Ting-An Chen

Homework #4

Deep Learning for Computer Vision

Problem 1: Prototypical Network (70%)

1. (20%) Describe the architecture & implementation details of your model. (Include but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot setting for meta-train and meta-test phase) Please report the accuracy on the validation set under 5-way 1-shot setting (during inference).

   Accuracy: 45.33% +- 0.84%

   Architecture: 4 convolutional layers

```
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)
    )
  )
)
```

Implementation details

- **The number of training episodes**
  - Meta-train phase: 100
  - Meta-test phase: 400
- **The number of epochs:** 200
- **Distance function:** Euclidean distance
- **Learning rate schedule:** 0.001 (decay step: 20)
- **Data augmentation:** No
- **Optimizer:** Adam
- **N-way K-shot setting**
  - Meta-train phase: 30-way 1-shot (15 queries)
  - Meta-test phase: 5-way 1-shot (15 queries)

2. (20%) When meta-train and meta-test under the same 5-way 1-shot setting, please report and discuss the accuracy of the prototypical network using 3 different distance functions (i.e., Euclidean distance, cosine similarity, and parametric function). You should also describe how you design your parametric function. Note: Parametric: use a learnable model to measure the distance between two features.

Euclidean distance	Cosine similarity	*Parametric function: FC layer + Gaussian kernel + Euclidean distance
45.33% +- 0.84%	43.86% +- 0.86%	38.51 +- 0.8%

*Parametric function:
FC layer + Gaussian kernel + Euclidean distance

We first use one fully connected layer & standardization to project features to the lower dimensional (dimension: from 1600 to 300) Gaussian kernel. Afterward, we calculate the Euclidean distances.

We observe that euclidean distance which is the most effective. The parametric function we designed might reduce to too small dimensions to be recognized.

3. (10%) When meta-train and meta-test under the same 5-way K-shot setting, please report and compare the accuracy with different shots. (K=1, 5, 10)

K = 1	K = 5	K = 10
45.33% +- 0.84%	35.88% +- 2.58%	48.46% +- 3.11%

10 shots obtain the best performance since many data to learn. However, 5 shots have lower accuracy than 1 shot. It might be due to the randomness in selecting meta training data. Specifically, 5 images can be much different from one another, leading to a challenge in learning.

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

1. (10%) Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (Include but not limited to the name of the SSL method you used, data augmentation for SSL, learning rate schedule, optimizer, and batch size setting for this pre-training phase)

Implementation details

- **SSL method:** BYOL
- **Learning rate schedule:** 0.001 (decay step: 30)
- **Batch size:** 32
- **Optimizer:** Adam
- **Data augmentation:**

```
self.transform1 = transforms.Compose([
        filenameToPILImage,
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                        [0.229, 0.224, 0.225])
        ])
```

```
self.transform2 = transforms.Compose([
        filenameToPILImage,
        transforms.ToTensor(),
        RandomApply(
           transforms.ColorJitter(0.8, 0.8, 0.8, 0.2),
           p = 0.3
        ),
        transforms.RandomGrayscale(p=0.2),
        transforms.RandomHorizontalFlip(),
        RandomApply(
           transforms.GaussianBlur((3, 3), (1.0, 2.0)),
           p = 0.2
        ),
        transforms.RandomResizedCrop((128, 128)),
        transforms.Normalize(
           mean=torch.tensor([0.485, 0.456, 0.406]),
           std=torch.tensor([0.229, 0.224, 0.225])),
        ])
```

2. (10%) Following Problem 2-1, please conduct the Image classification on the Office-Home dataset as the downstream tasks for your SSL method. Also, compare the results.

Setting	**Pre-training (Mini-ImageNet)**	**Fine-tuning (Office-Home dataset)**	**Validation Accuracy (Office-Home dataset)**
A	-	Train full model (backbone + classifier)	28.325%
B	w/ label (SL)	Train full model (backbone + classifier)	52.709%
C	w/o label (SSL)	Train full model (backbone + classifier)	**37.192%**
D	w/ label (SL)	Fix the backbone. Train classifier only	51.724%
E	w/o label (SSL)	Fix the backbone. Train classifier only	32.020%

*SL: supervised learning

*SSL: self-supervised learning

3. (10%) Discuss or analyze the results in Problem 2-2

From the table in Problem 2-2, we can conclude that the performances of supervised learning (SL) are better than self-supervised learning (SSL). The result is apparent since SL has labels to learn. In addition, training a full model outperforms training only classifiers since more parameters are updated in the former task.