Large-Batch DNN Training

Yang You

youyang@cs.berkeley.edu

Computer Science Division of UC Berkeley

Summary

1. There are 2 key difficulties in large batch training:
   - optimization and generalization
2. Facebook (“ImageNet in 1 hour”) suggested 2 techniques
   - linear scaling and warm-up for learning rate
   - ResNet-50 scales to Batch 8K (**state-of-the-art**)
3. Facebook recipe doesn’t work on Alexnet for batch > 1K
   - 5% Top-1 accuracy loss for Batch 4K
4. Proposed Solutions
   - Optimize the networks based on BatchNorm
   - “Layer-wise Adaptive Rate Scaling”(LARS) algorithm
5. Results of ImageNet training
   - Alexnet with batch size 8K
   - Alexnet-BN with batch size 8K
   - ResNet-50 with batch size 32K

Benefits of Large-Batch Training

AlexNet

batch size	accuracy	8-GPU speed	8-GPU time
512	0.588	5797 img/sec	6h 9m 0s
4096	**0.584**	16373 img/sec	**2h 10m 52s**

AlexNet-BN

batch size	accuracy	8-GPU speed	8-GPU time
512	0.602	5771 img/sec	6h 10m 30s
4096	**0.604**	15379 img/sec	**2h 19m 24s**

Large Batch gives ~ 3x speed-up

(* time is for 100 epochs)

Outline

- **Introduction**
- Background
- Our Approaches
- Results and Discussions

Motivation for Large Batch training

- **Mini-Batch SGD (Stochastic Gradient Descent):**
  1. Take mini-batch with B data samples each iteration
  2. Compute gradients of weights based on B
  3. Update the weights: $W = W - \gamma \nabla W$
     - W : weights
     - ∇W : gradients
     - γ: learning rate

How to speed up Mini-Batch SGD?

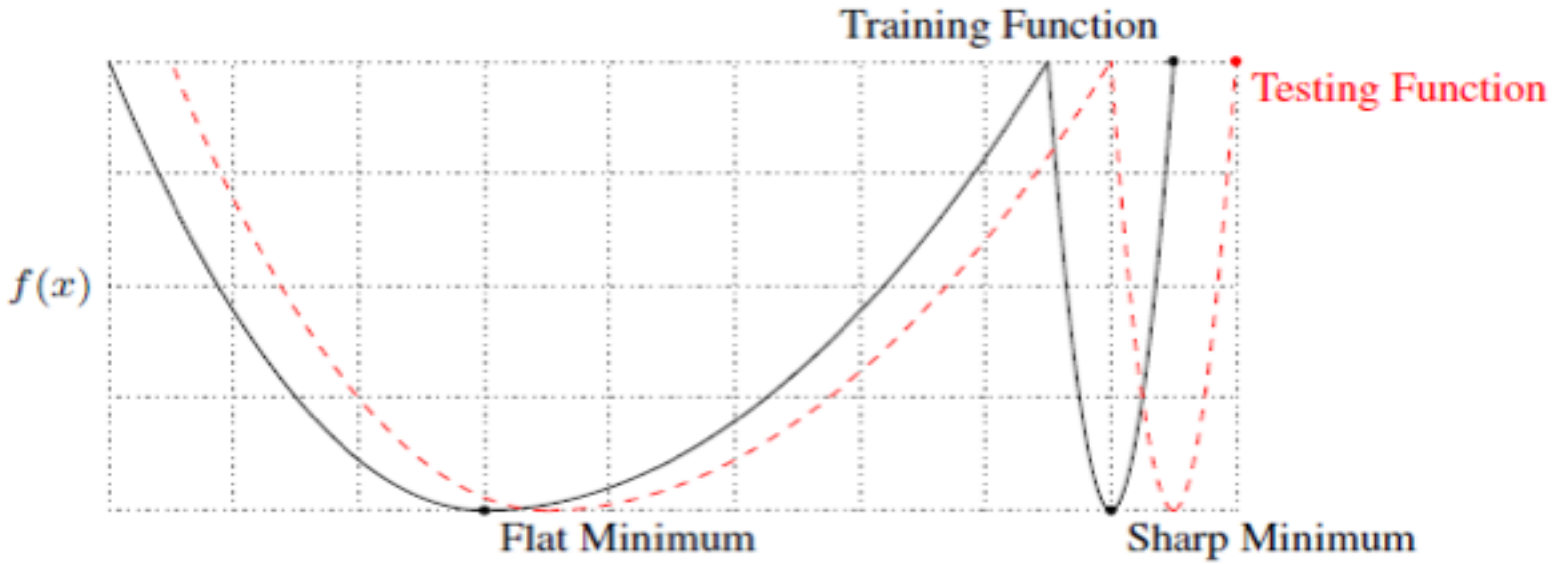
- Use more GPUs!
- Data parallelism: P GPUs - each GPU has B/P data samples
- Need large batch to utilize each GPU

Difficulties of Large-Batch Training

It’s difficult to keep the test accuracy, while increasing batch size:

1. **Generalization**
2. Optimization

CIFAR-10



“convergence to sharp minimizers gives rise to the poor generalization of large-batch methods for deep learning”

N. Keskar, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2017

Difficulties of Large-Batch Training

It's difficult to keep the test accuracy, while increasing batch size:

1. Generalization
2. **Optimization:**
   - a linear scaling of learning rate γ as a function of mini-batch size B
   - a learning rate "warm-up"

