Deep Learning Suite

Contact: Prasanna Balaprakash (pbalapra@anl.gov), Venkat Vishwanath (venkat@anl.gov), Kalyan Kumaran (kumaran@anl.gov), ALCF, Argonne National Laboratory

Summary Version

1.0

Purpose of Benchmark

The deep learning suite contains: CANDLE benchmark codes implement deep learning architectures that are relevant to problems in cancer; Convolutional Neural Networks (CNNs) that comprises convolutional layers followed by fully connected layers; LSTM recurrent neural network (RNN) architecture that remembers values over arbitrary intervals to deal with temporal and time-series prediction; and, resnet-50 distributed training code for classification in ImageNet dataset at scale.

Characteristics of Benchmark

CANDLE:
CANDLE benchmark codes implement deep learning architectures that are relevant to problems in cancer. These architectures address problems at different biological scales, specifically problems at the molecular, cellular and population scales. We use two diverse benchmark problems, namely: a) P1B1, a sparse autoencoder to compress the expression profile into a low-dimensional vector, and, b) P3B1 a multi-task deep neural net for data extraction from clinical reports.

Convolutional Networks:
Convolutional Neural Network (CNN) is comprised of convolutional layers followed by fully connected layers. In this benchmark we target CNN architecture to take advantage of the 2D structure for image data set. This benchmark includes AlexNet, GoogleNet, Overfeat, VGG reference examples.

Long short-term memory (LSTM):
LSTM is a recurrent neural network (RNN) architecture that remembers values over arbitrary intervals to deal with temporal and time-series prediction.

The Candle, Convolution networks and LSTM benchmarks are to be run in throughput mode individually on the entire system. We are interested in the total number of instances we can run concurrently for each benchmark across the entire system and the associated maximum time for

each instance in comparison to the time taken to run 14000 instances across a 20PF Titan system.

ResNet-50 for imagenet:
This benchmark code implements resnet-50 distributed training for imagenet data using Keras and Horovod.

Mechanics of Building Benchmark

All the benchmark codes are implemented in python.

1. Setup python 3.6.2 environment.
2. Install the following the python packages with the corresponding versions and dependencies
    a. Keras>=2.0.3
    b. tensorflow>=1.0.1
    c. horovod>= 0.11.3
3. For resnet-50 benchmark, download ImageNet Challenge 2012 training and validation data set. See the following script for an example:

https://github.com/avolkov1/keras_experiments/blob/master/examples/build_imagenet_data/download_imagenet.sh

Mechanics of Running Benchmark

Candle Benchmarks

> cd candle/Benchmarks/Pilot1/P1B1
> python p1b1_baseline_keras2.py

> cd candle/Benchmarks/Pilot3/P3B1
> python p3b1_baseline_keras2.py

Note: When you run the benchmarks for the first time, the benchmarks will download the required training and testing data. Algorithmic and implementation changes are **not allowed** for baseline runs.

For reference, you can also download the relevant Candle datasets from:
P1B1:
http://ftp.mcs.anl.gov/pub/candle/public/benchmarks/P1B1/P1B1.train.csv'
http://ftp.mcs.anl.gov/pub/candle/public/benchmarks/P1B1/P1B1.test.csv

P3B1:
http://ftp.mcs.anl.gov/pub/candle/public/benchmarks/P3B1/P3B1_data.tgz

Convnet Benchmarks

>cd convnet-benchmarks/tensorflow
>python benchmark_alexnet.py
>python benchmark_googlenet.py
>python benchmark_overfeat.py
>python benchmark_vgg.py

Note: By default, the code will run on a GPU because of NHWC mode. Vendors are only allowed to change the source code to port onto CPUs. Algorithmic and implementation changes are **not allowed** for baseline runs.

LSTM/RNN Benchmarks

> cd rnn-benchmarks/tensorflow
> python rnn.py -n basic_lstm -b 128 -l 1024 -s 30

Note: By default, the code will run on GPU because of NHWC mode. Vendors are only allowed to change the source code to port onto CPUs. Algorithmic and implementation changes are **not allowed** for baseline runs.

Resnet-50

```
keras_imagenet_resnet50.py is the source code for Resnet-50
```

Run Rules for Throughput Case - Candle, Convnets, and LSTM

Note: Reference timings and accuracy from Titan can be found at the end of the document and also in the spreadsheet.

Base run: Suite is run in single precision. Changes for porting to CPU are allowed. Vendors can use updated versions of Keras, Tensorflow and related dependencies. Vendors can use math libraries of their choice. The candle data set should be on a global namespace reachable from all nodes. The vendors are allowed to increase the number of epochs until the training reaches the reference accuracy. The benchmark source code with any modification (for CPUs) together with instructions for installing and running the code, including the the libraries used, should be reported.

Optimized run: Vendors can optimize the code and use different frameworks (for example mxnet, pytorch, CNTK etc instead of tensorflow). Vendors can also use different

implementations of alexnet, vgg, overfeat, vgg, LSTM, p1b1, p3b1. Vendors can run in reduced precision to obtain results. In this case, the minimum acceptable accuracy is within 1% of the baseline reference accuracy. The precision must be reported and the version of the routines must be supported in a library. The benchmark source code with any modification together with instructions for installing and running the code, including the the libraries used, should be reported.

