An Integrated Approach to Neural Network Design, Training, and Inference

Amir Gholami[1], Michael W. Mahoney[2], Kurt Keutzer[3]
University of California, Berkeley
{amirgh,mahoneymw,keutzer}@berkeley.edu

***Abstract*— Finding the right Neural Network model and training it for a new task requires considerable expertise and extensive computational resources. Moreover, the process often includes ad-hoc rules that do not generalize to different application domains. These issues have limited the applicability and usefulness of DNN models, especially for new learning tasks. This problem is becoming more acute, as datasets and models grow larger, which increases training time, making random/brute force search approaches quickly untenable. In large part, this situation is due to the first-order stochastic gradient descent (SGD) methods that are widely-used for training DNNs. Despite SGD's well-known benefits, vanilla SGD tends to perform poorly, and thus one introduces many (essentially ad-hoc) knobs and hyper-parameters to make it work. It has been found that these hyper-parameters are significantly more sensitive to tuning in large scale training with SGD, and this has impeded effective use of supercomputing systems. Here, we argue that a multi-faceted approach is needed to address these challenges by considering the full stack of neural network architecture design, large scale training, and efficient inference on edge platforms. This requires designing mechanisms to better understand NN training and bridge the gap between theoretical results for optimization, second order methods, and high performance computing.**

I. Background and Significance

Deep Neural Networks (DNNs) have proven to be very effective in diverse applications ranging from semantic segmentation [1, 2] and detection [3, 4] in computer vision to scientific applications such as astronomy [5], climate science [6], and medical image analysis [7, 8]. In these and many other applications of machine learning (ML) and artificial intelligence (AI), finding the right DNN architecture for a particular application and then training a high-quality model requires extensive hyper-parameter tuning and architecture search, often on very large data sets. The delay associated with training DNNs is often the main bottleneck in the design process, and this bottleneck limits the usefulness of DNNs in many applications.

The most straightforward method for accelerating training is to perform the so-called data parallel approach with large batches [9]. However, to efficiently utilize distributed processors, the batch size must grow with the number of processes. In the ideal case, the hope is to decrease the computational time proportional to the increase in batch size, without any drop in generalization quality. However, this is typically not the case; and, in fact, training with large batches often results in poor generalization [10, 11]. It has been found that large batch size training is more likely to converge to the so-called "sharp" local minima, which often do not generalize well. As opposed to this, small batch training has been found to converge to "flatter" local minima that do not have this problem. However, the latter cannot be efficiently scaled to parallel processes.

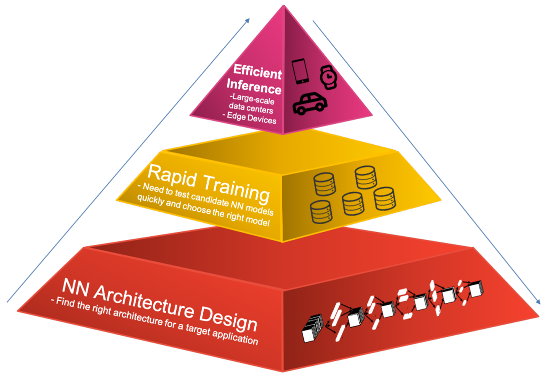
Efficient Inference
-Large-scale data centers
- Edge Devices
Rapid Training
- Need to test candidate NN models quickly and choose the right model
NN Architecture Design
- Find the right architecture for a target application

Fig. 1. Reaching the next milestone in large scale machine learning, requires a novel approach by integrating the full stack of designing Neural Network architecture, training, and inference on a target hardware platform. This requires development of new tools/hammers to gain more insight into the problem such as through second-order methods, scalable training methods that are robust to hyper-parameters, and direct integration of inference constraints such as latency/power on a target hardware platform.

