Project Adam: Building an Efficient and Scalable Deep Learning Training System

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible,
and Karthik Kalyanaraman, Microsoft Research
Traditional Machine Learning
Deep Learning

Data → Deep Learning → Prediction

Objective Function

Humans
Deep Learning

The network learns complex intermediate representations from data without explicit labels.
Problem with Deep Learning

Current computational needs on the order of petaFLOPS!
Accuracy scales with data and model size
Neural Networks

Activation function: $a_i(l) = F((\Sigma_{j=1,k} w_{ij}(l-1,l)*a_j(l-1)) + b_i)$
Convolutional Neural Networks with Max Pooling
Neural Network Training (with Stochastic Gradient Descent)

- Inputs processed one at a time in random order with three steps:
  1. Feed-forward evaluation
  2. Back propagation
  3. Weight updates

### Feed-forward evaluation:

The output of each neuron $i$ in a layer $l$, called its activation, $a_i$, is computed as a function of its $k$ inputs from neurons in the preceding layer $l-1$ (or input data for the first layer). If $w_{ij}(l-1,l)$ is the weight associated with a connection between neuron $j$ in layer $l-1$ and neuron $i$ in layer $l$:

$$a_i(l) = F(\sum_{j=1}^{k} w_{ij}(l-1,l)*a_j(l-1) + b_i)$$

where $b_i$ is a bias term for the neuron.

### Back-propagation:

Error terms, $\delta_i$, are computed for each neuron, $i$, in the output layer, $L_o$, first as follows:

$$\delta_i(L_o) = (t_i(L_o) - a_i(L_o))^{*}F'(a_i(L_o))$$

where $t_i(x)$ is the true value of the output and $F'(x)$ is the derivative of $F(x)$.

These error terms are then back-propagated for each neuron $i$ in layer $l$ connected to $m$ neurons in layer $l+1$ as follows:

$$\delta_i(l) = \sum_{j=1}^{m} \delta_j(l+1)*w_{ij}(l,l+1)^{*}F'(a_i(l))$$

### Weight updates:

These error terms are used to update the weights (and biases similarly) as follows:

$$\Delta w_{ij}(l-1,l) = \alpha^{*} \delta_i(l)*a_j(l-1)$$

for $j = 1 .. k$

where $\alpha$ is the learning rate parameter. This process is repeated for each input until the entire training dataset has been processed.
Project Adam

• Optimizing and balancing both computation and communication for this application through whole system co-design

• Achieving high performance and scalability by exploiting the ability of machine learning training to tolerate inconsistencies well

• Demonstrating that system efficiency, scaling, and asynchrony all contribute to improvements in trained model accuracy
Adam System Architecture
Fast Data Serving

• Large quantities of data needed (10-100TBs)
• Data requires transformation to prevent over-fit
• Small set of machines configured separately to perform transformations and serve data
• Data servers pre-cache images using nearly all of system memory as a cache
• Model training machines fetch data in advance in batches in the background
Multi Threaded Training

• Multiple threads on a single machine
• Different images assigned to threads that share model weights
• Per-thread training context stores activations and weight update values
Fast Weight Updates

• Weights updated locally without locks
• Race condition permitted
  • Weight updates are commutative and associative
  • Deep neural networks are resilient to small amounts of noise
• Important for good scaling
Reducing Memory Copies

• Pass pointers rather than copying data for local communication
• Custom network library for non local communication
  • Exploit knowledge of the static model partitioning to optimize communication
  • Reference counting to ensure safety under asynchronous network IO
Memory System Optimizations

• Partition so that model layers fit in L3 cache
• Optimize computation for cache locality
Mitigating the Impact of Slow Machines

- Allow threads to process multiple images in parallel
- Use a dataflow framework to trigger progress on individual images based on arrival of data from remote machines
- At end of epoch, only wait for 75% of the model replicas to complete
  - Arrived at through empirical observation
  - No impact on accuracy
Parameter Server Communication

Two protocols for communicating parameter weight updates

1. Locally compute and accumulate weight updates and periodically send them to the server
   • Works well for convolutional layers since the volume of weights is low due to weight sharing

2. Send the activation and error gradient vectors to the parameter servers so that weight updates can be computed there
   • Needed for fully connected layers due to the volume of weights. This reduces traffic volume from M*N to K*(M+N)
Evaluation

• Visual Object Recognition Benchmarks
• System Hardware
• Baseline Performance and Accuracy
• System Scaling and Accuracy
Visual Object Recognition Benchmarks