Resnet-50

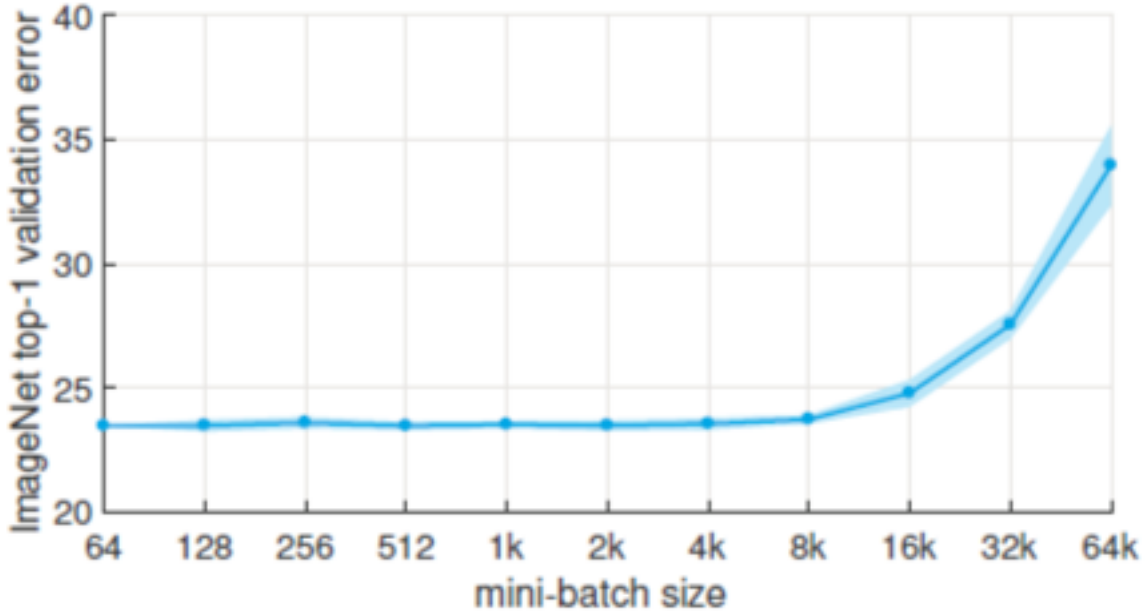


Figure 1. **ImageNet top-1 validation error *vs.* minibatch size.**

Optimization is not a problem if you get right hyper-parameters
Priya Goyal, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2017

Benchmarks

- ImageNet
  - AlexNet: Target - top1 = 58% (100 epochs)
  - ResNet-50*: Target - top1 = 73% (100 epochs)

*does not use data augmentation

Outline

- Introduction
- **Background**
- Our Approaches
- Results and Discussions

Learning Rate scaling for Large Batch training

- Learning rate policy wrt batch size:
  - If we increase B to kB, then increase learning rate by k
  - keep weight decay and momentum unchanged

Alexnet-OWT:

B=128: top1= 57.7%

B=1024: top1 = 56.7%

No results for B > 1024

Alex Krizhevsky One weird trick for parallelizing convolutional neural networks , 2014

Learning Rate scaling for Large Batch training

- Learning rate policy wrt batch size:
  - If we increase B to kB, then increase learning rate by k
  - keep weight decay and momentum unchanged
  - **Learning rate warmup: start from a small γ, increase γ in a few epochs**

ResNet-50

B = 256: top1 = 76.40%

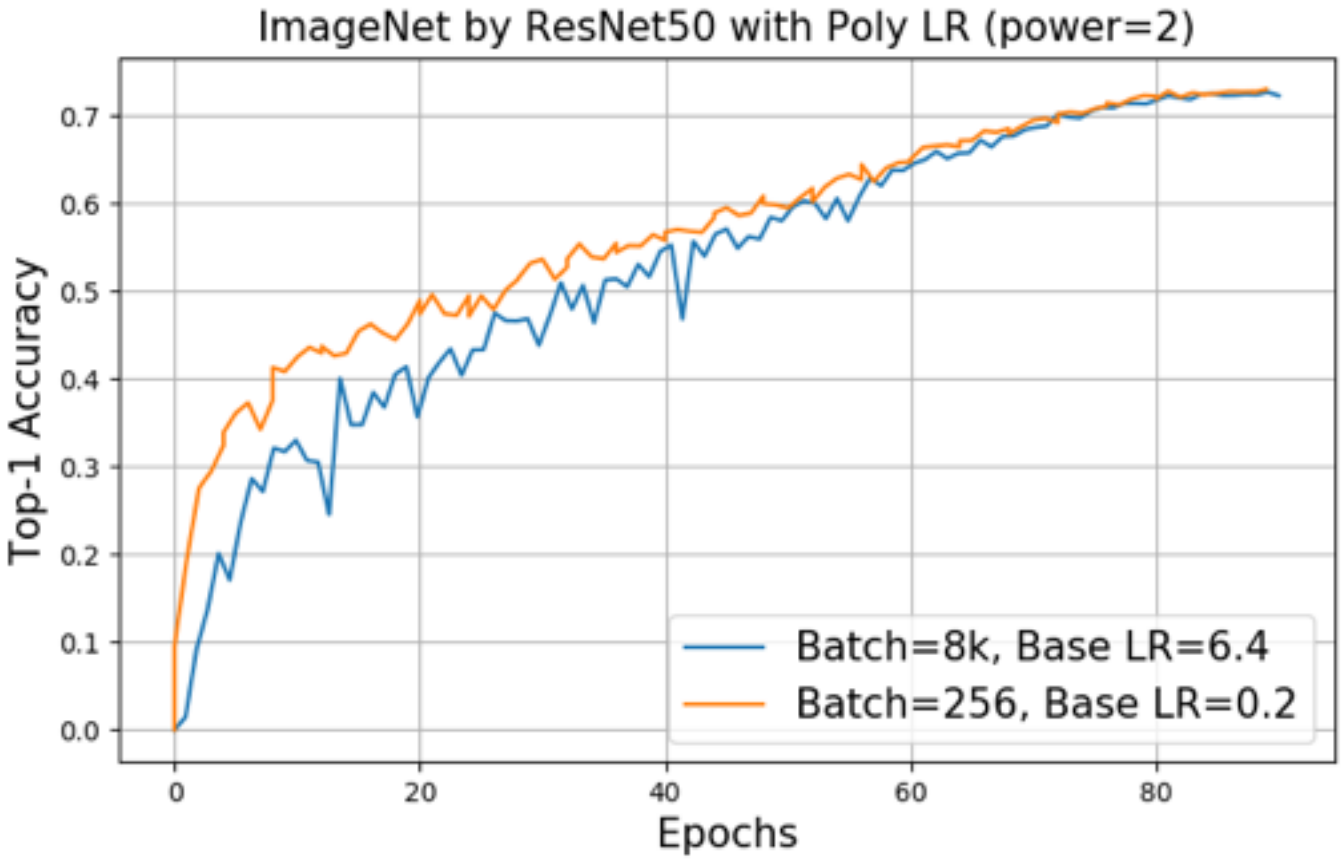
$B = 8192$: top1 = 76.26%

Priya Goyal, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2017:

