Scaling SGD Batch Size to 32K for ImageNet 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**

<table>
<thead>
<tr>
<th>batch size</th>
<th>accuracy</th>
<th>8-GPU speed</th>
<th>8-GPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.588</td>
<td>5797 img/sec</td>
<td>6h 9m 0s</td>
</tr>
<tr>
<td>4096</td>
<td>0.584</td>
<td>16373 img/sec</td>
<td>2h 10m 52s</td>
</tr>
</tbody>
</table>

**AlexNet-BN**

<table>
<thead>
<tr>
<th>batch size</th>
<th>accuracy</th>
<th>8-GPU speed</th>
<th>8-GPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.602</td>
<td>5771 img/sec</td>
<td>6h 10m 30s</td>
</tr>
<tr>
<td>4096</td>
<td>0.604</td>
<td>15379 img/sec</td>
<td>2h 19m 24s</td>
</tr>
</tbody>
</table>

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
     - $\nabla W$: gradients
     - $\gamma$: 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

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

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 $\gamma$ as a function of mini-batch size $B$
   • a learning rate “warm-up”

Resnet-50

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 $\gamma$, increase $\gamma$ 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 $\gamma=0.05$: 5% accuracy loss

<table>
<thead>
<tr>
<th>batch size</th>
<th>learning rate</th>
<th>warmup</th>
<th>momentum</th>
<th>epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.02</td>
<td>no</td>
<td>0.9</td>
<td>100</td>
<td>58.8%</td>
</tr>
<tr>
<td>1024</td>
<td>0.02</td>
<td>no</td>
<td>0.9</td>
<td>100</td>
<td>58.2%</td>
</tr>
<tr>
<td>4096</td>
<td>0.05</td>
<td>yes</td>
<td>0.9</td>
<td>100</td>
<td>53.1%</td>
</tr>
</tbody>
</table>

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 help. Net diverged.

<table>
<thead>
<tr>
<th>learning rate</th>
<th>warmup</th>
<th>epochs</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>Yes</td>
<td>100</td>
<td>0.509539</td>
</tr>
<tr>
<td>0.02</td>
<td>Yes</td>
<td>100</td>
<td>0.526504</td>
</tr>
<tr>
<td>0.03</td>
<td>Yes</td>
<td>100</td>
<td>0.520454</td>
</tr>
<tr>
<td>0.04</td>
<td>Yes</td>
<td>100</td>
<td>0.530026</td>
</tr>
<tr>
<td>0.05</td>
<td>Yes</td>
<td>100</td>
<td>0.531018</td>
</tr>
<tr>
<td>0.06</td>
<td>Yes</td>
<td>100</td>
<td>0.516101</td>
</tr>
<tr>
<td>0.07</td>
<td>Yes</td>
<td>100</td>
<td>0.001056</td>
</tr>
<tr>
<td>0.08</td>
<td>Yes</td>
<td>100</td>
<td>0.491709</td>
</tr>
<tr>
<td>0.09</td>
<td>Yes</td>
<td>100</td>
<td>0.001056</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>batch size</th>
<th>learning rate</th>
<th>Weight decay</th>
<th>Momentum</th>
<th>Epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.02</td>
<td>0.0005</td>
<td>0.9</td>
<td>128</td>
<td>60.2%</td>
</tr>
<tr>
<td>4096</td>
<td>0.18</td>
<td>0.0005</td>
<td>0.9</td>
<td>128</td>
<td>58.9%</td>
</tr>
<tr>
<td>8192</td>
<td>0.30</td>
<td>0.0005</td>
<td>0.9</td>
<td>128</td>
<td>58.0%</td>
</tr>
</tbody>
</table>

• 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 $\lambda$, but run into stability issues when $\left|\lambda \cdot \frac{\partial L}{\partial W_l}\right|$ 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)

$$\hat{G}_l = \rho \frac{G_l}{|G_l|} \cdot |W_l|,$$

where: $G_l = \frac{\partial L}{\partial W_l}$ - gradient of loss wrt layer weights $w_l$,
$\rho$ - “trust ratio” (user defined parameter).
## AlexNet: add LARS

### AlexNet

<table>
<thead>
<tr>
<th>batch size</th>
<th>learning rate rule</th>
<th>learning rate</th>
<th>learning rate</th>
<th>warmup</th>
<th>momentum</th>
<th>Epochs</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>Regular</td>
<td>0.02</td>
<td>no</td>
<td>0.9</td>
<td>100</td>
<td>58.8%</td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>LARS</td>
<td>0.10</td>
<td>13 epochs</td>
<td>0.9</td>
<td>100</td>
<td>58.4%</td>
<td></td>
</tr>
<tr>
<td>8192</td>
<td>LARS</td>
<td>0.10</td>
<td>8 epochs</td>
<td>0.9</td>
<td>100</td>
<td>58.3%</td>
<td></td>
</tr>
</tbody>
</table>

### AlexNet-BN

<table>
<thead>
<tr>
<th>batch size</th>
<th>learning rate rule</th>
<th>learning rate</th>
<th>learning rate</th>
<th>warmup</th>
<th>momentum</th>
<th>Epochs</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>LARS</td>
<td>0.05</td>
<td>2 epochs</td>
<td>0.9</td>
<td>100</td>
<td>60.2%</td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>LARS</td>
<td>0.10</td>
<td>2 epochs</td>
<td>0.9</td>
<td>100</td>
<td>60.4%</td>
<td></td>
</tr>
<tr>
<td>8192</td>
<td>LARS</td>
<td>0.14</td>
<td>2 epochs</td>
<td>0.9</td>
<td>100</td>
<td>60.1%</td>
<td></td>
</tr>
</tbody>
</table>
AlexNet-BN: Effects of 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
Large Batch vs Baseline
Benefits of Large-Batch Training

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