• MNIST digit recognition
Visual Object Recognition Benchmarks

- ImageNet 22k Image Classification

American Foxhound  
English Foxhound

http://www.juvomi.de/hunde/bilder/m/FOXEN01M.jpg
System Hardware

• 120 HP Proliant servers
• Each server has an Intel Xeon E5-2450L processor 16 core, 1.8GHZ
• Each server has 98GB of main memory, two 10Gb NICs, one 1 Gb NIC
• 90 model training machines, 20 parameter servers, 10 image servers
• 3 racks each of 40 servers, connected by IBM G8264 switches
Baseline Performance and Accuracy

- Single model training machine, single parameter server.
- Small model on MNIST digit classification task
Model Training System Baseline
Parameter Server Baseline

![Graph showing the comparison of different operations in billion parameter updates per second against the number of processor cores. The graph includes lines for weight updates, weight updates over network, local weight computation, and local weight computation with inputs over network.](image-url)
## Model Accuracy Baseline

<table>
<thead>
<tr>
<th>Systems</th>
<th>MNIST Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellow et al [12]</td>
<td>99.55%</td>
</tr>
<tr>
<td>Adam</td>
<td>99.63%</td>
</tr>
<tr>
<td>Adam (synchronous)</td>
<td>99.39%</td>
</tr>
</tbody>
</table>
System Scaling and Accuracy

• Scaling with Model Workers
• Scaling with Model Replicas
• Trained Model Accuracy
Scaling with Model Workers

![Bar chart showing the increase in billion connections with the number of machines.

<table>
<thead>
<tr>
<th># of Machines</th>
<th>Billion connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>35</td>
</tr>
</tbody>
</table>
Scaling with Model Replicas
<table>
<thead>
<tr>
<th>Systems</th>
<th>ImageNet 22K Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Le et al. [18]</td>
<td>13.6%</td>
</tr>
<tr>
<td>Le et al. (with pre-training) [18]</td>
<td>15.8%</td>
</tr>
<tr>
<td>Adam</td>
<td>29.8%</td>
</tr>
</tbody>
</table>
Trained Model Accuracy at Scale
Exascale Deep Learning for Climate Analytics

Thorsten Kurth*, Josh Romero*, Sean Treichler, Mayur Mudigonda, Nathan Luehr, Everett Phillips, Ankur Mahesh, Michael Matheson, Jack Deslippe, Massimiliano Fatica, Prabhat, Michael Houston

Credits: nersc, nvidia, Oak Ridge National Laboratory
Socio-Economic Impact of Extreme Weather Events

- Tropical cyclones and atmospheric rivers have major impact on modern economy and society

- CA: 50% of rainfall through atmospheric rivers

- FL: Flooding, influence on insurance premiums and home prices

- $200B worth of damage in 2017

- Costs of ~$10B/event for large events
Understanding Extreme Weather Phenomena

• will there be more hurricanes?
• will they be more intense?
• will they make landfall more often?
• will atmospheric rivers carry more water?
• can they help mitigate droughts and decrease risk of forest fires?
• will they cause flooding and heavy precipitation?
Impact Quantification of Extreme Weather Events

• detect hurricanes and atmospheric rivers in climate model projections

• enable geospatial analysis of EW events and statistical impact studies for regions around the world

• flexible and scalable detection algorithm

• gear up for future simulations with ~1 km² spatial resolution
Unique Challenges for Climate Analytics

• interpret as segmentation problem
  • 3 classes - background (BG), tropical cyclones (TC), atmospheric rivers (AR)
• deep learning has proven successful for these tasks
• climate data is complex
  • high imbalance - more than 95% of pixels are background
  • high variance - shape of events change
  • many input channels w/ different properties
• high resolution required
  • no static background, highly variable in space and time
Unique Challenges for Deep Learning

• need labeled data for supervised approach
  • can be leveraged from existing heuristic-based approaches
• define neural network architecture
  • balance between compute performance and model accuracy
  • employ high productivity/flexibility frameworks for rapid prototyping
  • performance optimization requires holistic approach
• hyper parameter tuning (HPO)
  • necessary for convergence and accuracy
Unique Challenges for Deep Learning at Extreme Scale

• data management
  • shuffling/loading/processing/feeding 20 TB dataset to keep GPUs busy
  • efficient use of remote filesystem

• multi-node coordination and synchronization
  • synchronous reduction of O(50)MB across 27360 GPUs after each iteration

• hyper parameter tuning (HPO)
  • convergence and accuracy challenging due to larger global batch sizes
1. The climate model predicts water vapor, wind speeds and humidity.