Outline

- Introduction
- Background
- **Our Approaches**
- Results and Discussions

Reproduce Facebook's results



73% accuracy in 90 epochs for B = 256 and $B = 8192$
warmup for 5 epochs
Our baseline's accuracy is lower than Facebook's: we did not use data augmentation

AlexNet-4K

We applied FB recipe to train Alexnet with B=4K

The best result was for γ=0.05: 5% accuracy loss

batch size	learning rate	warmup	momentum	epochs	test accuracy
512	0.02	no	0.9	100	**58.8%**
1024	0.02	no	0.9	100	58.2%
4096	0.05	yes	0.9	100	**53.1%**

We also tuned learning rate, momentum, weight decay, min learning rate tuning, etc

Alexnet-4K

We couldn’t increase learning rate using Linear Scaling.
Warm-up did not helped. Net diverged.

learning rate	warmup	epochs	accuracy
0.01	Yes	100	0.509539
0.02	Yes	100	0.526504
0.03	Yes	100	0.520454
0.04	Yes	100	0.530026
0.05	Yes	100	0.531018
0.06	Yes	100	0.516101
0.07	Yes	100	0.001056
0.08	Yes	100	0.491709
0.09	Yes	100	0.001056

AlexnetBN: add Batch Normalization

Replaced Local Response Norm (LRN) layers with Batch Norm (BN).

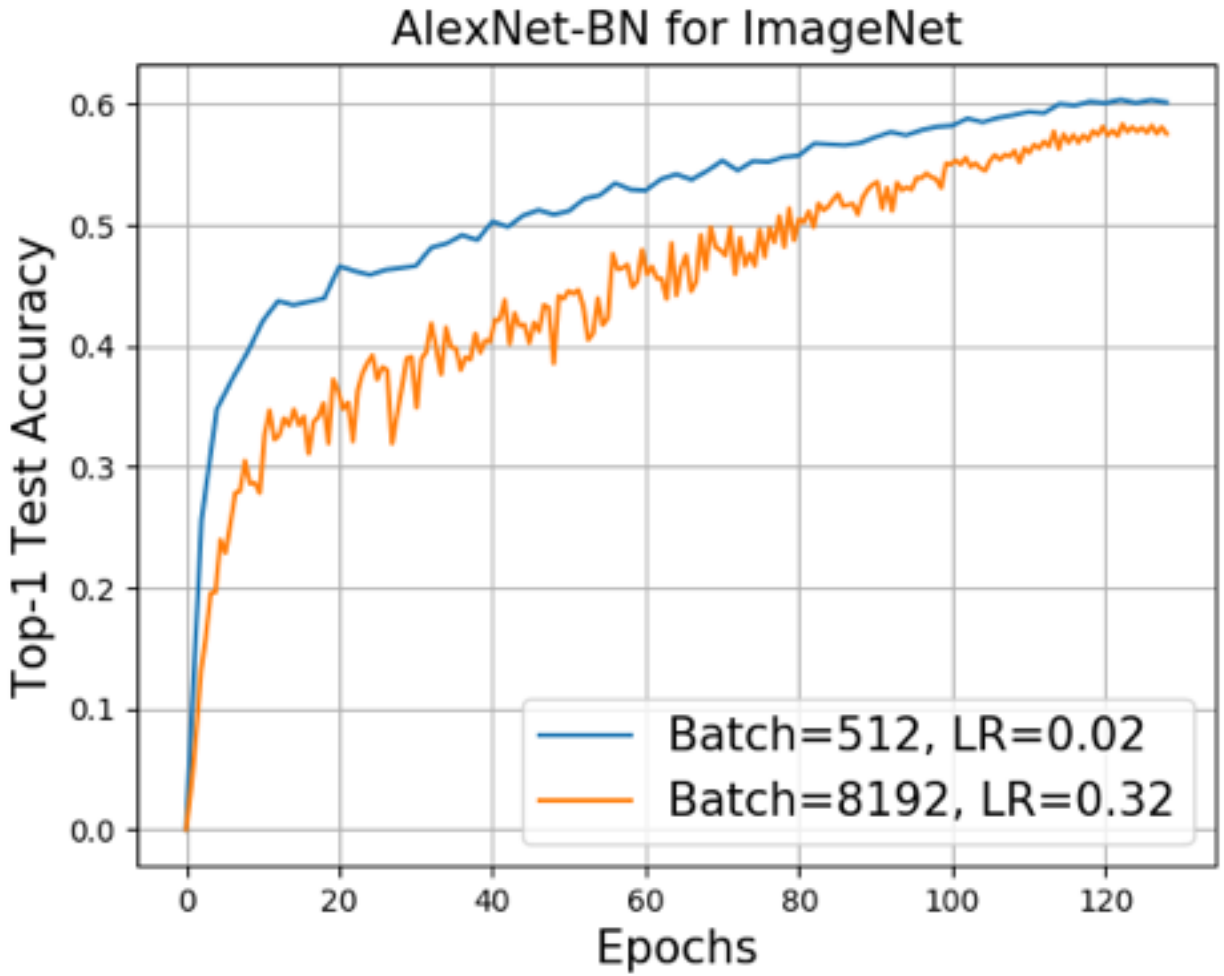
Now we can scale learning rate up!

