Deep Learning for Computer Vision - Homework 4

資工碩一 R10922005 李澤諺

January 3, 2022

Problem 1: Prototypical Network

以下爲我於本次作業中所實作的 prototypical network：

```python
class PrototypicalNetwork(nn.Module):
    def __init__(self):
        super(PrototypicalNetwork , self).__init__()

        self.convolution = nn.Sequential(
            nn.Conv2d(3 , 64 , kernel_size = 3 , stride = 1 , padding = 1) ,
            nn.BatchNorm2d(64) ,
            nn.ReLU() ,
            nn.MaxPool2d(2) ,
            nn.Conv2d(64 , 64 , kernel_size = 3 , stride = 1 , padding = 1) ,
            nn.BatchNorm2d(64) ,
            nn.ReLU() ,
            nn.MaxPool2d(2) ,
            nn.Conv2d(64 , 64 , kernel_size = 3 , stride = 1 , padding = 1) ,
            nn.BatchNorm2d(64) ,
            nn.ReLU() ,
            nn.MaxPool2d(2) ,
            nn.Conv2d(64 , 64 , kernel_size = 3 , stride = 1 , padding = 1) ,
            nn.BatchNorm2d(64) ,
            nn.ReLU() ,
            nn.MaxPool2d(2)
        )

    def forward(self , x):
        x = self.convolution(x)
        x = x.view(x.size(dim = 0) , -1)
        return x
```

在 data preprocessing 方面，我將 training dataset 中所有的圖片皆縮放至 84×84 的大小，並將其 pixel 的值 scale 到 0 和 1 之間，以此進行 data preprocessing，此外，我也使用了 torchvision 的 RandomHorizontalFlip、RandomAffine、ColorJitter 進行 data augmentation。接著，我使用了 Adam 作爲 optimizer，其中 learning rate 爲 0.001，並使用了 StepLR 作爲 learning rate scheduler，其中 gamma 爲 0.5，step_size 爲 20，以及使用 cross entropy 作爲 loss function，使

用 Euclidean distance 作爲 distance function，訓練 100 個 epoch，以此訓練 prototypical network。在 meta-train 時，我使用了 15-way 1-shot，並且每個 class 皆選出了 15 筆的 query data，而在 meta-test 時，我則使用了 5-way 1-shot，並且每個 class 皆選出了 75 筆的 query data，而不論是 meta-train 或 meta-test，episode 數量皆爲 600 個。最後，我所訓練出來的 prototypical network 在 validation dataset 上使用 5-way 1-shot 所得到的 accuracy 爲 $42.66 \pm 0.80\%$。

接著，以下爲我分別使用 Euclidean distance、cosine similarity、parametric function 作爲 distance function 時，在 validation dataset 上所得到的 accuracy：

Distance Function	Accuracy
Euclidean Distance	$42.66 \pm 0.80\%$
Cosine similarity	$29.85 \pm 0.45\%$
Parametric Function	$42.51 \pm 0.68\%$

其中，我所實作的 parametric function 如下：

```
class ParametricDistance(nn.Module):
    def __init__(self):
        super(ParametricDistance , self).__init__()

        self.distance = nn.Sequential(
            nn.Linear(3200 , 256 , bias = True) ,
            nn.ReLU() ,
            nn.Linear(256 , 1 , bias = True)
        )

    def forward(self , x , y):
        N , D = x.shape
        M , D = y.shape
        x = x.unsqueeze(dim = 1).expand(N , M , D)
        y = y.unsqueeze(dim = 0).expand(N , M , D)
        return self.distance(torch.cat((x , y) , dim = 2)).squeeze()
```

由上表可以看出，cosine similarity 的 performance 最差，推測其可能是因爲僅靠 feature vector 之間的夾角沒有辦法完全反映出兩者的相關性所致，而 Euclidean distance 和 parametric function 的 performance 相近，由於 parametric function 的 weight 特別設計過的話，可以特化爲 Euclidean distance，意即 parametric function 比起 Euclidean distance 更爲 general，complexity 更高，但 parametric function 的 performance 卻比 Euclidean distance 略差一些，推測其可能是因爲 overfitting 所致，也有可能是 hyper-parameter 沒有調整好的緣故。

接著，以下爲使用 5-way 1-shot、5-way 5-shot、5-way 10-shot 所訓練出來的 prototypical network 在 validation dataset 上所得到的 accuracy：

Setting	Accuracy
5-way 1-shot	$42.66 \pm 0.80\%$
5-way 5-shot	$59.48 \pm 0.87\%$
5-way 10-shot	$65.43 \pm 0.85\%$

由上表可以看出，當 support set 越大時，所得到的 performance 越好，推測其可能是因爲當 support set 有越多的 data 時，用越多的 feature vector 取平均之後才能越接近眞正的 prototypical vector，進而得到越正確的 classification 所致。

Problem 2: Visualization in Image Captioning

在 data preprocessing 方面，我將 dataset 中的圖片皆縮放至 128×128 的大小，並將其 pixel 的值 scale 到 0 和 1 之後，再使用助教所提供的 mean 和 standard deviation 進行 normalization，以此進行 data preprocessing，此外，在 pretraining phase 和 training phase 時，我也使用了 torchvision 的 RandomHorizontalFlip、RandomAffine、ColorJitter 進行 data augmentation。接著，在 pretraining phase 中，我使用了 BYOL 進行 self-supervised learning，其中，我使用了 Adam 作爲 optimizer，learning rate 爲 0.0003，batch size 爲 16，訓練了 100 個 epoch，以此得到了 pretrained weight，而在 training phase 中，我使用了 Adam 作爲 optimizer，其中 learning rate 爲 0.0005，並使用了 ExponentialLR 作爲 learning rate scheduler，其中 gamma 爲 0.9，以及使用 cross entropy 作爲 loss function，訓練 50 個 epoch，以此得到最後的 model。接著，以下爲我於 5 個 setting 之下所得到的 validation accuracy：

Setting	Accuracy
A	0.49015
B	0.54187
C	0.52463
D	0.20197
E	0.30542

由此可以看出，setting D 和 setting E，也就是固定 backbone 而只訓練 classifier 的 performance 最差，我認爲其原因爲 backbone 是 pretrain 在 Mini-ImageNet dataset 上，但 downstream task 則是使用 Office-Home dataset，若不進行 finetune 的話，適用於 Mini-ImageNet dataset 的 backbone 將難以直接適用於 Office-Home dataset 之上，而 setting B 和 setting C 的 performance 又比 setting A 的好了一些，意即有進行 pretraining 的 performance 較好，我認爲是因爲雖然 Mini-ImageNet dataset 和 Office-Home dataset 之間存在 bias，但兩個 dataset 之中的圖片在 feature 上仍然存有一定的相似度，故比起完全從頭訓練，使用在 Mini-ImageNet dataset 上得到的 pretrained weight 來進行 finetune，更能有效地得到較好的 performance，而 setting B 的 performance 又比 setting C 的好了一些，我認爲是因爲使用 label 來進行 pretraining 的話，所使用的 information 更多，自然能訓練出更好的 backbone，因而在 finetune 時達到更好的 performance。