In order to address these drawbacks, many solutions have been proposed [12–17]. However, these methods either work only for particular models on particular datasets, or they require massive hyper-parameter tuning. While extensive hyper-parameter turning may result in good tables for publications, it is antithetical to the original motivation of using large batch sizes to reduce training time in real applications. This is still an open problem, and it has limited the effective use of supercomputers for these computationally-intensive AI/ML tasks. While one could naively use a supercomputer and perform extensive hyper-parameter sweeps, this is not possible for many large-scale tasks, it is not an efficient use of computational resources, and it often undermines the goal of scientific insight. For example, in our prior work [18, 19] we showed that using large batch size training with SGD leads to diminishing returns. Sample results are shown in Fig. 2, where we show the achieved speed up by increasing batch size for reaching a testing loss threshold. It can be clearly seen that using larger batches does not lead to speed up for a fixed testing loss target. In fact, it has been observed that large batch size training with SGD requires more iterations [20] to recover accuracy which limits the overall speedups. That is even though larger batches can be parallelized more efficiently from a systems perspective, but

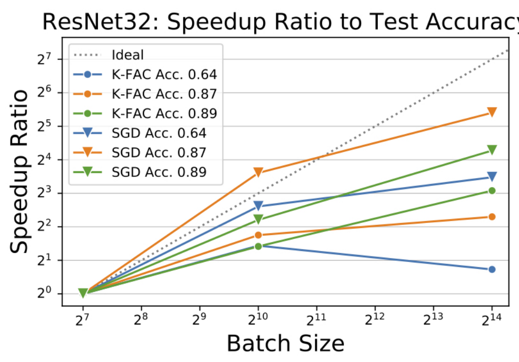

Fig. 2. Speed up to a target testing accuracy versus batch size is shown for both SGD and K-FAC for ResNet32 trained on CIFAR-10. The diminishing returns effect can be seen for both K-FAC (circles) and SGD (triangles). We can clearly see that using larger batches leads to significantly lower speedups as compared to ideal line. For more details please see [19].

more work/iterations is needed to reach a target accuracy.

The main source for many of these problems stems from the use of the first-order Stochastic Gradient Descent (SGD) algorithm in training DNNs. While SGD has several well-known benefits [21], it is also known to be very sensitive to hyper-parameters such as step size (a.k.a. learning rate), initialization, and momentum. These parameters vary widely from one DNN model to another, sometimes by orders of magnitude. This extreme sensitivity to tuning is exacerbated when performing training with large batches [19]. Therefore, one has to execute many (thousands of) tests to determine the right hyper-parameter values. For example, one of the problems with SGD training is that it uses the same step size for all of the parameters, irrespective of curvature (Hessian) information. Ideally, we want to use larger step sizes for parameters that have small curvature, and vice versa. This is illustrated in Fig. 3, where we show the top Hessian eigenvalue for different layers of Inception-V3 trained on ImageNet dataset. One can clearly see that there is an order of magnitude difference in the associated curvature information. For example, the loss landscape of the last layer of Inception-V3 has very small curvature, which means that a larger step size should be used for those parameters, as opposed to the second layer, which has three orders of magnitude larger curvature, and which thus needs a much smaller step size. Adaptive variants of SGD have been proposed to address this (e.g., AdaGrad and ADAM), but they work only somewhat reliably, and they only work for particular problems.

To address these problems, a multi-faceted approach is needed that can encapsulate the full stack of designing, training, and executing the DNN model on a target hardware platform. All of these stages are interconnected, and focusing only on one area will lead to sub-optimal solutions. This requires designing mechanisms to better understand DNN training and bridging the gap between (i) theoretical results for optimization, (ii) second order methods, and (iii) high performance computing.

