

# 10 Billion Parameter Neural Networks in your Basement

Adam Coates

Stanford University

## Overview: two parts

- Deep learning and feature learning.
  - Exciting topic in machine learning.
  - Major area of AI research.
- HPC and deep learning

# What do we want computers to do with our data?

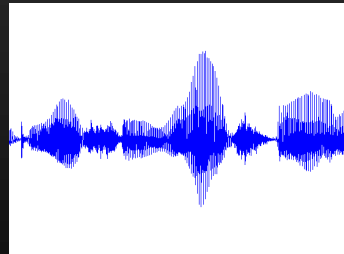
Images/video



Label: "Motorcycle"  
Suggest tags  
Image search

...

Audio



Speech recognition  
Music classification  
Speaker identification

...

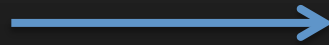
Text



Web search  
Anti-spam  
Machine translation

...

# Machine learning for image classification



"Motorcycle"

# Computer vision is hard!



Migma Systems, Inc.



# What do we want computers to do with our data?

Images/video



Label: "Motorcycle"  
Suggest tags

Machine learning performs well on many of these problems, but is a **lot** of work.

What is it about machine learning that makes it so hard to use?

Text



Web search

Anti-spam  
Machine translation  
...

# Why is this hard?



“Motorcycle”



177	153	118	91	85	100	124	145
151	124	93	77	86	115	148	168
115	93	78	83	108	145	177	191
88	79	84	104	136	168	190	197
82	85	103	127	152	170	180	182
91	101	120	138	150	157	159	159
103	114	127	136	140	140	140	141
111	119	126	130	130	129	128	130

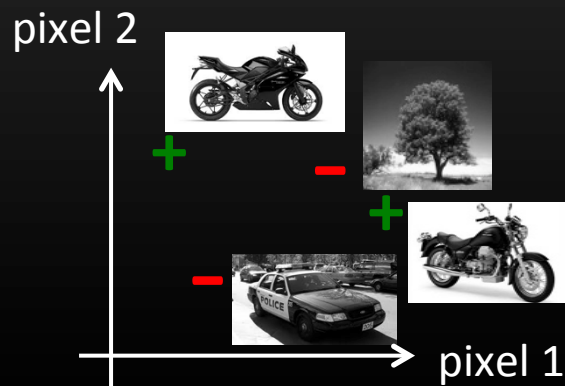
Pixel Intensity

Pixel intensity is a very difficult representation.

# Why is this hard?



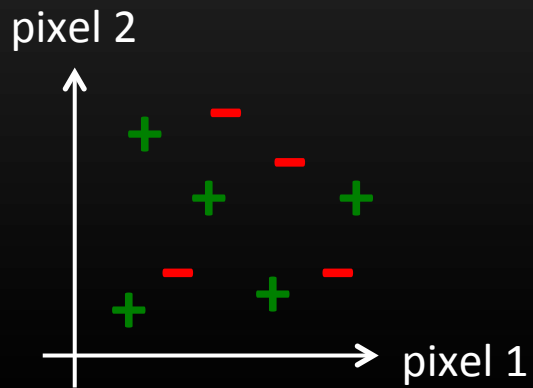
↓  
[72 160] Pixel Intensity



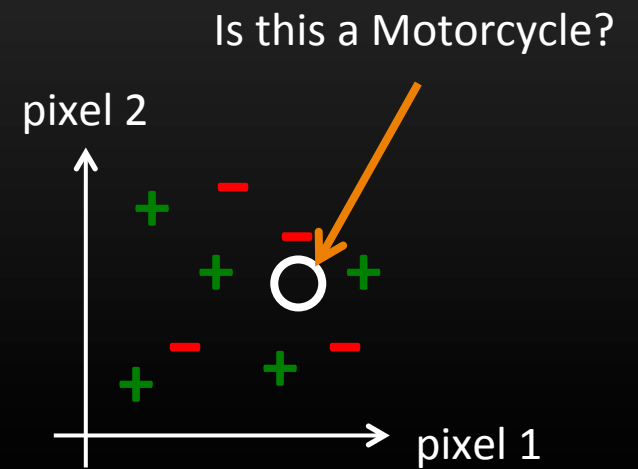
- + Motorcycle
- Not Motorcycle



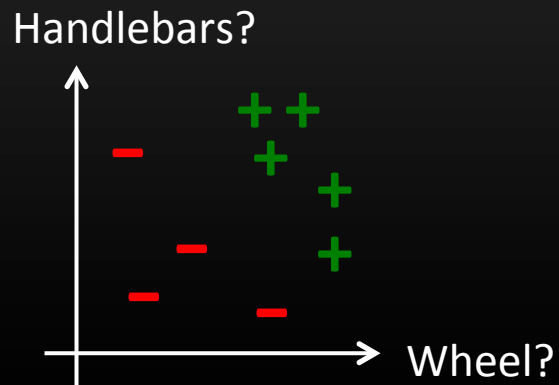
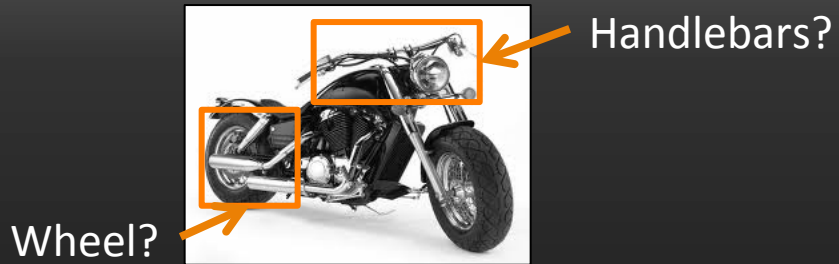
# Why is this hard?



