Scaling SGD Batch Size to 32K for ImageNet Training

Yang You

Computer Science Division of UC Berkeley

youyang@cs.berkeley.edu
Outline

- **Why large-batch training is important?**
- Why large-batch training is difficult?
- How to scale up batch size?
- Results and Benefits of large-batch training.
Mini-Batch SGD (Stochastic Gradient Descent)

- Take $B$ data points each iteration
- Compute gradients of weights based on $B$ data points
- Update the weights: $\mathcal{W} = \mathcal{W} - \eta \ast \nabla \mathcal{W}$
  - also used momentum and weight decay
  - $\mathcal{W}$: weights
  - $\nabla \mathcal{W}$: gradients
  - $\eta$: learning rate
  - $B$: batch size

Data-Parallelism on $P$ GPUs
- Each GPU has a copy of $\mathcal{W}_i$ and $\nabla \mathcal{W}_i$ ($i \in \{1, 2, ..., P\}$)
- Each GPU has $B/P$ data points to compute its own $\nabla \mathcal{W}_i$
- communication: an all-reduce sum each iteration ($\sum_{i=1}^{P} \nabla \mathcal{W}_i$)
- Each GPU does $\mathcal{W}_i = \mathcal{W}_i - \eta/P \ast \sum_{i=1}^{P} \nabla \mathcal{W}_i$
Single GPU: large batch size benefits

- $B = 512$, the GPU achieves peak performance
- If we have 16 GPUs, we need a batch size of $8192$ ($16 \times 512$)
  - make sure each GPU is efficient
Motivation

- **Pick a Commonly-Used Approach in DNN Training?**
  - Data-Parallelism Mini-Batch SGD (e.g. Caffe, Tensorflow, Torch)
  - recommended by Dr. Bryan Catanzaro (NVIDIA VP)

- **How to speedup Mini-Batch SGD?**
  - Use more processors (e.g. GPU)

- **How to make each GPU efficient if we use many GPUs?**
  - Give each GPU enough computations (find the right $B$)

- **How to give each GPU enough computations?**
  - Use large batch size (use $PB$)
Standard Benchmarks

- 1000-class ImageNet dataset by AlexNet
  - 58% accuracy in 100 epochs
- 1000-class ImageNet dataset by ResNet-50
  - 73% accuracy in 90 epochs

1 epoch: statistically touch all the data once \((n/B\) iterations)\)
  - \(n\) is the total number of data points
  - do not use data augmentation (preprocess the dataset)
We fix the number of operations as $90 \times 1.28$ Million $\times 7.72$ Billion

- 90 epochs for using ResNet-50 to process ImageNet-1k dataset
Why Large-Batch can speedup DNN training?

- Reduce the number of iterations
- Keep the single iteration time constant (roughly)
  - by using more processors
Why Large-Batch can speedup DNN training?

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Epochs</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>100</td>
<td>250,000</td>
</tr>
<tr>
<td>1024</td>
<td>100</td>
<td>125,000</td>
</tr>
<tr>
<td>2048</td>
<td>100</td>
<td>62,500</td>
</tr>
<tr>
<td>4096</td>
<td>100</td>
<td>31,250</td>
</tr>
<tr>
<td>8192</td>
<td>100</td>
<td>15,625</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,280,000</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

- ImageNet dataset: 1,280,000 data points
- Goal: get the same accuracy in the same epochs
  - fixed epochs = fixed number of floating point operations
  - needs much less iterations: speedup!
Why Large-Batch can speedup DNN training?

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Epochs</th>
<th>Iterations</th>
<th>GPUs</th>
<th>Iteration Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>100</td>
<td>250,000</td>
<td>1</td>
<td>$t_1$</td>
</tr>
<tr>
<td>1024</td>
<td>100</td>
<td>125,000</td>
<td>2</td>
<td>$t_1 + \log(2)t_2$</td>
</tr>
<tr>
<td>2048</td>
<td>100</td>
<td>62,500</td>
<td>4</td>
<td>$t_1 + \log(4)t_2$</td>
</tr>
<tr>
<td>4096</td>
<td>100</td>
<td>31,250</td>
<td>8</td>
<td>$t_1 + \log(8)t_2$</td>
</tr>
<tr>
<td>8192</td>
<td>100</td>
<td>15,625</td>
<td>16</td>
<td>$t_1 + \log(16)t_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,280,000</td>
<td>100</td>
<td>100</td>
<td>2500</td>
<td>$t_1 + \log(2500)t_2$</td>
</tr>
</tbody>
</table>

- ImageNet dataset: 1,280,000 data points
- use batch size = 512 for each GPU
- $t_1$: computation time, $t_2$: communication time $(\alpha + |W|/\beta)^1$
  - $t_1 \gg t_2$ is possible for ImageNet training by Infiniband$^2$

---

1. $\alpha$ is latency, $\beta$ is inverse of bandwidth
2. Goyal et al, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2017 (Facebook Report)
Difficulties of Large-Batch Training: much more epochs!