Greatly improved the accuracy:

batch size	learning rate	Weight decay	Momentum	Epochs	test accuracy
512	0.02	0.0005	0.9	128	60.2%
4096	0.18	0.0005	0.9	128	58.9%
8192	0.30	0.0005	0.9	128	58.0%

- 58% achieved, but the baseline is also higher!
- Still needs to improve large-batch's accuracy

Alexnet-BN: accuracy gap



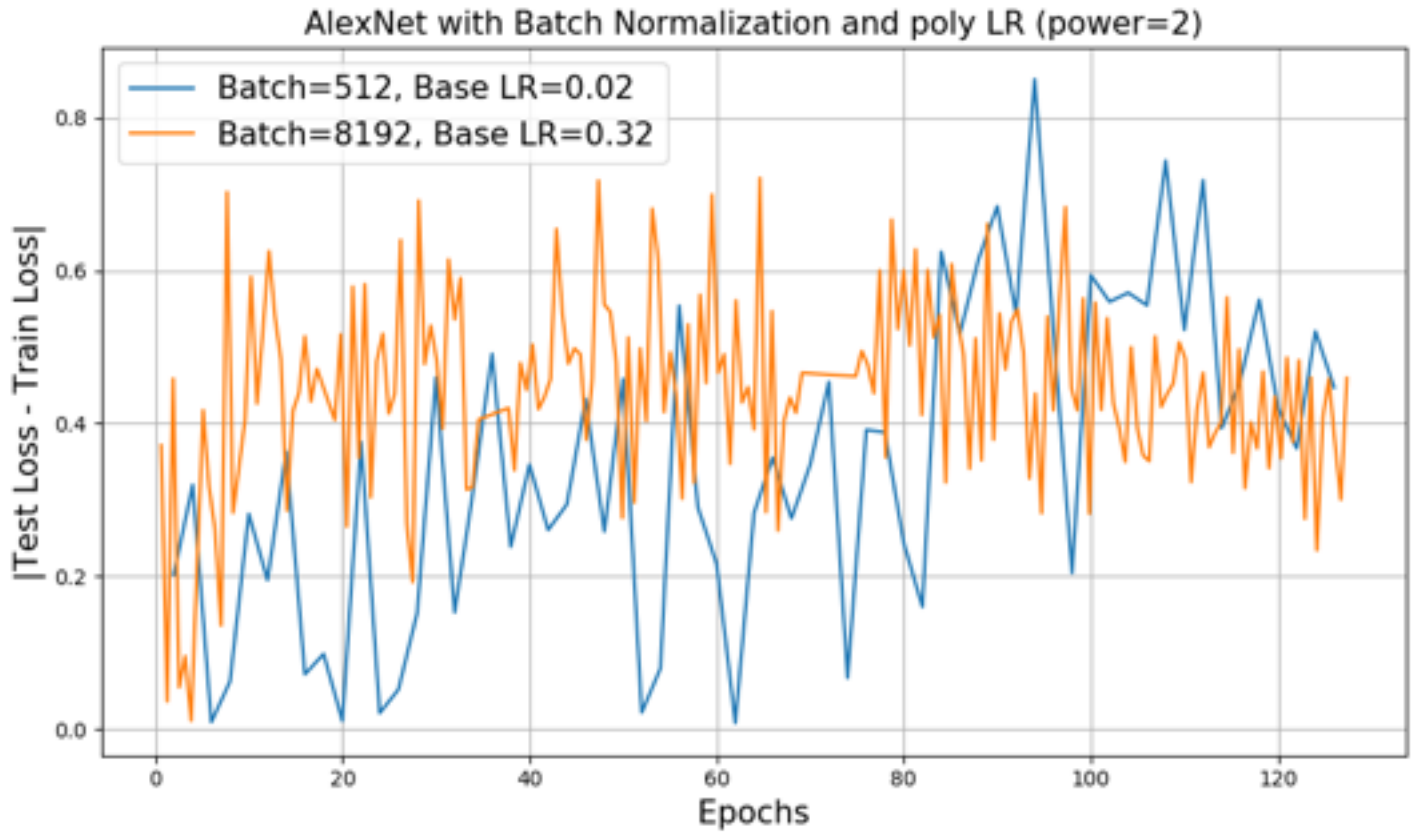
Increase epochs from 100 to 128

AlexNet-BN-8K: BN solved generalization problem

Batch Norm solved the generalization problem:

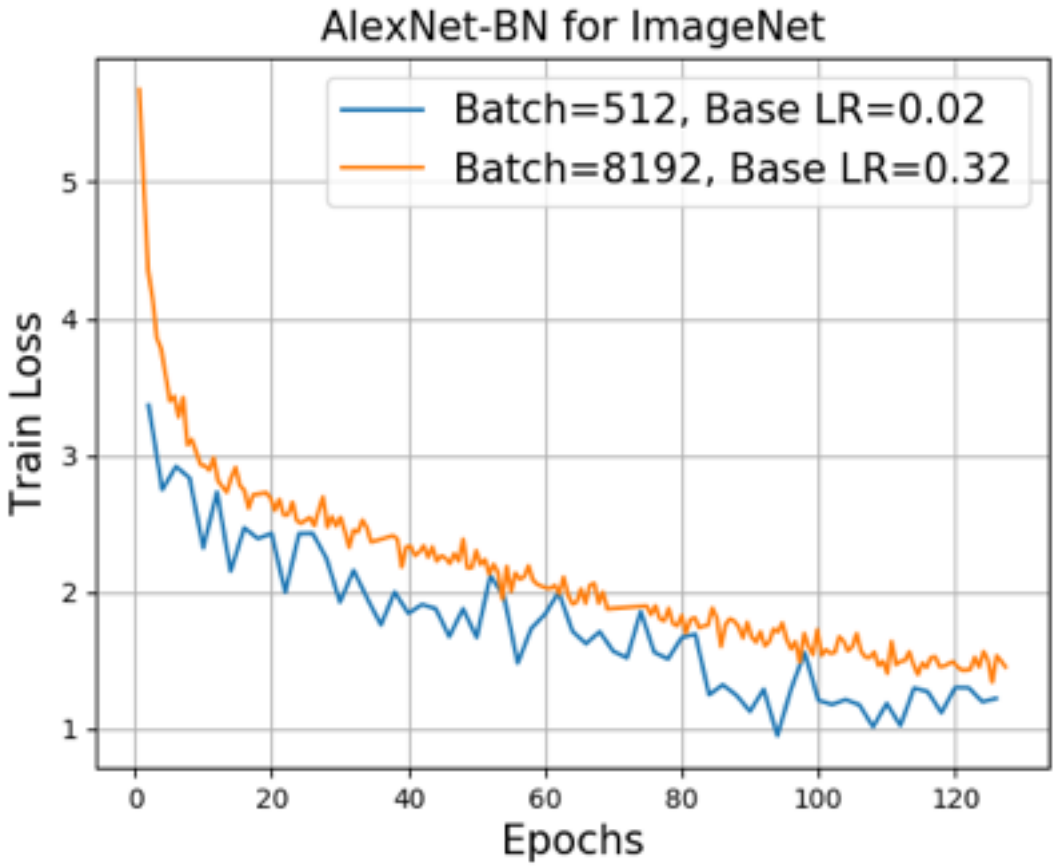
Regular batch: low train loss, low test loss

