Development of training environment for deep learning with medical images on supercomputer system based on asynchronous parallel Bayesian optimization

Yukihiro Nomura[1], Toshihiro Hanawa[2], Issei Sato[3,4,5], Shouhei Hanaoka[5], Takahiro Nakao[5], Masaki Murata[1], Tomomi Takenaga[1,6], Tetsuya Hoshino[2], Yuji Sekiya[2], Naoto Hayashi[1], Osamu Abe[5]

1 Dept. of CDRPM, The University of Tokyo Hospital, Tokyo, Japan
2 Information Technology Center, The University of Tokyo, Tokyo, Japan
3 Graduate School of Frontier Sciences, The University of Tokyo, Tokyo, Japan
4 Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan
5 Department of Radiology, The University of Tokyo Hospital Tokyo, Japan
6 Ibaraki Prefectural University of Health Sciences, Ibaraki, Japan

nomuray-tky@umin.ac.jp

Introduction

- Deep learning has been exploited in the field of medical image analysis.
  ex.) automated lesion detection, classification of abnormalities, segmentation, image enhancement, image generation
- Most of medical image data is volumetric.

Requirement for training deep learning with medical images:

- Large amounts of computational power including GPU cluster
- Careful tuning of numerous hyper-parameters

Objective of this study

To build an environment for training deep learning with medical images on the supercomputer system based on asynchronous parallel Bayesian optimization

Our training environment

1. Reedbush-H supercomputer system

- 120 compute nodes
  - Intel Xeon E5-2695v4 × 2
  - 256GB RAM
  - NVIDIA Tesla P100 (16GB) × 2
- Parallel file system (Lustre, 5.04PB)
- 100 Gbps InfiniBand
- Shared login node
- Dedicated login node
  - to ensure security using the anonymized medical image data

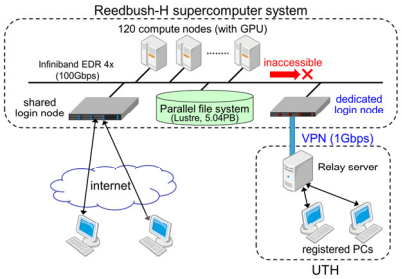

Fig. 1 Configuration of Reedbush-H supercomputer system. UTH: The University of Tokyo Hospital

2. Training framework

- Implemented into the dedicated login node
  - **hyper-parameter tuning module**
    - ➢ based on Bayesian optimization (BO) [1] written in C/C++
  - **job submission module**
    - ➢ written in Xcrypt [2]
- Training jobs are iterlatively executed at the compute nodes

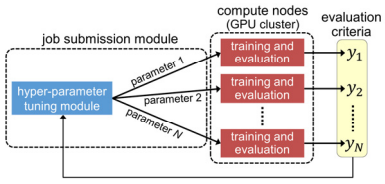

Fig. 2 Configuration of our training framework.

3. Bayesian optimization (BO) [1]

- Sequential algorithm
- Maximization of the black-box function
  - $f(\boldsymbol{x})$: drawn from Gaussian Process ($\boldsymbol{x}$: hyper-parameters)

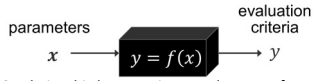

Fig. 3 Relationship between input and output of hyper-parameter search by Bayesian optimization.

- The sampling point was chosen by balancing exploitation and exploration

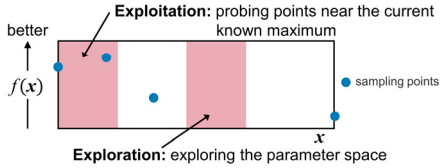

Fig. 4 Relationship between exploitation and exploration.

Asynchronous parallel BO (asyBO)[3]

- The parameters for new job were chosen using the result of the finished jobs.
  - expect to improve the efficiency for training of deep learning on the GPU cluster

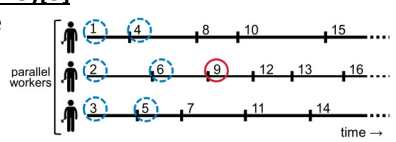

Fig. 5 Example of job sequence for asynchronous parallel BO using $M = 3$ workers. The parameters for ninth job were chosen using the result of the first six jobs. (The seventh and eighth jobs are ignored because they are not completed.)

