Taming Service Variability, Building Worldwide Systems, and Scaling Deep Learning

Jeff Dean
Google

Joint work with many collaborators
A Talk in Two Parts...

- **Part 1:** Building and deploying large-scale services
  - achieving low latency in shared environments
  - dealing with worldwide deployment and operations

- **Part 2:** Learning high-level representations via deep learning
  - automatically, from raw data
  - useful in wide variety of domains
Faster Is Better
Faster Is Better
Large Fanout Services

- Ad System
- Frontend Web Server
- Super root
- Images
- Local
- News
- Video
- Blogs
- Books
- Cache servers

query
Shared Environment

Linux
Shared Environment

- file system
- chunkserver

- Linux
Shared Environment

- file system
- chunkserver
- scheduling system
- Linux
Shared Environment

Various other system services
File system
Chunkserver
Scheduling system
Linux
Shared Environment

- Bigtable tablet server
- Various other system services
- File system chunkserver
- Scheduling system
- Linux
Shared Environment

cpu intensive job

random MapReduce #1

Bigtable tablet server

various other system services

file system chunkserver

scheduling system

Linux
Shared Environment

- random app #2
- cpu intensive job
- random app
- random MapReduce #1
- Bigtable tablet server
- various other system services
- file system chunks server
- scheduling system
- Linux
Shared Environment

- **Huge benefit:** greatly increased utilization

- **... but hard to predict effects increase variability**
  - network congestion
  - background activities
  - bursts of foreground activity
  - not just your jobs, but everyone else’s jobs, too
  - not static: change happening constantly

- **Exacerbated by large fanout systems**
The Problem with Shared Environments
The Problem with Shared Environments
The Problem with Shared Environments

• Server with 10 ms avg. but 1 sec 99%ile latency
  – touch 1 of these: 1% of requests take ≥ 1 sec
  – touch 100 of these: 63% of requests take ≥ 1 sec
Tolerating Faults vs. Tolerating Variability

• Tolerating faults:
  – rely on extra resources
    • RAIDed disks, ECC memory, dist. system components, etc.
    – *make a reliable whole out of unreliable parts*

• Tolerating variability:
  – use these same extra resources
    – *make a predictable whole out of unpredictable parts*

• Times scales are very different:
  – variability: 1000s of disruptions/sec, scale of **milliseconds**
  – faults: 10s of failures per day, scale of **tens of seconds**
Latency Tolerating Techniques

• **Cross request adaptation**
  – examine recent behavior
  – take action to improve latency of future requests
  – typically relate to balancing load across set of servers
  – time scale: 10s of seconds to minutes

• **Within request adaptation**
  – cope with slow subsystems in context of higher level request
  – time scale: right now, while user is waiting
Backup Requests w/ Cross-Server Cancellation

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one.
Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one.
Backup Requests w/ Cross-Server Cancellation

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one
Backup Requests w/ Cross-Server Cancellation

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one.

Each request identifies other server(s) to which request might be sent.
Backup Requests w/ Cross-Server Cancellation

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one.

Each request identifies other server(s) to which request might be sent.
Backup Requests w/ Cross-Server Cancellation

Server 1
- req 3
- req 6

Server 2
- req 9
  also: server 1

Client

“Server 2: Starting req 9”

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one

Each request identifies other server(s) to which request might be sent
Backup Requests w/ Cross-Server Cancellation

Each request identifies other server(s) to which request might be sent

Similar to Michael Mitzenmacher’s work on “The Power of Two Choices”, except send to both, rather than just picking “best” one
Backup Requests: Bad Case

Server 1

req 3

Server 2

req 5

Client
Backup Requests: Bad Case

Server 1

req 3

Server 2

req 5

req 9

req 9
Backup Requests: Bad Case

Server 1
- req 3
  - req 9
    - also: server 2

Server 2
- req 5

req 9
Backup Requests: Bad Case

Server 1

- req 3
- req 9
  also: server 2

Server 2

- req 5
- req 9
  also: server 1

Client
Backup Requests: Bad Case

Server 1

req 9
also: server 2

Client

Server 2

req 9
also: server 1
Backup Requests: Bad Case

Server 1

req 9
also: server 2

Server 2

req 9
also: server 1

“Server 1: Starting req 9”

“Server 2: Starting req 9”

Client
Backup Requests: Bad Case

Server 1

"Server 2: Starting req 9"