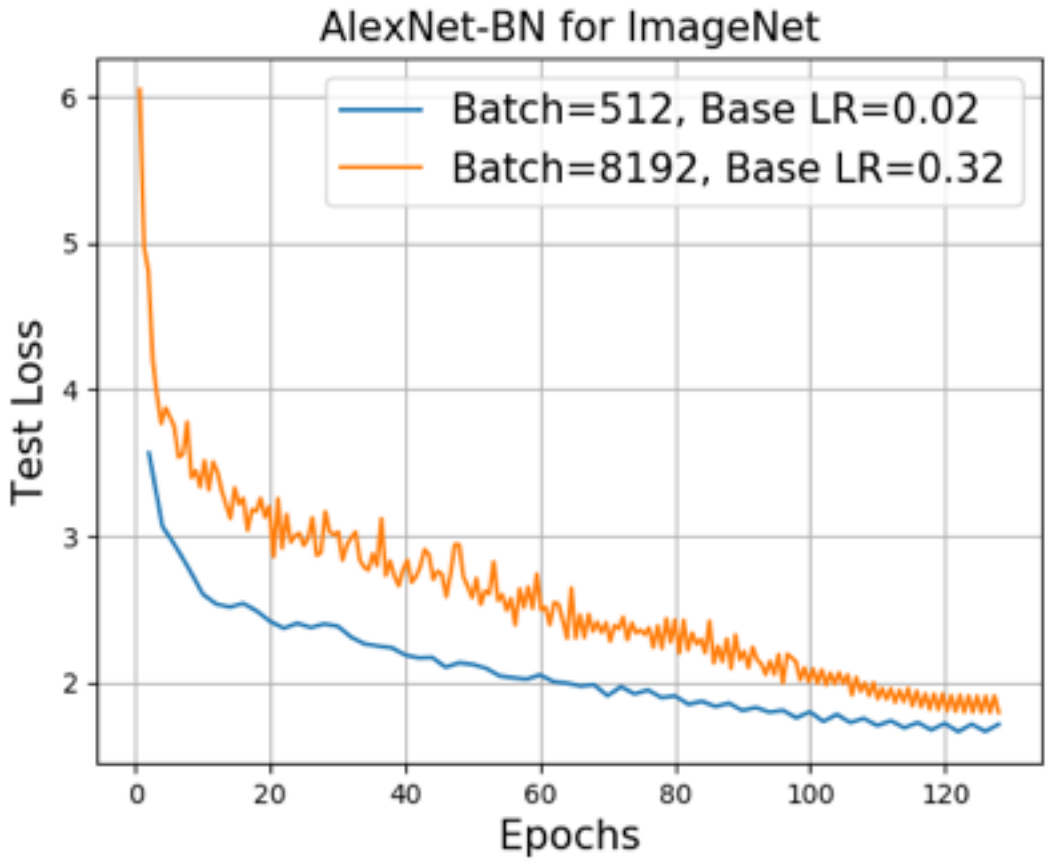
Large batch: low train loss, high test loss



