ChainerMN

Scalable Distributed Deep Learning with Chainer

Takuya Akiba | Preferred Networks, Inc.
ChainerMN

v1.0.0β

Released today!

https://git.io/chainermn
Agenda

1. Intro to distributed deep learning
2. Chainer and ChainerMN
3. Benchmark results
4. Future directions
Training Iterations
Training Iterations

Forward

2

→

6

→
Training Iterations

Forward → Backward

Gradient \( \frac{\delta E}{\delta w} \)

2
Training Iterations
Two Kinds of Parallelism

Data Parallel

Model Parallel
Synchronous vs. Asynchronous

**Synchronous**

- Computation

**Asynchronous**

- Parameter Server
  - Push grad
  - Pull model
Synchronous vs. Asynchronous

Synchronous

Which should we use?
Synchronous vs. Asynchronous

Synchronous

Lower throughput

Asynchronous

Higher throughput
Large scale **distributed deep** networks
J Dean, G Corrado, R Monga, K Chen ... - Advances in neural ... 2012 - papers.nips.cc
We found that Downpour SGD, a highly asynchronous variant of SGD works surprisingly well for training nonconvex **deep learning** models. Sandblaster L-BFGS, a **distributed** implementation of L-BFGS, can be competitive with SGD, and its more efficient use of network ...

**Deep learning**
Y LeCun, Y Bengio, G Hinton - Nature, 2015 - nature.com
**Deep learning** discovers intricate structure in large data sets by using the backpropagation algorithm ... RNN) taking, as extra input, the representation extracted by a **deep** convolution neural ... The **distributed** representations of words are obtained by using backpropagation to jointly ...

**Deep learning:** methods and applications
L Deng, D Yu - Foundations and Trends® in Signal ... 2014 - nowpublishers.com
... have enabled the **deep learning** methods to effectively exploit complex, compositional nonlinear functions, to learn **distributed** and hierarchical ... 7 Technology, University of Washington, and numerous other places; see http://deplearning.net/deep-learning-research-groups ...
Large Scale Distributed Deep Networks

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc' Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Andrew Y. Ng
{jeff, gcorrado}@google.com
Google Inc., Mountain View, CA

Abstract

Recent work in unsupervised feature learning and deep learning has shown that being able to train large models can dramatically improve performance. In this paper, we consider the problem of training a deep network with billions of parameters using tens of thousands of CPU cores. We have developed a software framework called DistBelief that can utilize computing clusters with thousands of machines to train large models. Within this framework, we have developed two algorithms for large-scale distributed training: (i) Downpour SGD, an asynchronous stochastic gradient descent procedure supporting a large number of model replicas, and (ii) Sandblaster, a framework that supports a variety of distributed batch optimization procedures, including a distributed implementation of L-BFGS. Downpour SGD and Sandblaster L-BFGS both increase the scale and speed of deep network training. We have successfully used our system to train a deep network 30x larger than previously reported in the literature, and achieves state-of-the-art performance on ImageNet, a visual object recognition task with 16 million images and 21k categories. We show that these same techniques dramatically accelerate the training of a more modestly-sized deep network for a commercial speech recognition service. Although we focus on and report performance of these methods as applied to training large neural networks, the underlying algorithms are applicable to any gradient-based machine learning algorithm.

1 Introduction
Synchronous vs. Asynchronous

Synchronous

Lower throughput

Asynchronous

Higher throughput
Synchronous vs. Asynchronous

- Synchronous
  - Lower throughput

- Asynchronous
  - Higher throughput

Only looking at throughput?
**Revisiting Distributed Synchronous SGD**

Jianmin Chen, Rajat Monga, Samy Bengio & Rafal Jozefowicz
Google Brain
Mountain View, CA, USA
{jmchen, rajatmonga, bengio, rafalj}@google.com
Gradient Staleness
Training Iterations

