The Nebula Benchmark Suite: Implications of Lightweight Neural Networks

Bogil Kim, Sungjae Lee, Chanho Park, Hyeonjin Kim, and William J. Song
School of Electrical Engineering, Yonsei University

Jan. 2020 (Version 1.0)

1. Introduction

Nebula is a lightweight benchmark suite for neural networks. Recent neural networks become increasingly deeper and larger. This trend puts great challenges on the modeling and analysis of computer systems in that it takes longer execution time to process a large number of operations and sizable data. Nebula tackles this challenge by taking the opposite direction based on an observation that the computations of neural networks are mainly comprised of matrix and vector operations. Nebula benchmarks drastically reduce the number of operations for an expectation that the lightened networks will still have similar architectural behaviors as the full-fledged neural networks, and thus it is not always necessary to run the complete networks with sizable data if the purpose is to characterize micro-architectural behaviors in computer systems. Results based on hardware measurements prove that Nebula benchmarks indeed meet the similarity and affordability goal.

2. Prerequisite, Download, and Installation

The Nebula benchmark suite implements a C++ framework that facilitates the composition of diverse neural networks, similar to the well-known Caffe framework. The framework standardizes the implementation of network and layer classes for easier extension to new type of neural networks. The standardization makes it convenient to interface with them and share acceleration libraries among the various implementations of neural networks.

Nebula requires g++ compiler to build on CPU, nvcc on GPU, and OpenCV for image processing. It utilizes several acceleration libraries encompassing OpenBLAS for CPU and CUDA libraries (e.g., cuBLAS and cuDNN) to speed up the computations of neural networks, but the implementations are not limited to those external libraries. To install prerequisite to run the Nebula benchmark, use the following command in terminal.

```
[required] * g++
[required] * OpenCV
$ sudo apt-get install libopencv-dev
[optional] * nvcc
[optional] * cuBLAS for GPU
[optional] * cuDNN for GPU
[optional] * OpenBLAS for CPU
$ sudo apt-get install libopenblas-dev
```

It has been validated in Ubuntu 16.04 and 18.04. To obtain a copy of Nebula v1.0 (as of July, 2020), use the following command in terminal. Then, enter the *nebula/* directory and build the benchmark using shell script *nebula.sh*.

```
$ git clone --branch v1.0 https://github.com/yonsei-icsl/nebula
$ cd nebula/
$ ./nebula.sh build all
```

After downloading the Nebula framework, you can build Nebula with various options in shell script *nebula.sh*.

```
# nebula.sh
##### Optional arguments #####
# Use gdb for debug.
  use_debug=0
# Use GPU.
  gpu_enabled=1
# Use custom blas.
  custom_blas=0
# Do not load weight for training by default.
  load_weight=0
```

If you want to build specific network, build the benchmark using following commands.

```
$ ./nebula.sh build lib
$ ./nebula.sh build <benchmark>
```

The Nebula suite consists of seven representative neural networks including AlexNet, VGG, ResNet, MLP, DBN, RNN, and LSTM. Each network offers three size options; Large, Medium, and Small.

3. Running Nebula Framework

Nebula runs the neural network with network configuration file, dataset image, and weight file. Network configuration file is located in benchmark directory (benchmarks/vgg/network.cfg). You can obtain dataset image and weight file using the command *get_data.sh* and *get_weight.sh*, respectively.

