FUNDAMENTALS OF DEEP LEARNING FOR MULTI-GPUS

LAB 3, PART 2: OPTIMIZATION STRATEGIES

NVIDIA | DEEP LEARNING INSTITUTE

WHAT CAN WE DO TO IMPROVE THE OPTIMIZATION PROCESS?

- Manipulate the learning rate?
- Add noise to the gradient?
- Manipulate the batch size?
- Change the learning algorithm?

WHAT CAN WE DO ABOUT IT?

Early approaches: scaling the learning rate

"Theory suggests that when multiplying the batch size by k, one should multiply the learning rate by √(k) to keep the variance in the gradient expectation constant.

$$\mathrm{cov}\left(\Delta\mathbf{w}, \Delta\mathbf{w}\right) \approx \frac{\eta^2}{M}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{g}_n\mathbf{g}_n^\top\right) \longrightarrow \eta \propto \sqrt{M}$$

…

Theory aside, for the batch sizes considered in this note, the heuristic that I found to work the best was to multiply the learning rate by k when multiplying the batch size by k. I can't explain this discrepancy between theory and practice."

In practice linear scaling is still frequently used.

Krizhevsky, A. (2014). One weird trick for parallelizing convolutional neural networks. arXiv:1404.5997

WHAT CAN WE DO ABOUT IT?

Warmup strategies

- A lot of networks will diverge early in the learning process
- Warmup strategies address this challenge

Gradual warmup. We present an alternative warmup that *gradually* ramps up the learning rate from a small to a large value. This ramp avoids a sudden increase of the learning rate, allowing healthy convergence at the start of training. In practice, with a large minibatch of size kn, we start from a learning rate of η and increment it by a constant amount at each iteration such that it reaches $\hat{\eta} = k\eta$ after 5 epochs (results are robust to the exact duration of warmup). After the warmup, we go back to the original learning rate schedule.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., ... & He, K. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:1706.02677

WHAT CAN WE DO ABOUT IT?

Batch Normalization

Batch normalization improves the learning process by minimizing drift in the distribution of inputs to a layer

It allows higher learning rates and reduces the need to use dropout

The idea is to normalize the inputs to all layers in every batch (this is more sophisticated than simply normalizing the input dataset)

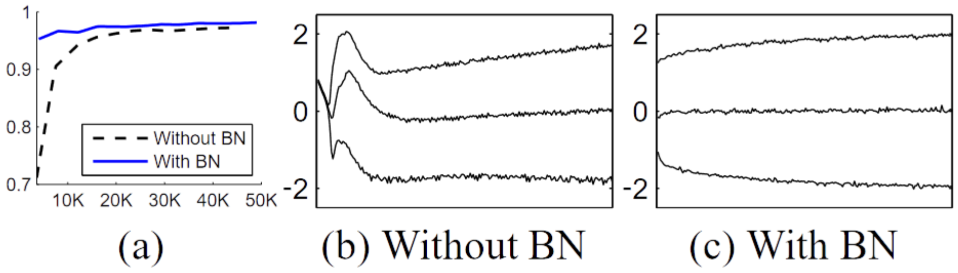


Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

Ioffe and Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167

WHAT CAN WE DO ABOUT IT?

Ghost Batch Normalization

- The original batch normalization paper suggests using the statistics for the entire batch, but what should that mean when we have multiple GPUs?
- We can introduce additional noise by calculating smaller batch statistics ("ghost batches").
- Batch normalization is thus carried out in isolation on a per-GPU basis.

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

Adding noise to the gradient

- Keeps the covariance constant with changing batch size (as $\sigma^2 \propto M$)
- Does not change the mean

Furthermore, we can match both the first and second order statistics by adding multiplicative noise to the gradient estimate as follows:

