Deep Learning Systems at Scale

Soumith Chintala

Facebook AI Research

IBM Supercomputing PIC, T.J. Watson Research Center

8th June, 2016

Talk Overview

1.Deep Learning

2. Algorithms
- Convolution Nets
- Recurrent Nets

3. Hardware
- CPUs
- GPUs
  - NVIDIA
  - AMD
- Other
  - Xeon Phi

4. Scale & Optimizations
- Multi-core CPU
- Single GPU
- Single Machine
- Multi-machine

5. Computing Paradigms
- Collectives / MPI
  - Torch, Caffe
- Graph Compilation
  - Theano, MXNet, TensorFlow,
  - Caffe2, Purine

Deep Learning

Image Intelligence: Classification

Image Intelligence : Detection

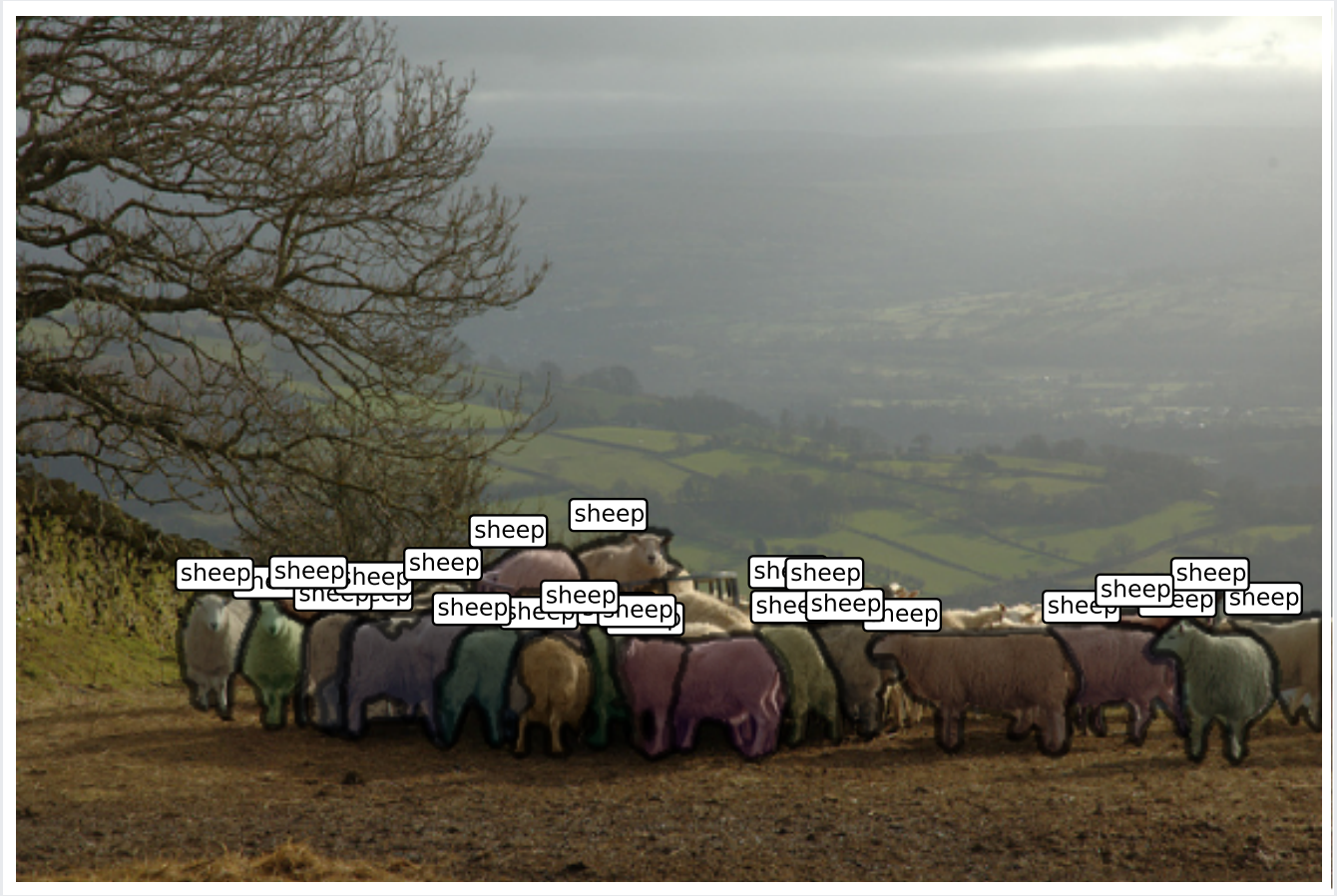


Image Intelligence : Detection

Image Intelligence

https://code.facebook.com/posts/accessibility/

Video Intelligence

Image and Video Generation

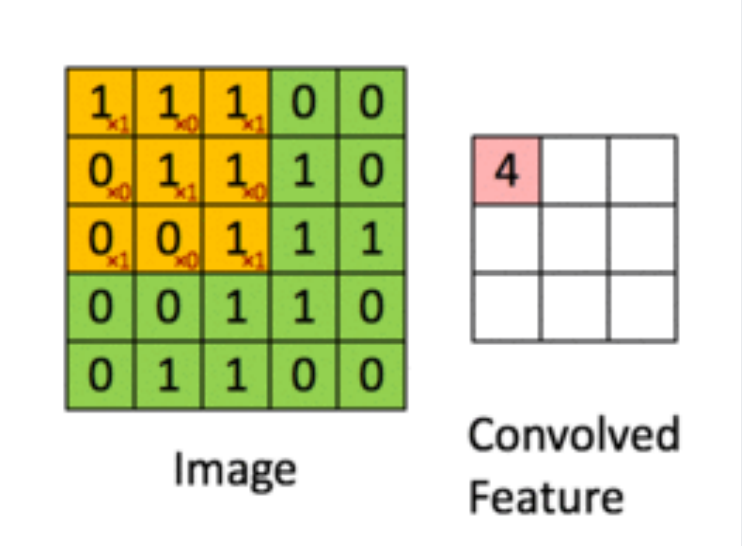
Predicting the Future

Natural Language Understanding

- Language Translation
- Reading, Writing and answering Questions
- Chatbots / Personal assistants

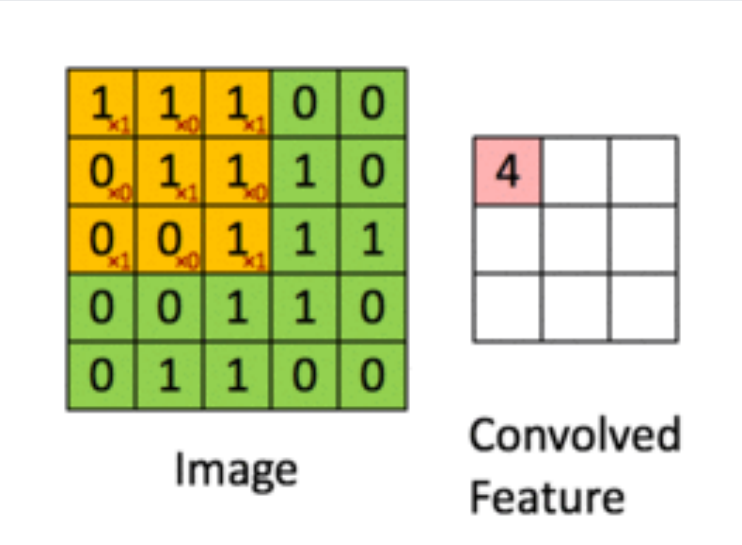
Algorithms

Convolution Neural Networks (ConvNets)