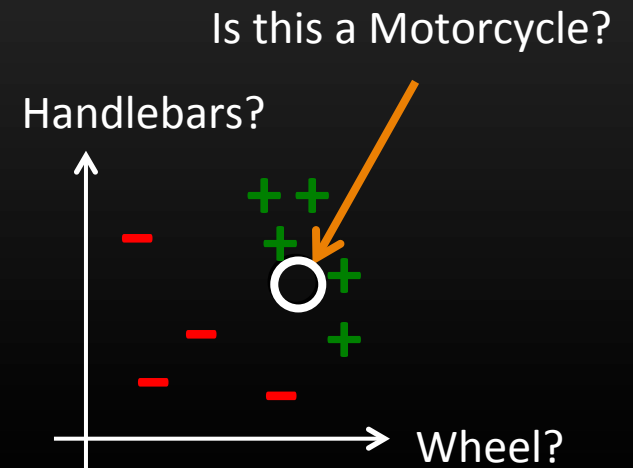
Learning Algorithm



# Features



Learning Algorithm



# Why do features help?

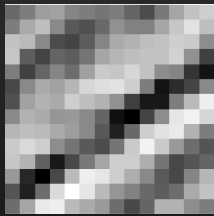
Provide knowledge that system can't learn on its own.



Can we acquire this knowledge from data?

# Learning features

14 x 14 pixel image patch



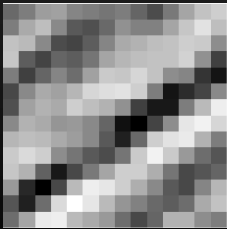
196 pixel intensities

$$\begin{bmatrix} 108 \\ 168 \\ 195 \\ 214 \\ 187 \\ 97 \\ \dots \end{bmatrix}$$

Can we learn a “better” feature vector?

## Example: Sparse coding

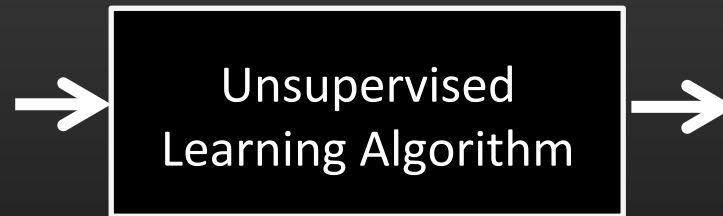
- Try to find a set of “basis” images so that *any* 14x14 patch can be built from just a few of them.  
[Olshausen & Field, '96]


$$= h_1 \begin{array}{|c|} \hline ? \\ \hline \end{array} + h_2 \begin{array}{|c|} \hline ? \\ \hline \end{array} + h_3 \begin{array}{|c|} \hline ? \\ \hline \end{array} \dots$$

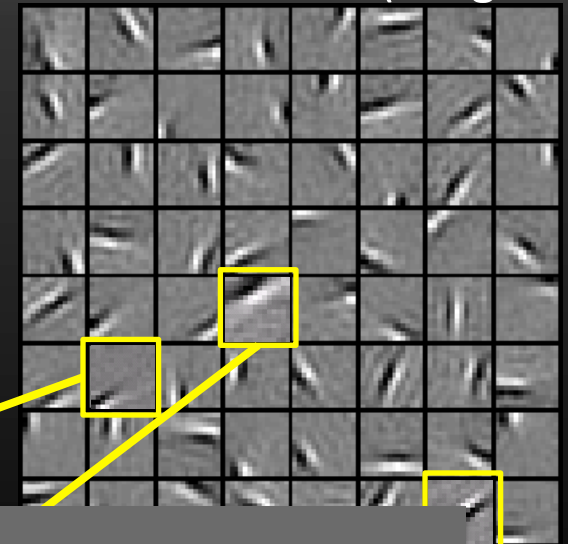
such that most of  $h_1, h_2, h_3, \dots$  are zero (“sparse”).

# Example: Sparse auto-encoder

Unlabeled Image Patches



Learned Features ("Edges")



Feature Vector: a "higher level" representation.

$$h = [0 \dots 0 \ 0.8 \ 0 \dots 0 \ 0.3 \ 0 \dots 0 \ 0.5 \ 0 \dots 0]$$

# Features as neural networks

- We have mathematical principles to find features  $h$  that are better than original pixels.
- Often use “neural networks” to generate these features:

Features



$$h = g(Wx)$$

Pixels

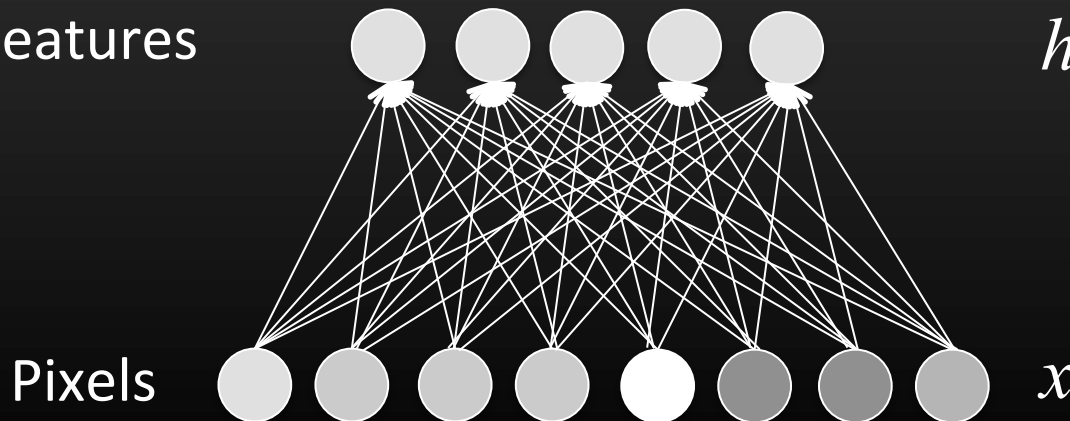


$x$

# Learning features as neural networks

- We have mathematical principles to find features  $h$  that are better than original pixels.
- Often use “neural networks” to generate these features:

Features

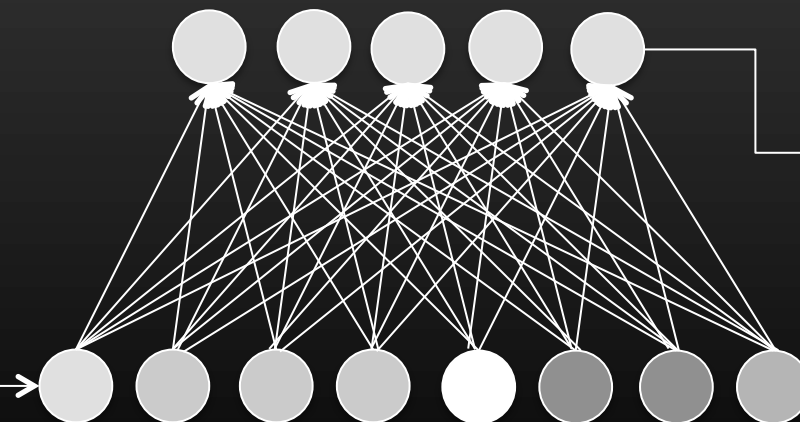
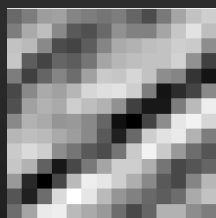


$$h = g(Wx)$$



# Learning features as neural networks

14 x 14 pixel image patch

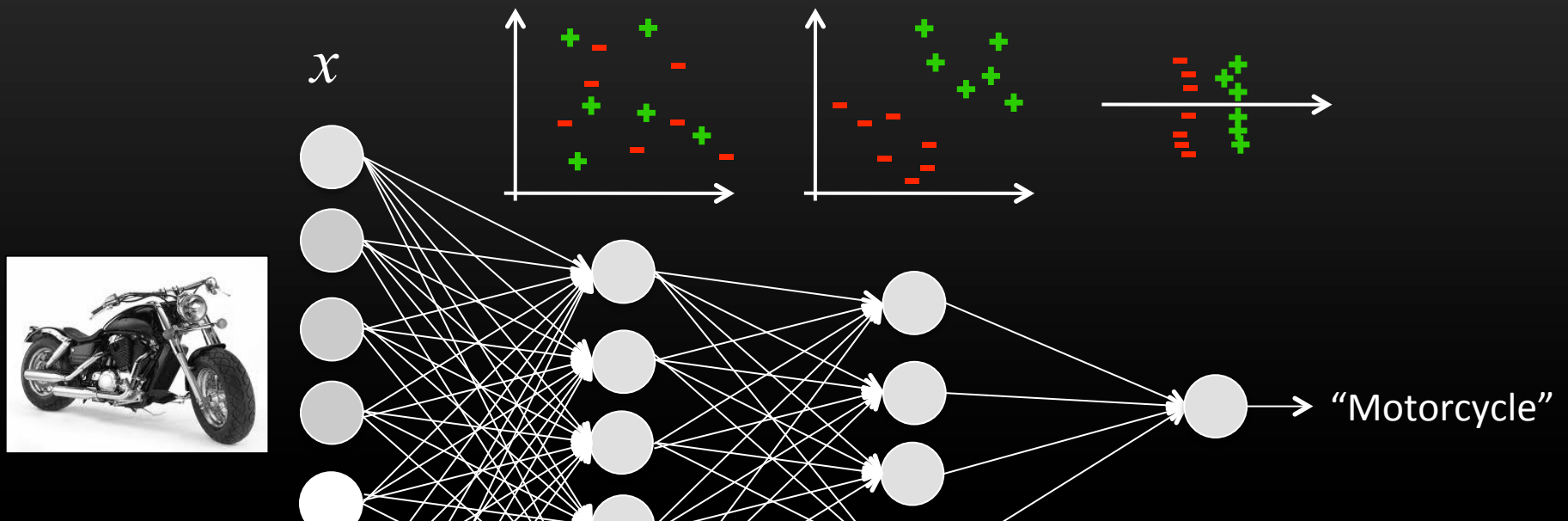


Features

[ 0.0  
0.8  
0.0  
...  
0.5  
0.0 ]

# Deep learning

- Try to build deep neural networks that compute higher and higher level abstractions.