Along the first direction, we need to develop a more practical theory for training NNs to enable large scale

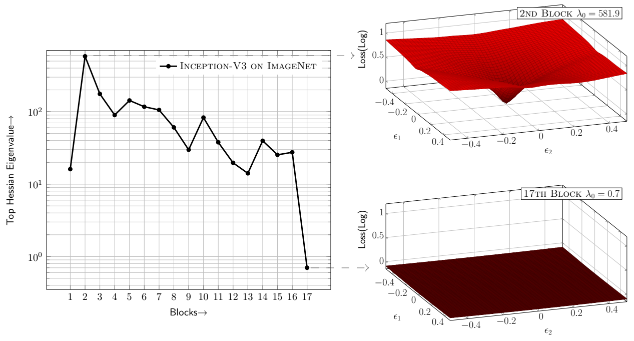

Fig. 3. Top eigenvalue of each individual block of pre-trained Inception-V3 on ImageNet. Note that the magnitudes of eigenvalues of different blocks varies by orders of magnitude [30].

training of NNs with more robust and interpretable methods. For example, our recent work has developed new theoretical results in the use of higher order optimization methods and in particular their strength and weaknesses as compared to SGD [22–26]. Along the second direction, we have developed the PyHessian framework, which is an open source library that enables fast computation of Hessian spectrum. This includes the top eigenvalue, trace, and even the full Eigenvalue Spectral Density of the Hessian for DNNs [11, 27]. This has lead to new insights in large scale training of DNNs [11, 28], adversarial robustness [29], and development of a novel Hessian AWare Quantization framework [30–33]. The latter has become the state-of-the-art for compressing DNN models through quantization.

Along the third direction, we have developed new methods to scale training by using the so-called integrated parallelism which is based on communication-avoiding algorithms [9]. The algorithm enables distributing the computations by finding optimal partitioning of the data and model, and avoids the problems of large batch size training. Mesh TensorFlow library is a recent work from Google that has used this approach and implemented it in TensorFlow [34].

II. Conclusions

One promising solution to address the challenges associated with large scale training of DNN models is to incorporate recent advances in theory, second-order optimization, and high performance computing. Our recent work, has focused on developing the infrastructure required to address these challenges. In particular, we have developed PyHessian [27, 35] a novel library for second-order based analysis of DNN models, HAWQ [30–33, 36] a library for compressing DNN models for efficient inference at the edge, and integrated parallelism [37] a new algorithm which enables scaling training without changing hyper-parameters.

The next milestone in enabling efficient large scale training of DNN models can be achieved by encapsulating the full stack of training, designing, and executing the DNN model on a target hardware platform. These three phases are intricately related and only focusing on one aspect, will not lead to optimal solutions.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[2] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *In Review*, 2017.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[4] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 129–137.

[5] B. Naul, J. S. Bloom, F. Pérez, and S. van der Walt, "A recurrent neural network for classification of unevenly sampled variable stars," *Nature Astronomy*, vol. 2, no. 2, p. 151, 2018.

[6] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica *et al.*, "Exascale deep learning for climate analytics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. IEEE Press, 2018, p. 51.

[7] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle, "Brain tumor segmentation with deep neural networks," *Medical image analysis*, vol. 35, pp. 18–31, 2017.

[8] A. Mang, S. T. A. Gholami, N. Himthani, S. Subramanian, J. Levitt, M. Azmat, K. Scheufele, M. Mehl, C. Davatzikos, B. Barth, and G. Biros, "SIBIA-GlS: Scalable biophysics-based image analysis for glioma segmentation," *The multimodal brain tumor image segmentation benchmark (BRATS), MICCAI*, 2017.

[9] A. Gholami, A. Azad, P. Jin, K. Keutzer, and A. Buluc, "Integrated model, batch and domain parallelism in training neural networks," *ACM Symposium on Parallelism in Algorithms and Architectures(SPAA'18)*, 2018, [PDF].

[10] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[11] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney, "Hessian-based analysis of large batch training and robustness to adversaries," *Advances in Neural Information Processing Systems*, 2018.