LIMITS OF DATA PARALLELISM
The elusive optimum

- Some amount of data parallelism is optimum
- This amount depends to change based on:
  - Model
  - Dataset
  - Optimization algorithm
- Generally we run at 512-2048 samples

slide from Dr. Bryan Catanzaro (Feb 13, 2017 at Berkeley)
Difficulties of Large-Batch Training

- **Lose Accuracy by running the same epochs!**
- Without accuracy, this was well studied 20 years ago
  - Standard Divide-and-Conquer approach
    - Divide: partition a batch of data points to different machines
    - Conquer: an all-reduce operation at each iteration
Why large-batch training is important?

**Why large-batch training is difficult?**

How to scale up batch size?

Results and Benefits of large-batch training.
Difficulties of Large-Batch Training

- Why lose accuracy?
  - Generalization Problem\(^3\)
    - High training accuracy, but low test accuracy
  - Optimization Difficulty\(^4\)
    - Hard to get the right hyper-parameters

---


\(^4\) Goyal et al, *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*, 2017 (Facebook Report)
Large-batch training is a sharp minimum problem\(^5\)
- even you can train a good model, it is hard to generalize
- high training accuracy :-( but low test accuracy :-(

Optimization Problem

- You can keep the accuracy, but it is hard to optimize\(^6\)
- Facebook scales to 8K (able to use 256 NVIDIA P100 GPUs!)

\(^6\)Goyal et al, Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, 2017 (Facebook Report)
Most effective techniques (Facebook’s recipe)

- Control the learning rate ($\eta$)
- Linear Scaling rule\(^7\)
  - if you increase $B$ to $kB$, then increase $\eta$ to $k\eta$
    - # iterations reduced by $k\times$, # updates reduced by $k\times$
    - each update should enlarged by $k\times$
- Warmup rule\(^8\)
  - start from a small $\eta$, increase $\eta$ in a few epochs
    - avoid the network diverges in the beginning

---
\(^7\) Alex Krizhevsky, *One weird trick for parallelizing convolutional neural networks*, 2014 (Google Report)
\(^8\) Goyal et al, *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*, 2017 (Facebook Report)
### State-of-the-art Large-Batch ImageNet Training

<table>
<thead>
<tr>
<th>Team</th>
<th>Model</th>
<th>Baseline Batch</th>
<th>Large Batch</th>
<th>Baseline Accuracy</th>
<th>Large Batch Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google(^9)</td>
<td>AlexNet</td>
<td>128</td>
<td>1024</td>
<td>57.7%</td>
<td>56.7%</td>
</tr>
<tr>
<td>Amazon(^10)</td>
<td>ResNet-152</td>
<td>256</td>
<td>5120</td>
<td>77.8%</td>
<td>77.8%</td>
</tr>
<tr>
<td>Facebook(^11)</td>
<td>ResNet-50</td>
<td>256</td>
<td>8192</td>
<td>76.40%</td>
<td>76.26%</td>
</tr>
</tbody>
</table>

---

\(^9\) Alex Krizhevsky, *One weird trick for parallelizing convolutional neural networks*, 2014 (Google Report)
\(^10\) Mu Li, *Scaling Distributed Machine Learning with System and Algorithm Co-design*, 2017 (CMU Thesis)
\(^11\) Goyal et al, *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*, 2017 (Facebook Report)
Reproduce Facebook’s results

- $B = 256$ and $B = 8192$: achieve 73% accuracy in 90 epochs
- Our baseline’s accuracy is lower than Facebook’s
  - we didn’t use data augmentation
Facebook’s recipe does not work for AlexNet

- Can only scale batch size to 1024, tried everything:
  - Warmup + Linear Scaling
  - Tune $\eta$ + Tune momentum + Tune weight decay
  - data shuffle, data scaling, min $\eta$ tuning, etc