2. These observables are used to compute the Integrated Water Vapor Transport.
Label Creation: Atmospheric Rivers

3. Binarization by thresholding at 95th percentile

4. Flood fill algorithm generates AR candidates by masking out regions in mid-latitudes
Label Creation: Tropical Cyclones

1. Extract cyclone center and radius using thresholds for pressure, temperature, and vorticity

2. Binarize patch around cyclone center using thresholds for water vapor, wind, and precipitation
Systems

Piz Daint

• Cray XC50 HPC system at CSCS, 5th on top500
• 5320 nodes with Intel Xeon E5-2695v3 and 1 NVIDIA P100 GPU
• Cray Aries interconnect in diameter 5 dragonfly topology
• ~54.4 PetaFlop/s peak performance (FP32)

Summit

• leadership class HPC system at OLCF, 1st on top500
• 4609 nodes with 2 IBM P9 CPU and 6 NVIDIA V100 GPU
• 300 GB/s NVLink connection btw. 3 GPUs in a group
• 800 GB available NVMe storage/node
• dual-rail EDR Infiniband in fat-tree topology
• ~3.45 ExaFlop/s theoretical peak performance (FP16)
Single GPU

• Things to consider:
  • Is my TensorFlow model efficiently using GPU resources?
  • Is my data input pipeline keeping up?
  • Is my TensorFlow model providing reasonable results?
Single Node

• Things to consider:
  • Is my data input pipeline still keeping up?
  • Is my data-parallel TensorFlow model providing reasonable results?
  • How is my performance scaling over PCIe/NVLink using Horovod?
Multi Node

- Things to consider:
  - Is my data-parallel TensorFlow model still providing reasonable results?
  - How is my performance scaling over NVLink + InfiniBand using Horovod?
  - How do I distribute my data across so many nodes?
Deep Learning Models for Extreme Weather Segmentation

Tiramisu, 35 layers, 7.8M parameters, 4.2 TF/sample

DeepLabv3+, 66 layers, 43.7M parameters, 14.4 TF/sample
On-Node I/O Pipeline

- files are in HDF5 with single sample + label/file
- list of filenames passed to TensorFlow Dataset API (tf.data)
- HDF5 serialization bottleneck addressed with multiprocessing + h5py
- extract and batch using tf.data input pipeline
Improvements to Horovod: Original Control Plane

W1: \{1, 2, 5, 8, 13\}
W2: \{1, 3, 5, 7, 13\}
W3: \{2, 3, 5, 10, 13\}
W4: \{1, 3, 5, 10, 13, 14\}

Gather → Intersect lists → Broadcast

W1: \{5, 13\}

Allreduce 5, 13
Improvements to Horovod: Tree-based Control Plane

asynchronous
gather + intersect

tree-based
broadcast

allreduce 5, 13
Improvements to Horovod: Hybrid All-Reduce

• NCCL uses NVLink for high throughput, but ring-based algorithms are latency-limited at scale

• hybrid NCCL/MPI strategy uses strengths of both

• one inter-node allreduce per virtual NIC

• MPI work overlaps well with GPU computation
Gradient Pipelining (Lag)

lag-0 (fully synchronous)

W1 → g1k → g̃k → W1
W2 → g2k → g̃k → W2
W3 → g3k → g̃k → W3
W_N → g_Nk → g̃k → W_N

lag-1

Q1 → g1k → Q1
Q_N → g_Nk → Q_N
Q1 → g̃_k-1 → Q1
Q_N → g̃_k-1 → Q_N
Scaling Tiramisu

- FP16-model sensitive to communication
- FP16-model BW-bound (only 2.5x faster than FP32)
- almost ideal scaling for both precisions on Summit when gradient lag is used
Scaling DeepLabv3+

- FP16-model sensitive to communication
- FP16-model BW-bound (only 2.5x faster than FP32)
- excellent scaling for both precisions on Summit when gradient lag is used
Concurrency/Precision and Convergence

- **Tiramisu (FP16, 384 GPUs)**
- **Tiramisu (FP32, 384 GPUs)**
- **Tiramisu (FP16, 1536 GPUs)**
- **Tiramisu (FP32, 1536 GPUs)**
Concurrency/Precision and Convergence

-2.1x improvement in time to solution

Tiramisu (FP16, 384 GPUs)
Tiramisu (FP32, 384 GPUs)
Tiramisu (FP16, 1536 GPUs)
Tiramisu (FP32, 1536 GPUs)
Model/Lag and Convergence

- Tiramisu (no lag)
- DeepLabv3+ (no lag)
- DeepLabv3+ (lag=1)
Segmentation Analysis

- best result for intersection-over-union (IoU) obtained: \(~73\%\)
- result at large scale (batch-size > 1500): IoU \(~55\%\)
Conclusions

• deep learning and HPC converge, achieving *exascale* performance
• compute capabilities of contemporary HPC systems can be utilized to tackle challenging scientific deep learning problems
• HPO and convergence at scale still an open problem — but now we can do it
• software enhancements benefit deep learning community at large
• deep learning-powered techniques usher in a new era of precision analytics for various science areas
Thank You. Questions?