Client

Server 2

"Server 1: Starting req 9"

req 9 also: server 2

req 9 also: server 1
Backup Requests: Bad Case

Server 1
- req 9
- also: server 2

Server 2
- req 9
- also: server 1

Client
- reply
Backup Requests: Bad Case

Server 1

- req 9
- also: server 2

Server 2

- req 9
- also: server 1

reply
Backup Requests: Bad Case

Server 1

req 9
also: server 2

reply

Server 2

req 9
also: server 1

Likelihood of this bad case is reduced with lower latency networks
Backup Requests w/ Cross-Server Cancellation

• Read operations in distributed file system client
  – send request to first replica
  – wait 2 ms, and send to second replica
  – servers cancel request on other replica when starting read
• Measure higher-level monitoring ops that touch disk
Backup Requests w/ Cross-Server Cancellation

• Read operations in distributed file system client
  – send request to first replica
  – wait 2 ms, and send to second replica
  – servers cancel request on other replica when starting read
• Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td>Backup after 2 ms</td>
<td></td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
</tbody>
</table>
Backup Requests w/ Cross-Server Cancellation

- Read operations in distributed file system client
  - send request to first replica
  - wait 2 ms, and send to second replica
  - servers cancel request on other replica when starting read
- Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td>Backup after 2 ms</td>
<td></td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
</tbody>
</table>

-43%
Backup Requests w/ Cross-Server Cancellation

• Read operations in distributed file system client
  – send request to first replica
  – wait 2 ms, and send to second replica
  – servers cancel request on other replica when starting read

• Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
<tr>
<td>+Terasort</td>
<td>No backups</td>
<td>24 ms</td>
<td>56 ms</td>
<td>108 ms</td>
<td>159 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>19 ms</td>
<td>35 ms</td>
<td>67 ms</td>
<td>108 ms</td>
</tr>
</tbody>
</table>
• **Read operations in distributed file system client**
  – send request to first replica
  – wait 2 ms, and send to second replica
  – servers cancel request on other replica when starting read
• **Measure higher-level monitoring ops that touch disk**

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
<tr>
<td>+Terasort</td>
<td>No backups</td>
<td>24 ms</td>
<td>56 ms</td>
<td><strong>108 ms</strong></td>
<td>159 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>19 ms</td>
<td>35 ms</td>
<td><strong>67 ms</strong></td>
<td>108 ms</td>
</tr>
</tbody>
</table>

-38%
Backup Requests w/ Cross-Server Cancellation

- Read operations in distributed file system client
  - send request to first replica
  - wait 2 ms, and send to second replica
  - servers cancel request on other replica when starting read
- Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
<tr>
<td>+Terasort</td>
<td>No backups</td>
<td>24 ms</td>
<td>56 ms</td>
<td>108 ms</td>
<td>159 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>19 ms</td>
<td>35 ms</td>
<td>67 ms</td>
<td>108 ms</td>
</tr>
</tbody>
</table>

Backups cause about ~1% extra disk reads
Backup Requests w/ Cross-Server Cancellation

- Read operations in distributed file system client
  - send request to first replica
  - wait 2 ms, and send to second replica
  - servers cancel request on other replica when starting read
- Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
<tr>
<td>+Terasort</td>
<td>No backups</td>
<td>24 ms</td>
<td>56 ms</td>
<td>108 ms</td>
<td>159 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>19 ms</td>
<td>35 ms</td>
<td>67 ms</td>
<td>108 ms</td>
</tr>
</tbody>
</table>
Backup Requests w/ Cross-Server Cancellation

- Read operations in distributed file system client
  - send request to first replica
  - wait 2 ms, and send to second replica
  - servers cancel request on other replica when starting read
- Measure higher-level monitoring ops that touch disk

<table>
<thead>
<tr>
<th>Cluster state</th>
<th>Policy</th>
<th>50%ile</th>
<th>90%ile</th>
<th>99%ile</th>
<th>99.9%ile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly idle</td>
<td>No backups</td>
<td>19 ms</td>
<td>38 ms</td>
<td>67 ms</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>16 ms</td>
<td>28 ms</td>
<td>38 ms</td>
<td>51 ms</td>
</tr>
<tr>
<td>+Terasort</td>
<td>No backups</td>
<td>24 ms</td>
<td>56 ms</td>
<td>108 ms</td>
<td>159 ms</td>
</tr>
<tr>
<td></td>
<td>Backup after 2 ms</td>
<td>19 ms</td>
<td>35 ms</td>
<td>67 ms</td>
<td>108 ms</td>
</tr>
</tbody>
</table>

Backups w/big sort job gives same read latencies as no backups w/ idle cluster!
Cluster-Level Services