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Base $\eta$</th>
<th>poly power</th>
<th>momentum</th>
<th>epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.02</td>
<td>2</td>
<td>0.9</td>
<td>100</td>
<td>0.588</td>
</tr>
<tr>
<td>1024</td>
<td>0.02</td>
<td>2</td>
<td>0.9</td>
<td>100</td>
<td>0.582</td>
</tr>
<tr>
<td>4096</td>
<td>0.05</td>
<td>2</td>
<td>0.9</td>
<td>100</td>
<td>0.531</td>
</tr>
</tbody>
</table>
Facebook’s recipe does not work for AlexNet

- We couldn’t scale up the learning rate
- Warmup did help (1, 2, 3, ..., 10 epochs)
- Network diverged at $\eta = 0.07$

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Base $\eta$</th>
<th>warmup</th>
<th>epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>0.01</td>
<td>yes</td>
<td>100</td>
<td>0.509</td>
</tr>
<tr>
<td>4096</td>
<td>0.02</td>
<td>yes</td>
<td>100</td>
<td>0.527</td>
</tr>
<tr>
<td>4096</td>
<td>0.03</td>
<td>yes</td>
<td>100</td>
<td>0.520</td>
</tr>
<tr>
<td>4096</td>
<td>0.04</td>
<td>yes</td>
<td>100</td>
<td>0.530</td>
</tr>
<tr>
<td>4096</td>
<td>0.05</td>
<td>yes</td>
<td>100</td>
<td>0.531</td>
</tr>
<tr>
<td>4096</td>
<td>0.06</td>
<td>yes</td>
<td>100</td>
<td>0.516</td>
</tr>
<tr>
<td>4096</td>
<td>0.07</td>
<td>yes</td>
<td>100</td>
<td>0.001</td>
</tr>
</tbody>
</table>
Outline

- Why large-batch training is important?
- Why large-batch training is difficult?
- **How to scale up batch size?**
- Results and Benefits of large-batch training.
Solve the generalization problem by Batch Normalization

- Generalization problem\(^\text{12}\)
  - regular batch: \(|\text{Test loss - Train Loss}|\) is small
  - large batch: \(|\text{Test loss - Train Loss}|\) is large

\(^{12}\text{Keskar et al, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2017 (ICLR)}\)
Solve the generalization problem by Batch Normalization

Generalization problem\textsuperscript{13}

- regular batch: $|\text{Test loss} - \text{Train Loss}|$ is small
- large batch: $|\text{Test loss} - \text{Train Loss}|$ is large

\textsuperscript{13}Keskar et al, \textit{On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima}, 2017 (ICLR)
Solve the generalization problem by Batch Normalization

- Optimize the model
  - Batch Norm (BN) instead of Local Response Norm (LRN)
  - BN after Convolutional layers
- Run more epochs (100 epochs to 128 epochs)

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Base LR</th>
<th>poly power</th>
<th>momentum</th>
<th>weight decay</th>
<th>epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.02</td>
<td>2</td>
<td>0.9</td>
<td>0.0005</td>
<td>128</td>
<td>0.602</td>
</tr>
<tr>
<td>4096</td>
<td>0.18</td>
<td>2</td>
<td>0.9</td>
<td>0.0005</td>
<td>128</td>
<td>0.589</td>
</tr>
<tr>
<td>8192</td>
<td>0.30</td>
<td>2</td>
<td>0.9</td>
<td>0.0005</td>
<td>128</td>
<td>0.580</td>
</tr>
</tbody>
</table>

- Higher accuracy, but the baseline is also higher
- Still needs to improve large-batch’s accuracy
Still needs to improve AlexNet’s accuracy

- Reduce epochs from 128 to 100
  - Clearly an accuracy gap
Reason: different Gradient-Weight ($|\nabla W|/|W|$) Ratios

