytorch

3. Prototypical network paper: https://arxiv.org/abs/1703.05175
4. BYOL paper: https://arxiv.org/abs/2006.07733