Project Adam and Mesh-TensorFlow

Ashkan Bagheri Jeddi
CSE 5194.01
AU 2019
Papers

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


Mesh-TensorFlow: Deep Learning for Supercomputers

Outline

• Neural Networks
• Giant Models
• Project Adam
• Mesh-TensorFlow
• Summary
Machine Learning and Deep Learning

Data → Objective Function → Humans → Deep Learning → Prediction
## Deep Learning

<table>
<thead>
<tr>
<th>Core AI</th>
<th>Data Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Recognition</td>
<td>Movie Recommendation</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td></td>
</tr>
<tr>
<td>Machine Translation</td>
<td></td>
</tr>
<tr>
<td>Representing Data is <em>Difficult</em></td>
<td>Representing Data is <em>Easy</em></td>
</tr>
<tr>
<td>Deep learning is <em>state-of-the-art</em></td>
<td>Deep learning has <em>no benefit</em></td>
</tr>
</tbody>
</table>
Neural Networks

Network learns complex intermediate representations without explicit labels!

https://www.tensorflow.org/about
Neural Networks

https://www.tensorflow.org/about
Giant Models

https://www.usenix.org/node/186213
Giant Models

https://www.usenix.org/node/186213
Project Adam:
Building an Efficient and Scalable Deep Learning Training System
Adam: Scalable Deep Learning Platform

Scalable Training Algorithm
• Asynchronous SGD

Scalable Model Partitioning
• Model-Parallelism

Scalable Model Parameter Store
• Data-Parallelism

Scalable Data Transformation
• Data Processing and Augmentation

High-value Capabilities with Broad Applicability
• Rectified Linear Units (ReLU)
• Momentum, Adaptive Gradient
• Dropout
System Architecture

Model-Parallelism

Data-Parallelism

Global Model Parameter Store

Model Replica

Model Workers

Data Shards

W \Delta W
Asynchronous Weight Update

- Maximize the performance on a single machine.
- Completely Asynchronous
- No locks on the shared weights
- Weight Updates are associative and commutative.

\[
\Delta W = \Delta W_7 + \Delta W_{24} + \Delta W_6 + \ldots
\]
Model Partitioning

- How much of the model can be fitted on a single machine?
- Memory Bandwidth
Asynchronous Batch Updates

\[ \Delta W = \Delta W_3 + \Delta W_1 + \Delta W_2 + \cdots \]
Local Weight Computation

\[ \Delta W = \alpha \delta \alpha \]

\[ \Delta W [O(N^2)] \]

\[ (\delta, \alpha) [O(kN)] \]

\[ \Delta W = \alpha \delta \alpha \]
Model Size Scaling
Parameter Server Performance

The diagram illustrates the performance of parameter updates with respect to the number of processor cores. It shows the comparison between local weight computation and weight updates over the network, as well as local weight computation with inputs over the network. The x-axis represents the number of processor cores, while the y-axis shows the billion parameter updates per second.
Scaling: ImageNet Training

[Graph showing the comparison between System with PS and Linear (no PS) in terms of Billion connections trained per second versus the number of machines.]
Baseline: MNIST

- 0.6B Connections
- 50K Examples
- Single-Box Performance was evaluated.
- Why does asynchrony help?

<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>
Results

2 times more accurate than prior best on ImageNet 22K.

<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>

![Graph showing Top-1 accuracy vs model size in millions.](chart.png)
Mesh-TensorFlow:
Deep Learning for Supercomputers
Synchronous Data-Parallelism

• Most Common Method of training NN on distributed hardware
• Parameters replicated on every core
• Batch Split Between Cores
• Sum (allreduce) Parameter Gradients
• Fast Communication on Different Networks
Synchronous Data-Parallelism

- Universal Model Architecture
- Fast to Compile
- Full Utilization
- Allreduce is fast on locally connected cores

- Cannot train giant models, entire model has to fit on every device.
Giant Models

• Applications: Machine Translation, Question Answering, Dialogue, Sentimental Analysis, etc
• Quality Improves with Model Size (e.g. OpenAI GPT-2)
• Infinite Training data freely available
• Train One big Model, Fine-tune for many tasks
  • Radford et al. 2018, OpenAI
  • BERT: Devlin et al. 2018, Google
Giant Models: Transformer LM

(0.14B Variables, Trained on Billion-Word Language Modeling Benchmark, According to Ray Kurzweil ...)

• ..., professor of musical economics at the University of California, Berkeley, the military’s "biggest challenge" might have been growing the region’s growing urban sprawl.

• ..., the chief executive of ITC Emerging Markets, the metals sector should contribute about $210 billion by year-end 2008 – a figure that has now risen to $3 billion.

