Deep Learning for Computer Vision

Homework #4

經濟四 李品樺 **B07303024**

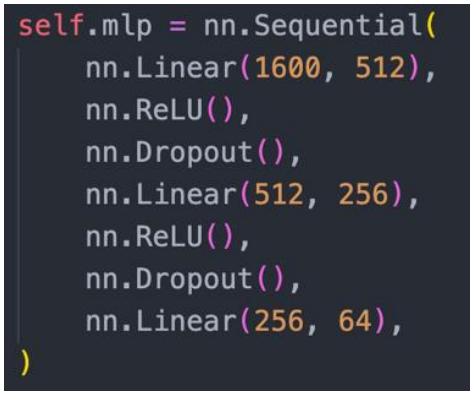
Collaborator: None

Problem 1: Prototypical Network

- **Report**

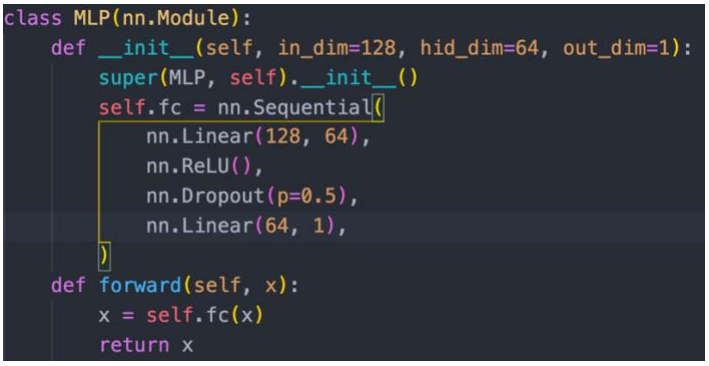
1. **Describe the architecture & implementation details of your model. (Include but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot setting for meta-train and meta-test phase)**

   Using the provided feature extraction backbone and adding MLP after as the figure shows, we initialize the model with Gaussian noise. For distance function, we choose Euclidean distance. Choose Adam as optimizer with 1e-4 learning rate and 1e-4 weight decay. We conduct 10-way 1-shot experiment setting to train the model first, then we finetune the model under 5-way 1-shot setting with learning rate 8e-5. Without data augmentation, we train for 500 epochs, and each epoch has 256 episodes. The mean accuracy can reach about 42.63%.



2. **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 function (i.e., Euclidean distance, cosine similarity and parametric function). You should also describe how you design your parametric function.**

For Problem 1-2 and 1-3, we train for 100 epochs, and with learning rate 1e-4. For the parametric function, we implemented the parametric function using MLP as the figure below shows. From the table, we can find out that accuracy of using parametric function and cosine similarity as loss function is far below using Euclidean distance, and the accuracy of using parametric function is slightly higher than using cosine similarity. It is probably because the model design of parametric function is not complicated enough, or the choice of hypermeters are not good enough, so the it does not perform well.



	Euclidean distance	Cosine similarity	Parametric function
Accuracy	37.08 +- 0.77 %	19.95 +- 0.33 %	22.45 +- 0.36 %

3. **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)**

It is intuitive that if we give more shots at the training phases, the accuracy will be higher. From the table below, we notice that the experimental results proved this idea. Notice that when we increase K from 1 to 5, the accuracy improves about 30%. On the other hand, when K increases from 5 to 10, the accuracy only raises about 6%.

	K=1	K=5	K=10
Accuracy	37.103%	66.519%	72.402%

- **Model Performance**

Accuracy: 42.63 +- 0.84 %

Problem 2: Self-Supervised Pre-training for Image Classification

- **Report**

1. **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)**

   We choose BYOL for SSL pretraining. The data augmentation for SSL is shown in the following figure. For pretraining state, we choose Adam as optimizer with learning rate 7e-5. with no learning rate scheduler. Train after 1000 epochs with batch size 128, we use the pretrained model for downstream tasks.

```
self.transform = transforms.Compose([
    filenameToPILImage,
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.CenterCrop(64),
    transforms.RandomRotation(20),
    transforms.Resize(128),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
```

2. **Following Problem 2-1, please conduct the Image classification on Office-Home dataset as the downstream task for your SSL method. Also, please complete the following Table, which contains different image classification setting, and compare the results.**

   For downstream tasks, we fix the classifier structure across setting A-E as the following figure shows. We train for 500 epochs with Adam as optimizer and fixed learning rate at 2e-4. Our batch size is 32 for all settings. Notice that for the optimizer, we need to filter out the parameters that does not require gradients in setting D and setting E.

```
resnet.fc = nn.Sequential(
    nn.Linear(2048, 1024),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(1024, 512),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(512, 256),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(256, 65),
)
```

Since the provided labels in Office-Home dataset are in text data, thus we need to convert those labels into numerical class (0 to 64). When we are loading data at training stage, we save two dictionaries "label2class.pkl" and "class2label.pkl". During inference stage, we load the saved "class2label.pkl" and convert the predicted result back to text labels.

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Classification accuracy on valid set (Office-Home dataset)
A	-	Train full model (backbone + classifier)	**37.438%**
B	w/ label (TAs have provided this backbone)	Train full model (backbone + classifier)	**41.379%**
C	w/o label (Your SSL pre-trained backbone)	Train full model (backbone + classifier)	**37.685%**
D	w/ label (TAs have provided this backbone)	Fix the backbone. Train classifier only	**37.931%**
E	w/o label (Your SSL pre-trained backbone)	Fix the backbone. Train classifier only	**17.241%**

3. Discuss or analyze the results in Problem 2-2.

From the table above shows, setting B performs the best and the accuracy is over 41.3%. We can analyze the result in three aspects: 1. Pretrain or train the model from scratch 2. Pretrain with labels or without labels 2. Finetune full model or classifier only. First, comparing setting A, settting B, and setting C, accuracy only increased by 0.25% if pretrained without labels. If pretrained with

label, the accuracy can increase about 4%. Secondly, pretrain with labels is critical to improve the peerformance of this problem. Accuracy of setting D is above 20% higher than setting E, and that of setting B is almost 4% higher than setting C. Finally, finetune on the entire model instead of fixing the backbone is also important for better performance. Accuracy of setting B is 3% higher than setting D; setting C 20% has an accuracy 20% higher than setting E.

Reference: BYOL, fix backbone

- **Model Performance**

  Setting C Accuracy: **37.685%**