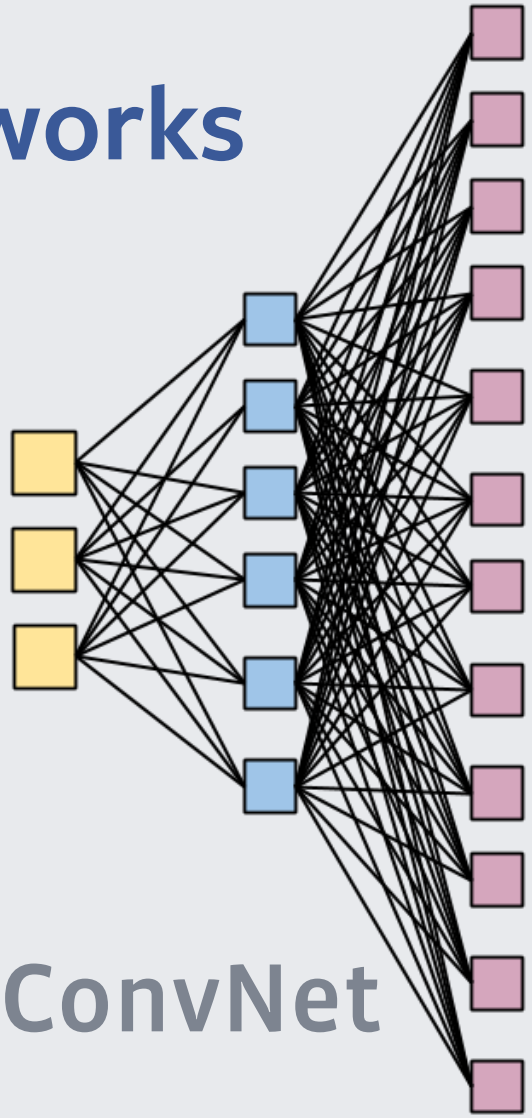


2D Convolution

Convolution Neural Networks (ConvNets)

