
Takuya Akiba, Shuji Suzuki, Keisuke Fukuda, and Kota Uenishi
Preferred Networks, Inc.
Who are we?

Preferred Networks, Inc. (PFN):
A Tokyo-based Deep Learning & IoT company
Research and engineering in PFN

- Strong Engineering partnership

- Active research
  - Constantly publish papers in top-tier ML conferences
  - Including 3 papers in ICLR’18

and more!
“Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions”

arXiv:1710.06280
Distributed Deep Learning
Training time of ResNet-50 (90 epochs) on ImageNet

<table>
<thead>
<tr>
<th></th>
<th>Time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goyal et al.</td>
<td>60min.</td>
</tr>
<tr>
<td>Codreanu et al.</td>
<td>62min.</td>
</tr>
<tr>
<td>Cho et al.</td>
<td>50min.</td>
</tr>
<tr>
<td>You et al.</td>
<td>31min.</td>
</tr>
<tr>
<td>Akiba et al.</td>
<td>15min.</td>
</tr>
</tbody>
</table>

Details later!
Jen-Hsun Huang
NVIDIA CEO, at SC’17
What we want:

Shorter training time

It is always better

No questions? 😊
Answer: Not Really.

Even if training time is faster…

• Model accuracy is degraded => 😥

• Programming is hard => 😥

Increasing the training throughput is easy…
But it does not necessarily make R&D faster
What we **really** want:

Shorter training time

Faster R&D cycle

- Design a new model quicker
- Train faster
- Get a better (or equivalent) model
Background of the ImageNet challenge
A Powerful, Flexible, and Intuitive Framework for Neural Networks

GET STARTED

LEARN MORE

https://chainer.org/
Chainer: A Flexible Deep Learning Framework

Define-and-Run

Define
- Model definition
  - Computational graph
  - Gradient function

Run
- Training data
  - Computational graph
  - Gradient function

Define-by-Run

Define-by-Run
- Model definition
  - Computational graph
  - Gradient function

- Training data
  - Computational graph
  - Gradient function

Caffe2, TensorFlow etc.

PyTorch, TensorFlow (Eager Execution) etc.
ChainerMN: Distributed Training with Chainer

- Add-on package for Chainer
- Enables multi-node distributed deep learning using NVIDIA NCCL2

Features
- **Scalable**: Near-linear scaling with hundreds of GPUs
- **Flexible**: Even GANs, dynamic NNs, and RL are applicable

**Distributed Training with ChainerMN**

```
Forward  Backward  All-Reduce  Forward  Backward  Optimize
Forward  Backward  Optimize
Forward  Backward  Optimize
```
MN-1: an in-house supercomputer

- NVIDIA Tesla P100 × 1024
  8 GPUs per node, 128 nodes in total
- Inter-connected by InfiniBand FDR
  2 HCAs per node, tree-like topology

The number of employees is about 120, so this is relatively very large for us!

Fun!
(Do you think it’s crazy?)
OK, let’s tackle the ImageNet problem with our 1024 P100 GPUs!
Our goal: **15 min.**

- Training CNNs on ImageNet is very time consuming
- Original ResNet-50 paper: **29 hours using 8 GPUs**
- Notable achievement by Goyal et al.: **1 hour using 256 GPUs.**

⇒ We can use 1024 GPUs. 1 hour / 1024 * 256 = 15 mins. 🤔

Sounds easy? ABSOLUTELY NO!

**Technical Challenges:**
1. Large batch problem
2. Performance Scalability (while keeping flexibility)
3. Troubles 😃

“Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes”
(armx:1711.04325)
Challenges in the “ImageNet-15min challenge”

1. The “large batch” problem
   – “Sharp minima”
   – Fewer training iterations

2. Performance scalability

3. Technical issues 😞
Challenges in the “ImageNet-15min challenge”

1. The “large batch” problem
   - “Sharp minima”
   - Fewer training iterations

2. Performance scalability

3. Technical issues 😞
Challenge 1: The “large batch” problem

“It has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize”

From Keskar et al.
“On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima”

1. Computed gradients in each iteration is an average of larger number of samples → gradients are “less stochastic”, which makes it difficult to escape from local minima

2. Total number of iterations (=updates) is smaller
• Linear scaling rule:
  • “If minibatch-size is $k$ times larger, increase learning rate by $k$ times”
• Gradual warmup scheme
Additional techniques for 1024 GPUs:

- We needed to go further: $32 \times 1024 = 32k$ batchsize!

- RMSprop Warmup
  - SGD: generalizes well, but converges slower.
  - We start the training with RMSprop, then gradually transition to SGD.

- Batch normalization without moving averages
Challenges in the “ImageNet-15min challenge”

1. The “large batch” problem
   – “Sharp minima”
   – Fewer training iterations

2. Performance scalability

3. Technical issues 😞
Challenge 2: 
Performance scalability

Allreduce operation is critical for scalability
How to overcome scalability challenge?

Improve the All-reduce bottleneck

- Use faster communication routines
- Reduce communication data
How to overcome scalability challenge?

Improve the All-reduce bottleneck

- Use faster communication routines
- Reduce communication data
Faster communication routines

• ChainerMN is built on top of MPI
  
  – Just call MPI_Allreduce() and nothing else to do? (MPI should be well tuned… Agreed?)
  