Run Rules for Data Parallel Training with Resnet for ImageNet

Base run: Suite is run in single precision. Changes for porting to CPU are allowed. Vendors can use updated versions of Keras, Tensorflow and related dependencies. Vendors can use math libraries of their choice. Vendors are allowed to change the batch size, learning rate, learning schedule, and number of epochs. Vendors can make changes to allow for code portability across tensorflow versions and they should report all changes. The imagenet data set should be on a global namespace reachable from all nodes. The benchmark source code with any modification (for CPUs) together with instructions for installing and running the code, including the the libraries used, should be reported.

Optimized run: Vendors can optimize the code and use different frameworks (for example mxnet, pytorch, CNTK etc instead of tensorflow). Vendors can also use different implementations of Resnet-50. Vendors can also run in reduced precision to obtain results. In this case, the minimum acceptable accuracy (top-1) is 74.9%.

The precision must be reported and the version of the routines must be supported in a library. The benchmark source code with any modification together with instructions for installing and running the code, including the the libraries used, should be reported.

Computing the FOM

For the two candle benchmarks, the runs should satisfy the reference accuracy (for p1b1 test correlation of 0.7816 and for p3b1 0.5523, given by the geometric mean of 6 accuracy values). For each candle benchmark, we are interested in the total number of instances we can run concurrently across the entire proposed system and the maximum time across all instances. We will compare these with the maximum time when we run 14000 instances of each candle benchmark across a 20PF Titan system. First, for each instance of a benchmark, we calculate the time. Next, we calculate the maximum time (max_time) across all instances. To compute the FOM of each candle benchmark, we compute the ratio of the titan time (max_titan) to the time on the proposed system (max_proposed), and multiply this by the ratio of the number of concurrent instances on the proposed system to the number of instances on titan (14000). The FOM of candle benchmark is given by the geometric mean of the two FOMs.

For each convnet benchmark, we are interested in the total number of instances we can run concurrently across the entire proposed system and the maximum time across all instances. We will compare these with the maximum time when we run 14000 instances of each benchmark across a 20PF Titan system. First, for each instance of a benchmark, we calculate the average time per batch for forward + backward (tfbw) across 100 steps. Next, we calculate the maximum tfbw (max_tfbw) across all instances. To compute the FOM of the benchmark, we compute the ratio of the titan time (max_tfbw_titan) to the time on the proposed system (max_tfbw_proposed), and multiply this by the ratio of the number of concurrent instances on the proposed system to the number of instances on titan (14000). The FOM of convnet benchmark is given by the geometric mean of the four individual FOMs.

For the LSTM benchmark, we are interested in the total number of instances we can run concurrently across the entire proposed system and the maximum time across all instances. We will compare these with the maximum time when we run 14000 instances of LSTM benchmark across a 20PF Titan system. First, for each instance of a benchmark, we calculate the average time per sample for forward + backward (tfbw) across 100 steps. Next, we calculate the maximum tfbw (max_tfbw) across all instances. To compute the FOM, we compute the ratio of the titan time (max_tfbw_titan) to the time on the proposed system (max_tfbw_proposed), and multiply this by the ratio of the number of concurrent instances on the proposed system to the number of instances on titan (14000).

Note:Convnet and LSTM benchmarks do not target accuracy. They are primarily used for measuring the operations involved in convolutional neural nets and LSTMs.

RESNET

For the ResNet-50 for Imagenet benchmark, reference values are computed based on 20% of the 20 PF titan system run. The ResNet-50 runs on the proposed system should satisfy the reference accuracy (74.9% top-1 test accuracy). The improvement in runtime is given by the ratio of the resnet_time_ref to the resnet_time, where resnet_time_ref and resnet_time are maximum of the runtimes obtained on the reference system and proposed system across all resnet instances, respectively. The resnet_time_ref is using 5 instances of resnet concurrently on a 20PF titan system and taking the maximum runtime across all instances. The resnet_time on the proposed system should be obtained by running N instances of resnet concurrently on the full proposed system and taking the maximum runtime across all instances. Each of the N instance should occupy 20% or lower on the proposed system. The throughput factor is given by 1 + exp(1-(N/5)), where N is the number of ResNet-50 instances that can be run concurrently on the proposed full system. The FOM of ResNet-50 benchmark is given by the product of the improvement in runtime and throughput factor.

For the optimized FOM, we will consider both the improvement in the time to solution and the solution quality. The improvement in runtime is given by the ratio of resnet_time_ref to the resnet_time. The improvement in accuracy is given by 1+ ((resnet_acc - 74.9)/(100-74.9))^3 where resnet_acc is the top-1 test accuracy obtained on the proposed system, respectively. The

optimized FOM is given by the product of improvement in runtime, accuracy, and throughput factor.

The overall FOM is computed separately for base and optimized run as weighted mean of the four FOM values from candle, convnet, RNN, ResNet-50 for Imagenet, where the weights are 5.0, 1.0, 2.0, 5.0, respectively.

Reference timings and accuracy from Titan

candle		**time**	**accuracy**
	p1b1	2675.2415	0.7808
	p3b1	60.0076	0.5637
		Forward across 100 steps	**Forward-backward across 100 steps**
convnets	alexnet	0.0715 s/batch	0.2116 s/batch
	googlenet	0.3428 s/batch	1.1633 s/batch
	overfeat	0.2112 s/batch	0.6434 s/batch
	vgg	0.4264 s/batch	1.6375 s/batch
LSTM		0.0005268 s/sample	0.001411 s/sample
		time	**accuracy**
Imagnet	Resnet50	1500s	74.9%(top-1 test accuracy)