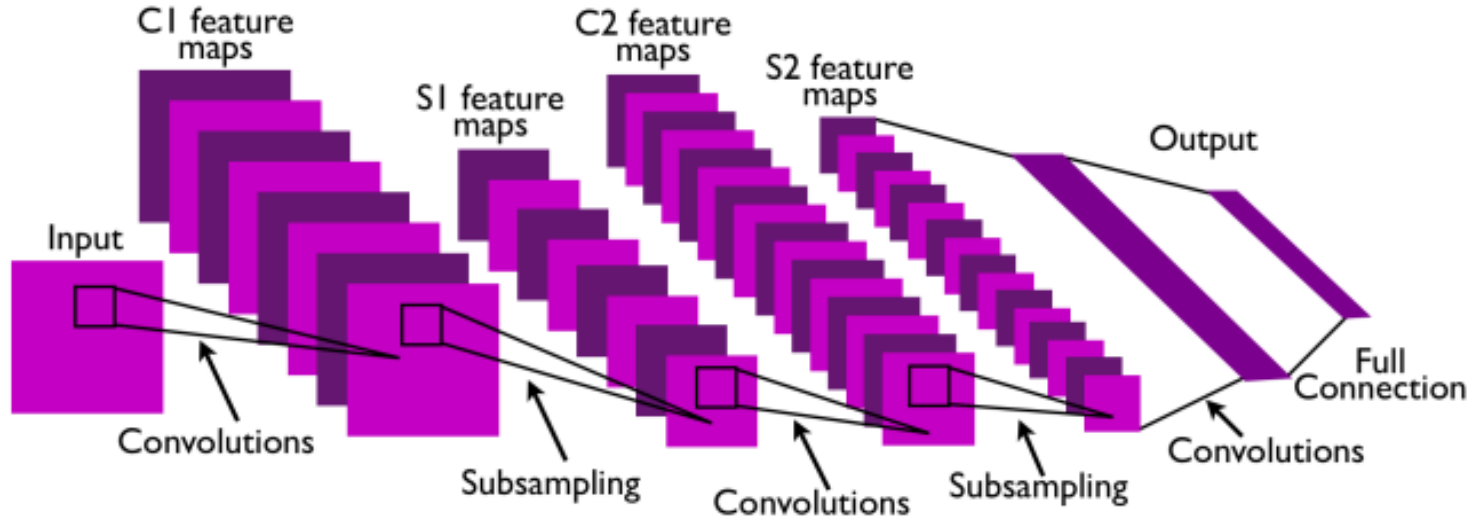


2D Convolution



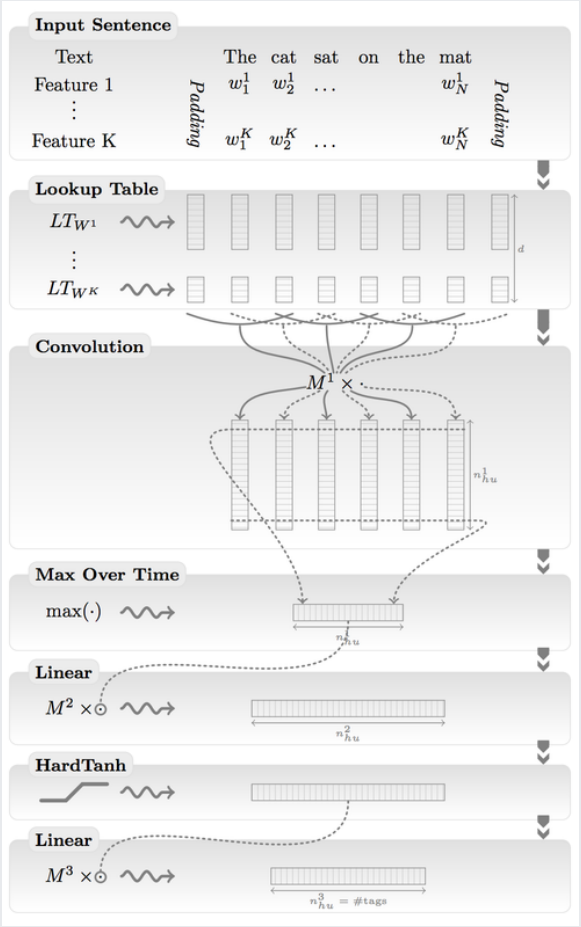
ConvNet

Convolution Neural Networks (ConvNets)