• ..., the head of PMI, he has now written off all his employees fairly.

• ..., senior vice president of sales and marketing at Aspen, there is a percentage of the online response for the auto company escalating to 18 customers and increasing to 30 days a year.

• ..., director of the Economic Policy Institute, India ranks 21st out of 221 nations in the current budget deficit, which ranks 13th globally.

• ..., Victor Hesketh’s food agency that produces, sends and distributes the raw beef and pork through the state’s Food and Drug Administration, and they’re also allowed to take the lamb by their own account alongside the chicken and veal.

• ..., the author of "Smoking: The Letters of Hysteria and Reclining the State of South Carolina" (2007), 30 percent of liquor’s coconut is sold on the first batch.

• ..., an MIT student who is not involved in anything more than a stock-market move, the latest system of debt and bankruptcy parallels the completely unregulated collection of debt that emerged in the early 1990s, when a financial industry boom brought a financial crisis that spilled almost everywhere from the United States.
Giant Models: Transformer LM

(4.9B Variables, Trained on Billion-Word Language Modeling Benchmark, According to Ray Kurzweil ...)

• ... , chief technology officer for the US Department of Energy, aviation has "potential to be the largest and fastest growing source of consumer and commercial emissions."

• ... , the futurist son of the futurist who wrote The Singularity is Near, the "early days" of Google are not the right time to push forward with the next great leap.

• ... , creator of the first modern computer, the sexy cyborg was the brainchild of an MIT professor, Thomas Harris, and a former banking entrepreneur, Henry Lee, who was looking for an UNK, a light that could be recovered and used to light up the Internet.

• ... , the inventor of the modern personal computer, the shrinking human brain could eventually replace the Internet as a tool of human intelligence and imagination.

• ... , the expert and co-author of "The Singularity is Near: Comprehending the Technological Future of Engineering," people are looking for ways to protect and make their lives better.

• ... , creator of the Gaia hypothesis, earlier computer systems should become not just more efficient, but more-efficient, increasing their efficiency by reducing human errors (the unexpected, but often the regrettable) and even the number of errors.

• ... , who will make an appearance at this year’s Consumer Electronics Show in Las Vegas next week, these mobile gadgets will be able to "talk" to each other.

• ... , the futurist turned futurist, the onset of Alzheimer’s coincided precisely with the rate of unemployment in America.
Model Parallelism
(Splitting the Model between Different Devices)

• Can train giant Models
• Low Inference Latency (Split Computation)
• Tricky to Design
Model Parallelism by Device-Placement

- Assign Operations to Different Processors: Using TensorFlow
- Tricky to design efficient algorithm
- Giant Graph
Model Parallelism by Device-Placement

Example: GNMT (Wu et al. 2016)

https://arxiv.org/abs/1609.08144
Model Parallelism by Mesh-TensorFlow
(inspired by synchronous Data-Parallelism)

- Every Processor involved in every operation.
- SPMD (Single Program Multiple Devices)
- Collective Communication (e.g. allreduce)
Abilities

• Data-Parallelism (Batch-Splitting)
• Model-Parallelism (Model-Splitting)
• Spatial Splitting of large inputs (e.g. Images or Videos)
• Combinations of above
Hardware

• Homogenous set of processors
  • TPU-pods (two-dimensional supercomputers at Google)
  • multi-GPU
  • multi-CPU

• View as a logical n-dimensional mesh (Not Physical)

• Performance Depends on the architecture
Dimensions to Split

• Data-Parallelism: Split *Batch* Dimension
  • Tensors with *batch* dimension (e.g. activations) are split.
  • Tensors with no *batch* dimension (e.g. parameters) are replicated.

• Model Parallelism: Split Other Dimensions
  • Choose Different Dimensions to Split (e.g. dimensions representing the size of the hidden layers)
Operations

• *Usually* operations does not involve communication and are local.
  • Each processor computes its slice of the output from its slice of the inputs

• Some operations require collective communication
  • allreduce
Example: Perceptron

\[ Y = HV \]

\[ H = \text{ReLU}(XW) \]

- **Input Layer**
- **Hidden Layer**
- **Output Layer**

Weight Matrices:
- \( V_{h,d} \)
- \( W_{d,h} \)
Example: Perceptron
(Data Parallelism: Split batch dimension “b”)

- Processor 0
  - $X_{b/2,d}$
  - $H_{b/2,h}$
  - $Y_{b/2,d}$
  - $V_{h,d}$
  - $W_{d,h}$

