Scaling Deep Learning

Bryan Catanzaro
@ctnzr
Computer vision: Find coffee mug
Computer vision: Find coffee mug
Why is computer vision hard?

Andrew Ng
Artificial Neural Networks

Neurons in the brain  Deep Learning: Neural network

Andrew Ng
Computer vision: Find coffee mug
What is a neural network?

Data (image)

\[ x_1 \in \mathbb{R}^5, \quad x_2 \in \mathbb{R}^5 \]

\[ x_2 = (W_1 x_1)_+ \]

\[ x_3 = (W_2 x_2)_+ \]

Yes/No

(Mug or not?)

Andrew Ng
Supervised learning
(learning from tagged data)

\[ X \rightarrow Y \]

Input: Image
Output tag: Yes/No
(Is it a coffee mug?)

Data:
- Coffee mug: Yes
- Fork: No

Learning $X \rightarrow Y$ mappings is hugely useful
What do we want AI to do?

Drive us to work

Serve drinks?

Help us find things

Guide us to content

Scientists See Promise in Deep-Learning Programs

Keep us organized

Help us communicate

Drive us to work

帮助我们沟通
OCR-based Translation App
Baidu IDL
AskADoctor can assess 520 different diseases, representing ~90 percent of the most common medical problems.
Image Captioning
Baidu IDL

A yellow bus driving down a road with green trees and green grass in the background.

Living room with white couch and blue carpeting. Room in apartment gets some afternoon sun.
Sample questions and answers
Natural User Interfaces

• Goal: Make interacting with computers as natural as interacting with humans

• AI problems:
  – Speech recognition
  – Emotional recognition
  – Semantic understanding
  – Dialog systems
  – Speech synthesis
Demo

- Deep Speech public API
Machine learning in practice

- Enormous amounts of research time spent inventing new features.

Think really hard...  Hack up in Matlab

Run on workstation

Idea

Test

Code
Why Deep Learning?

1. Scale Matters
   - Bigger models usually win

2. Data Matters
   - More data means less cleverness necessary

3. Productivity Matters
   - Teams with better tools can try out more ideas
Scaling up

• Make progress on AI by focusing on systems
  – Make models bigger
  – Tackle more data
  – Reduce research cycle time
    • Accelerate large-scale experiments
Training Deep Neural Networks

\[ y_j = f \left( \sum_i w_{ij} x_i \right) \]

- Computation dominated by dot products
- Multiple inputs, multiple outputs, batch means GEMM
  - Compute bound
- Convolutional layers even more compute bound
Computational Characteristics

• High arithmetic intensity
  – Arithmetic operations / byte of data
  – $O(\text{Exaflops}) / O(\text{Terabytes}) : 10^6$
    • In contrast, some other ML training jobs are
      $O(\text{Petaflops})/O(\text{Petabytes}) = 10^0$

• Medium size datasets
  – Generally fit on 1 node
  – HDFS, fault tolerance, disk I/O not bottlenecks

Training 1 model: ~20 Exaflops
Deep Neural Network training is HPC

• Turnaround time is key

• Use most efficient hardware
  – Parallel, heterogeneous computing
  – Fast interconnect (PCIe, Infiniband)

• Push strong scalability
  – Models and data have to be of commensurate size
Training: Stochastic Gradient Descent

\[ w' = w - \frac{\gamma}{n} \sum_i \nabla_w Q(x_i, w) \]

• Simple algorithm
  – Add momentum to power through local minima
  – Compute gradient by backpropagation

• Operates on minibatches
  – This makes it a GEMM problem instead of GEMV

• Choose minibatches stochastically
  – Important to avoid memorizing training order

• Difficult to parallelize
  – Prefers lots of small steps
  – Increasing minibatch size not always helpful
Limitations of batching

Spending 2x the work picking a direction
Doesn’t reduce iteration count by 2x
SVAIL Infrastructure

- Software: CUDA, MPI, Majel (SVAIL internal library)
- Hardware:
  - NVIDIA GeForce GTX Titan X
  - Titan X x8
  - Mellanox Interconnect

@ctnzr
Node Architecture

- All pairs of GPUs communicate simultaneously over PCIe Gen 3 x16
- Groups of 4 GPUs form Peer to Peer domain
- Avoid moving data to CPUs or across QPI
Parallelism

Model Parallel

Data Parallel

MPI_Allreduce()
Speech Recognition: Traditional ASR

- Getting higher performance is hard
- Improve each stage by engineering

![Graph showing the relationship between data and model size with accuracy. The graph illustrates that as data and model size increase, accuracy improves, but the rate of improvement slows down. There is an annotation indicating expert engineering is needed to continue improving accuracy.]
Speech recognition: Traditional ASR

- Huge investment in features for speech!
  - Decades of work to get very small improvements
Speech Recognition 2: Deep Learning!

• Since 2011, deep learning for features

“The quick brown fox jumps over the lazy dog.”
Speech Recognition 2: Deep Learning!

