Why Data Science and Machine Learning Need Silicon Photonics

Benjamin Klenk, Larry Dennison
NVIDIA, 2788 San Tomas Expressway Santa Clara, CA 95051
bklenk@nvidia.com, ldennison@nvidia.com

Abstract: Training deep neural networks demands vast amounts of computation, provided by large distributed systems. The increasing demand for bandwidth will exceed the limits of electrical and non-integrated optical signaling and will require integrated optics in the future. © 2020 The Author(s)

OCIS codes: 000.0000, 999.9999.

1. Introduction

The combination of vast amounts of available data and processing power of today's computing systems has enabled Machine Learning (ML) techniques to surpass human capabilities in a wide range of tasks, including image and speech recognition [1, 2]. Consequently, ML is transforming almost every industry and has become the focus of all major internet companies, such as Google, Microsoft, or Amazon.

In particular, deep learning (DL) has been shown to be able to excel at various tasks, including computer vision, speech recognition, and natural language processing (NLP). DL implements deep networks of neurons that can learn higher order functions and representations. There are two basic operations: training and inference. In supervised learning, the training process presents labeled data to the neural network, which makes a prediction that is compared with the label. The error of the prediction is backpropagated to update the neural network's parameters. This process is repeated numerous times until the error becomes minimal. After training, the network is deployed and fed with unlabeled data, on which predictions are made. This process is known as inference.

The training of deep neural networks (DNNs) is compute intensive and requires many iterations over vast amounts of data. Without parallel computing and accelerators like GPUs or Google's specialized TPU [3], the training of such DNNs would not be practical. In fact, today's training systems comprise hundreds if not thousands of accelerators [4]. As accelerator performance improves and systems scale out to such large numbers of accelerators, the interconnection network becomes critical. The remainder of this work discusses the role of the network and why we will see an increased demand for bandwidth. This demand cannot be fulfilled by existing electrical or optical solutions and requires us to move towards silicon photonics.

2. Parallel Training Algorithms

The following describes how DNNs are trained on parallel and distributed systems.

2.1. Data Parallelism

The algorithm used to train DNNs is stochastic gradient descent (SGD). Instead of calculating and backpropagating the error for every sample in the training set, multiple samples are drawn stochastically to form a *mini-batch*. Parameters of the DNN are updated once per mini-batch. One pass through all mini-batches of the training set is called an *epoch*.

The most common and scalable way to parallelize the training is to split the mini-batch into p *sub-batches*, in which p is the number of parallel workers. Each worker calculates the weight gradients, which indicate how the parameters need to change, and then exchanges the gradients with all other workers, commonly implemented with an *All-Reduce*.

The maximum size of the mini-batch is the size of the training set. However, if parameters are only updated once per epoch, many epochs may be required until the error of the DNN converges to a minimum. Conversely, using small mini-batches will result in more frequent parameter updates, but can lead to a poor quality of result. As a consequence, there is an optimal mini-batch size for a network to converge using the least amount of computation [5]. As we scale out horizontally to large numbers of workers under a given maximal mini-batch, the sub-batch per worker becomes smaller and smaller. In practice the sub-batch often needs to be larger than one sample to allow for efficient computations.

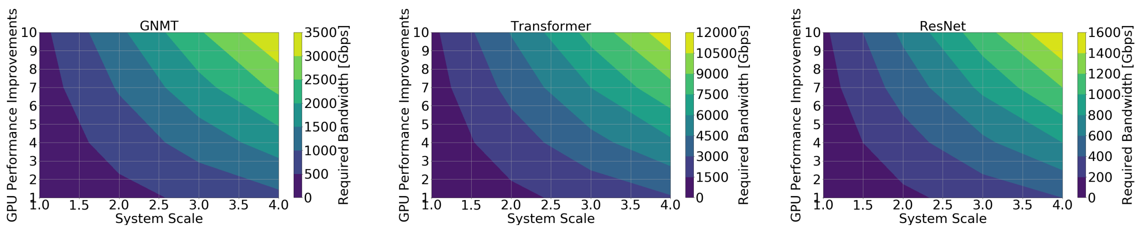

Fig. 1. Required bandwidth to perfectly overlap the weight gradient reduction with the backward pass as GPUs become faster and systems larger. The data assumes software All-Reduce algorithms, which achieve half of the available raw bandwidth (shown in the graphs).

While some networks like *ResNet* for computer vision tasks allow for large mini-batches [6] and therefore good scale-out, others like *Transformer networks* [7] for NLP are more restricted. In order to accelerate training beyond the limit of data parallelism, model parallelism needs to be applied.

2.2. *Model Parallelism*

While data parallelism splits the input and trains replicas of the model on sub-batches, model parallelism distributes the model and its parameters across a number of workers. This becomes necessary when we reach the limits of data parallelism, but also when models are simply too large to fit into a worker's memory. In general, we observe that larger models tend to perform better at their tasks and achieve higher accuracy, both in computer vision [8] and NLP [9]. We can therefore assume that models will continue to grow and require a plurality of workers, rendering communication between them even more important.