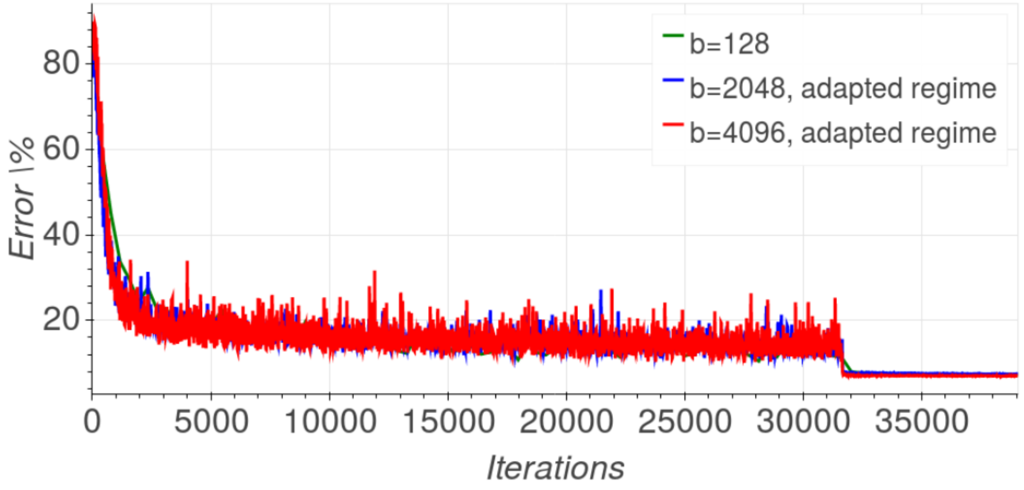
$$\hat{\mathbf{g}} = \frac{1}{M} \sum_{n \in B}^{N} \mathbf{g}_n z_n \,,$$

where $z_n \sim \mathcal{N}\left(1, \sigma^2\right)$ are independent random Gaussian variables for which $\sigma^2 \propto M$. This can be verified by using similar calculation as in appendix section A. This method keeps the covariance constant when we change the batch size, yet does not change the mean steps $\mathbb{E}\left[\Delta \mathbf{w}\right]$.

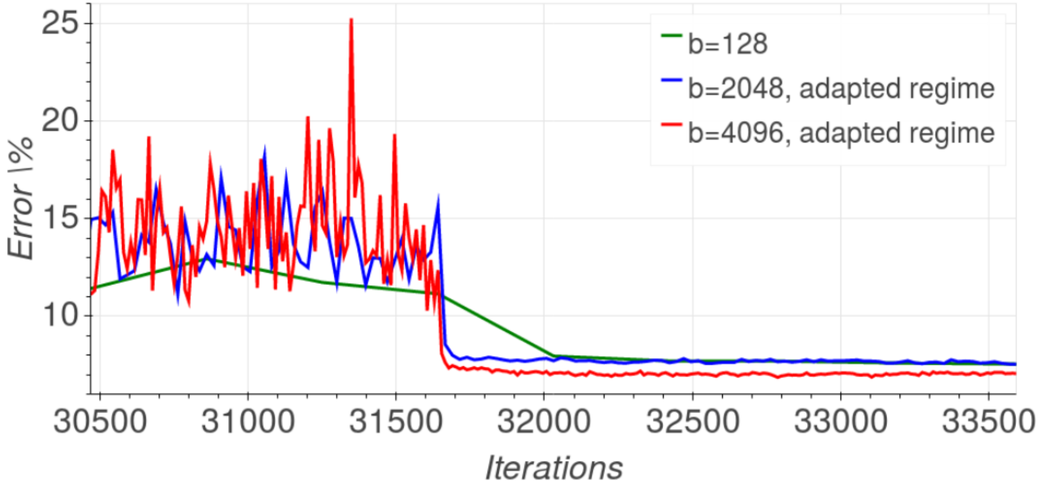
Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