4. Xcrypt [2]

- Job level parallel script language based on Perl
- Easy to manage asynchronously running jobs

Example of Xcrypt script:

```
use base qw (sandbox limit core);
&limit::initialize(4);          ← Set the number of workers (M ≥ 1)
my $PARAM_NUM = 10;
my $HISTORY_FILE = "$ENV{'PWD'}/parallel_bo4_01.out";
%template = (
  'RANGE0' => [1..32],
  'id' => "parallel_bo4_01",
  'JS_phnode' => '1',
  'JS_queue' => 'queue',
  'JS_group' => 'group',
  'JS_limit_time' => '24:00:00',
  'exe1@' => sub { "python training_unet3d.py" },
  'arg1_0@' => sub { " $ENV{'PWD'}/$self->{id}" },
                                  ← Set information to submit the job
  'before' => sub {
    my $param_file = "$ENV{'PWD'}/$self->{id}/params.txt";
    my $ret = `get_bo_params $PARAM_NUM $HISTORY_FILE $param_file`;
    sleep 5;
  },                              ← Procedure invoked before submitting the job (get new hyper-parameters)
  'after' => sub {
    my $results = `cat $ENV{'PWD'}/$self->{id}/results.txt`;
    open(OUT, ">> $BO_HISTORY_FILE");
    print OUT $results . "¥n";
    close(OUT);
  },                              ← Procedure invoked after the job is completed (save the combination of the hyper-parameters and the evaluation value to the history file)
  'after_aborted' => sub {
    my $params = `cat $ENV{'PWD'}/$self->{id}/params.txt`;
    open(OUT, ">> $BO_HISTORY_FILE");
    print OUT $params ." 0.000000¥n";
    close(OUT);
  }                               ← Procedure invoked after the job is aborted (save the combination of the hyper-parameters and the evaluation value (=0.000000) to the history file)
);
prepare_submit_sync (%template);
```

Evaluation

Automated detection of lung nodule in chest CT using 3D U-Net

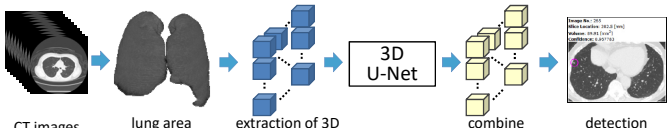

Fig. 6 Flowchart of automated detection of lung nodule in chest CT using 3D U-Net.

Conditions

- Dataset: 948 sets of chest CT images
  - training: 843 sets, validation: 105 sets
- Evaluation criterion
  - partial AUC value of FROC curve ([0,1])
    - ➢ upper limit: 5 false positives per case
- 11 types of hyper-parameters (Table 1)
- Number of workers (M): 1*, 2, 4, 8, 16
- Experiment: 3 times
  - 32 training jobs per experiment
  - initialization: randomly picking 3 sets of hyper-parameters

*** works as sequential BO (seqBO)**

Table 1 Search range of the hyper-parameters.

Parameter	Search range
size of input cube patch (N)	32 or 64
depth of U-Net	2/3/4
number of filters at the first layer	4-32 (step:2)
filter size of convolution layer	3 or 5
using residual unit	True/False
minibatch size	2-16 (step:2)
learning rate (Adam)	10^{-3}-10^{-6}
β_1 (Adam)	0.9-0.99
number of data augmentation for each patch	0-4
maximum size of random shift for data augmentation	0-N/2
random rotation in augmentation※	True/False

※ 0/90/180/270 degree on the axial, coronal, and sagittal plane

Results

- Performance after tuning:
  - **seqBO (M = 1) > asyBO ($M \leq 8$) > asyBO (M = 16) ≈ random search**
  - asyBO can realize of training with hyper-parameter tuning in a short time

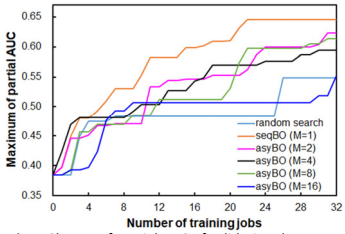

Fig. 7 Change of partial AUC of validation data where each value is maximum value in past training jobs.

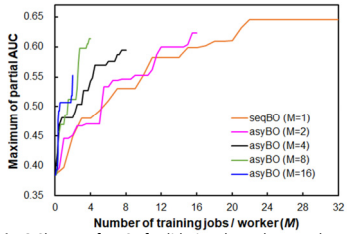

Fig. 8 Change of AUC of validation data where each value is maximum value in past trials. The horizontal axis indicates the number of training jobs divided by the number of workers.

Conclusion

The constructed environment enabled to efficiently train deep learning with hyper-parameter tuning on the Reedbush-H supercomputer system.

Acknowledgement

This work was supported by the Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures and High Performance Computing Infrastructure projects in Japan (Project IDs: jh170036-DAH, jh180073-DAH, and jh190047-DAH)

References

[1] J. Snoek, et al., Advances in NIPS, pp. 2951-2959, 2012.
[2] T. Hiraishi, et al., Proceedings of the WHIST, 2012.
[3] K. Kandasamy, et al., arXiv:1705.09236v1, 2017.
[4] Y. Nomura, et al., Jpn J Radiol, vol.37, issue 3, pp.264-273, 2019.