3. Bandwidth Requirements in Machine Learning

In this section, the bandwidth requirements of the training of DNNs are discussed in more detail.

3.1. *Deep Learning Training*

In data parallelism the weight gradient reductions can be overlapped with the backpropagation of the error, for example by reducing gradients of layer i while we calculate gradients for layer $i-1$ (note that we go backwards through the DNN). Therefore, the bandwidth of the network needs to be sufficiently high to hide communication behind the backward pass. The required bandwidth B_{req} in data parallelism therefore depends on the model size M and floating point operations (FLOPs) F, the number of workers p, the performance of a worker P and efficiency α, as well as the mini-batch size b_m. Note that the efficiency and FLOPs are non-linear functions of the sub-batch $\frac{b_m}{p}$ and model size M and increase as either gets larger.

$$B_{req} \propto \frac{pM \cdot P\alpha(\frac{b_m}{p},M)}{F(b_m,M)} \xrightarrow{M=const} \frac{pP\alpha(\frac{b_m}{p})}{F(b_m)} \qquad \begin{array}{l} \leftarrow \text{increases as we improve the GPU performance} \\ \leftarrow \text{decreases as we increase the scale} \end{array} \tag{1}$$

Equation (1) shows that the bandwidth increases as the number of workers and the performance of a worker increase and we observe that both of these factors grow rapidly. Although Moore's Law in terms of single processor performance has slowed down if not ended, parallel processing and specialization keep on increasing compute performance at exponential rates. Furthermore, we already deploy hundreds to thousands of accelerators to train large models and as models become larger and tasks more complex, even larger systems are required to keep training duration reasonable. As a result, the bandwidth requirements between accelerators keep on growing to the point in which electrical and non-integrated optical signaling become impractical. Figure 1 shows the calculated required bandwidth based on a network's size and the time spent in the backward pass. As we increase the scale (x-axis) or the performance of the GPU (y-axis) the backward pass time decreases while the size of the weight gradients stay constant. The graphs assume a linear decrease in compute time as we increase scale or performance. As a result, the required bandwidth increases dramatically as GPUs get faster and systems larger.

As aforementioned, model parallelism will become inevitable to overcome limits of data parallelism, but also memory capacity constraints of processors. Unlike data parallelism, it is difficult to hide communication. One attempt is to map layers of the DNN onto different processors and operate the chain of workers in a pipeline fashion. However, this requires to find a careful partitioning in which each worker has about the same amount of work to avoid 'pipeline bubbles'. A more promising albeit non-overlapping case is to process individual layers collaboratively on multiple workers. This requires synchronization and collective data exchanges for every split layer in the forward and backward pass. Again, this too requires careful partitioning and effort to optimize data placement for locality.

Each layer's time is a linear combination of time spent on computation and communication. Consequently, with compute improving at exponential rates the bandwidth needs to keep up to avoid the network to become the bottleneck. For example, NVIDIA's Pascal architecture achieved 20 TFLOP/s at 150 GB/s NVLink bandwidth, while the current Votla architecture achieves 120 TFLOP/s at 300 GB/s NVLink bandwidth. That is a 3x increase in the ratio of compute performance to I/O bandwidth.

The reason why model parallelism is complex and hard to implement is the effort involved in optimizing for locality to minimize the overhead of communication. This is caused by the much lower I/O bandwidth compared to the local memory bandwidth. For example, NVIDIA's Volta architecture [10] provides 900GB/s to local HBM (and much higher bandwidth to the L2 cache), but only 300GB/s to NVLink-connected peers. However, the NVLink domain is limited to 16 GPUs today as it requires an indirect network with full-bisection bandwidth. Scaling model parallelism further requires not only higher bandwidth but also larger domains (e.g. rack scale). Co-packaged silicon photonics can address the predicted bandwidth density, power and reach requirements.

4. Conclusion

Machine Learning is becoming the dominant workload in data centers and accuracy of models improve rapidly. As models keep increasing in size, training them becomes challenging and requires tremendous amount of compute resources. While processor performance improves at exponential rate and we continue to increase our systems, the interconnection network's bandwidth needs to keep up. For example, the training of a Transformer model on a 2x larger systems with 5x faster GPUs than today already requires almost 3 Tbps bandwidth per GPU (i.e. 8x 400GigE), pushing the limits of electrical and traditional optical signaling. Practical model parallelism demands network bandwidths close to the accelerator-attached memory bandwidth, further arguing for in-package silicon photonics.

References

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
2. Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.
3. Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12. IEEE, 2017.
4. Mlperf v0.5 and v0.6 Training Closed; Retrieved from `www.mlperf.org` 11 October 2019, all entries. MLPerf name and logo are trademarks. See `www.mlperf.org` for more information.
5. Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.
6. Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. Extremely large minibatch sgd: training resnet-50 on imagenet in 15 minutes. *arXiv preprint arXiv:1711.04325*, 2017.
7. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
8. Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.
9. Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
10. Jack Choquette, Olivier Giroux, and Denis Foley. Volta: Performance and programmability. *IEEE Micro*, 38(2):42–52, 2018.