Longer training with larger learning rate



(a) Validation error

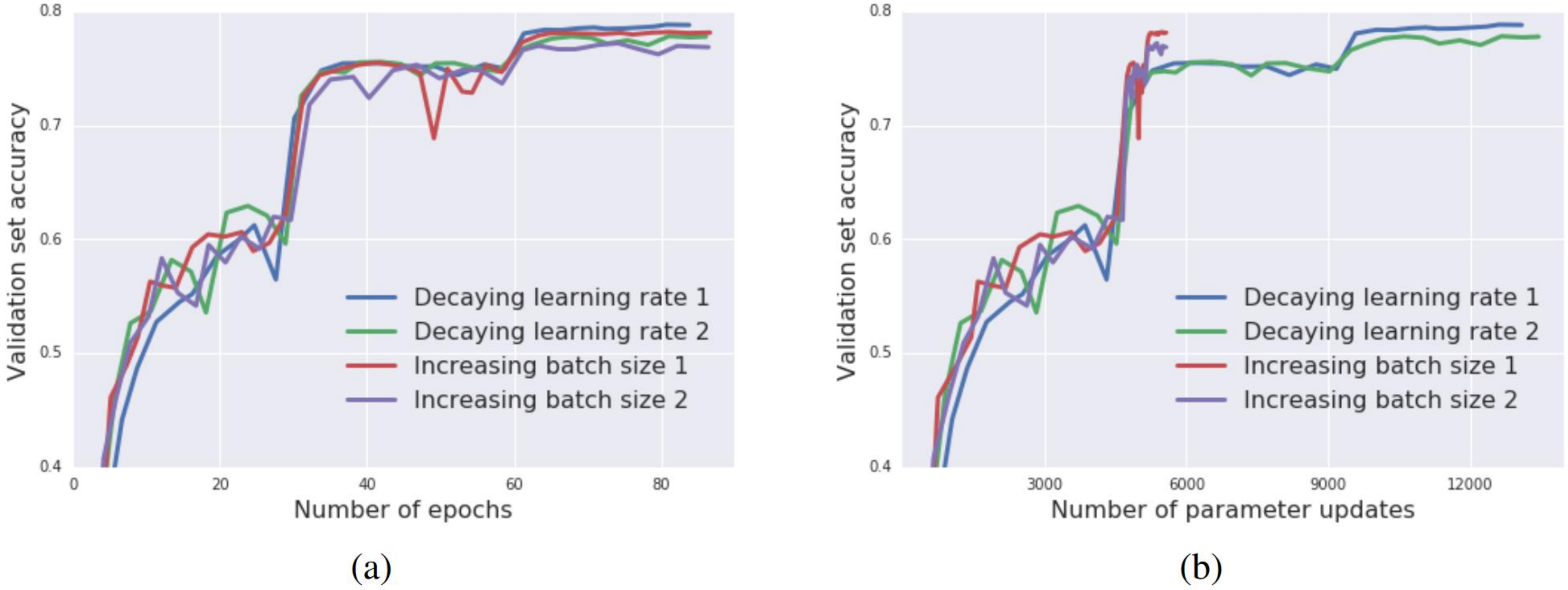


(b) Validation error - zoomed

Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv:1705.08741

WHAT CAN WE DO ABOUT IT?

Increasing the batch size, instead of learning rate decay



Smith, S. L., Kindermans, P. J., & Le, Q. V. (2017). Don't Decay the Learning Rate, Increase the Batch Size. arXiv:1711.00489

WHAT CAN WE DO ABOUT IT?