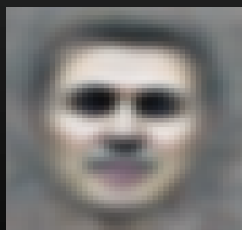
# Large-scale deep learning

- Historically, bigger models have tended to make way for improved results.
  - Big networks can represent more complex concepts.
- What types of “high level” concepts can big networks learn?

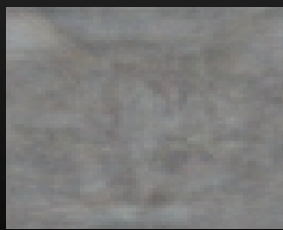
## “High-level features”

- 1 billion parameter, 9 layer neural network trained by Google.
  - Trained on 10 million YouTube video frames.
- Some features represent “objects” in images.
  - System has no prior knowledge of the concept of “objects”.

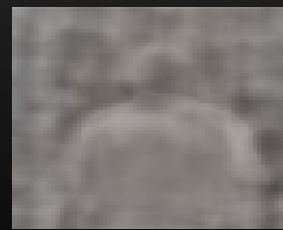
Faces:



Cats:



Bodies:



➤ **1000 machines for 1 week. (16000 cores.)**

[Le et al., ICML 2012; Dean et al., NIPS 2012]

**Large-scale DL in your basement**

# What can the rest of us do??

- Millions of \$\$ hardware.
- Extensive engineering to handle node failures and network traffic.
- Hard to scale further (10,000 machines?!)



# Scaling

- Scale up: Make use of GPUs.

~~➤ Limited GPU memory.~~

~~➤ Hard to put more than ~4 GPUs in machine.~~

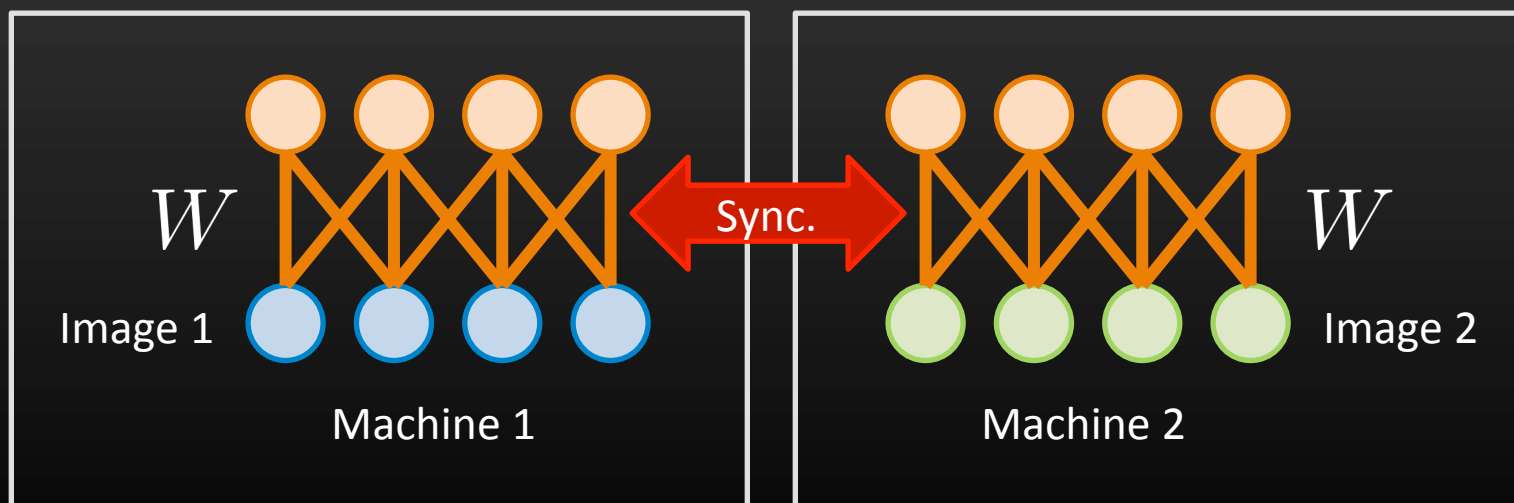
- Scale out: Use many machines.

~~➤ More than ~10-20 machines uses too many resources.~~

➤ Networking infrastructure a recurring bottleneck.

# Two ways to scale neural networks

- Simple solution: “data parallelism”
  - Parallelize over several training images at once.