AlexNet-BN-8K: optimization problem

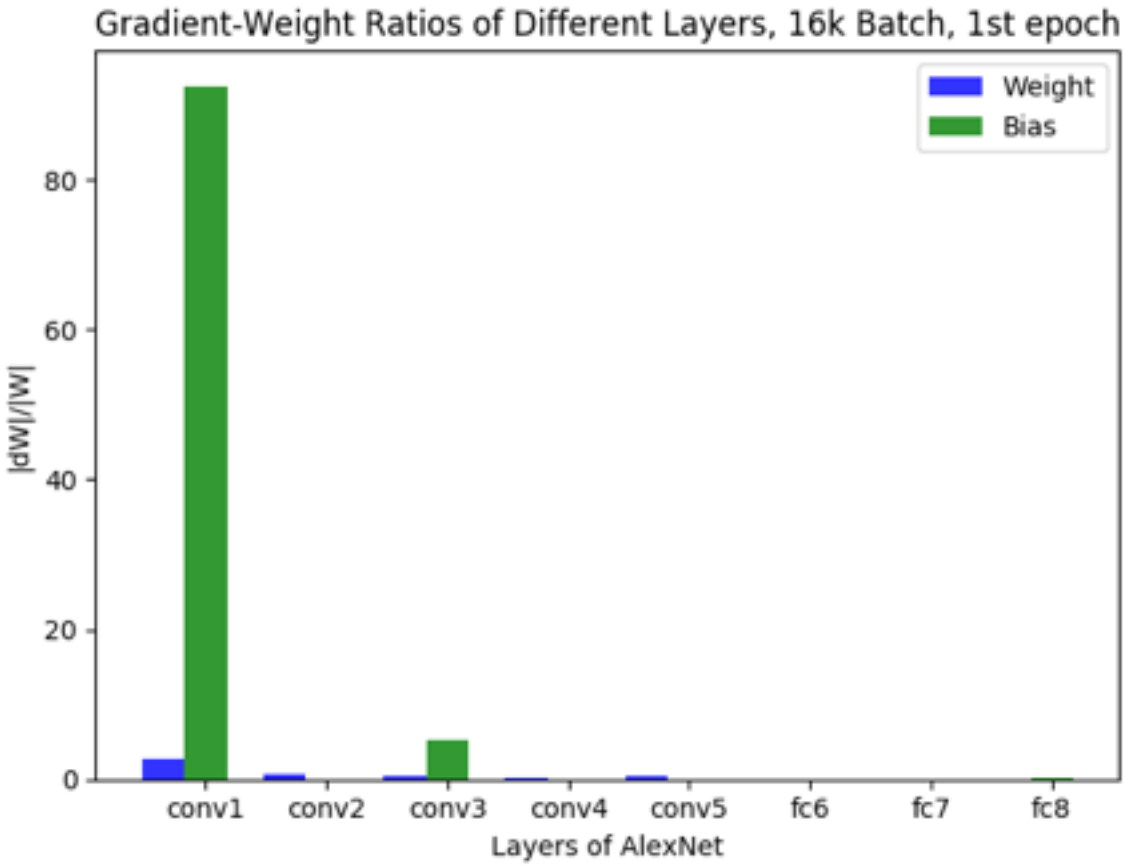
Need better optimization solution

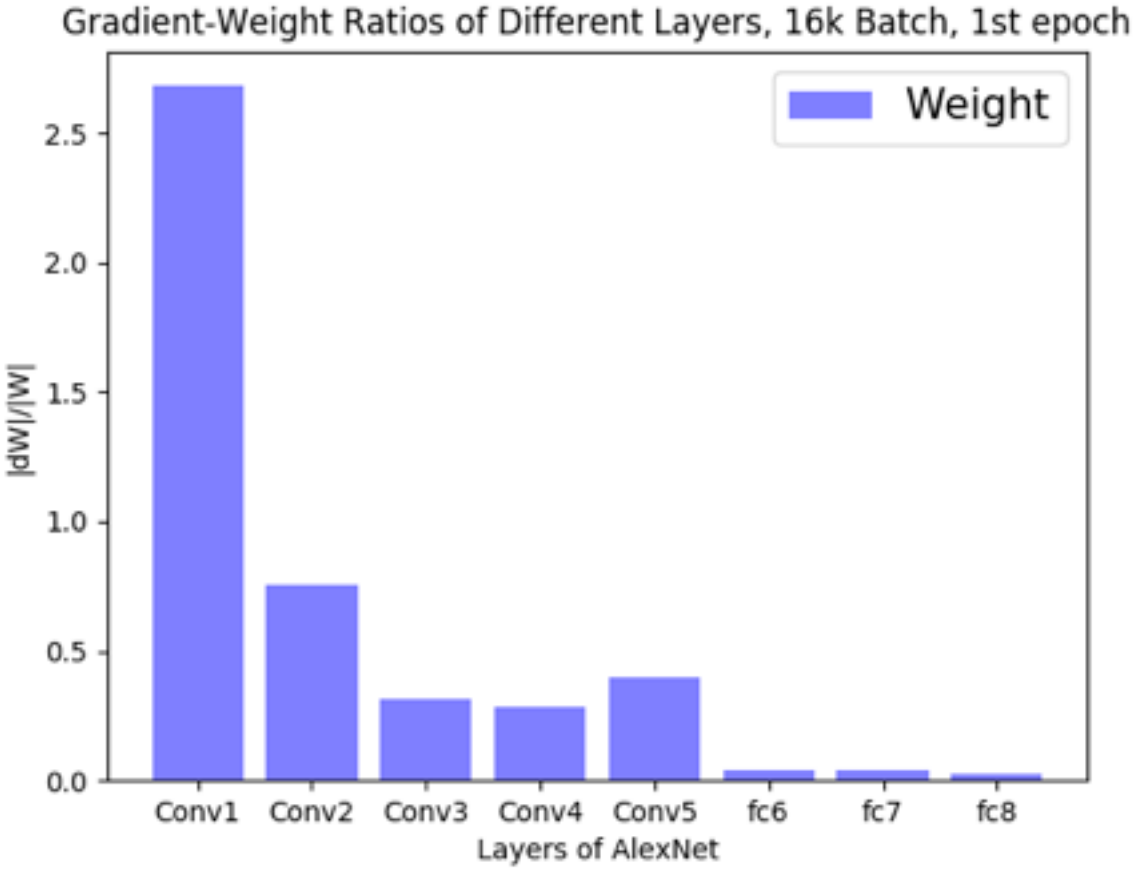




AlexNet: Optimization difficulties

- Let's compare $\frac{|\Delta W|}{|W|}$ for different layers in the beginning and the end training





Layer-wise Adaptive Rate Scaling (LARS)

- **Issues with one global learning rate**
  1. We would like to increase learning rate λ, but run into stability issues when $\left|\lambda * \frac{\partial L}{\partial W_l}\right|$ becomes the same order as $|W_l|$
  2. Maximal learning rate is controlled by weakest layer
  3. Should use ramp with small initial learning rate to overcome the dependency on weights initialization

Layer-wise Adaptive Rate Scaling (LARS)

$$\widehat{G_l} = \rho \frac{G_l}{|G_l|} * |W_l|,$$

where: $G_l = \frac{\partial L}{\partial W_l}$ - gradient of loss wrt layer weights w_l,

ρ - "trust ratio" (user defined parameter).

AlexNet: add LARS

AlexNet

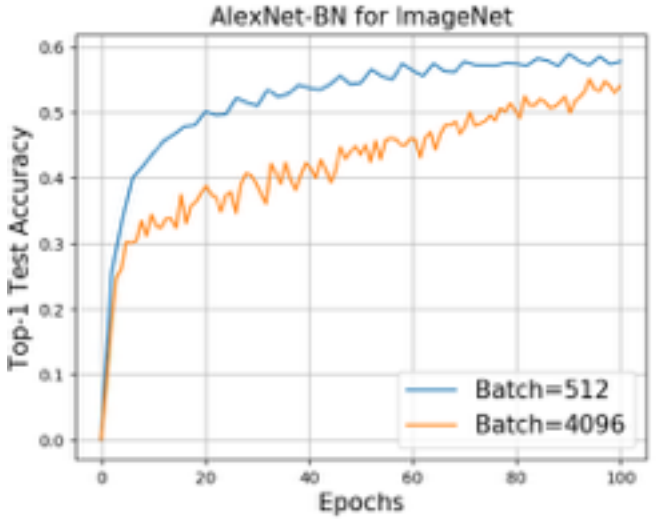
batch size	learning rate rule	learning rate	warmup	momentum	Epochs	accuracy
512	Regular	0.02	no	0.9	100	58.8%
4096	LARS	0.10	13 epochs	0.9	100	58.4%
8192	LARS	0.10	8 epochs	0.9	100	58.3%