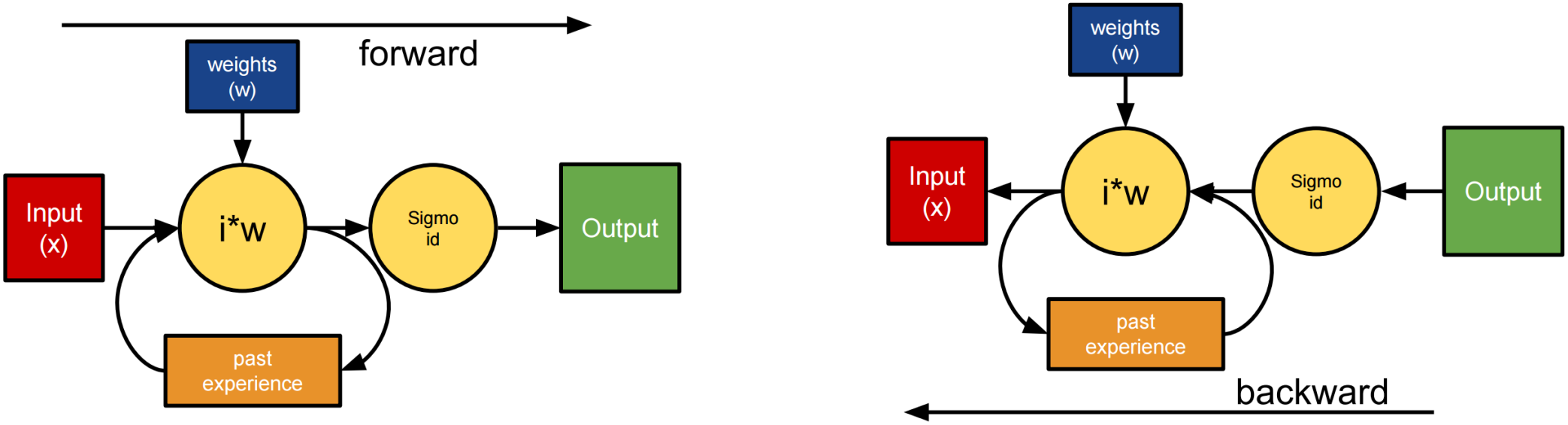


ConvNet

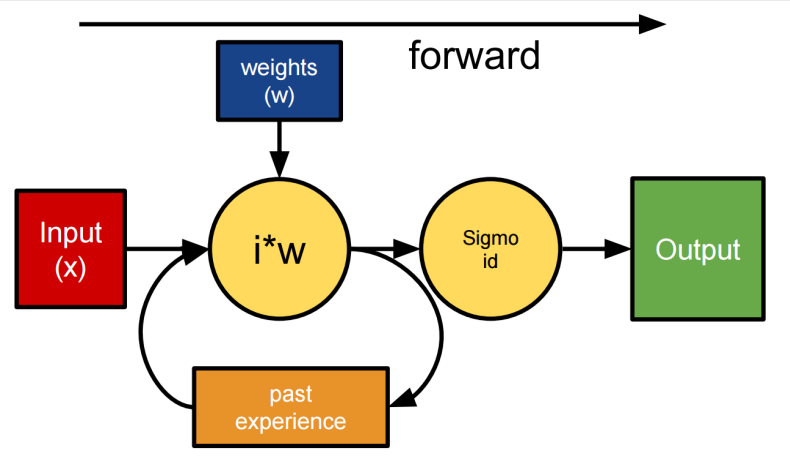
Convolution Neural Networks (ConvNets)