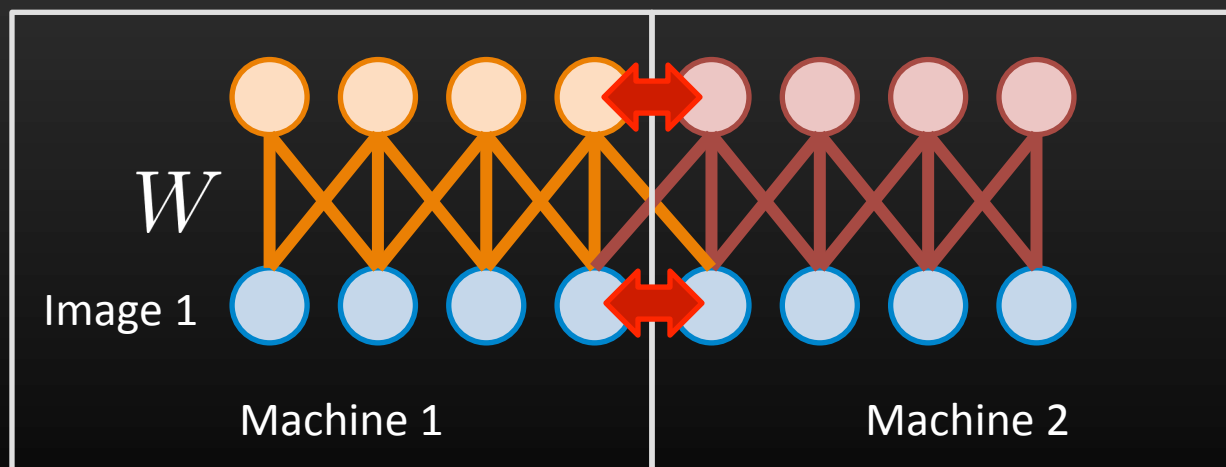


- Need to synchronize parameters across machines.
- Difficult to fit big models on GPUs.



# Two ways to scale neural networks

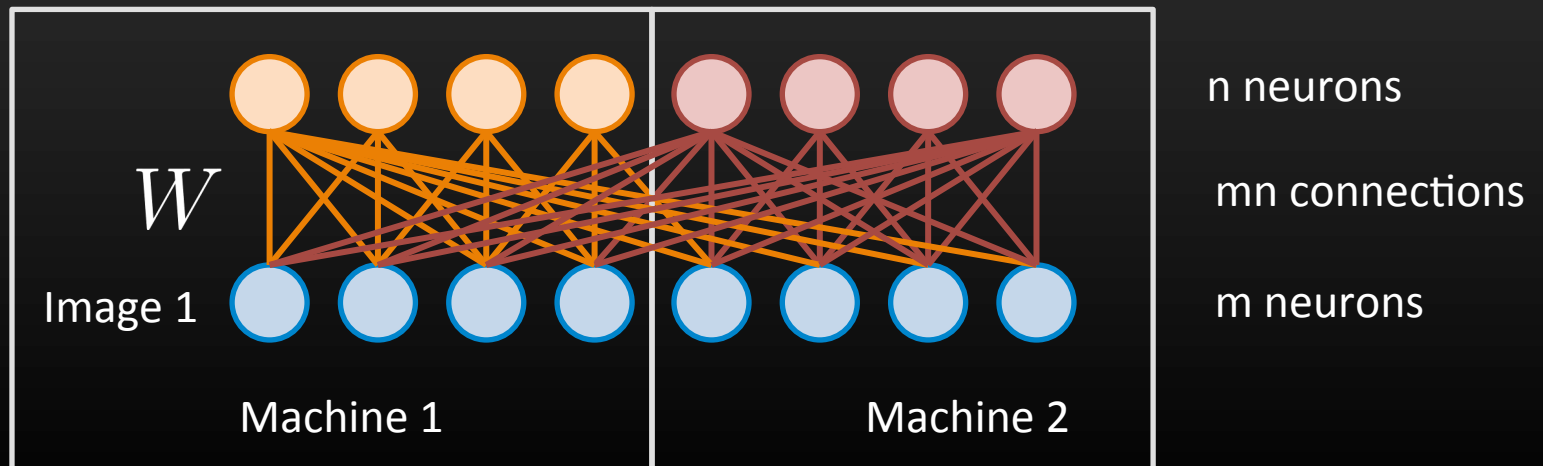
- “Model parallelism”
  - Parallelize over neurons / features in the network.



- Scales to much larger models.
- Much more frequent synchronization.

# Efficiency

- Why should this work?
  - Number of neurons to move:  $O(m + n)$
  - Amount of computation to do:  $O(mn)$



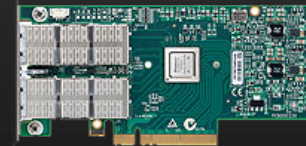
- Big networks end up bottlenecked on compute!

## One catch: Network bottleneck

- Still need to move those neurons...
  - Move 1MB of neurons for 100 images at 1Gbps = **0.8 seconds**
    - Must do this for *every* layer (e.g., 10 or more).
    - Typically  $\gg 10$  times slower than computation.
- Hard to make “m” and “n” big.
  - How do we scale out efficiently??

# COTS HPC Hardware

- Infiniband:
  - FDR Infiniband switch.
  - 1 network adapter per server.  
56 Gbps; microsecond latency.
- GTX 680 GPUs
  - 4 GPUs per server.
  - > 1 TFLOPS each for ideal workload.



