|  |  |
| --- | --- |
| Lab 3: Few-Shot Classification | |
| Student ID: | Student Name: |

Table 1. Accuracies on the Omniglot dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Atlantean | Hiragana | Katakana | Korean | ULOG | Average |
| Baseline (random) | 0.7615 | 0.8423 | 0.8383 | 0.8900 | 0.9154 | 0.8495 |
| Baseline (finetuning) | 0.9846 | 0.9346 | 0.8723 | 0.9600 | 0.9538 | 0.9411 |
| Prototypical networks |  |  |  |  |  |  |

**1. Baseline (1pt)**

1. Run "baseline.py". Report the accuracy of baseline (random) model on Table 1
2. What do you think is the main factor of Baseline's poor performance?

Because the size of training set is small.

**2. Transfer Learning (2pt)**

1. Now, we want to exploit the advantage of meta training set consisting of 1,423 characters from 45 different alphabets. We replace the output layer of baseline network to perform 1,423 classes classification. Run "transfer\_pretrain.py" to perform pretraining stage of transfer learning
2. After the pretraining stage, we want to finetune the baseline model for test tasks. Run "transfer\_finetune.py". To perform fine-tuning, you should implement the code of loading pretrained weights. Then report the accuracy of baseline (finetune) model on Table 1
3. Compare two baseline models. Which one gives better performance on test tasks? What do you think is driving the performance improvement?

Baseline(finetuning) model has better performance than baseline(random) model. Since "transfer\_finetune.py" was learned using the parameters of the "transfer\_pretrain.py" model, the learning time and accuracy of the model were improved.

**3. Prototypical Networks** **(2pt)**

1. Implement the omniglot\_prototype\_collate\_fn function in "datasets.py" to perform the episodic training algorithm
2. Implement the prototypical network in "models.py"
3. Run "prototype\_meta.py" and "prototype\_eval.py" sequentially. Report the performance of prototypical network to Table 1. Compare three models. Which one is the best? What is the main factor of its performance?

The model works fine, but the prototype\_meta.py file failed to run through because 1 epoch took more than 2-3 minutes per fork. It seems to be a problem caused by running moel in a local environment, and Pytorch has not supported mps normally yet.(I runed models of questions 1 and 2 in the colab environment, and question 3 in the local environment.)  
  
The expected performance ranking of the network is thought to be as follows.  
1) 1) Baseline(finetuning)  
2) 2) Prototypical Network  
3) 3) Baseline(random)  
  
The reason why I predicted Finetuning to perform better than Prototypic Network is because of the feature of omniglot dataset. The omniglot dataset is a data set with a relatively small input(1x32x32) and a large class number(143). the Prototypic Network adopted a 20-way 5-shot metal learning method, and even if each epoch was repeated 1000 times, I think the parameters of each class would not have been sufficiently learned.

**4. Advanced programming technique (functional programming) (1pt)**

1. Explain the line 28-34 in the following code. Also, explain the map function:

텍스트이(가) 표시된 사진

자동 생성된 설명]

Line 28 : Define 'self.data' list

Line 29 : Define for loop with containing all file names in the path under root/ into list format

Line 30 : Return all file names in the path under root/character in list format

Line 31 : Using map() method, apply read\_image() method to each element of 'images' list.

Line 32 : Using map() method, apply to\_tensor() method to each element of 'images' list.

Line 33 : Wrap image tensor in list format. Then stack tensor expanding into a 0-dimensional space.

Line 34 : Append image tensor using data.append() method.

Given list object, map() function returns a map object of the results after applying the given function to each item of list.