- With more data, DL acoustic models perform better than traditional models
Speech Recognition 3: “Deep Speech”

- End-to-end learning

“The quick brown fox jumps over the lazy dog.”
Speech Recognition 3: “Deep Speech”

- We believe end-to-end DL works better when we have big models and lots of data
End-to-end speech with DL

- Deep neural network predicts characters directly from audio
Recurrent Network

- RNNs model temporal dependence
- Various flavors used in many applications
  - LSTM, GRU, Bidirectional, ...
  - Especially sequential data (time series, text, etc.)
- Sequential dependence complicates parallelism
Connectionist Temporal Classification

\[ P(\_\_T H \_\_\_E \_\_C \_\_A A A \_\_T T \_\_\_) \]

\[ P(\_T \_\_H \_\_E E \_\_\_\_C \_\_A A \_\_T \_\_\_) \]

\[ P(\text{THE—CAT—}) \]
warp-ctc

- Recently open sourced our CTC implementation (sorts from ModernGPU)
- Efficient, parallel CPU and GPU backend
- 100-400X faster than other implementations
- Apache license, C interface

https://github.com/baidu-research/warp-ctc

ctcStatus_t compute_ctc_loss(const float* const activations,
float* gradients,
const int* const flat_labels,
const int* const label_lengths,
const int* const input_lengths,
int alphabet_size,
int minibatch,
float *costs,
void *workspace,
ctcComputeInfo info);
Training sets

- Train on 45k hours (~5 years) of data
  - Still growing

- Languages
  - English
  - Mandarin

- End-to-end deep learning is key to assembling large datasets
All-reduce

- We implemented our own all-reduce out of send and receive
- Several algorithm choices based on size
- Careful attention to affinity and topology

<table>
<thead>
<tr>
<th>GPU</th>
<th>OpenMPI all-reduce</th>
<th>Our all-reduce</th>
<th>Performance Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>55359.1</td>
<td>2587.4</td>
<td>21.4</td>
</tr>
<tr>
<td>8</td>
<td>48881.6</td>
<td>2470.9</td>
<td>19.8</td>
</tr>
<tr>
<td>16</td>
<td>21562.6</td>
<td>1393.7</td>
<td>15.5</td>
</tr>
<tr>
<td>32</td>
<td>8191.8</td>
<td>1339.6</td>
<td>6.1</td>
</tr>
<tr>
<td>64</td>
<td>1395.2</td>
<td>611.0</td>
<td>2.3</td>
</tr>
<tr>
<td>128</td>
<td>1602.1</td>
<td>422.6</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Scalability

- Batch size is hard to increase
  - algorithm, memory limits
- Performance at small batch sizes leads to scalability limits
Performance for RNN training

- 55% of GPU FMA peak using a single GPU
- ~48% of peak using 8 GPUs in one node
- Weak scaling very efficient, albeit algorithmically challenged
Strong scaling RNNs with Persistent Kernels

- Strong scaling is hard!

![Graph showing Teraflop/s vs GPU Count with linear and standard trends. Credit: G. Diamos, ICML 2016](image)
Strong scaling RNNs with Persistent Kernels

• Small batch size bad for standard RNN:
  – Less reuse of parameters
  – Bad SIMD efficiency

• But RNNs reuse parameters across time
  – Can we stash them in register file?
  – Make RNNs compute limited at small batch?
  – Enable strong scaling?

• Persistent Kernels hard to implement:
  – Require global synchronization (CUDA hates this)
  – MB of parameters pinned to register file

• Limitations:
  – Size, architecture (RNN vs. LSTM vs. GRU etc.)

[G. Diamos, ICML 2016]
Persistent Kernel implementation

- SASS assembly for Maxwell (using Maxas)
- Supports up to 1152 neurons on TitanX
  - 5 MB of user data pinned in registers!
- SW Pipeline following limiters:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication latency</td>
<td></td>
</tr>
<tr>
<td>Barrier latency</td>
<td></td>
</tr>
<tr>
<td>Memory bandwidth</td>
<td></td>
</tr>
<tr>
<td>Memory latency</td>
<td></td>
</tr>
<tr>
<td>FP throughput</td>
<td></td>
</tr>
</tbody>
</table>

[Source: G. Diamos, ICML 2016]
Persistent Kernel Performance (TitanX)

- [G. Diamos, ICML 2016]

- [https://github.com/baidu-research/persistent-rnn](https://github.com/baidu-research/persistent-rnn)
Strong scaling RNNs with Persistent Kernels

![Graph showing performance comparison between Linear, Persistent, and Standard RNNs with varying GPU counts.]

[G. Diamos, ICML 2016]
Determinism

- Determinism very important
- So much randomness, hard to tell if you have a bug
- Networks train despite bugs, although accuracy impaired
- Reproducibility is important
  - For the usual scientific reasons
  - Progress not possible without reproducibility

- We use synchronous SGD
Precision

- FP16 mostly works
  - Use FP32 for softmax and weight updates
- More sensitive to labeling error
Batch Dispatch

CPU Server

CPU
- Thread 1
- Thread 2
- Thread 3
- Thread 4

[C. Fougner]
Batch Dispatch
Batch Dispatch Performance

212M parameter RNN - 98%ile latency

Latency (ms)

Number of concurrent users

4X more users at acceptable latency

[C. Fougner]
Thoughts

- Computationally dense processors (like GPUs) required
- Programmability
  - We don’t know the algorithms of the future
- Lower precision
  - But not too low
  - Interesting algorithm/dataset engineering here
- We need better support for multi-GPU
  - E.g. Atomics between GPUs, collectives
  - Looking forward to NVLink
Conclusion

• Deep Learning is solving many hard problems

• Training deep neural networks is an HPC problem

• Scaling brings AI progress!
Thanks

• Andrew Ng, Adam Coates, Awni Hannun, Patrick LeGresley, Greg Diamos, Chris Fougner ... and all of SVAIL

Bryan Catanzaro
@ctnzr