  – Bandwidth efficiency of MPI_Allreduce with GPUDirect: 10%
    (as of the experiment, Open MPI 2.1.2, Infiniband FDR)
NCCL : Nvidia Collective Communication Library

64MB Allreduce (MPI_SUM), 2 processes,
- Open MPI 2.1.2
  (default configuration: no advanced tuning)
- Over Infiniband FDR(4x)

"MPI" :
Allreduce an array on host memory
(ordinary MPI_Allreduce)

“MPI-CUDA” :
Allreduce an array on GPU's device memory
(You can pass device memory pointer to MPI routines)

NCCL is 5.9x faster!
Further optimizations for NCCL:

- Improve network performance
  - GPU Direct P2P & RDMA
  - Manual ring configuration
How to overcome scalability challenge?

Improve the All-reduce bottleneck

• Use faster communication routines
• Reduce communication data
Reduce communication data: use FP16

Compute gradients

Convert FP32 to FP16

Allreduce (with NCCL)

Convert FP16 to FP32 and update

The accuracy degradation is negligible!!
Challenges in the “ImageNet-15min challenge”

1. The “large batch” problem
   - “Sharp minima”
   - Fewer training iterations

2. Performance scalability

3. Technical issues 😞
Crash... Crash... Crash...

• The more you buy, the more you crash

• \( \geq 192 \) GPUs: Crash \( \rightarrow \) NCCL2: too many file descriptors
• \( \geq 784 \) GPUs: Crash \( \rightarrow \) Bug in ChainerMN
• \( \geq 944 \) GPUs: Crash \( \rightarrow \) NCCL2: stack overflow

• Some GPUs were broken, as well

(As of NCCL 2.0.5)
Crash... Crash... Crash...

Tips for users of NCCL v2 with >1000 GPUs:

• NCCL v2 opens a large number of file descriptors.
  – `ulimit -n unlimited`, or will see ’unhandled system error’
• NCCL v2 uses huge amount of stack.
  – `ulimit -s unlimited`, or will see SEGV
• When it suddenly starts to claim ‘unhandled system error’, just `reboot` all nodes.

(As of NCCL 2.0.5)
Training time of ResNet-50 (90 epochs) on ImageNet

- Goyal et al. (Facebook): 60min.
- Codreanu et al. (IBM): 62min.
- Cho et al. (IBM): 50min.
- You et al.: 31min.
- Akiba et al. (This work): 15min.

Faster
Training ResNet-50 on ImageNet in 15 mins

<table>
<thead>
<tr>
<th>Team</th>
<th>Hardware</th>
<th>Software</th>
<th>Batchsize</th>
<th>Time</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>He et al.</td>
<td>P100 × 8</td>
<td>Caffe</td>
<td>256</td>
<td>29 hr</td>
<td>75.3 %</td>
</tr>
<tr>
<td>Goyal et al.</td>
<td>P100 × 256</td>
<td>Caffe2</td>
<td>8,192</td>
<td>1 hr</td>
<td>76.3 %</td>
</tr>
<tr>
<td>Codreanu et al.</td>
<td>KNL 7250 × 720</td>
<td>Intel Caffe</td>
<td>11,520</td>
<td>62 min</td>
<td>75.0 %</td>
</tr>
<tr>
<td>Cho et al.</td>
<td>P100 × 256</td>
<td>Torch</td>
<td>8,192</td>
<td>50 min</td>
<td></td>
</tr>
<tr>
<td>You et al.</td>
<td>Xeon 8160 × 1600</td>
<td>Intel Caffe</td>
<td>16,000</td>
<td>31 min</td>
<td>75.3 %</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td><strong>P100 × 1024</strong></td>
<td><strong>Chainer</strong></td>
<td><strong>32,768</strong></td>
<td><strong>15 min</strong></td>
<td><strong>74.9 %</strong></td>
</tr>
</tbody>
</table>


- **Dataset**: ImageNet-1k
- **Accuracy**: single-crop top-1 validation accuracy
- **Training duration**: 90 epochs (common configuration for ResNet50)

We achieved a total training time of 15 minutes while maintaining a comparable accuracy of 74.9%.
Maybe you are thinking:

We don’t have so many GPUs…
Our GPU cluster does not have Infiniband…

It’s not for us 😞
ChainerMN is for you.
Want to try Chainer + ChainerMN?

Cloud formation support is coming soon!
Optimization technique for non-IB environment: Double buffering

- Each update uses the gradients from previous iteration (1-step stale grad.)
Computing time of ImageNet training with Double Buffering + FP16 communication

- Local batchsize: 64
- 32 processes
- NCCL for Allreduce
Validation Accuracy of ResNet-50 model on ImageNet

Accuracy

epoch

IB  10 Gb Eth  10 Gb Eth w/ double buffering and FP16 all-reduce
95% scalability up to 32 GPUs!!

ResNet-50 on ImageNet training
- 25Gbps Ethernet
- Double buffering
- FP16 communication (NCCL)
- V100 GPUs
- Batchsize: 64/GPU
Next step?

“ImageNet is the new MNIST”
by Chris Ying (Google Brain)

How to move towards larger, more complex models?
Taxonomy of distributed deep learning

Data-parallelism
- Synchronous
- Asynchronous

Model-parallelism
- Fine-grained
- Coarse-grained

Main focus (currently)
Data parallel: sync vs. async

Synchronous:

Asynchronous:
Model parallelism

Find-grained

Coarse-grained

Example:
Mixture-of-Experts [Shazeer+(Google Brain), ICLR’17]
ChainerMN’s focus, now and future

Data-parallelism
- Synchronous (Main focus (currently))
- Asynchronous

Model-parallelism
- Fine-grained
- Coarse-grained

Under active development
Basic components ready to use
Conclusion

- We finished training of ResNet-50 on ImageNet in 15 min.
- We achieved both of speed, accuracy, and productivity

We will continue tackling hard problems!
Thank you!