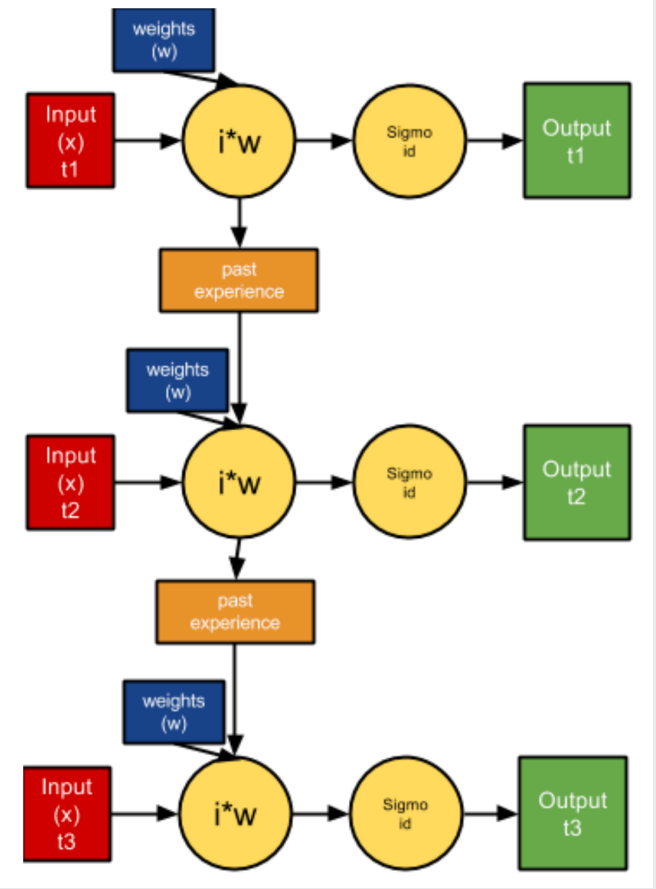
Recurrent Neural Networks



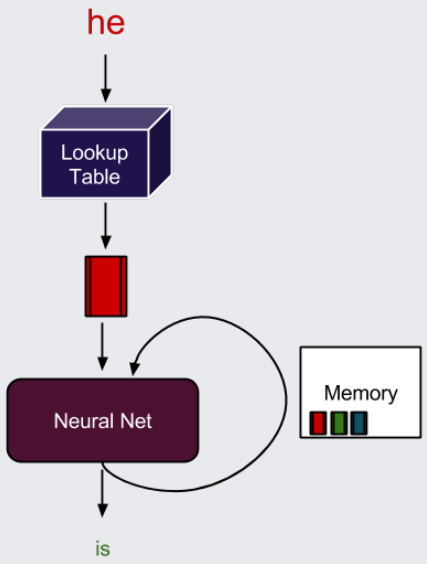
Recurrent Neural Networks

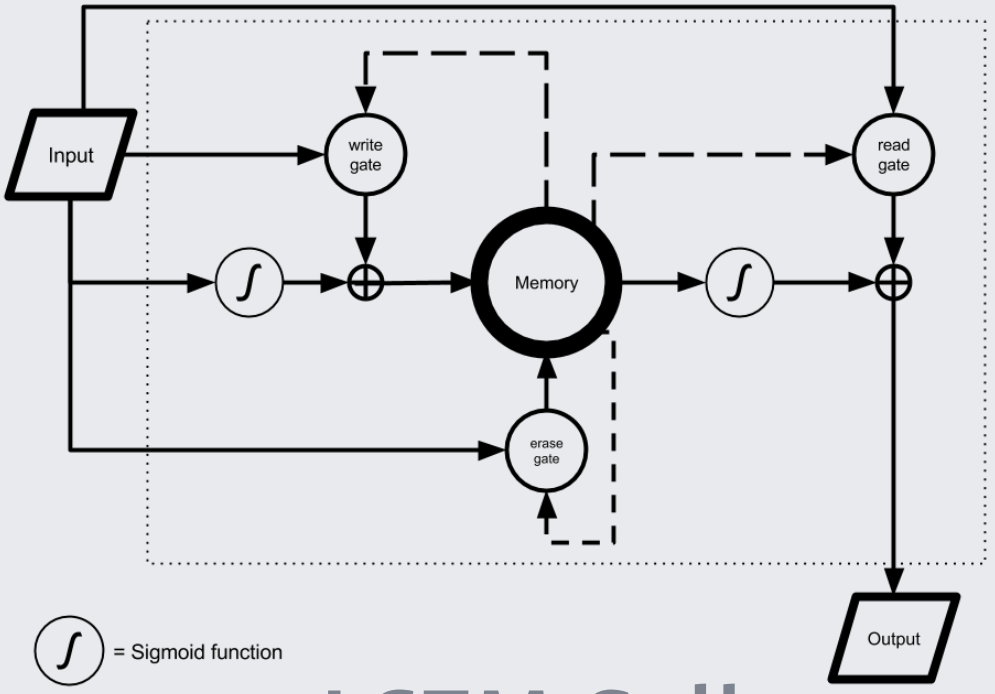


Unfolded in Time



Recurrent Neural Networks





LSTM Cell

Hardware