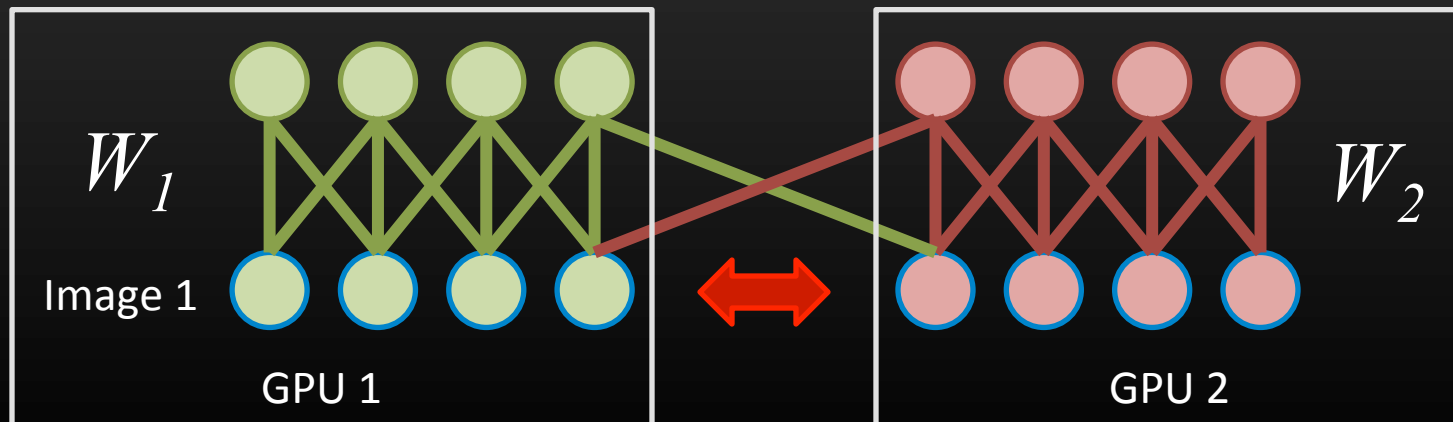
# OTS HPC Software Infrastructure

- Infiniband (“IB”): Use MPI
  - MPI = Message Passing Interface
    - Standard mid-level API usually supporting IB.
  - MVAPICH2: GPU-aware MPI implementation.
    - Enables message passing across GPUs with MPI.
    - Transparently handle GPUs in different machines.
- GPUs: NVIDIA CUDA
  - All GPU operations are local. No RDMA, etc.

# Model parallelism in MPI

MPI starts a single process for each GPU.

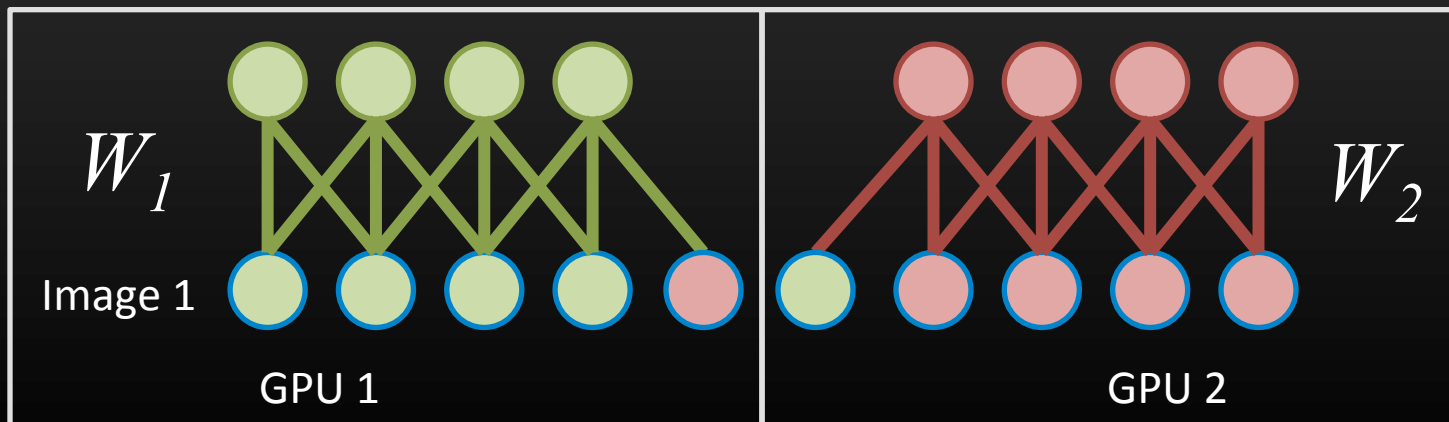
- Enables message passing, but this is pretty unnatural.



# Model parallelism in MPI

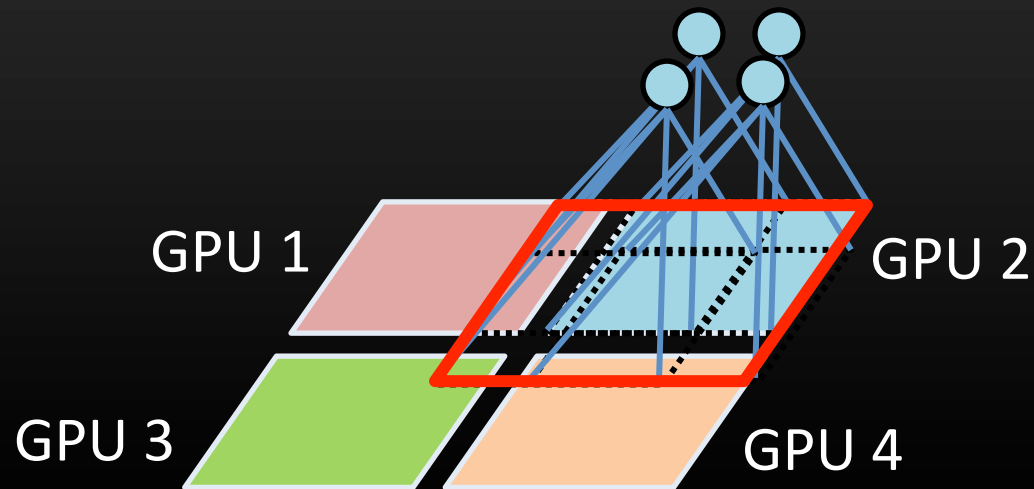
MPI starts a single process for each GPU.