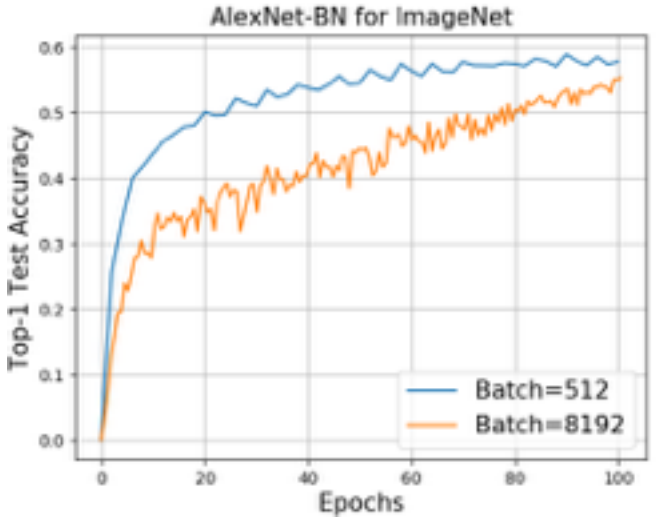
AlexNet-BN

batch size	learning rate rule	learning rate	warmup	momentum	Epochs	accuracy
512	LARS	0.05	2 epochs	0.9	100	60.2%
4096	LARS	0.10	2 epochs	0.9	100	60.4%
8192	LARS	0.14	2 epochs	0.9	100	60.1%