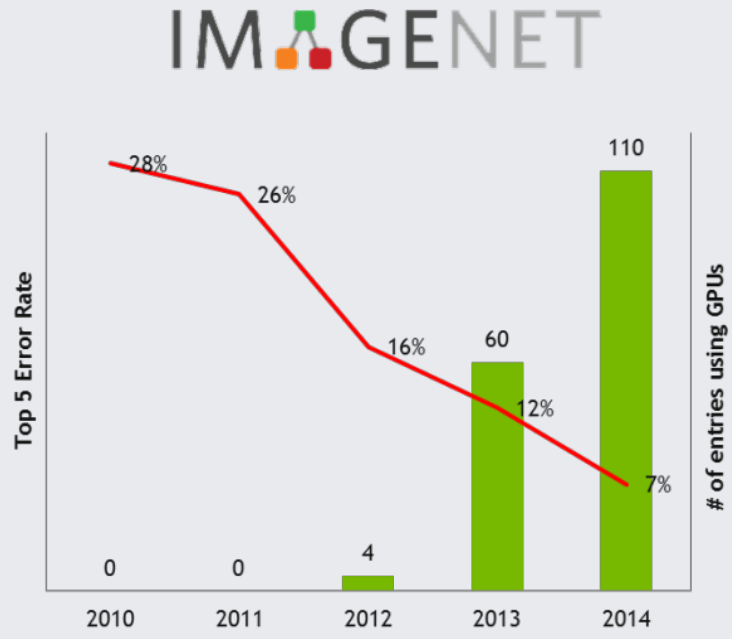
Hardware

- CPUs
- GPUs
  - NVIDIA
  - AMD
- Other

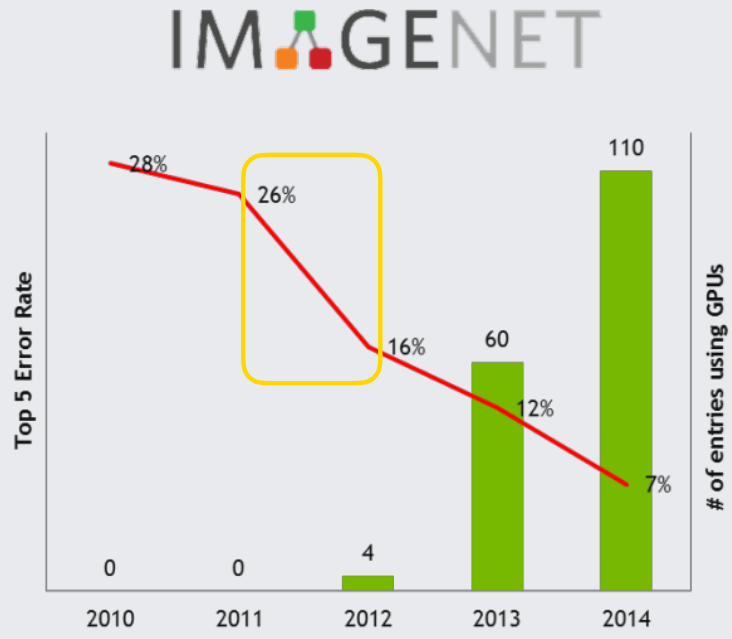
CPUs

- Multi-core CPU evaluations
- Limitations
  - 1 or 2 CPUs per machine
  - Poor theoretical (and practical) peak