- Our earliest systems made things easier within a cluster:
  - GFS/Colossus: reliable cluster-level file system
  - MapReduce: reliable large-scale computations
  - Cluster scheduling system: abstracted individual machines
  - BigTable: automatic scaling of higher-level structured storage
Cluster-Level Services

• Our earliest systems made things easier within a cluster:
  – GFS/Colossus: reliable cluster-level file system
  – MapReduce: reliable large-scale computations
  – Cluster scheduling system: abstracted individual machines
  – BigTable: automatic scaling of higher-level structured storage

• Solve many problems, but leave many cross-cluster issues to human-level operators
  – different copies of same dataset have different names
  – moving or deploying new service replicas is labor intensive
- Google Docs depends on 50+ services

- Want to perform high-level operations:
  - run it in half a dozen clusters
  - release a new version
  - fix it on the fly in an emergency
  - move some (possibly shared) sub-services to another cluster
Dependencies

• Google Docs depends on 50+ services

• Want to perform high-level operations:
  – run it in half a dozen clusters
  – release a new version
  – fix it on the fly in an emergency
  – move some (possibly shared) sub-services to another cluster

Important that scheduling service has worldwide view
World-Wide Systems

- Future scale: ~$10^6$ to $10^7$ machines, spread at 100s to 1000s of locations around the world, ~$10^9$ client machines

- zones of semi-autonomous control
- consistency after disconnected operation
- power adaptivity
Adaptivity in World-Wide Systems

• Challenge: automatic, dynamic world-wide placement of data & computation to minimize latency and/or cost, given constraints on:
  – bandwidth
  – packet loss
  – power
  – resource usage
  – failure modes
  – ...

• Users specify high-level desires:
  “Store this data on at least 2 disks in EU, 2 in U.S. & 1 in Asia”
  “99%ile latency for accessing this data should be <50ms”
Higher Level Systems

• Systems that provide high level of abstraction that “just works” are incredibly valuable:
  • GFS, MapReduce, BigTable, world-wide systems, transparent variability reduction techniques, etc.

• Can we build high-level systems that just work in other domains like machine learning?
Scaling Deep Learning

• Much of Google is working on approximating AI. AI is hard
  • Many people at Google spend countless person-years hand-engineering complex features to feed as input to machine learning algorithms

• Is there a better way?

• Deep Learning: Use very large scale brain simulations
  • improve many Google applications
  • make significant advances towards perceptual AI
Deep Learning

- Algorithmic approach
  - automatically learn high-level representations from raw data
  - can learn from both labeled and unlabeled data

- Recent academic deep learning results improve on state-of-the-art in many areas:
  - images, video, speech, NLP, ...
  - ... using modest model sizes (<= ~50M parameters)

- We want to scale this approach up to much bigger models
  - currently: ~2B parameters, want ~10B-100B parameters
  - general approach: parallelize at many levels
Deep Networks
Deep Networks

Input Image
(or video)
Deep Networks

Some scalar, nonlinear function of local image patch

Input Image (or video)
Deep Networks

Some scalar, nonlinear function of local image patch

Input Image (or video)
Deep Networks

Some scalar, nonlinear function of local image patch

Input Image (or video)
Deep Networks

Many responses at a single location. In many models these are independent, but some allow strong nonlinear interactions.

Some scalar, nonlinear function of local image patch.

Input Image (or video)
Deep Networks

Input Image
(or video)
Deep Networks

Input Image
(or video)
Deep Networks

Multiple “maps”

Input Image (or video)
Deep Networks
Unsupervised Training

Core idea: try to reconstruct input from just the learned representation

Due to Geoff Hinton, Yoshua Bengio, Andrew Ng, and others
Layer 1

Input Image
(or video)
Layer 1

Layer 2

Input Image
(or video)
Layer 1

Traditional ML tools

Layer 2

Output feature vector

Input Image (or video)
Supports arbitrary models, but best for large models with mostly local connections.
Supports arbitrary models, but best for large models with mostly local connections.
Partition model across machines

Partition assignment in vertical silos.
Partition model across machines

Partition assignment in vertical silos.

Minimal network traffic: The most densely connected areas are on the same partition
Partition model across machines

Partition assignment in vertical silos.

Minimal network traffic: The most densely connected areas are on the same partition.

Our biggest models: 144 machines, ~2300 cores
Making a single model bigger and faster was the right first step.

But training is still slow with large data sets if a model only considers tiny minibatches (10-100s of items) of data at a time.