- Processor 1
  - $X_{b/2,d}$
  - $H_{b/2,h}$
  - $Y_{b/2,d}$
  - $V_{h,d}$
  - $W_{d,h}$

- Split $b$
- Replicate
- Split $b$

$X_{b,d}$
$Y_{b,d}$
$H_{b,h}$
$V_{d,h}$
$W_{d,h}$
**Example: Perceptron**
(Model Parallelism: Split dimension “h”)

\[
Y = Y_0 + Y_1
\]

\[
Y_0 = H_{b,:h/2} V_{d,:h/2}
\]

\[
Y_1 = H_{b,h/2} V_{d,h/2}
\]

\[
V_{d,:h/2}
\]

\[
W_{d,:h/2}
\]

\[
X_{b,d}
\]

**Processor 0**

\[
Y = Y_0 + Y_1
\]

\[
Y_0 = H_{b,:h/2} V_{d,:h/2}
\]

\[
Y_1 = H_{b,h/2} V_{d,h/2}
\]

\[
V_{d,h/2}
\]

\[
W_{d,h/2}
\]

\[
X_{b,d}
\]

**Processor 1**

\[
Y_{b,d}
\]

\[
V_{d,h}
\]

\[
H_{b,h}
\]

\[
W_{d,h}
\]

\[
X_{b,d}
\]

\[
allreduce
\]

\[
Y_0 = H_{b,:h/2} V_{d,:h/2}
\]

\[
V_{d,:h/2}
\]

\[
W_{d,:h/2}
\]

\[
X_{b,d}
\]

\[
allreduce
\]

\[
Y_1 = H_{b,h/2} V_{d,h/2}
\]

\[
V_{d,h/2}
\]

\[
W_{d,h/2}
\]

\[
X_{b,d}
\]

\[
allreduce
\]

\[
Y = Y_0 + Y_1
\]

\[
V_{d,h}
\]

\[
H_{b,h}
\]

\[
W_{d,h}
\]

\[
X_{b,d}
\]

\[
allreduce
\]

\[
Y = Y_0 + Y_1
\]

\[
V_{d,h}
\]

\[
H_{b,h}
\]

\[
W_{d,h}
\]

\[
X_{b,d}
\]

\[
allreduce
\]

\[
Y = Y_0 + Y_1
\]

\[
V_{d,h}
\]

\[
H_{b,h}
\]

\[
W_{d,h}
\]

\[
X_{b,d}
\]
Example: Perceptron
(Model Parallelism: Split dimension “d”)

\[
Y_{b,::d/2} \\
V_{h,::d/2} \\
H = ReLU(H_0 + H_1) \\
H_0 = X_{b,::d/2} W_{::d/2,h} \\
W_{::d/2,h} \\
X_{b,::d/2} \\
\text{Processor 0}
\]

\[
Y_{b,d/2:} \\
V_{h,d/2:} \\
H = ReLU(H_0 + H_1) \\
H_1 = X_{b,d/2:} W_{d/2::h} \\
W_{d/2::h} \\
X_{b,d/2:} \\
\text{Processor 1}
\]

\[
Y_{b,d} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce}
\]

\[
H = ReLU(H_0 + H_1) \\
H_0 = X_{b,::d/2} W_{::d/2,h} \\
W_{::d/2,h} \\
X_{b,::d/2} \\
\text{Processor 0}
\]

\[
H = ReLU(H_0 + H_1) \\
H_1 = X_{b,d/2:} W_{d/2::h} \\
W_{d/2::h} \\
X_{b,d/2:} \\
\text{Processor 1}
\]

\[
Y_{b,d} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce}
\]

\[
H = ReLU(H_0 + H_1) \\
H_0 = X_{b,::d/2} W_{::d/2,h} \\
W_{::d/2,h} \\
X_{b,::d/2} \\
\text{Processor 0}
\]

\[
H = ReLU(H_0 + H_1) \\
H_1 = X_{b,d/2:} W_{d/2::h} \\
W_{d/2::h} \\
X_{b,d/2:} \\
\text{Processor 1}
\]

\[
Y_{b,d} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce} \\
\Rightarrow \text{allreduce}
\]
Example: Perceptron
(Data and Model Parallelism on 2D Mesh)

\[ Y_{b,d} \]
\[ V_{h,d} \]
\[ H_{b,h} \]
\[ W_{d,h} \]

- Split \( b \) across rows
- Replicate across columns

- Split \( h \) across columns
- Replicate across rows

- Tiled across all processors

Model-Parallelism (Dimension \( h \))

