DLCV hw4 report

B07902054 資工四 林子權

Problem 1

Architecture and implementation detail

模型架構上，我使用助教提供的Conv-4來當作feature extractor。丟進去的images沒有經過resize，大小為84x84，丟入feature extractor會得到一根1600維的feature vector。

在meta-train的階段，我先去extract support data的feature，然後計算每個classes對應的prototype，接著去extract query data的feature，並且去計算query features跟prototype之間的distance(這邊我使用parametric function來測量distance，詳細的說明在下面)，得到一個大小為(N, N_way)的matrix(N為query data的數量)。我把這個matrix取負號，當作每個query data對N_way個classes的prediction score，然後去算cross entropy loss來做optimization。

訓練的參數以及方式細節如下：

- number of epoch: 100
- number of training episodes: 100
- distance function: parametric function(跟下面那個section所描述的方式一樣)
- learning rate: 1e-3
- learning rate schedule: torch.optim.lr_scheduler.StepLR(step_size=20, gamma=0.5)
- data augmentation: None
- optimizer: Adam
- 5-way 1-shot setting

由助教提供的eval.py算出的模型正確率：
Accuracy: 55.98 ± 0.91 %

Compare between different distance function

我使用了下面三種方式來當作我的distance function，並用了跟上一個section完全一樣的訓練參數。

下面的accuracy，是用助教給的val_testcase.csv來做測試得到的：

- Euclidean distance: 46.35 ± 0.90 %
- Cosine similarity: 44.68 ± 0.90 %
- Parametric function: 55.98 ± 0.91 %

Parametric function的設計如下：假設我有兩個向量x和y，則x跟y的距離為$\frac{x^T M y}{max(|x|_2 \cdot |y|_2, \epsilon)}$，$\epsilon$為一個接近0的很小的值，用來避免分母為0。若$x$跟$y$為$n$維的vector，則$M$為$n$ x n的matrix，為可以透過訓練更新的參數。M的初始值為identity matrix，即訓練一開始的distance function其實為cosine similarity。

可以看到Parametric function的正確率高過另外兩個非常多，這個結果是可以猜得到的，因為人工設計的distance function不一定能很好地描述兩個點在feature space裡頭的距離。讓Parametric function的初始function為cosine similarity是一個不錯的設計，這樣可以大概率保證訓練出來的distance function會至少比cosine similarity還要好。

Compare between different shots

我測試了當K=1, 5, 10時，5-way K-shot setting的accuracy。所有實驗都使用上一個section提到的parametric function來當作我的distance function。

Meta-train on 5-way K-shot, meta-test on 5-way K-shot的accuracy如下。測試的資料是我自己在validation set當中sample出來的，共600個episodes，每個episode共有5*15個query data：

- K=1: 55.98 ± 0.91 %
- K=5: 84.47 ± 0.51 %
- K=10: 87.71 ± 0.42 %

可以觀察到，基本上K值的大小和正確率呈現正相關，尤其是從K=1到K=5，進步的幅度相當的大，從K=5到K=10則是只有些微的進步。由此可以猜測調整K值這個improvement的方法，正確率應該會有一個upper bound。

Problem 2

Implementation details of your SSL method

我使用BYOL ssl method來pre-training我的ResNet50 backbone。 我沒有使用任何data augmentation還有learning rate schedule。 Optimizer使用的是Adam，batch size為128，訓練了600個epoch約24小時。input image則是如助教規定的resize成128x128。

Experiment on different fine-tuned setting

下面的五個實驗我使用了完全一樣的training hyperparameters以及classifier的架構：

Training hyperparameters:

- image size = (128 x 128)
- batch size = 128
- learning rate = 1e-4

- number of epoch = 40
- classifier hidden dimension = 4096

Classifier model structure:

```
import torch.nn as nn

classifier = nn.Sequential(
    nn.Linear(2048, 4096),
    nn.BatchNorm1d(4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, 65)
)
```

Experiment result:

Setting	Pre-training	Fine-tuning	Acc
A	none	Train full model	0.2315270935960591
B	w/ label	Train full model	0.3645320197044335
C	w/o label	Train full model	0.5320197044334976
D	w/ label	Fix the backbone	0.3448275862068966
E	w/o label	Fix the backbone	0.5369458128078818

Experiment Result Analysis

- 首先很顯然地，setting A的表現是最差的，因為它沒有使用任何pre-training的model，可見pre-training對於model performance是很有幫助的。
- 再來可以發現，使用self-supervised pre-training model的setting，表現比使用supervised pre-training model的setting還要好非常多。這點我認為可能的原因有下面兩種：
  - supervised比起self-supervised的pre-training方式，感覺更容易overfit在training dataset上，或許學出來的representation沒有辦法很好的transfer到其他domain
  - 也有可能是supervised model訓練得不夠久，學出來的representation不夠好。這點因為supervised pre-training model是助教提供的，無法確定
- supervised pre-training model搭配上train full model的訓練方式，結果會比fix住backbone還要好約2個百分點，我覺得還蠻合理的，因為原本的backbone是pre-training在不同的dataset上，要做一點fine-tuned結果才會比較好
- self-supervised pre-training model搭配上train full model的訓練方式，結果會比fix住backbone還要差約0.5個百分點，那代表在pre-training階段，這個backbone就已經收斂到了一個非常好的位置了，在僅僅只訓練40個epochs的情況下，train full model的訓練方式還沒有辦法比較好，可能要再多訓練久一點才有機會超越。