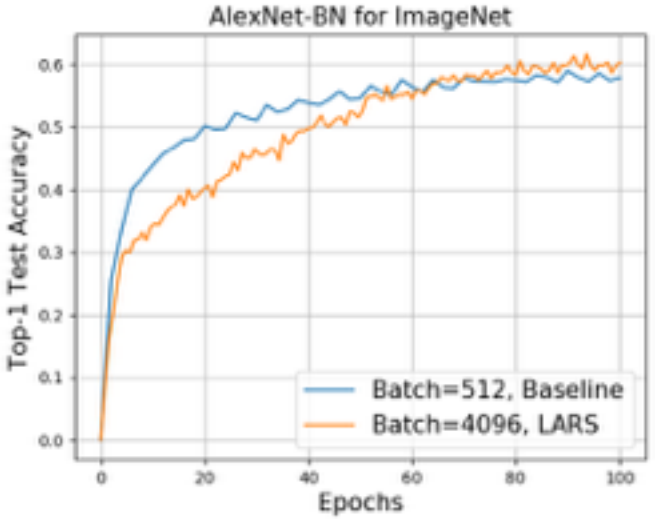
AlexNet-BN: Effects of LARS

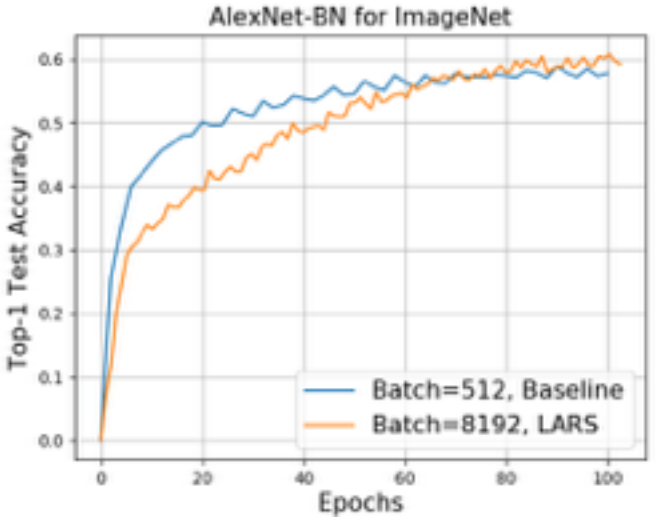
Baseline





LARS





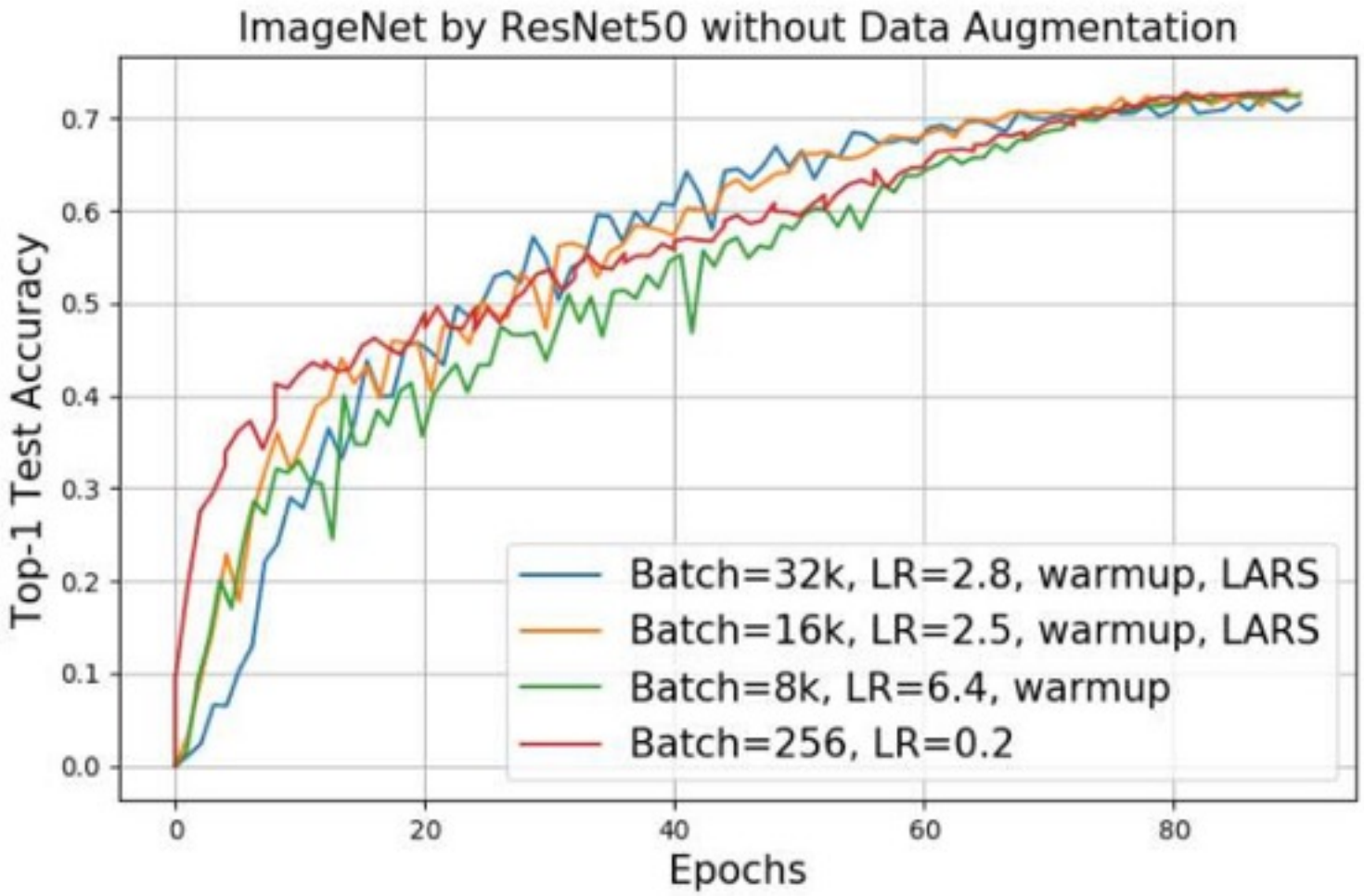
Outline

- Introduction
- Background
- Our Approaches
- **Results and Discussions**

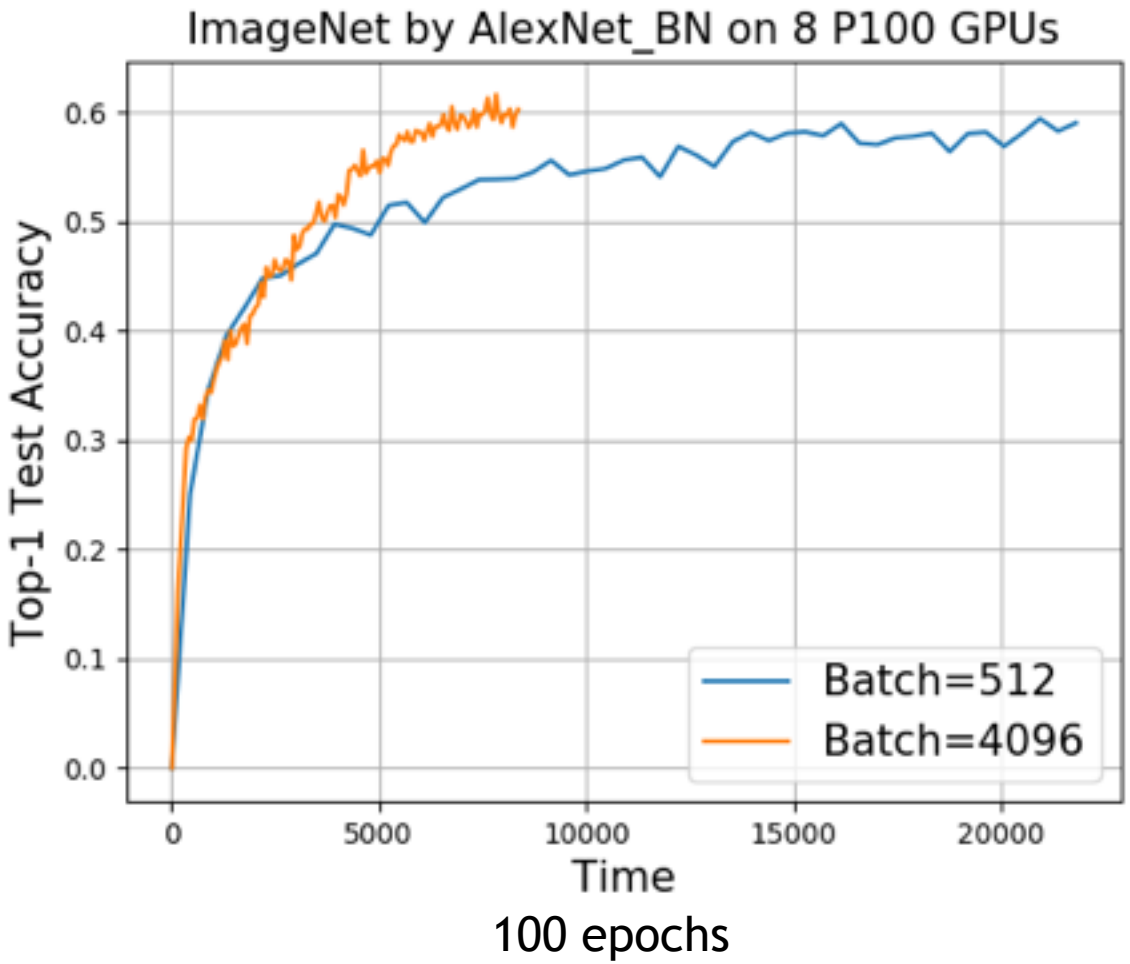
Summary

1. There are 2 key difficulties in large batch training:
   - optimization and generalization
2. Facebook (“ImageNet in 1 hour”) suggested 2 techniques
   - linear scaling and warm-up for learning rate
   - ResNet-50 scales to Batch 8K (**state-of-the-art**)
3. Facebook recipe doesn’t work on Alexnet for batch > 1K
   - 5% Top-1 accuracy loss for Batch 4K
4. Proposed Solutions
   - Optimize the networks based on BatchNorm
   - “Layer-wise Adaptive Rate Scaling”(LARS) algorithm
5. Results of ImageNet training
   - Alexnet with batch size 8K
   - Alexnet-BN with batch size 8K
   - ResNet-50 with batch size 32K

Scales to 32K, beats state-of-the-art



Benefits of Large-Batch Training



100 epochs

Large-Batch can make full use of the increased computational powers

Thanks!

- Scaling SGD Batch Size to 32K for ImageNet Training
- https://arxiv.org/abs/1708.03888