Usual Training with Chainer

- Forward
- Backward
- Optimize

Distributed Training with ChainerMN
Training Iterations

Usual Training with Chainer

Forward → Backward → Optimize

Distributed Training with ChainerMN

Forward → Forward → Forward
Training Iterations

Usual Training with Chainer

Distributed Training with ChainerMN
Training Iterations

Usual Training with Chainer

Distributed Training with ChainerMN
Training Iterations

Usual Training with Chainer

Distributed Training with ChainerMN
Chainer & ChainerMN
A Flexible Deep Learning Framework
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
ChainerMN’s building blocks

CUDA-Aware MPI

Node

Node

NVIDIA NCCL
NVIDIA NCCL

```c
ncclBcast(
    dev_buff, n, ncclFloat, 0, comm, stream);
```

- MPI-like interface
- Optimized inner-node GPU-GPU communication
Non-CUDA-Aware MPI

```c
cudaMemcpy(
    host_buf, dev_buf, n, cudaMemcpyDeviceToHost);
MPI_Send(
    host_buf, n, MPI_CHAR, 1, 100, MPI_COMM_WORLD);
```

CUDA-Aware MPI

```c
MPI_Send(
    dev_buf, n, MPI_CHAR, 1, 100, MPI_COMM_WORLD);
```
<table>
<thead>
<tr>
<th>Non-CUDA-Aware MPI</th>
<th>CUDA-Aware MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>memcpy D→H</td>
<td>memcpy H→D</td>
</tr>
<tr>
<td>MPI_Sendrecv</td>
<td>MPI_Sendrecv</td>
</tr>
</tbody>
</table>

Non-CUDA-Aware MPI

CUDA-Aware MPI +GPUDirect

Images are from Parallel Forall | https://devblogs.nvidia.com/parallelforall/introduction-cuda-aware-mpi/
Two-Dimensional All-Reduce
Two-Dimensional All-Reduce

① NCCL Reduce-Scatter
Two-Dimensional All-Reduce

② MPI All-Reduce
Two-Dimensional All-Reduce

③ NCCL All-Gather
Benchmark Results
GeForce GTX TITAN X (Maxwell)  
× 4  
× 32  
Infiniband FDR 4X
Training Speedup for ImageNet Classification (ResNet-50)

- Chainer
- Ideal speedup

Speedup vs. Number of GPUs
All that glitters is not gold…

We achieved XX times speedup!!

- Improvement only of throughput is actually too easy
  - Too large batch sizes
  - Infrequent synchronization

- Need to check accuracy of resulting models
Training Throughput for ImageNet Classification
(ResNet-50)

#### #Samples per second

<table>
<thead>
<tr>
<th>#GPUs</th>
<th>Chainer</th>
<th>MXNet</th>
<th>CNTK</th>
<th>TensorFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Directions
Limits of Data Parallelism

- **Problem 1**: Fewer iterations
- **Problem 2**: Sharp minima
Problem 1: Fewer iterations

1 epoch
(Same amount of work)

1 GPU

1 iteration

Multi GPUs
Problem 2: Sharp minima [Keskar+, ICLR’17]
Limits of Data Parallelism

- **Problem 1**: Fewer iterations
- **Problem 2**: Sharp minima
Simple model parallelism is not the solution

Standard Training

Batch 1 | Batch 2

Overlapped Model Parallelism

Staleness!
Synthetic Gradients [Jaderberg(DeepMind)+, '16]
Synthetic Gradients [Jaderberg(DeepMind)+, ’16]

Standard Training

- fwd
- bwd

Batch 1
Batch 2

Synthetic Gradient

- ...
Virtual Forward-Backward Networks [Miyato (PFN) et. al, ICLRw’17]

Successfully decoupled 100-layer CNN
Preferred Networks
A startup that applies DL to industrial IoT

Deep learning  ×  Industrial IoT
Robot autonomously learns bin picking without human instruction
Agenda

1. Intro to distributed deep learning
2. Chainer and ChainerMN
3. Benchmark results
4. Future directions
Thank you, team!
ChainerMN
v1.0.0β
Released today!
https://git.io/chainermn