DLCV HW4

b07901169 楊宗桓

1 Prototypical Network

1.1 Model architecture & Implementation

1.1.1 Model

Model架構如圖一: Conv-4是依照給定的Conv-4的架構；Parametric Function由MLP實現。

```
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)
    )
  )
)
MLP(
  (fc): 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()
  )
)
```

Figure 1: model architecture

1.1.2 Implementation Detail

Hyperparameter	value
learning rate	0.0001
beta1	0.9
beta2	0.999

(a) training parameters

Meta-train	
n_way	30
n_shot	1
n_query	15
total episodes	8000

(b) meta training parameters

Meta-test	
n_way	5
n_shot	1
n_query	15

(c) meta testing parameters

選用的parameter如table 1，為了加强的model的能力，加大meta train的n_way到30並限制n_shot為1，同時另外做data augmentation:包括RandomResizedCrop(0.75,1.0)、ColorJitter(0.4,0.4,0.4,0.2)、RandomHorizontalFlip(0.5)，optimizer選用Adam(lr=1e-4,beta=(0.9,0.999))，distance metric最後選用parametric(MLP)

1.1.3 best model

最好的model在validation testcase上的accuracy: ***47.62 +- 0.85%***

1.2 Different distance function

Distance Metric	Acc
parametric	***47.62 +- 0.85%***
cosine	39.54 +- 0.71%
euclidian	43.98 +- 0.81

Table 2: Accuracy on validation testcase with different distance metric

parametric function實作的方式如figure.1的mlp所示的三層linear+ReLU，我實作時發現太淺層的linear表現不太好同時太多層會在validation set上overfit，所以最後的Parametric Function採用三層的linear。另外mlp的input為prototype x - prototype y，而prototype從前面的convnet對image抽取而來。結果可以發現parametric的表現最好，euclidian為其次，parametric表現最好可能是因為他可以搭配convnet學到prototype不同維度的feature差距，因而得到更多資訊；我推測cosine之所以表現較差可能是因為他算出來的similarity限縮於0到1之間因此壓縮了資訊。

1.3 5-way k-shot

k-shot	Acc
1	44.97%
5	56.83%
10	***64.56%*** %

Table 3: Accuracy on validation set with different k-shot training

Figure 2: Training curve on different k-shot

這裡的setting都採用parametric function並在meta train跟meta test都做相同的5-way k-shot最後在validation set上得到的accuracy，由table.3可以發現在k=10時表現最好，我認為可能是:當做one-shot training時Model看到的prototype僅來自一張圖片平均(也就是其本身)，所以對於圖片會有比較大的bias也導致model表現不好，figure.2是不同k-shot的training curve，可以發現5-shot其實跟10-shot差異不大，不過因為作業要求的meta test是5-way 1-shot最終採用30-way 1-shot的meta train。

2 SSL pretraining for Image classification

2.1 Implementation

Hyperparameter	value
learning rate	0.0001
beta1	0.9
beta2	0.999
batch_size	64
epochs	100

(a) training Hyperparameters

BYOL parameter	value
projection_size	256
projection_hidden_size	4096
moving_average_decay	0.99

(b) BYOL parameters

我使用BYOL這個REPO的套件實現SSL[1]，我採用他REPO下提及的Optimal hyperparameter如table 4.(b)，training時使用的hyperparameter如table 4.(a)，optimizer選用Adam沒有另外做data augementation因為這個套件下就有另外包括Data augmentation，我只在downstream的classification才加做data augmentation包括:RandomHorizontalFlip、ColorJitter(0.4,0.4,0.4)

2.2 different pretrain & finetune setting

Setting	Pretraining(MiniImageNet)	Fine-tuning(Office-Home)	Classification accuracy(Office-Home validation)
A	-	Train full model	33.67%
B	w/ label	Train full model	43.35%
C	w/o label	Train full model	***44.33%***
D	w/ label	Fix Backbone	36.21%
E	w/o label	Fix Backbone	40.10%

Table 5: Accuracy on validation set with different k-shot training

table.5的setting皆為epochs=100(若已達到收斂會提早結束training)且皆用相同的data augmentation(見2.1),可以發現setting C的accuracy最高，代表model在pretrain時學到的資訊能夠有效transfer到downstream task由此可以看到SSL的强大:即使fine-tune在不同dataset仍能有高accuracy，甚至比pretrain在相同的dataset(setting B)表現還要好，也有可能是因為setting B/D雖然是pretrain在相同的dataset上但仍會因為pretrain的data選擇而overfit，另外可以發現在supervised learning的pretrain下且finetune時fix backbone會導致model表現明顯變差(setting B v.s setting D，-7.14%)；而用SSL做pretrain且finetune時fix backbone(setting C v.s setting E)得到的accuracy並沒有下降太多(-3.25%)可能是因為SSL pretrain出來的convnet能夠抽取有效的跨dataset的feature，所以在finetune時model會著重在classifier的學習上因此有沒有fix backbone對model影響不大。figure.3是不同setting的training curve，可以發現setting E的training loss降的最慢，而有經過supervised pretraining的B跟D最快達到收斂。

Figure 3: Training curve on different setting

3 Reference

[1] https://github.com/lucidrains/byol-pytorch