2021 DLCV
HW4
R09942171 電信所碩一 黃繼綸

Problem 1: Few-Shot Learning – Prototypical Network

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

1.

Model architecture : After feature extraction, I put in a MLP module, which is mapping the extracted features into 256 dimensions.

Training episodes : 100 episodes

Distance function : Euclidean distance

Learning rate scheduler : StepLR with step_size=40 and gamma=0.412

Data augmentation : No

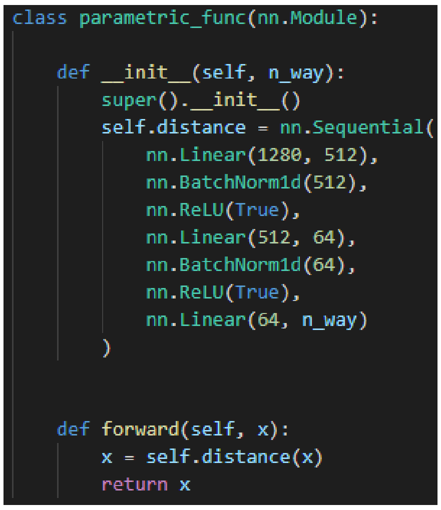
Optimizer : Adam

N-way K-shot setting: 30-way 1-shot for meta-train, 5-way 1-shot for meta-test

The accuracy in the validation set is *46.97* +- *0.88%*

2.

Design of the parametric function: First, I feed the support images and query images into the feature extractor to obtain the extracted features. The setting is 5way-1shot. Therefore, I calculate the Euclidean distance between the feature of 5 support images and the feature of query images. Finally, such features proceed through the Multi Perceptron Layers (MLP) to do classification. Because the dimensions of the extracted features are 256, we have a total of 256 * 5 dimensions in the input channels of the MLP. The MLP layers are demonstrated below:



```
class parametric_func(nn.Module):

    def __init__(self, n_way):
        super().__init__()
        self.distance = nn.Sequential(
            nn.Linear(1280, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(True),
            nn.Linear(512, 64),
            nn.BatchNorm1d(64),
            nn.ReLU(True),
            nn.Linear(64, n_way)
        )


    def forward(self, x):
        x = self.distance(x)
        return x
```

In this experiment, we can see that the performances of the non-parametric function and the parametric function are similar. But, the parametric function performs a little better than the non-parametric function. The reason is that the parametric function can learn how to compute the relationship between the support set features and the query set features.

5way-1shot	Euclidean	Cosine_similarity	Parametric func.
Accuracy	43.57%	43.97%	43.98%

3.

In this experiment, we can see that if the more images of each category, the higher the classification accuracy. This phenomenon is reasonable since the representative's information can be more representative.

Euclidean metric	5way-1shot	5way-5shot	5way-10shot
Accuracy	43.11%	61.44%	65.58%

Problem 2: Self-supervised learning

Ref : https://github.com/lucidrains/byol-pytorch

1.

SSL method: Bootstrap Your Own Latent (BYOL)

Data augmentation : ColorJitter(0.8,0.8,0.8,0.2), RandomGrayscale, RandomHorizontalFlip, GaussianBlur, RandomResizeCrop

Learning rate schedule : No

Optimizer: Adam with lr=3e-4

Batch_size: 64

N_epochs: 100

2.

Settings	Accuracy(%)
A	45.32
B	45.07
C	**51.48**
D	46.55
E	45.57

3. In Problem 2-2, we conduct the extensive experiments about supervised pretraining and self-supervised pretraining. First, we can see that the model trained from scratch is better than the model trained on miniImage using supervised learning. But if we fixed the backbone of the model, the performance

will increase a little. This phenomenon only appeared in the pretrained using supervised learning. Second, the model pretrained using self-supervised learning and didn’t fix the backbone has the best performance in this experiment. I think the reason this happens is that the domain of the miniImagenet is not very similar to the domain of the office dataset. If we apply the supervised learning on miniImagenet, the model will overfit the domain of miniImagenet. However, if we apply self-supervised learning on miniImagenet, the model just learns the relative relation on the input data. After that, the model can be more powerful to generalize to the domain of the office dataset.