Large-Scale Deep Learning for Intelligent Computer Systems

Jeff Dean

Google Brain team in collaboration with many other teams
Growing Use of Deep Learning at Google

Across many products/areas:
- Android
- Apps
- GMail
- Image Understanding
- Maps
- NLP
- Photos
- Robotics
- Speech
- Translation
- many research uses..
- YouTube
- … many others …

# of directories containing model description files

Time (quarterly)
Outline

Two generations of deep learning software systems:

- 1st generation: DistBelief [Dean et al., NIPS 2012]
- 2nd generation: TensorFlow (unpublished)

An overview of how we use these in research and products

Plus, ...a new approach for training (people, not models)
Google Brain project started in 2011, with a focus on pushing state-of-the-art in neural networks. Initial emphasis:

- use large datasets, and
- large amounts of computation

to push boundaries of what is possible in perception and language understanding
Plenty of raw data

- **Text**: trillions of words of English + other languages
- **Visual data**: billions of images and videos
- **Audio**: tens of thousands of hours of speech per day
- **User activity**: queries, marking messages spam, etc.
- **Knowledge graph**: billions of labelled relation triples
- ...

How can we build systems that truly understand this data?
This movie should have NEVER been made. From the poorly done animation, to the beyond bad acting. I am not sure at what point the people behind this movie said "Ok, looks good! Lets do it!" I was in awe of how truly horrid this movie was."
Turnaround Time and Effect on Research

● Minutes, Hours:
  ○ Interactive research! Instant gratification!

● 1-4 days
  ○ Tolerable
  ○ Interactivity replaced by running many experiments in parallel

● 1-4 weeks:
  ○ High value experiments only
  ○ Progress stalls

● >1 month
  ○ Don’t even try
Important Property of Neural Networks

Results get better with

more data +
bigger models +
more computation

(Better algorithms, new insights and improved techniques always help, too!)
How Can We Train Large, Powerful Models Quickly?

- Exploit many kinds of parallelism
  - Model parallelism
  - Data parallelism
Model Parallelism

Representation

Layer 2

Layer 1

Input Image
Model Parallelism

(Sometimes) Local Receptive Fields

Layer N

... 

Layer 1

Input data

Representation
Model Parallelism: Partition model across machines
Data Parallelism

Parameter Servers

Model Replicas

Data
Data Parallelism

Parameter Servers

Model Replicas

Data

$p$
Data Parallelism

Parameter Servers

Model Replicas

Data

$p$

$\Delta p$
Data Parallelism

\[ p' = p + \Delta p \]

Parameter Servers

Model Replicas

Data

\( \Delta p \)

\( p \)
Data Parallelism

Parameter Servers

$p' = p + \Delta p$

Model Replicas

Data
Data Parallelism

Parameter Servers

\[ p' \]
\[ \Delta p' \]

Model Replicas

Data

...
Data Parallelism

Parameter Servers

$p'' = p' + \Delta p$

\[ \Delta p' \]

$p'$

Model
Replicas

Data
Data Parallelism

Parameter Servers

\[ p'' = p' + \Delta p \]

Model Replicas

Data
Data Parallelism Choices

Can do this **synchronously**:

- N replicas equivalent to an N times larger batch size
- Pro: No noise
- Con: Less fault tolerant (requires recovery if any single machine fails)

Can do this **asynchronously**:

- Con: Noise in gradients
- Pro: Relatively fault tolerant (failure in model replica doesn’t block other replicas)

(Or **hybrid**: M asynchronous groups of N synchronous replicas)
Data Parallelism Considerations

Want model computation time to be large relative to time to send/receive parameters over network

Models with fewer parameters, that reuse each parameter multiple times in the computation

- Mini-batches of size $B$ reuse parameters $B$ times

Certain model structures reuse parameter many times within each example:

- **Convolutional models** tend to reuse hundreds or thousands of times per example (for different spatial positions)
- **Recurrent models** (LSTMs, RNNs) tend to reuse tens to hundreds of times (for unrolling through $T$ time steps during training)
What are some ways that deep learning is having a significant impact at Google?
Sequence to Sequence Models

Oriol Vinyals, Ilya Sutskever & Quoc Le started looking at how to map one sequence to another sequence:

The lazy brown fox jumped over the ...
The lazy brown fox jumped over the ...
Connect two, you get a machine translation system

The quick brown fox jumped over the ...