Data-Parallelism (Dimension \( b \))
More General

• Name all the tensor dimensions.
• Layout: partial map from tensor dimension name to mesh dimension across which that tensor dimension is split
  Example:
  layout = {“batch”: “processor_rows”, “hidden size”: “processor_columns”}
Example: Transformer Model

Data-Parallelism
- batch
- d_ff
- num_heads
- vocab_size

Model-Parallelism

Data-Parallelism and Model-Parallelism

https://arxiv.org/abs/1706.03762
Rules

• Picking a good layout: non-trivial
  • All expensive matmul, einsum (matrix multiplications and tensor contractions) operations should be split across all dimensions
  • Not allowed to split two dimensions of the same tensor across same dimensions of the mesh.
  • Splitting small sizes may result in more communication time rather than computation time.
    • Big Batch
    • Wide Layers
How To

• Mesh-TensorFlow does not work with TensorFlow 2.0
• Build a graph in python with named dimensions
• Define a mesh
• Choose layout
## Results:

Billion Word Language Modeling Benchmark
`languagemodel_wiki_noref_v128k_l1k`

Table 2: Transformer-Decoder Language Models: $d_{model} = 1024$, $d_k = d_v = 256$

<table>
<thead>
<tr>
<th>$d_f$</th>
<th>$heads$</th>
<th>Parameters (Billions)</th>
<th>Billion-Word Benchmark Word-Perplexity</th>
<th>Wikipedia Subword-Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>4</td>
<td>0.14</td>
<td>35.0</td>
<td>8.74</td>
</tr>
<tr>
<td>8192</td>
<td>8</td>
<td>0.22</td>
<td>31.7</td>
<td>8.03</td>
</tr>
<tr>
<td>16384</td>
<td>16</td>
<td>0.37</td>
<td>28.9</td>
<td>7.44</td>
</tr>
<tr>
<td>32768</td>
<td>32</td>
<td>0.67</td>
<td>26.8</td>
<td>6.99</td>
</tr>
<tr>
<td>65516</td>
<td>64</td>
<td>1.28</td>
<td>25.1</td>
<td>6.55</td>
</tr>
<tr>
<td>131072</td>
<td>128</td>
<td>2.48</td>
<td>24.1</td>
<td>6.24</td>
</tr>
<tr>
<td>262144</td>
<td>256</td>
<td>4.90</td>
<td>24.0 (23.5)</td>
<td>6.01</td>
</tr>
<tr>
<td>Prev Best DNN [20]</td>
<td></td>
<td>6.5</td>
<td>28.0</td>
<td></td>
</tr>
<tr>
<td>Best DNN Ensemble [17]</td>
<td></td>
<td></td>
<td>26.1</td>
<td></td>
</tr>
<tr>
<td>Best Ensemble (different methods)[17]</td>
<td></td>
<td>$&gt; 100$</td>
<td>23.7</td>
<td></td>
</tr>
</tbody>
</table>
Results:
WMT14 En-Fr and WMT14 En-De

Table 3: Transformer Machine-Translation Results. $d_{model} = 1024$, $d_k = d_v = 128$

<table>
<thead>
<tr>
<th>$d_{ff}$</th>
<th>heads</th>
<th>$d_k, d_v$</th>
<th>Parameters (Billions)</th>
<th>WMT14 EN-DE BLEU</th>
<th>WMT14 EN-FR BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048</td>
<td>4</td>
<td>128</td>
<td>0.15</td>
<td>25.5</td>
<td>41.8</td>
</tr>
<tr>
<td>4096</td>
<td>8</td>
<td>128</td>
<td>0.24</td>
<td>26.5</td>
<td>42.5</td>
</tr>
<tr>
<td>8192</td>
<td>16</td>
<td>128</td>
<td>0.42</td>
<td>27.1</td>
<td>43.3</td>
</tr>
<tr>
<td>16384</td>
<td>32</td>
<td>128</td>
<td>0.77</td>
<td>27.5</td>
<td>43.5</td>
</tr>
<tr>
<td>32768</td>
<td>64</td>
<td>128</td>
<td>1.48</td>
<td>27.5</td>
<td>43.8</td>
</tr>
<tr>
<td>65536</td>
<td>128</td>
<td>128</td>
<td>2.89</td>
<td>26.7</td>
<td><strong>43.9</strong></td>
</tr>
</tbody>
</table>
| 4096    | 16    | 64        | 0.21                  | **28.4**         | 41.8             | [21]
Summary

• Adam is a scalable deep learning training system built from commodity PCs.
• Adam achieves scalability and efficiency via whole-system co-design and exploiting asynchrony.
• Mesh-TensorFlow is built for training giant models on TPU pods.
• Both methods use Data-Parallelism and Model-Parallelism.