<table>
<thead>
<tr>
<th>Layer</th>
<th>conv1.1</th>
<th>conv1.0</th>
<th>conv2.1</th>
<th>conv2.0</th>
<th>conv3.1</th>
<th>conv3.0</th>
<th>conv4.0</th>
<th>conv4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>W</td>
<td>_2$</td>
<td>1.86</td>
<td>0.098</td>
<td>5.546</td>
<td>0.16</td>
<td>9.40</td>
<td>0.196</td>
</tr>
<tr>
<td>$</td>
<td>\nabla W</td>
<td>_2$</td>
<td>0.22</td>
<td>0.017</td>
<td>0.165</td>
<td>0.002</td>
<td>0.135</td>
<td>0.0015</td>
</tr>
<tr>
<td>$</td>
<td>W</td>
<td>_2$</td>
<td>8.48</td>
<td>5.76</td>
<td>33.6</td>
<td>83.5</td>
<td>69.9</td>
<td>127</td>
</tr>
<tr>
<td>$</td>
<td>\nabla W</td>
<td>_2$</td>
<td>8.48</td>
<td>5.76</td>
<td>33.6</td>
<td>83.5</td>
<td>69.9</td>
<td>127</td>
</tr>
</tbody>
</table>

- L2 norm of layer weights and gradients of AlexNet
  - Batch = 4096 at 1st iteration
  - Bad: the same $\eta$ for all the layers ($W = W - \eta \nabla W$)
    - layer fc6.0’s best $\eta$ leads to divergence for layer conv1.0
Layer-wise Adaptive Rate Scaling (LARS)

\[ \eta = l \times \gamma \times \frac{||W||_2}{||\nabla W||_2} \]

- \( l \): scaling factor, 0.001 for AlexNet and ResNet training
- \( \gamma \): input LR, a tuning parameter for users
- We usually tune \( \gamma \) from 1 to 50
Effects of LARS

- **AlexNet**

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>LR rule</th>
<th>poly power</th>
<th>warmup</th>
<th>weight decay</th>
<th>momentum</th>
<th>Epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>regular</td>
<td>2</td>
<td>N/A</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.588</td>
</tr>
<tr>
<td>4096</td>
<td>LARS</td>
<td>2</td>
<td>13 epochs</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.584</td>
</tr>
<tr>
<td>8192</td>
<td>LARS</td>
<td>2</td>
<td>8 epochs</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.583</td>
</tr>
</tbody>
</table>

- **AlexNet-BN**

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>LR rule</th>
<th>poly power</th>
<th>warmup</th>
<th>weight decay</th>
<th>momentum</th>
<th>Epochs</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>LARS</td>
<td>2</td>
<td>2 epochs</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.602</td>
</tr>
<tr>
<td>4096</td>
<td>LARS</td>
<td>2</td>
<td>2 epochs</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.604</td>
</tr>
<tr>
<td>8192</td>
<td>LARS</td>
<td>2</td>
<td>2 epochs</td>
<td>0.0005</td>
<td>0.9</td>
<td>100</td>
<td>0.601</td>
</tr>
</tbody>
</table>
Outline

- Why large-batch training is important?
- Why large-batch training is difficult?
- How to scale up batch size?
- Results and Benefits of large-batch training.
Implementation Details

- NVIDIA Caffe 0.16 with our own modification (Auto LR)
- 1 Intel Xeon CPU E5-2698 v4 @ 2.20GHz
- 8 NVIDIA P100 GPUs interconnected by NVIDIA NVLink

- Batch 8192 by ResNet-50: out of memory
  - partition the 8192-batch into 32 256-batches
  - compute 32 pieces of gradients sequentially
  - do an average operation after we get all the gradients
Effects of LARS

AlexNet-BN for ImageNet

- Batch=512
- Batch=4096

AlexNet-BN for ImageNet

- Batch=512, Baseline
- Batch=4096, LARS

AlexNet-BN for ImageNet

- Batch=512
- Batch=8192

AlexNet-BN for ImageNet

- Batch=512, Baseline
- Batch=8192, LARS
Effects of LARS

ImageNet by ResNet50 without Data Augmentation

Top-1 Test Accuracy vs. Epochs

- Blue line: Batch=32k, LR=2.9, warmup, LARS
- Green line: Batch=256, LR=0.2
Effects of LARS

ImageNet by ResNet50 without Data Augmentation

Top-1 Test Accuracy

Epochs

Batch=32k, LR=2.9, warmup, LARS
Batch=16k, LR=2.5, warmup, LARS
Batch=8k, LR=6.4, warmup
Batch=256, LR=0.2
Benefits of Large-Batch Training

- **AlexNet-BN**: $3 \times$ speedup by just increasing the batch size

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Stable 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>

- **AlexNet**: $3 \times$ speedup by just increasing the batch size

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Stable 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>

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