Minibatching Offers Improved Performance for Second Order Optimizers

Eric Silk [1] Swarnita Chakraborty [2] Dasgupta Nairanjana [2] Andrew Lumsdaine [3] Tony Chiang [3,4]

[1]University of Washington/University of Illinois Urbana-Champaign [2]Washington State University [3]Pacific Northwest National Laboratory [4]University of Washington Dept. of Mathematics

Abstract

Training deep neural networks (DNNs) used in modern machine learning is computationally expensive. Large datasets, therefore, rely on stochastic, first order methods for training; however, these often require significant tuning by hand to produce good performance. We empirically study the variability of different stochastic algorithms, including second-order methods, by treating their performance as an experiment and retraining the same architecture many times. Using 2-factor Analysis of Variance (ANOVA) with interactions, we show that batch size has a statistically significant effect on the peak accuracy of the methods, and that the full batch was largely the worst of them. Additionally, second order optimizers exhibited significantly lower variance, implying they may require less hyperparameter tuning, leading to a reduced overall training time.

Motivation

- Training models in Machine Learning is time consuming and expensive
- Traditional Optimization techniques have not seen significant penetration into this field
- In particular, Second Order Optimizers (i.e. incorporating curvature information) have been largely written off as too expensive
- Results in Machine Learning often present “best case” results without full empirical characterization
- We desired an empirically sound investigation into the convergence behaviors of two second order methods and to more fully characterize them and determine if their benefits outweigh their cost
- Additionally, we desired an empirically sound investigation into the effects of minibatching on gradient-based optimizers

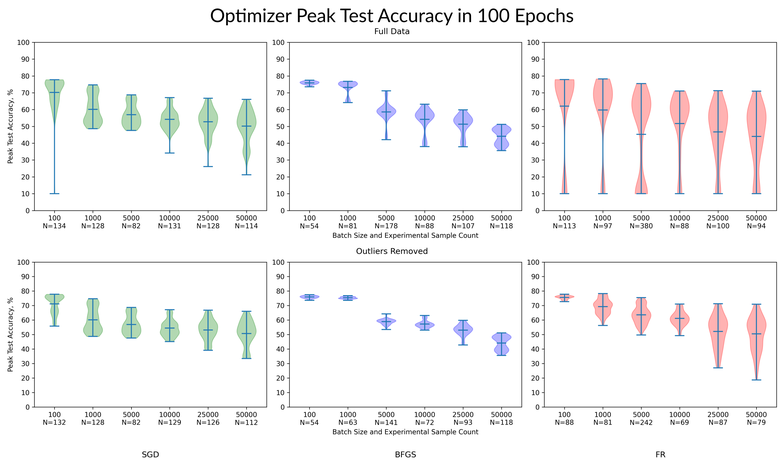
Method

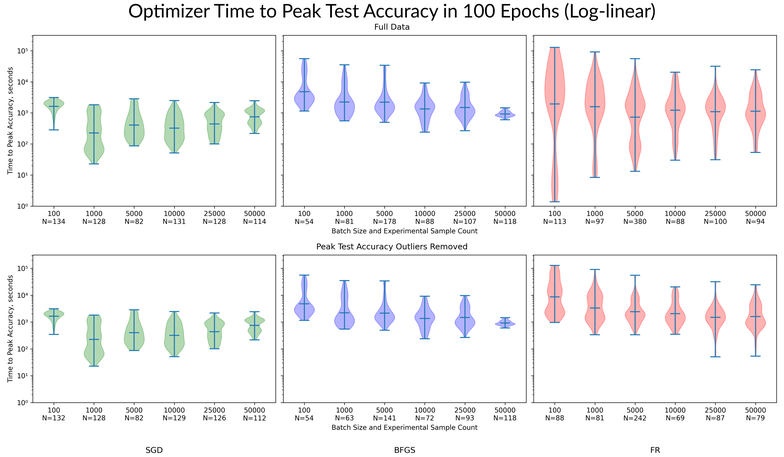
- We developed a library of Second Order Optimizers (SOOs) in PyTorch including nonlinear Conjugate Gradient (NLCG) and Hessian-Free Quasi-Newton (HFQN) methods
- We trained a ResNet-18 model on CIFAR10 to balance the need for a large number of trials with a more complex model
- We contrasted (Stochastic) Gradient Descent (SGD) with Fletcher-Reeves (FR) NLCG method and Limited Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) HFQN method
- To mimic a real-world hyperparameter tuning process, a range of reasonable hyperparameters (e.g. SGD’s learning rate was set to values well under 1, the maximum number of line searches allowed per step was kept to 10 or less) for each optimizer was selected and a randomized grid search was used to select the hyperparameters per training session
- Each trial consisted of training the model with a random initialization for 100 epochs
- Hundreds of trials were conducted for each optimizer over the course of several weeks on PNNL’s Marianas GPU cluster
- A two-factor ANOVA test was conducted to assess the relationship between optimizer, batch sizes, and peak test accuracy in a training session

System Information

- Nvidia DGX-2 GPU AI Server
- 16 Nvidia Tesla V100 GPU’s, 1 per Training Session
- Dual Intel Xeon Platinum 8168, 2.7GHz, 24-core

Results





ANOVA Table with Outlier Treatment

Source	DF	Type III SS	Mean Square	F Value	PR>F
Batch Size	5	35.92	7.18	480.66	< 0.001
Optimizer	2	1.37	0.684	45.78	< 0.001
Interaction	10	4.11	0.41	27.51	< 0.001

Results, continued

Pairwise Tests for Interaction Effects with Outlier Treatment

Batch Size	Treatment	LS Means	Letter Grouping
100	L-BFGS	4.3292	A
100	FR	4.3256	A
100	SGD	4.2597	A
1000	L-BFGS	4.3222	A
1000	FR	4.2354	B
1000	SGD	4.0879	C
5000	FR	4.1471	A
5000	L-BFGS	4.0456	B
5000	SGD	4.0358	B
10000	FR	4.2354	A
10000	L-BFGS	4.0456	A
10000	SGD	3.9925	B
25000	L-BFGS	3.9687	A
25000	SGD	3.9644	A
25000	FR	3.9288	A
50000	SGD	3.9089	A
50000	FR	3.8894	A
50000	L-BFGS	3.782	B

Open Source Implementation

All source code is available at: github.com/pnnl/pytorch_soo
Please feel free to use, cite, and open PR’s!

Conclusion

We have provided **strong, empirical** evidence for the often discussed effect of “implicit regularization” afforded by minibatching: decreased batch size produces increased peak accuracy. This effect **generalizes across optimizers**. This implies that acceleration via increased hardware utilization through larger batch sizes is likely not a viable research direction under the current paradigm. SOOs demonstrate **improved convergence results in peak accuracy**, both through **higher mean and lower variance**. While they exhibit slower wall-clock performance, this insensitivity to hyperparameters suggests that **hyperoptimization efforts may not be needed**, providing **amortized time savings**. This reduces cost, complexity, and increases confidence in the results of a training session.

Aknowledgements

The authors wish to thank PNNL for providing the funding and compute resources to make this scale of a study possible.

A special thanks to Dr. Tony Chiang, Dr. Andrew Lumsdaine, and Dr. Nairanjana Dasgupta for their unrelenting support on this project.

esilk2@illinois.edu