- Enables message passing, but this is pretty unnatural.



# HPC Software Infrastructure: Communication

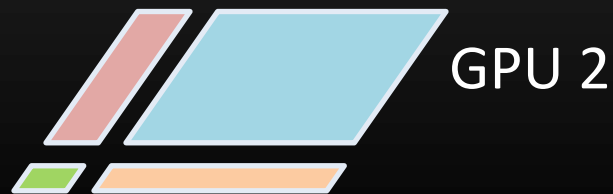
- Moving neuron responses around is confusing.
  - Hide communication inside “distributed array”.





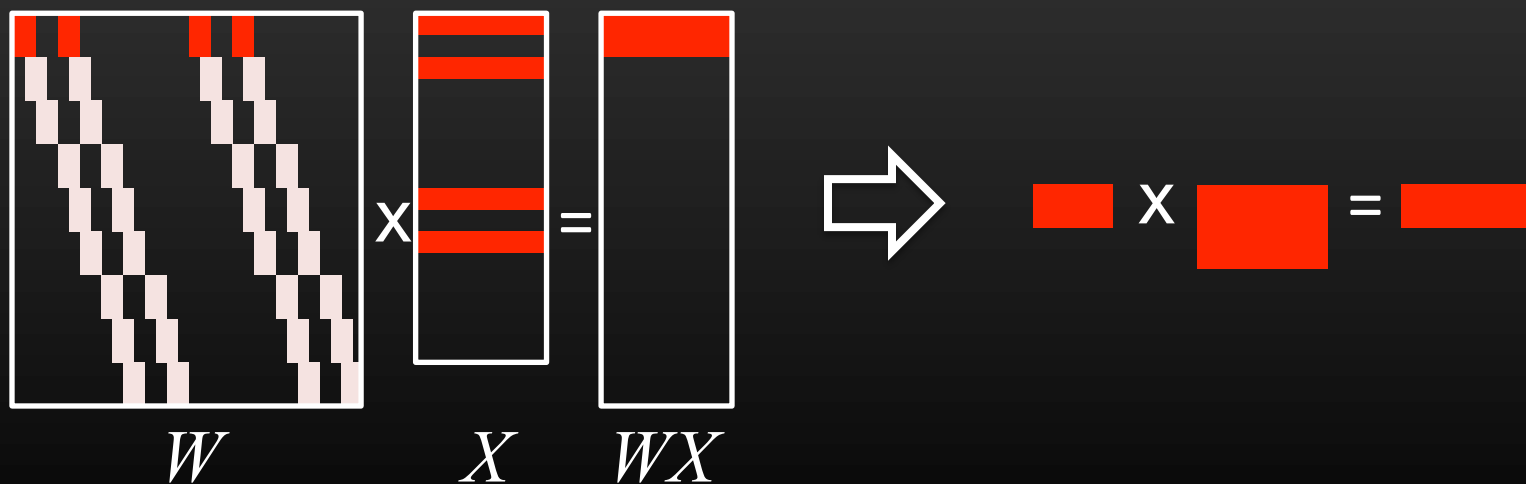
# HPC Software Infrastructure: Communication

- After some hidden communication, GPU 2 has all the input data it needs.
  - GPU code not much different from 1 GPU.



# HPC Software Infrastructure: GPU

- Bottleneck operations in large networks:
  - Dealing with sparse connectivity patterns.



- Trick: leverage optimized BLAS code for small dense multiplies.
  - Need to pick networks with big blocks of neurons sharing connectivity.

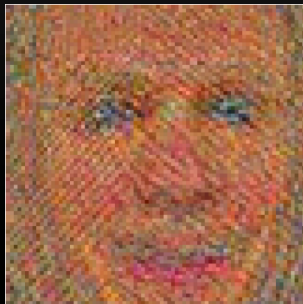
# Results: Unsupervised Learning

- Duplicated results from Le et al., 2012.
  - 3 machines, 12 GPUs

Object	Guessing	Random net	1.8B param net
Human faces	64.7%	64.8%	<b>88.2%</b>
Upper body	64.7%	64.8%	<b>80.2%</b>
Cats	64.7%	64.8%	<b>73.0%</b>

Visualizations of object-selective neurons:

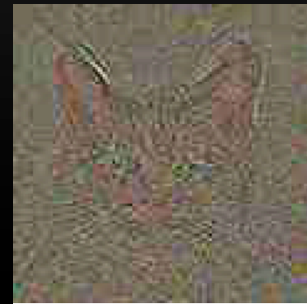
Faces:



Bodies:

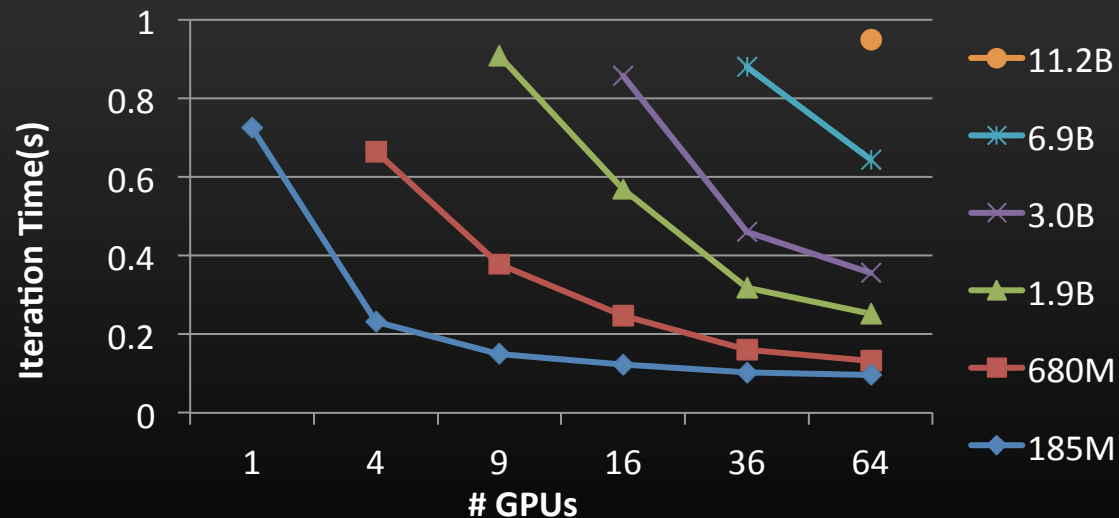


Cats:



# Results: Scaling

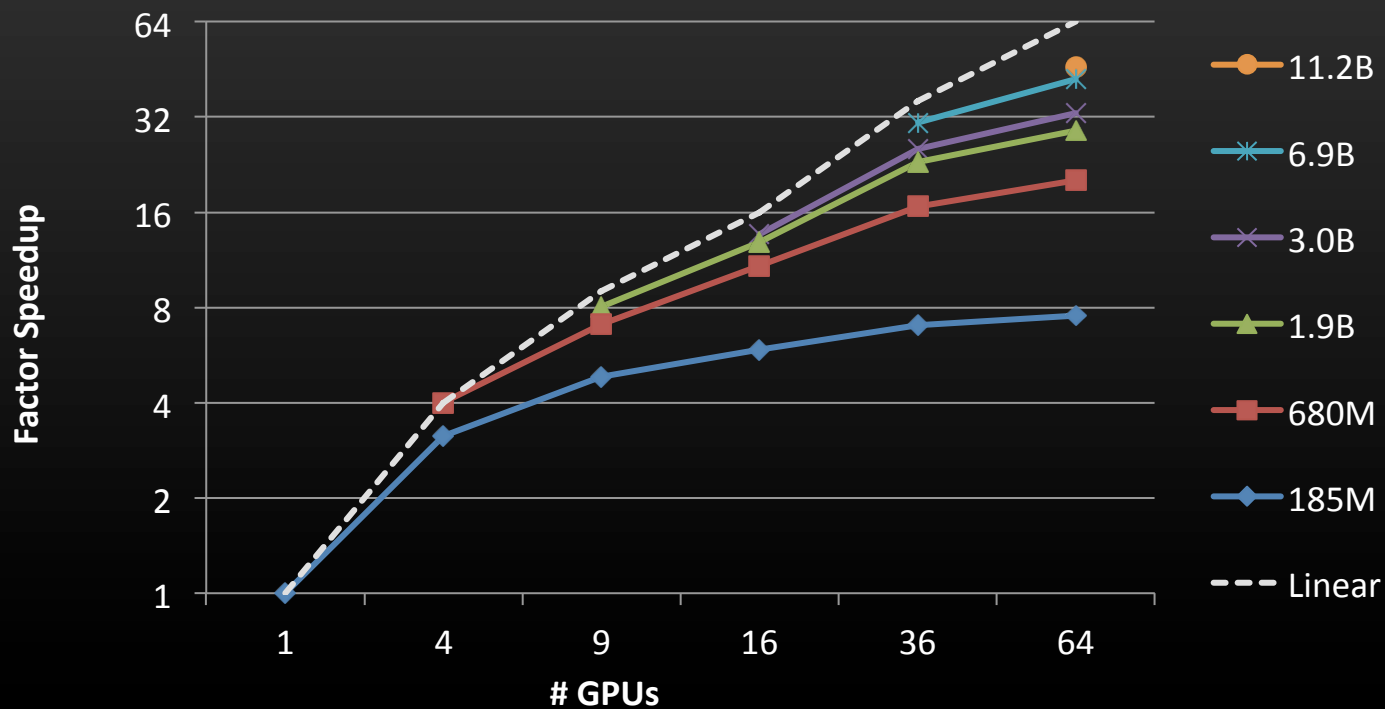
- 9-layer neural network from Le et al., 2012.
  - Compute “fine-tuning” update. (Most demanding step.)



- Up to **11.2B** parameter networks.
  - Update time similar to **185M** parameter network on 1 GPU.

# Results: Scaling

- Up to 47x increase in throughput:



# Conclusion

- “Tera-scale” deep learning now possible in a typical research lab.
  - Duplicated results from 1000 machines with 3 GPU servers.
- Simple abstractions and OTS software sufficient for a scalable implementation.
- 6.5x larger networks (up to 11.2B parameters).
  - What ideas are we missing to capture more complex concepts?
    - Hardware is suddenly not our bottleneck!

**Thanks to:** Quoc Le, Bill Daly, Cliff Woolley, Michael Bauer, Geoffrey Fox, Stanford CSD-CF, and the authors of MAGMA BLAS and MVAPICH2.