Le renard brun rapide saute par dessus ...
It works well

<table>
<thead>
<tr>
<th></th>
<th>WMT’14</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-of-the-art</td>
<td></td>
<td>37.0</td>
</tr>
<tr>
<td>Neural Translation Model</td>
<td></td>
<td>37.3</td>
</tr>
</tbody>
</table>

Sequence to Sequence Learning with Neural Networks
Ilya Sutskever, Oriol Vinyals, Quoc V. Le (NIPS 2014)

Addressing Rare Word Problems in Neural Translation Models (arxiv.org/abs/1410.8206)
Thang Luong, Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Wojciech Zaremba
... or a chatbot.

Hi Shiva, how are you?

I'm fine, thank you!
User> hello, i have a problem with my machine
BrainStop> hi
BrainStop> this is shiva.
BrainStop> how are you doing today?
User> hi shiva, how are you?
BrainStop> i'm fine, thank you!
BrainStop> how may i assist you today?
User> i want to access using vpn
BrainStop> currently the computer is connected to corp network?
User> no
BrainStop> check this solution:
BrainStop> https://INTERNAL_HOSTNAME_REDACTED/articles/connecting-to-disk-connecting-from-ssl-vpn#heading-4
User> thanks, bye!
BrainStop> thank you.
... or a parser.

Allen is locked in, regardless of his situ...
It works well

Completely learned parser with no parsing-specific code

State of the art results on WSJ 23 parsing task


http://arxiv.org/abs/1412.7449
… or something that can learn graph algorithms

output:
Convex Hull
(or Delauney Triangulation)
(or Travelling Salesman tour)

input:
collection of points

Object Recognition Improvement Over Time

Predicted Human Performance

ImageNet Challenge Winners

“cat”
Image Models

Module with 6 separate convolutional layers

24 layers deep

“cat”

Going Deeper with Convolutions
Szegedy et al. CVPR 2015

2014 ImageNet winner: 6.66% top-5 error rate
Good Fine-Grained Classification

“hibiscus”

“dahlia”
Good Generalization

Both recognized as “meal”
Sensible Errors

“snake”

“dog”
Works in practice… for real users

Wow.

The new Google plus photo search is a bit insane.

I didn’t tag those... ;)}
Google Plus photo search is awesome. Searched with keyword 'Drawing' to find all my scribbles at once :D

Works in practice... for real users
Maria’s Bakery Inn 超群饼屋
CIANO MOTOR ENGINEERS
MECHANICAL REPAIRS TO ALL MAKES AND MODELS

Specialising in BMW, MINI & TOYOTA

8 REGATTA ROAD FIVE DOCK 9745 3173
Connect sequence and image models, you get a captioning system

“A close up of a child holding a stuffed animal”
It works well (BLEU scores)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Previous SOTA</th>
<th>Show &amp; Tell</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS COCO</td>
<td>N/A</td>
<td>67</td>
<td>69</td>
</tr>
<tr>
<td>FLICKR</td>
<td>49</td>
<td>63</td>
<td>68</td>
</tr>
<tr>
<td>PASCAL (xfer learning)</td>
<td>25</td>
<td>59</td>
<td>68</td>
</tr>
<tr>
<td>SBU (weak label)</td>
<td>11</td>
<td>27</td>
<td>N/A</td>
</tr>
</tbody>
</table>

*Show and Tell: A Neural Image Caption Generator, Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan (CVPR 2015)*
TensorFlow: Second Generation Deep Learning System
Motivations

DistBelief (1st system) was great for scalability

Not as flexible as we wanted for research purposes

Better understanding of problem space allowed us to make some dramatic simplifications
TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more
graph = tf.Graph()
with graph.as_default():
    examples = tf.constant(train_dataset)
    labels = tf.constant(train_labels)

W = tf.Variable(tf.truncated_normal([image_size * image_size, num_labels]))  # Variables
b = tf.Variable(tf.zeros([num_labels]))

logits = tf.matmul(examples, W) + b  # Training computation
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits, labels))

optimizer = tf.train.GradientDescentOptimizer(0.5).minimize(loss)  # Optimizer to use
prediction = tf.nn.softmax(logits)  # Predictions for training data
TensorFlow Example (Batch Logistic Regression)

graph = tf.Graph()
with graph.AsDefault():
  examples = tf.constant(train_dataset)
  labels = tf.constant(train_labels)