How can we add another dimension of parallelism, and have multiple model instances train on data in parallel?
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server

Model

Data
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server

Model

Data
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server

$\Delta \phi$

Model

Data
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server \( p' = p + \Delta p \)
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server

Model

Data

$p'$
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server

Model

Data

Δp'
Asynchronous Distributed Stochastic Gradient Descent

Parameter Server \[ p'' = p' + \Delta p' \]**

Model

Data
Asynchronous Distributed Stochastic Gradient Descent

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

Parameter Server

Model Workers

Data Shards
Deep Learning Design Tradeoffs

- Lots of tradeoffs can be made to improve performance. Which ones are possible without hurting learning performance too much?

- For example:
  - Use lower precision arithmetic
  - Send 1 or 2 bits instead of 32 bits across network
  - Drop results from slow partitions

- What’s the right hardware for training and deploying these sorts of systems?
  - GPUs? FPGAs? Lossy computational devices?
Applications

- Acoustic Models for Speech
- Unsupervised Feature Learning for Still Images
- Neural Language Models
Acoustic Modeling for Speech Recognition

8000-label Softmax

One or more hidden layers of a few thousand nodes each.

11 Frames of 40-value Log Energy Power Spectra and the label for central frame

Trained in <5 days on cluster of 800 machines
Acoustic Modeling for Speech Recognition

8000-label Softmax

One or more hidden layers of a few thousand nodes each.

11 Frames of 40-value Log Energy Power Spectra and the label for central frame

Trained in <5 days on cluster of 800 machines

Major reduction in Word Error Rate ("equivalent to 20 years of speech research")
Applications

- Acoustic Models for Speech
- Unsupervised Feature Learning for Still Images
- Neural Language Models
Purely Unsupervised Feature Learning in Images

- 1.15 billion parameters (50x larger than largest deep network in the literature)
- Trained on 16k cores for 1 week using Async-SGD
- Do **unsupervised** training on one frame from each of 10 million YouTube videos (200x200 pixels)
- No labels!

Details in our ICML paper [Le et al. 2012]
Top level neurons seem to discover high-level concepts. For example, one neuron is a decent face detector:
Purely Unsupervised Feature Learning in Images

Most face-selective neuron

Top 48 stimuli from the test set
Purely Unsupervised Feature Learning in Images

Most face-selective neuron

Top 48 stimuli from the test set

Optimal stimulus by numerical optimization
Purely Unsupervised Feature Learning in Images

It is YouTube... We also have a cat neuron!

Top stimuli from the test set
Purely Unsupervised Feature Learning in Images

It is YouTube... We also have a cat neuron!

Top stimuli from the test set

Optimal stimulus
We made a cat detector!

It uses a few CPUs!
Semi-supervised Feature Learning in Images

Are the higher-level representations learned by unsupervised training a useful starting point for supervised training?

We do have some labeled data, so let’s fine tune this same network for a challenging image classification task.
Semi-supervised Feature Learning in Images

Are the higher-level representations learned by unsupervised training a useful starting point for supervised training?

We do have some labeled data, so let’s fine tune this same network for a challenging image classification task.

**ImageNet:**
- 16 million images
- ~21,000 categories
- Recurring academic competitions
Aside: 20,000 is a lot of categories....

01496331 electric ray, crampfish, numbfish, torpedo
01497118 sawfish
01497413 smalltooth sawfish, Pristis pectinatus
01497738 guitarfish
01498041 stingray
01498406 roughtail stingray, Dasyatis centroura
01498699 butterfly ray
01498989 eagle ray
01499396 spotted eagle ray, spotted ray, Aetobatus narinari
01499732 cownose ray, cow-nosed ray, Rhinoptera bonasus
01500091 manta, manta ray, devilfish
01500476 Atlantic manta, Manta birostris
01500854 devil ray, Mobula hypostoma
01501641 grey skate, gray skate, Raja batis
01501777 little skate, Raja erinacea
01501948 thorny skate, Raja radiata
01502101 barndoor skate, Raja laevis
01503976 dickeybird, dickey-bird, dickybird, dicky-bird
01504179 fledgling, fledgeling
01504344 nestling, baby bird
Aside: 20,000 is a lot of categories....
Semi-supervised Feature Learning in Images

Example top stimuli after fine tuning on ImageNet:

<table>
<thead>
<tr>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /> <img src="image2.png" alt="Image" /> <img src="image3.png" alt="Image" /> <img src="image4.png" alt="Image" /> <img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /> <img src="image7.png" alt="Image" /> <img src="image8.png" alt="Image" /> <img src="image9.png" alt="Image" /> <img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /> <img src="image12.png" alt="Image" /> <img src="image13.png" alt="Image" /> <img src="image14.png" alt="Image" /> <img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /> <img src="image17.png" alt="Image" /> <img src="image18.png" alt="Image" /> <img src="image19.png" alt="Image" /> <img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /> <img src="image22.png" alt="Image" /> <img src="image23.png" alt="Image" /> <img src="image24.png" alt="Image" /> <img src="image25.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Semi-supervised Feature Learning in Images

Example top stimuli after fine tuning on ImageNet:
Semi-supervised Feature Learning in Images

Example top stimuli after fine tuning on ImageNet:

<table>
<thead>
<tr>
<th>Neuron 10</th>
<th>Neuron 11</th>
<th>Neuron 12</th>
<th>Neuron 13</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Zebra" /> <img src="image2.png" alt="Zebra" /> <img src="image3.png" alt="Zebra" /> <img src="image4.png" alt="Zebra" /> <img src="image5.png" alt="Zebra" /> <img src="image6.png" alt="Zebra" /> <img src="image7.png" alt="Zebra" /> <img src="image8.png" alt="Zebra" /> <img src="image9.png" alt="Zebra" /></td>
<td><img src="image10.png" alt="Keyboard" /> <img src="image11.png" alt="Keyboard" /> <img src="image12.png" alt="Keyboard" /> <img src="image13.png" alt="Keyboard" /> <img src="image14.png" alt="Keyboard" /> <img src="image15.png" alt="Keyboard" /> <img src="image16.png" alt="Keyboard" /> <img src="image17.png" alt="Keyboard" /> <img src="image18.png" alt="Keyboard" /></td>
<td><img src="image19.png" alt="Wine" /> <img src="image20.png" alt="Wine" /> <img src="image21.png" alt="Wine" /> <img src="image22.png" alt="Wine" /> <img src="image23.png" alt="Wine" /> <img src="image24.png" alt="Wine" /> <img src="image25.png" alt="Wine" /> <img src="image26.png" alt="Wine" /> <img src="image27.png" alt="Wine" /></td>
<td><img src="image28.png" alt="Pizza" /> <img src="image29.png" alt="Pizza" /> <img src="image30.png" alt="Pizza" /> <img src="image31.png" alt="Pizza" /> <img src="image32.png" alt="Pizza" /> <img src="image33.png" alt="Pizza" /> <img src="image34.png" alt="Pizza" /> <img src="image35.png" alt="Pizza" /> <img src="image36.png" alt="Pizza" /></td>
</tr>
</tbody>
</table>
Semi-supervised Feature Learning in Images

ImageNet Classification Results:

ImageNet 2011 (20k categories)
• Chance: 0.005%
• Best reported: 9.5%
• Our network: 16% (+70% relative)
Applications

• Acoustic Models for Speech
• Unsupervised Feature Learning for Still Images
• Neural Language Models
Embeddings

\sim 100-D joint embedding space

porpoise
dolphin
Embeddings

∼100-D joint embedding space

Porpoise  Dolphin
Embeddings

~100-D joint embedding space

porpoise

dolphin

SeaWorld
Embeddings

~100-D joint embedding space

Obama
porpoise
SeaWorld
dolphin
Embeddings

~100-D joint embedding space

Obama

porpoise

dolphin

SeaWorld

Paris
Neural Language Models

Hinge Loss // Softmax

Hidden Layers?

Word Embedding Matrix

\[ E \] is a matrix of dimension ||Vocab|| \( \times d \)

Top prediction layer has ||Vocab|| \( \times h \) parameters.

Most ideas from Bengio et al 2003, Collobert & Weston 2008
Neural Language Models

Hinge Loss // Softmax

Hidden Layers?

Word Embedding Matrix

\[
E \text{ is a matrix of dimension } ||Vocab|| \times d
\]

Top prediction layer has \(||Vocab|| \times h\) parameters.

\{100s of millions of parameters, but gradients very sparse\}

Most ideas from Bengio et al 2003, Collobert & Weston 2008
Embedding sparse tokens in an N-dimensional space