[12] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[13] Y. You, I. Gitman, and B. Ginsburg, "Scaling sgd batch size to 32k for imagenet training," *arXiv preprint arXiv:1708.03888*, 2017.

[14] A. Devarakonda, M. Naumov, and M. Garland, "Adabatch: Adaptive batch sizes for training deep neural networks," *arXiv preprint arXiv:1712.02029*, 2017.

[15] S. L. Smith, P.-J. Kindermans, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv preprint arXiv:1711.00489*, 2017.

[16] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu *et al.*, "Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes," *arXiv preprint arXiv:1807.11205*, 2018.

[17] Y. You, J. Hseu, C. Ying, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large-batch training for lstm and beyond," *arXiv preprint arXiv:1901.08256*, 2019.

[18] N. Golmant, N. Vemuri, Z. Yao, V. Feinberg, A. Gholami, K. Rothauge, M. W. Mahoney, and J. Gonzalez, "On the computational inefficiency of large batch sizes for stochastic gradient descent," *CoRR*, vol. abs/1811.12941, 2018. [Online]. Available: http://arxiv.org/abs/1811.12941

[19] L. Ma, G. Montague, J. Ye, Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "Inefficiency of k-fac for large batch size training," *AAAI'20 (arXiv:1903.06237)*, 2020.

[20] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1731–1741.

[21] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, 2010, pp. 177–187.

[22] P. Xu, F. Roosta-Khorasan, and M. W. Mahoney, "Second-order optimization for non-convex machine learning: An empirical study," *arXiv preprint arXiv:1708.07827*, 2017.

[23] P. Xu, F. Roosta-Khorasani, and M. W. Mahoney, "Newton-type methods for non-convex optimization under inexact hessian information," *arXiv preprint arXiv:1708.07164*, 2017.

[24] F. Roosta-Khorasani and M. W. Mahoney, "Sub-sampled newton methods i: globally convergent algorithms," *arXiv preprint arXiv:1601.04737*, 2016.

[25] ——, "Sub-sampled newton methods ii: Local convergence rates," *arXiv preprint arXiv:1601.04738*, 2016.

[26] F. Roosta, Y. Liu, P. Xu, and M. W. Mahoney, "Newton-mr: Newton's method without smoothness or convexity," *arXiv preprint arXiv:1810.00303*, 2018.

[27] (2019, Sep.) https://github.com/amirgholami/pyhessian.git.

[28] Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney, "Large batch size training of neural networks with adversarial training and second-order information," *arXiv preprint arXiv:1810.01021*, 2018.

[29] Z. Yao, A. Gholami, P. Xu, K. Keutzer, and M. Mahoney, "Trust region based adversarial attack on neural networks," *Computer Vision and Pattern Recognition (CVPR'19)*, 2018.

[30] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," *Accepted in International Conference on Computer Vision (ICCV) preprint arXiv:1905.03696*, 2019.

[31] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," *Accepted in AAAI-20 (arXiv:1909.05840)*, 2019.

[32] Z. Dong, Z. Yao, D. Arfeen, Y. Cai, A. Gholami, M. Mahoney, and K. Keutzer, "Trace weighted hessian-aware quantization," *NeurIPS'19 workshop on Beyond First-Order Optimization Methods in Machine Learning*, 2019.

[33] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "Zeroq: A novel zero shot quantization framework," *arXiv preprint arXiv:2001.00281*, 2020.

[34] N. Shazeer, Y. Cheng, N. Parmar, D. Tran, A. Vaswani, P. Koanantakool, P. Hawkins, H. Lee, M. Hong, C. Young *et al.*, "Mesh-tensorflow: Deep learning for supercomputers," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 414–10 423.

[35] Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney, "PyHessian: Neural Networks through the lens of the Hessian," *under review*, 2019.

[36] (2020, Jan.) https://github.com/amirgholami/zeroq.git.

[37] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "Squeezenext: Hardware-aware neural network design," *Workshop paper in CVPR*, 2018.