2019 APAC HPC/AI Competition
Final Report

Team Name	GIST
Captain Name	Jabin Koo
Captain Email	jbkoo@smartx.kr
Member name	**Email address**
JungSu Han	jshan@smartx.kr
Soohyun Choi	shchoi@smartx.kr
Yujin Hong	hyj2508@smartx.kr

1. Introduction

This paper describes how team GIST optimized the original base codes and depicts methods used to obtain optimal training result, such as epoch count, images/sec, accuracy and loss.

2. Training Environment

a. Software environment

Frameworks and versions we used are:

i. Nvidia TensorFlow Release 18.09-py2
ii. TensorFlow 1.10.0
iii. Horovod 0.13.10
iv. OpenMPI 3.0.0
v. CUDA 10.0.130

Since there were some compatibility issues on DGX-1 server, we used fixed versions.

b. Dataset

ImageNet 2012 was used as a dataset. Since the original .jpg images take a long time to process, we changed the format into *tf-records* code to reduce the time elapsed for training. This transformation is done by running *build_imagenet_data.pbs* on the server, which is connected to *setDataset.sh*. The baseline code is from *Tensorflow/models* and the dataset is preprocessed in machine learning(ML) code by distortion, resizing, shifting and splitting.

c. PBS scripts

Two versions of scripts were used; one is *HorovodTrain.pbs* and another is *DockerSingleTrain.pbs*. For *DockerSingleTrain.pbs*, it is much faster to run hyperparameter tuning since it uses simple ML model and a single DGX node.

3. Method of Optimization

a. Data optimization

As a method of data preprocessing, the format of input data was changed into *tf-records* so that make it run faster and obtain better accuracy.

b. ML optimization

To enhance learning, hyperparameter tuning was used. It was found that there are some adjustable parameters: *batch_size, optimizer, num_epochs and weight_decay.* There are some optimizers such as *Momentum, Adagrad, Adadelta, AdagradDA, RMSprop and Adam* but *Momentum, SGD, RMSprop and Adam* are only utilizable optimizers in this environment. Besides, if *num_epochs* and *weight_decay* are well-adjusted, it is possible to gain optimal accuracy and total images per second.

```
python tf_cnn_benchmarks.py --data_format=NCHW --batch_size=256 \
--model=resnet50 --optimizer=momentum --variable_update=replicated \
--nodistortions --gradient_repacking=8 --num_gpus=8 \
--num_epochs=90 --weight_decay=1e-4 --data_dir=${DATA_DIR} --use_fp16 \
--train_dir=${CKPT_DIR}
```

Figure 1. Adjustable hyperparameters

Analyzing the training result in *Figure 2*, it is noticeable that *total images/sec* tends to increase as *batch_size* increases. This suggests that optimal batch_size can be found by increasing its size before falloff occurs. By increasing *total images/sec*, *batch_size* gets bigger as expected, but at *batch_size=512*, limit error occurs. Thus, it is concluded that the most appropriate *batch_size* is *256*.

result file	batch_size	num_epochs	weight_decay	optimizer	steps	max accuracy at(step)	walltime requested	walltime used	max accuracy(%)	total images/sec
_128_1_3.o8760365	128	1	1.00E-03	momentum	1250	1250	0:10:00	0:07:16	89.3[illegible]7	4092.82
_128_1_4.o8760366	128	1	1.00E-04	momentum	1250	1240	0:10:00	0:07:16	91.[illegible]8	4114.72
_128_1_5.o8761150	128	1	1.00E-05	momentum	1250	910	0:10:00	0:07:15	92.9[illegible]7	4114.47
_128_2_3.o8761153	128	2	1.00E-03	momentum	2500	2500	0:20:00	0:12:33	93.0[illegible]3	4092.03
_128_2_4.o8761155	128	2	1.00E-04	momentum	2500	2500	0:20:00	0:12:22	92.2[illegible]7	4131.52
_128_2_5.o8761156	128	2	1.00E-05	momentum	2500	2450	0:20:00	0:12:29	92.9[illegible]2	4112.14
_64_1_3.o8757627	64	1	1.00E-03	momentum	2500	2500	0:10:00	0:09:39	89.3[illegible]5	2805.49
_64_1_4.o8757628	64	1	1.00E-04	momentum	2500	2500	0:10:00	0:09:41	91.8[illegible]7	2777.51
_64_1_5.o8757629	64	1	1.00E-05	momentum	2500	2270	0:10:00	0:09:47	92.[illegible]6	2764.76

Figure 2. Total images per second proportional to batch size

batch_size	num_epochs	weight_decay	optimizer	steps	max accuracy at(step)	walltime requested	walltime used	max accuracy(%)	total images/sec
128	5	1.00E-03	momentum	6250				93.08	4094.93
128	5	1.00E-04	momentum	6250				93.03	4125.88
128	5	1.00E-05	momentum	6250				92.994	4115.4
64	5	1.00E-03	momentum	12510				93.086	2799.54
64	5	1.00E-04	momentum	12510				93.029	2815.56
64	5	1.00E-05	momentum	12510				92.99	2813.15
256	10	1.00E-03	momentum						
256	5	1.00E-04	momentum	3120				93.034	5392.48
256	10	1.00E-04	momentum	6250				93.06	5444.86
256	5	1.00E-05	momentum						
512	-> limit : error occurs								

Figure 3. Limit error occurs at *batch_size=512*

With *num_epochs=1*, the training results were very undertrained so now *num_epochs* was set to 20. As training progresses with *batch_size=128* and *weight_decay=1e-3,* accuracy became stable(about 93%) after step 4700, illustrated in *Figure 4*. Since *batch_size * steps / num_epochs = 160,000*, it can be inferred that *batch_size / num_epochs* should be smaller than 32.

Putting these all together, setting *batch_size=256*, *num_epoch=10* and *weight_decay=1e-3*, it can be expected that the training will not be underfitted, since *batch_size / num_epoch* is smaller than 32.

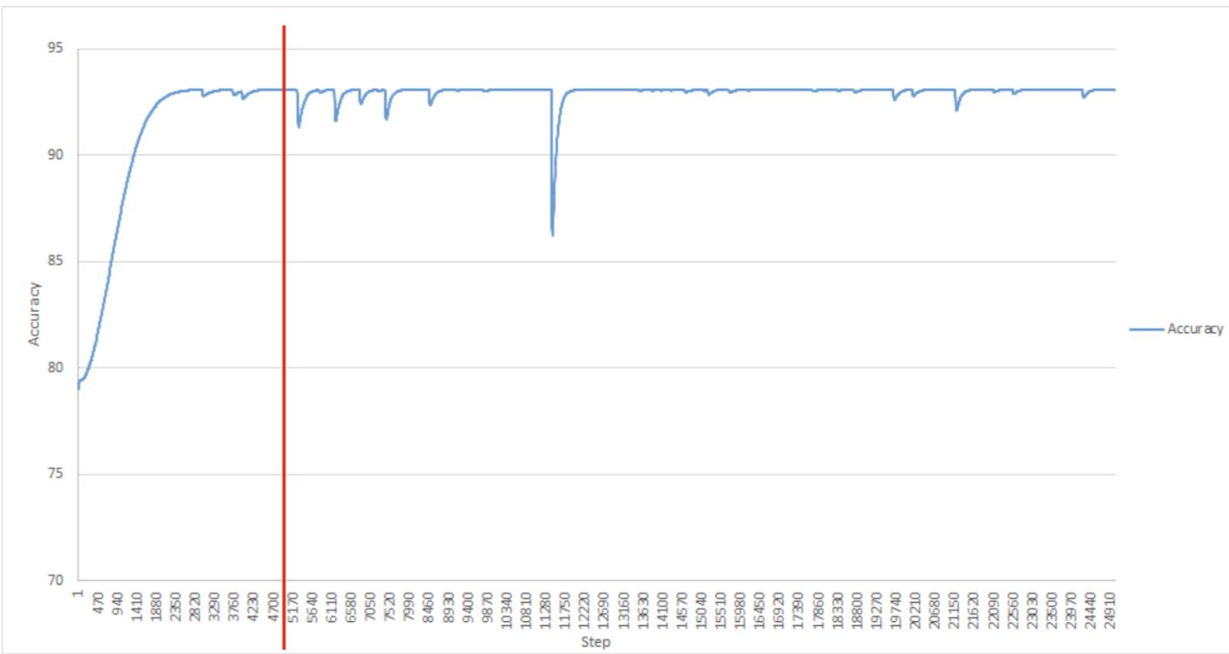

Figure 4. Training with *optimizer=momentum, batch_size=128*, *num_epoch=20* and *weight_decay=1e-3*

Besides, if *weight_decay=1e-4*, accuracy becomes stable after step 7500(*Figure 5*) so *batch_size / num_epochs* should be smaller than 24.

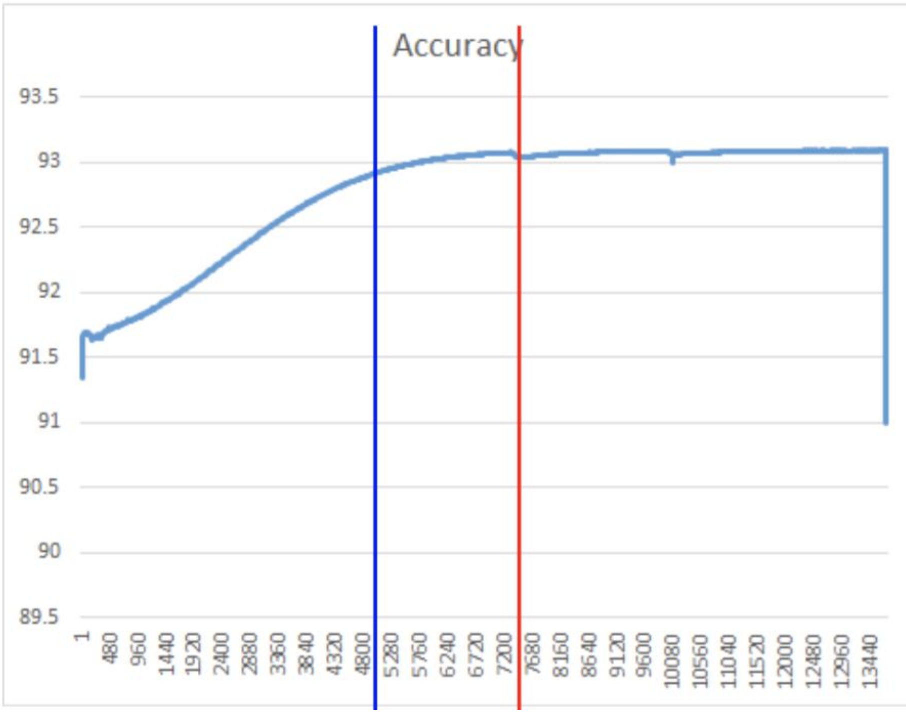

Figure 5. Training with *optimizer=momentum, batch_size=128*, *num_epoch=20* and *weight_decay=1e-4*

4. Results

From some experimental results, we expect maximum accuracy and total images/sec at *batch_size=256, num_epochs=10, weight_decay=1e-3* only in case of *optimizer=momentum*. In that experiment, maximum accuracy was about 93.088 and total images per second is 5436.71. Doing experiments with other optimizers, it was found that SGD optimizer gave the best performance(*Figure 6, 7*) where other conditions are the same.

However, since we could not write complete code of distributed training code based on Horovod, we need more practice and effort to complete this competition.

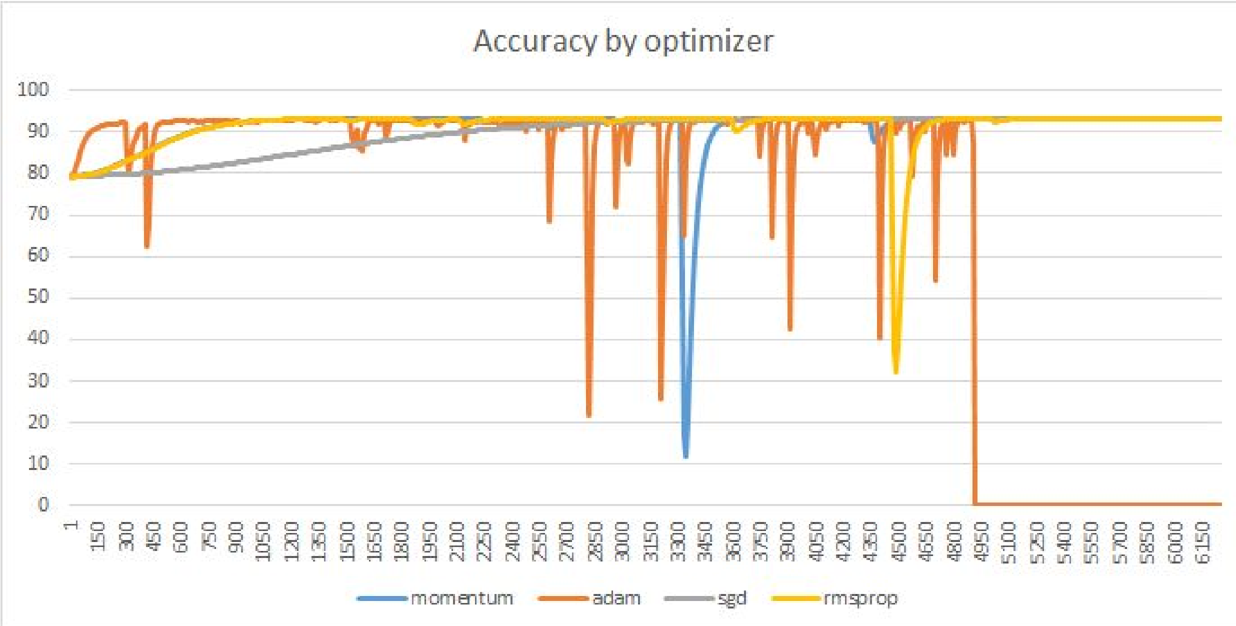

Figure 6. Training accuracy with various optimizers, *batch_size=256*, *num_epoch=10* and *weight_decay=1e-3*

batch_size	num_epochs	weight_decay	optimizer	steps	max accuracy at(step)	walltime requested	walltime used	max accuracy(%)	total images/sec
256	10	1.00E-03	adam	6250	3660			93.048	5356.18
256	10	1.00E-03	sgd	6250				93.089	5457.91
256	10	1.00E-03	momentum	6250	4120			93.088	5436.71
256	10	1.00E-03	rmsprop	6250				93.09	5378.85

Figure 7. Training results with various optimizers, *batch_size=256*, *num_epoch=10* and *weight_decay=1e-3*