Example: 50-D embedding trained for semantic similarity

Cluster 1:

apple

<table>
<thead>
<tr>
<th>Id</th>
<th>Distance</th>
<th>Adjust</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>11114</td>
<td>0.000000</td>
<td>Remove</td>
<td>apple</td>
</tr>
<tr>
<td>5026</td>
<td>0.652580</td>
<td>Add</td>
<td>fruit</td>
</tr>
<tr>
<td>14080</td>
<td>0.699192</td>
<td>Add</td>
<td>apples</td>
</tr>
<tr>
<td>48657</td>
<td>0.717818</td>
<td>Add</td>
<td>melon</td>
</tr>
<tr>
<td>28498</td>
<td>0.722390</td>
<td>Add</td>
<td>peach</td>
</tr>
<tr>
<td>39795</td>
<td>0.729893</td>
<td>Add</td>
<td>blueberry</td>
</tr>
<tr>
<td>35570</td>
<td>0.730500</td>
<td>Add</td>
<td>berry</td>
</tr>
<tr>
<td>25974</td>
<td>0.739561</td>
<td>Add</td>
<td>strawberry</td>
</tr>
<tr>
<td>46156</td>
<td>0.745343</td>
<td>Add</td>
<td>pecan</td>
</tr>
<tr>
<td>11907</td>
<td>0.756422</td>
<td>Add</td>
<td>potato</td>
</tr>
<tr>
<td>33847</td>
<td>0.759111</td>
<td>Add</td>
<td>pear</td>
</tr>
<tr>
<td>30895</td>
<td>0.763317</td>
<td>Add</td>
<td>mango</td>
</tr>
<tr>
<td>17848</td>
<td>0.768230</td>
<td>Add</td>
<td>pumpkin</td>
</tr>
<tr>
<td>39133</td>
<td>0.770143</td>
<td>Add</td>
<td>almond</td>
</tr>
<tr>
<td>14395</td>
<td>0.773105</td>
<td>Add</td>
<td>tomato</td>
</tr>
<tr>
<td>18163</td>
<td>0.782610</td>
<td>Add</td>
<td>onion</td>
</tr>
<tr>
<td>10470</td>
<td>0.782994</td>
<td>Add</td>
<td>pie</td>
</tr>
<tr>
<td>3023</td>
<td>0.787229</td>
<td>Add</td>
<td>tree</td>
</tr>
<tr>
<td>20340</td>
<td>0.793602</td>
<td>Add</td>
<td>bean</td>
</tr>
<tr>
<td>34968</td>
<td>0.794979</td>
<td>Add</td>
<td>watermelon</td>
</tr>
</tbody>
</table>
Embedding sparse tokens in an N-dimensional space