3.1 Preparing Database

You can download the dataset folder using the command *get_data.sh*. Nebula includes ImageNet for convolutional networks, NIST for fully connected networks, and PTB for recurrent networks. And each dataset supports variable-sized options from large to small datasets for various types of neural networks. Following shows the example of downloading small size of ImageNet.

```
$ ./get_data.sh
Which dataset? [ImageNet[I] / NIST[N] / MNIST[M]] I
Which size? [Large[L] / Medium[M] / Small[S]] S
Get data ID of ImageNet_S
--2020-04-16 19:11:14--  https://docs.google.com/uc?export=download&confirm=2bpp&id=1z
CTB82sCmBcf0tCapOa0cd7tAcPp9ADe
Resolving docs.google.com (docs.google.com)... 172.217.25.206, 2404:6800:4004:81a::200
e
Connecting to docs.google.com (docs.google.com)|172.217.25.206|:443... connected.
rk] train network.cfg data.cfg input.wgh(optional) output.wgh(optional)
 41

HTTP request sent, awaiting response... 302 Moved Temporarily

...

Connecting to doc-0o-a0-docs.googleusercontent.com (doc-0o-a0-docs.googleusercontent.c
om)|172.217.161.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-tar]
Saving to: ImageNet_S.tar

ImageNet_S.tar               [               <=>               ]   2.82G  5.15MB/s
```

After downloading the dataset, you should copy dataset list file to the benchmark folder. Dataset list includes *train.lst*, *test.lst*, and *labels.lst*. *train.lst* and *test.lst* contain the path of data file for train and test, and *labels.lst* contains the label data of dataset.

```
$ cd nebula/
$ cp data/ImageNet_S/*.lst benchmarks/resnet_S/
```

3.2 Preparing Weight for Inference

When you want to run the neural network in inference phase, you need a weight file. You can download weight file using the shell script *get_weight.sh*. The weight file is downloaded as *input.wgh* at each benchmark directory. Following shows the example of downloading weight file of Small VGG.

```
$ ./get_weight.sh vgg_S

Get file ID of vgg_S
/home/bogil/Publication/nebula/benchmarks/vgg_S
--2020-04-16 19:23:32--  https://docs.google.com/uc?export=download&confirm=&id=1L6GzG
0Je43jd6sVWICFC8oCVEtP507ee
Resolving docs.google.com (docs.google.com)... 172.217.25.206, 2404:6800:4004:81a::200
e
Connecting to docs.google.com (docs.google.com)|172.217.25.206|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily

...

Connecting to doc-00-2g-docs.googleusercontent.com (doc-00-2g-docs.googleusercontent.c
om)|172.217.161.33|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: input.wgh

input.wgh                       [           <=>                 ]  11.74M  5.31MB/s
```

3.3 Running the Neural Network

With input files (i.e., network configuration file, dataset image, and weight file of network), you can run variable-sized neural networks. After executing the network, Nebula prints out the result of execution depending on the execution model (i.e., inference and training). The result of inference includes instantaneous accuracy, cumulative accuracy, and execution time. On the other hands, an instantaneous loss, and a cumulative loss, and runtime are printed out at the end of every iteration of training phase.

```
$ ./nebula.sh test vgg_S
 ____________________________________________________________
|                                                            |
|    * *       #   #  #####  ###     #   #  #       ##     *   |
|        *    ##  #  #      #   #   #   #  #       # #   *   * |
|     *       # # #  #####  #####   #   #  #      ####     *   |
|   *        #  ##  #      #    #  #   #  #      #   #      *  |
|      *   #   #  #####  ######  #####  #####  #    #    *    |
|                                                            |
|                                                            |
|                                                            |
|  Nebula: A Neural network framework                        |
|  Intelligent Computing System Lab(ICSL)                    |
```

```
|  School of Electrical Engineering, Yonsei University        |
|  Version: 0.1                                               |
|_____________________________________________________________|
Initializing network ...
Running network ...
Iteration #0 (data #32):
  - accuracy: 50.000000%
  - runtime: 49.327999msec
Iteration #1 (data #64):
  - accuracy: 53.125000%
  - runtime: 49.601002msec
Iteration #2 (data #96):
  - accuracy: 53.125000%
  - runtime: 50.285999msec
Iteration #3 (data #128):
  - accuracy: 52.343750%
  - runtime: 49.383999msec
Iteration #4 (data #160):
  - accuracy: 57.500000%
  - runtime: 50.063000msec

...

Total runtime: 3.677841sec

Network test done.
```

4. Contact

In case you notice a bug or have a question regarding the use of Nebula benchmark suite, please feel free to contact via email, bogilkim@yonsei.ac.kr.