LARS: Layer-wise Adaptive Rate Scaling

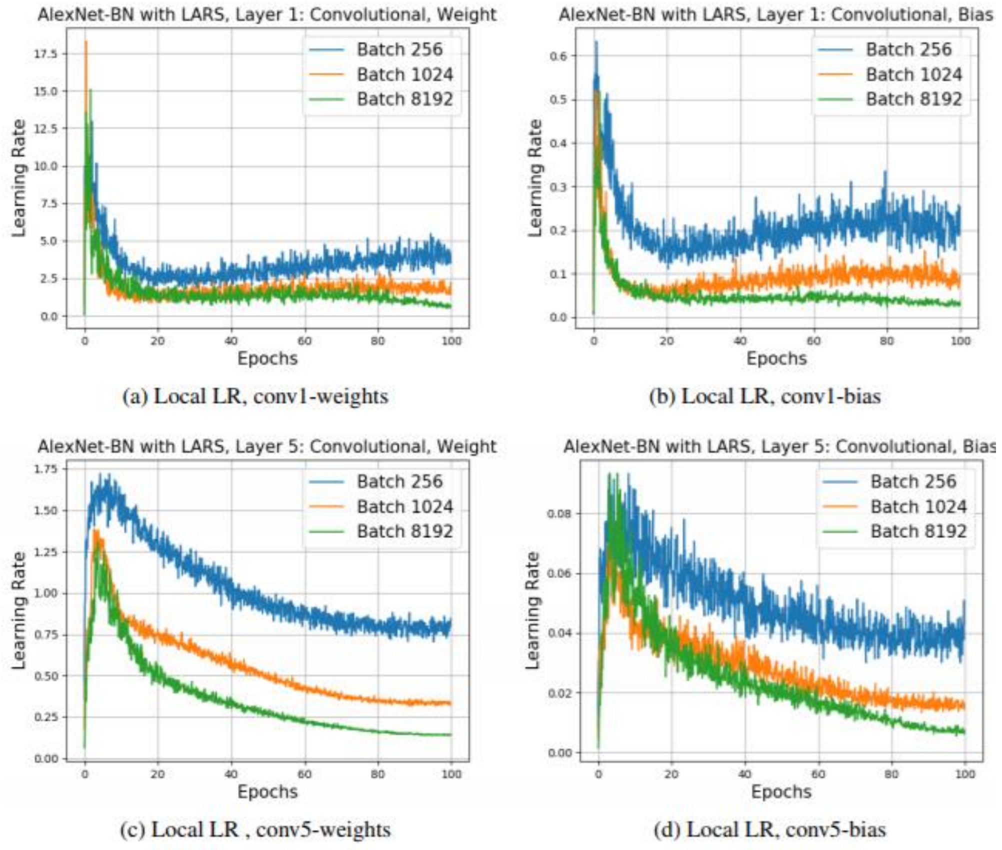
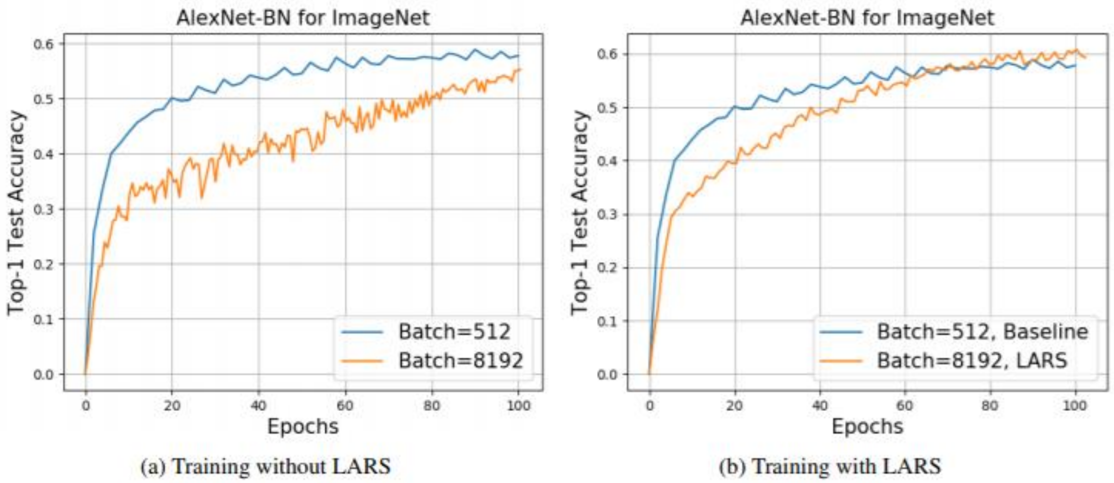


Figure 2: LARS: local LR for different layers and batch sizes



You, Y., Gitman, I., & Ginsburg, B. Large batch training of convolutional networks. [arXiv:1708.03888]

DEEP LEARNING INSTITUTE

WHAT CAN WE DO ABOUT IT?

LARS: Layer-wise Adaptive Rate Scaling

Control magnitude of the layer *k* update through local learning rate λ_k:

$$\Delta\, w_k(t+1) = \lambda_k * G_k(w(t))$$

where:

$G_k(w(t))$: stochastic gradient of L with respect to w_k,

λ_k: local learning rate for layer k, defined as

$$\lambda_k = min(\gamma,\ \eta \cdot \frac{||w_k(t)||_2}{||G_k(w(t))||_2})$$

where

η is trust coefficient (how much we trust stochastic gradient)

γ is global learning rate policy (steps, exponential decay, …)

You, Y., Gitman, I., & Ginsburg, B. Large batch training of convolutional networks. arXiv:1708.03888

WHAT CAN WE DO ABOUT IT?

LARC: Layer-wise learning rates with clipping; SGD with momentum is base optimizer

LAMB: Layer-wise learning rates; Adam as base optimizer

- More successful than LARC at language models like BERT

NovoGrad: Moving averages calculated on a per-layer basis

- Also useful in several different domains

NVIDIA

DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli