Distributed BERT Pre-Training & Fine-Tuning With Intel® Optimized TensorFlow on Intel® Xeon® Scalable Processors

Muhammed Emin Ozturk▼❖, Wei Wang❖, Maciej Szankin❖, Lei Shao❖

The Ohio State University ▼, Intel® ❖

ABSTRACT

Distributed computing has become a key component in the field of Data Science, allowing for faster prototyping and accelerated time to market of numerous workloads.

This work examines performance of BERT, a state-of-the-art language model for Neural Language Processing (NLP), in the tasks of pre-training and fine-tuning on a general-purpose Intel x86 CPU.

It offers an overview of performance benefits of using TensorFlow optimizations for Intel® Architecture along with novel support for bfloat16 precision on Intel CPU. Insights on training BERT at scale with different node configuration and with various optimization options is included in the analysis.

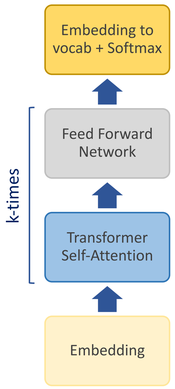
BERT

BERT, which stands for Bidirectional Encoder Representation from Transformer, is a model that can be applied to a variety of NLP tasks:

- Masked Language Model, Question answering
- Natural Language Inference, Named Entity Recognition etc.

Two Different Architecture of BERT (BERT-Large is used for MLPerf)

- BERT Base: 12 Layers(Transformer blocks), 12 attention heads, H (hidden size) =768 and 110 million parameters
- BERT Large: 24 Layers, 16 attention heads, H=1024 and 340 million parameters.



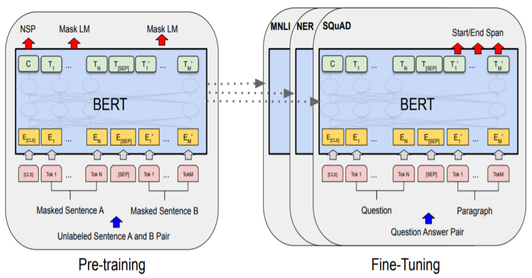


Image Source: https://arxiv.org/pdf/1810.04805.pdf

METHODOLOGY

Lower Numerical Precision (BFloat16) Optimizations

For fast trainings we use Cooper Lake-based CPU (CPX-SP) with bfloat16 support and Intel-optimized TensorFlow. The software stack features numerous Deep Neural Network (DNN) optimizations, including improved GELU, Softmax and various op fusions.

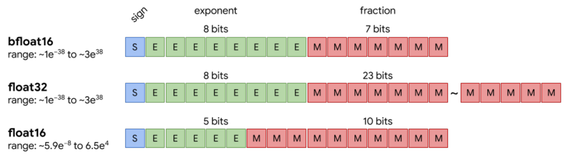


Image Source: https://cloud.google.com/tpu/docs/bfloat16

- reduced size of data in memory
- improved speed of the operations due to less data being transferred
- encode int8 as bfloat16 without loss of accuracy
- same range as fp32
- does not require range scaling (fp16 does)

Data Parallelism

Data is distributed among the process equally and DNN is replicated across process

- Broadcast operation is used to synchronize the weights in beginning
- AllReduce operation is used to synchronize the gradients across the multiple processes during the backward propagation.

The main issue of Data Parallelism is the linear increase of the batch size (Global BS) as we increase #processes.

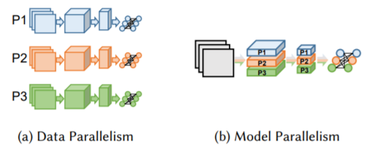


Image Source: https://dl.acm.org/doi/fullHtml/10.1145/3320060

Horovod

- **Horovod** is an open source library for distributed DNN training developed by Uber
- Provides a unified interface to train TensorFlow, Keras, PyTorch and MXNet models on HPC cluster using MPI, NVIDIA NCCL and Intel MLSL
- Horovod with:
  - Use Intel-MPI on Cascade Lake SP (CLX-SP) platform
  - OpenMPI on CPX-SP platform

Large Batch Stochastic Optimization

(Data Parallelism) The Linear increase of the batch size as we increase # of processes

↓

Large Batch Size (LBS) offer better performance due to late synchronization

↓

However, LBS cause degradation in accuracy while offering better performance

Is it possible to benefit from large batch size without losing accuracy?

YES

LAMB Optimizer

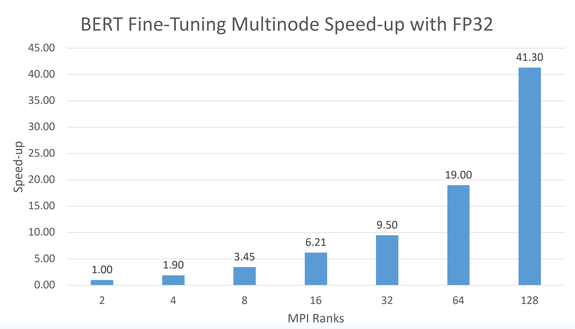
- Use of a large batch size without accuracy loss is possible with “LAMB Optimizer” as indicated by You *at al.* (“Large Batch Optimization for Deep Learning Training BERT in 76 Minutes”)
- Validated our implementation of Intel-BERT with LAMB Optimizer in Distributed Fine-Tuning experiments on up to 64 CLX-SP nodes

RESULTS

BERT Fine-Tuning Achieved Accuracy with FP32 precision

Platform Type, Sockets	Nodes	PPN	MPI	BS per MPI_Rank	Global Batch Size	LR	Best F1	Exact Match	Optimizer
CLX-SP 2S	1	2	2	32	64	0.00018	93.01	86.74	LAMB
CLX-SP 4S	1	4	4	32	128	0.00018	93.16	86.96	LAMB
CLX-SP 4S	2	4	8	32	256	0.00018	92.91	86.67	LAMB
CLX-SP 2S	8	2	16	32	512	0.00018	92.89	86.68	LAMB
CLX-SP 4S	8	4	32	32	1024	0.00018	92.35	86.18	LAMB
CLX-SP 2S	32	2	64	32	2048	0.00018	92.11	85.7	LAMB
CLX-SP 2S	64	2	128	32	4096	0.00018	91.82	85.22	LAMB

BERT Fine-Tuning Achieved Speedup with FP32 precision



BERT Fine-Tuning Speedup FP32 VS BF16

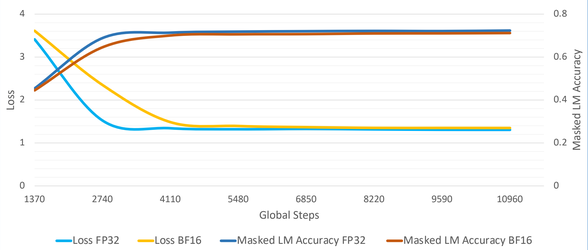
FP32 BERT Fine-Tuning

Platform Type, Sockets	Nodes	PPN	MPI	BS per MPI_Rank	Global Batch Size	LR	Best F1	Exact Match	Speedup	Optimizer
CLX-SP 2S	1	2	2	32	64	1.80E-04	93.01	86.74	1x	LAMB
CLX-SP 4S	1	4	4	32	128	1.80E-04	93.16	86.96	~1.9x	LAMB
CLX-SP 4S	2	4	8	32	256	1.80E-04	92.91	86.67	~3.45x	LAMB
CLX-SP 2S	8	2	16	32	512	1.80E-04	92.89	86.68	~6.19x	LAMB

BF16 BERT Fine-Tuning

Platform Type, Sockets	Nodes	PPN	MPI	BS per MPI_Rank	Global Batch Size	LR	Best F1	Exact Match	Speedup	Optimizer
CPX-SP 8S	1	2	2	32	64	1.80E-04	92.96	86.58	~3.8x	LAMB
CPX-SP 8S	1	4	4	32	128	1.80E-04	93.04	86.79	~7.6x	LAMB
CPX-SP 8S	1	8	8	32	256	1.80E-04	93.02	86.98	~12.6x	LAMB
CPX-SP 8S	1	16	16	32	512	1.80E-04	92.44	86.42	~12.95	LAMB

BERT Pre-Training Convergence FP32 VS BF16



CONCLUSION

This work examined the scalability of BERT’s training with software and hardware optimizations on a modern Intel CPU with a novel support of bfloat16 precision format on up to 128 MPI workers. The conducted experiments show that there is a visible performance gain from using CPU-oriented TensorFlow optimizations and a low-precision bfloat16 numerical format in both pre-training and fine-tuning phases without hurting overall accuracy of the trained model.

REFERENCES

[1] I. Tenney, D. Das, and E. Pavlick, “Bert rediscovers the classical nlp pipeline,” arXiv preprint arXiv:1905.05950, 2019.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in neural information processing systems, 2017, pp. 5998–6008.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[4] Z. Li, Y. Wang, T. Zhi, and T. Chen, “A survey of neural network accelerators,” Frontiers of Computer Science, vol. 11, no. 5, pp. 746– 761, 2017.

[5] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf et al., “Mlperf training benchmark,” arXiv preprint arXiv:1910.01500, 2019.

[6] P. Mattson, V. J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang et al., “Mlperf: An industry standard benchmark suite for machine learning performance,” IEEE Micro, vol. 40, no. 2, pp. 8–16, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[8] A. Sergeev and M. D. Balso, “Horovod: fast and easy distributed deep learning in TensorFlow,” arXiv preprint arXiv:1802.05799, 2018.

[9] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, , and C.-J. Hsieh, “Large batch optimization for deep learning: Training bert in 76 minutes,” ICLR, 2020.