W = tf.Variable(tf.truncated_normal([image_size * image_size, num_labels]))  # Variables
b = tf.Variable(tf.zeros([num_labels]))

logits = tf.matmul(examples, W) + b  # Training computation
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits, labels))

optimizer = tf.train.GradientDescentOptimizer(0.5).minimize(loss)  # Optimizer to use
prediction = tf.nn.softmax(logits)  # Predictions for training data

with tf.Session(graph=graph) as session:
  tf.initialize_all_variables().run()
  for step in xrange(num_steps):
    _, l, predictions = session.run([optimizer, loss, prediction])  # Run & return 3 values
    if (step % 100 == 0):
      print 'Loss at step', step, ':', l
      print 'Training accuracy: %.1f%%' % accuracy(predictions, labels)
Computation is a dataflow graph

Graph of Nodes, also called Operations or ops.

- biases
- weights
- examples
- labels

MatMul → Add → Relu → Xent
Computation is a dataflow graph with tensors.

Edges are N-dimensional arrays: Tensors

- biases
- weights
- examples
- labels
- MatMul
- Add
- Relu
- Xent
Computation is a dataflow graph with state

'Biases' is a variable

Some ops compute gradients

-= updates biases

biases

... Add ...

Mul -=

learning rate
Computation is a dataflow graph

Devices: Processes, Machines, GPUs, etc
TensorFlow: Expressing High-Level ML Computations

Automatically runs models on range of platforms:

from phones ...

to single machines (CPU and/or GPUs) ...

to distributed systems of many 100s of GPU cards
What is in a name?

● **Tensor**: N-dimensional array
  ○ 1-dimension: Vector
  ○ 2-dimension: Matrix
  ○ Represent many dimensional data flowing through the graph
    ■ e.g. Image represented as 3-d tensor rows, cols, color

● **Flow**: Computation based on data flow graphs
  ○ Lots of operations (nodes in the graph) applied to data flowing through

● **Tensors flow through the graph** → “**TensorFlow**”
  ○ Edges represent the tensors (data)
  ○ Nodes represent the processing
Flexible

- General computational infrastructure
  - Deep Learning support is a set of libraries on top of the core
  - Also useful for other machine learning algorithms
  - Possibly even for high performance computing (HPC) work
  - Abstracts away the underlying devices/computational hardware
Extensible

- Core system defines a number of standard *operations* and *kernels* (device-specific implementations of operations)
- Easy to define new operators and/or kernels
Deep Learning in TensorFlow

- Typical neural net “layer” maps to one or more tensor operations
  - e.g. Hidden Layer: \( \text{activations} = \text{Relu}(\text{weights} \times \text{inputs} + \text{biases}) \)

- Library of operations specialized for Deep Learning
  - Dozens of high-level operations: 2D and 3D convolutions, Pooling, Softmax, ...
  - Standard losses e.g. CrossEntropy, L1, L2
  - Various optimizers e.g. Gradient Descent, AdaGrad, L-BFGS, ...

- Auto Differentiation

- Easy to experiment with (or combine!) a wide variety of different models:
  - LSTMs, convolutional models, attention models, reinforcement learning, embedding models, Neural Turing Machine-like models, ...
No distinct Parameter Server subsystem

- Parameters are now just stateful nodes in the graph
- Data parallel training just a more complex graph
Synchronous Variant

update

add

gradient

model computation

parameters
Nurturing Great Researchers

- We’re always looking for people with the potential to become excellent machine learning researchers
- The resurgence of deep learning in the last few years has caused a surge of interest of people who want to learn more and conduct research in this area
Google Brain Residency Program

New one year immersion program in deep learning research

Learn to conduct deep learning research w/experts in our team

- Fixed one-year employment with salary, benefits, ...
- Goal after one year is to have conducted several research projects
- Interesting problems, TensorFlow, and access to computational resources
Google Brain Residency Program

Who should apply?

- people with BSc or MSc, ideally in computer science, mathematics or statistics
- completed coursework in calculus, linear algebra, and probability, or equiv.
- programming experience
- motivated, hard working, and have a strong interest in Deep Learning
Google Brain Residency Program

Program Application & Timeline

- **Applications Open!**
  - Oct 22 2015 - Jan 15 2016

- **Applications Review**
  - Selected candidates will be contacted in February for interviews

- **Phone & Onsite Interviews**

- **Application Result Announcement!**

- **Program Start & Orientation!**
  - June 6 2016
Google Brain Residency Program

For more information:

g.co/brainresidency

Contact us:

brain-residency@google.com
Questions?