Example: 50-D embedding trained for semantic similarity

<table>
<thead>
<tr>
<th>Id</th>
<th>Distance</th>
<th>Adjust</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>11114</td>
<td>0.000000</td>
<td>Remove</td>
<td>apple</td>
</tr>
<tr>
<td>5026</td>
<td>0.652580</td>
<td>Add</td>
<td>fruit</td>
</tr>
<tr>
<td>14080</td>
<td>0.699192</td>
<td>Add</td>
<td>apples</td>
</tr>
<tr>
<td>48657</td>
<td>0.717818</td>
<td>Add</td>
<td>melon</td>
</tr>
<tr>
<td>28498</td>
<td>0.722390</td>
<td>Add</td>
<td>peach</td>
</tr>
<tr>
<td>39795</td>
<td>0.729893</td>
<td>Add</td>
<td>blueberry</td>
</tr>
<tr>
<td>35570</td>
<td>0.730500</td>
<td>Add</td>
<td>berry</td>
</tr>
<tr>
<td>25974</td>
<td>0.739561</td>
<td>Add</td>
<td>strawberry</td>
</tr>
<tr>
<td>46156</td>
<td>0.745343</td>
<td>Add</td>
<td>pecan</td>
</tr>
<tr>
<td>11907</td>
<td>0.756422</td>
<td>Add</td>
<td>potato</td>
</tr>
<tr>
<td>33847</td>
<td>0.759111</td>
<td>Add</td>
<td>pear</td>
</tr>
<tr>
<td>30895</td>
<td>0.763317</td>
<td>Add</td>
<td>mango</td>
</tr>
<tr>
<td>17848</td>
<td>0.768230</td>
<td>Add</td>
<td>pumpkin</td>
</tr>
<tr>
<td>39133</td>
<td>0.770143</td>
<td>Add</td>
<td>almond</td>
</tr>
<tr>
<td>14395</td>
<td>0.773105</td>
<td>Add</td>
<td>tomato</td>
</tr>
<tr>
<td>18163</td>
<td>0.782610</td>
<td>Add</td>
<td>onion</td>
</tr>
<tr>
<td>10470</td>
<td>0.782994</td>
<td>Add</td>
<td>pie</td>
</tr>
<tr>
<td>3023</td>
<td>0.787229</td>
<td>Add</td>
<td>tree</td>
</tr>
<tr>
<td>20340</td>
<td>0.793602</td>
<td>Add</td>
<td>bean</td>
</tr>
<tr>
<td>34968</td>
<td>0.794979</td>
<td>Add</td>
<td>watermelon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Distance</th>
<th>Adjust</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>14979</td>
<td>0.000000</td>
<td>Remove</td>
<td>stab</td>
</tr>
<tr>
<td>7728</td>
<td>0.868853</td>
<td>Add</td>
<td>punch</td>
</tr>
<tr>
<td>469</td>
<td>0.909304</td>
<td>Add</td>
<td>shot</td>
</tr>
<tr>
<td>12820</td>
<td>0.909750</td>
<td>Add</td>
<td>thrust</td>
</tr>
<tr>
<td>8934</td>
<td>0.939908</td>
<td>Add</td>
<td>shell</td>
</tr>
<tr>
<td>10880</td>
<td>0.951466</td>
<td>Add</td>
<td>hammer</td>
</tr>
<tr>
<td>6975</td>
<td>0.951679</td>
<td>Add</td>
<td>bullet</td>
</tr>
<tr>
<td>1848</td>
<td>0.962053</td>
<td>Add</td>
<td>push</td>
</tr>
<tr>
<td>10888</td>
<td>0.962319</td>
<td>Add</td>
<td>eyed</td>
</tr>
<tr>
<td>718</td>
<td>0.965448</td>
<td>Add</td>
<td>hand</td>
</tr>
<tr>
<td>5865</td>
<td>0.966663</td>
<td>Add</td>
<td>grab</td>
</tr>
<tr>
<td>4611</td>
<td>0.967574</td>
<td>Add</td>
<td>swing</td>
</tr>
<tr>
<td>302</td>
<td>0.975696</td>
<td>Add</td>
<td>hit</td>
</tr>
<tr>
<td>869</td>
<td>0.976967</td>
<td>Add</td>
<td>force</td>
</tr>
<tr>
<td>1597</td>
<td>0.977625</td>
<td>Add</td>
<td>attempt</td>
</tr>
<tr>
<td>5977</td>
<td>0.978384</td>
<td>Add</td>
<td>finger</td>
</tr>
<tr>
<td>6162</td>
<td>0.978776</td>
<td>Add</td>
<td>knife</td>
</tr>
<tr>
<td>3434</td>
<td>0.980028</td>
<td>Add</td>
<td>sharp</td>
</tr>
<tr>
<td>1504</td>
<td>0.980160</td>
<td>Add</td>
<td>struck</td>
</tr>
<tr>
<td>39157</td>
<td>0.980219</td>
<td>Add</td>
<td>slug</td>
</tr>
</tbody>
</table>
Embedding sparse tokens in an N-dimensional space

Example: 50-D embedding trained for semantic similarity

<table>
<thead>
<tr>
<th>Cluster 1:</th>
<th>apple</th>
<th>Cluster 1:</th>
<th>stab</th>
<th>Cluster 1:</th>
<th>iPhone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td><strong>Distance↑</strong></td>
<td><strong>Adjust</strong></td>
<td><strong>Word</strong></td>
<td><strong>Id</strong></td>
<td><strong>Distance↑</strong></td>
</tr>
<tr>
<td>11114</td>
<td>0.000000</td>
<td><strong>Remove</strong></td>
<td>apple</td>
<td>14979</td>
<td>0.000000</td>
</tr>
<tr>
<td>5026</td>
<td>0.652580</td>
<td><strong>Add</strong></td>
<td>fruit</td>
<td>7728</td>
<td>0.868853</td>
</tr>
<tr>
<td>14080</td>
<td>0.699192</td>
<td><strong>Add</strong></td>
<td>apples</td>
<td>469</td>
<td>0.909304</td>
</tr>
<tr>
<td>48657</td>
<td>0.717818</td>
<td><strong>Add</strong></td>
<td>melon</td>
<td>12820</td>
<td>0.909750</td>
</tr>
<tr>
<td>28498</td>
<td>0.722390</td>
<td><strong>Add</strong></td>
<td>peach</td>
<td>8934</td>
<td>0.939908</td>
</tr>
<tr>
<td>39795</td>
<td>0.729893</td>
<td><strong>Add</strong></td>
<td>blueberry</td>
<td>10880</td>
<td>0.951466</td>
</tr>
<tr>
<td>35570</td>
<td>0.730500</td>
<td><strong>Add</strong></td>
<td>berry</td>
<td>6975</td>
<td>0.951679</td>
</tr>
<tr>
<td>25974</td>
<td>0.739561</td>
<td><strong>Add</strong></td>
<td>strawberry</td>
<td>1848</td>
<td>0.962053</td>
</tr>
<tr>
<td>46156</td>
<td>0.745343</td>
<td><strong>Add</strong></td>
<td>pecan</td>
<td>10888</td>
<td>0.962319</td>
</tr>
<tr>
<td>11907</td>
<td>0.756422</td>
<td><strong>Add</strong></td>
<td>potato</td>
<td>718</td>
<td>0.965448</td>
</tr>
<tr>
<td>33847</td>
<td>0.759111</td>
<td><strong>Add</strong></td>
<td>pear</td>
<td>5865</td>
<td>0.966663</td>
</tr>
<tr>
<td>30895</td>
<td>0.763317</td>
<td><strong>Add</strong></td>
<td>mango</td>
<td>4611</td>
<td>0.967574</td>
</tr>
<tr>
<td>17848</td>
<td>0.768230</td>
<td><strong>Add</strong></td>
<td>pumpkin</td>
<td>302</td>
<td>0.975696</td>
</tr>
<tr>
<td>39133</td>
<td>0.770143</td>
<td><strong>Add</strong></td>
<td>almond</td>
<td>869</td>
<td>0.976967</td>
</tr>
<tr>
<td>14395</td>
<td>0.773105</td>
<td><strong>Add</strong></td>
<td>tomato</td>
<td>1597</td>
<td>0.977625</td>
</tr>
<tr>
<td>18163</td>
<td>0.782610</td>
<td><strong>Add</strong></td>
<td>onion</td>
<td>5977</td>
<td>0.978384</td>
</tr>
<tr>
<td>10470</td>
<td>0.782994</td>
<td><strong>Add</strong></td>
<td>pie</td>
<td>6162</td>
<td>0.978776</td>
</tr>
<tr>
<td>3023</td>
<td>0.787229</td>
<td><strong>Add</strong></td>
<td>tree</td>
<td>3434</td>
<td>0.980028</td>
</tr>
<tr>
<td>20340</td>
<td>0.793602</td>
<td><strong>Add</strong></td>
<td>bean</td>
<td>1504</td>
<td>0.980160</td>
</tr>
<tr>
<td>34968</td>
<td>0.794979</td>
<td><strong>Add</strong></td>
<td>watermelon</td>
<td>39157</td>
<td>0.980219</td>
</tr>
</tbody>
</table>
Neural Language Models

- 7 Billion word Google News training set
- 1 Million word vocabulary
- 8 word history, 50 dimensional embedding
- Three hidden layers each w/200 nodes
- 50-100 asynchronous model workers
Neural Language Models

- 7 Billion word Google News training set
- 1 Million word vocabulary
- 8 word history, 50 dimensional embedding
- Three hidden layers each w/200 nodes
- 50-100 asynchronous model workers

Perplexity Scores

<table>
<thead>
<tr>
<th></th>
<th>Traditional 5-gram</th>
<th>NLM</th>
<th>5-gram + NLM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XXX</td>
<td>+15%</td>
<td>-33%</td>
</tr>
</tbody>
</table>

The diagram illustrates the process of predicting words in a sequence using neural language models.
Summary
Summary

- Two interesting “traditional systems” areas:
  - Building low-latency services out of variable-latency parts
  - Single systems that coordinate activity across the whole world, rather than separate instances per cluster/datacenter
Summary

- Two interesting “traditional systems” areas:
  - Building low-latency services out of variable-latency parts
  - Single systems that coordinate activity across the whole world, rather than separate instances per cluster/datacenter

- Systems work to scale deep learning gives promising results:
  - Work at intersection of systems and machine learning
  - Speech: Supervised model gives major improvement in WER.
  - Images: State of the art performance on ImageNet 21K classes
  - Neural language models: complementary to N-gram models
Many Thanks!

Variability reduction collaborators: Luiz Barroso + many others

World-wide storage system collaborators: Jay Corbett, Mike Epstein, Andrew Fikes, Chris Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Chris Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alex Lloyd, Sergey Melnik, David Mwauara, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Chris Taylor, Ruth Wang, Dale Woolford + others

Deep learning collaborators: Samy Bengio, Kai Chen, Greg Corrado, Tom Dean, Matthieu Devin, Quoc Le, Mark Mao, Rajat Monga, Andrew Ng, Patrick Nguyen, Marc’Aurelio Ranzato, Paul Tucker, Vincent Vanhoucke, Xiaoyun Wu, Peng Xe, Ke Yang + others