GPU-powered ConvNets



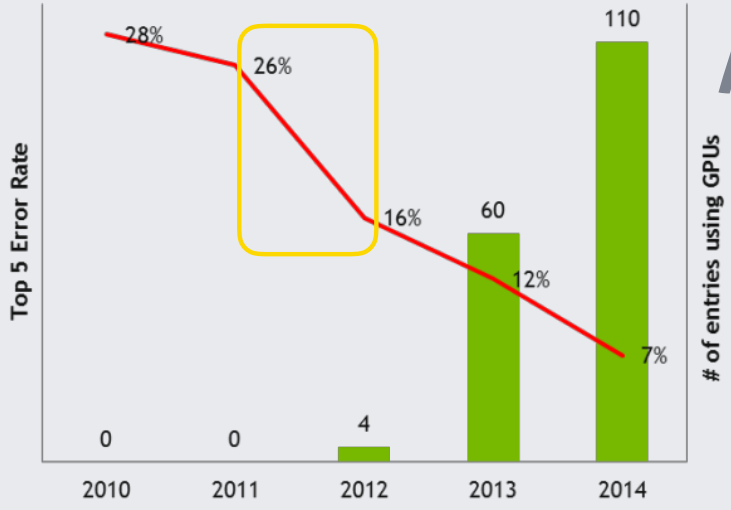
GPU-powered ConvNets



GPU-powered ConvNets

IMAGENET

cuda-convnet
High-performance C++/CUDA implementation of convolutional neural networks

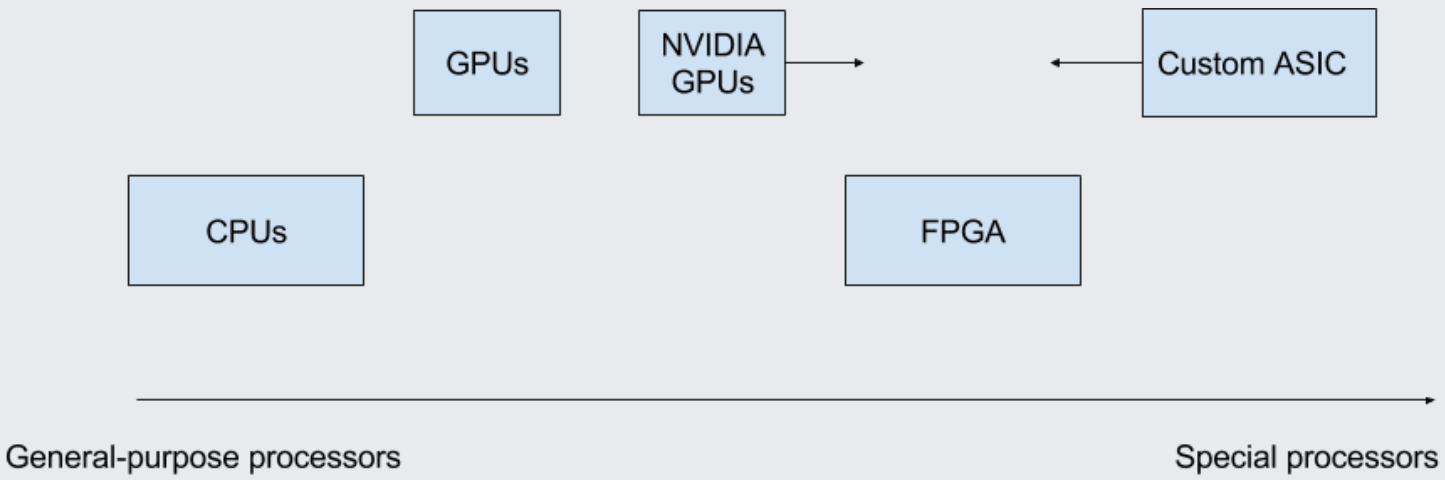


Alex Khrizevsky

Other Hardware

- Intel Xeon Phi
- Nervana Unnamed
- Google TPU
- GraphCore
- Teradeep
- Other custom chips (Yunji Chen et. al.)

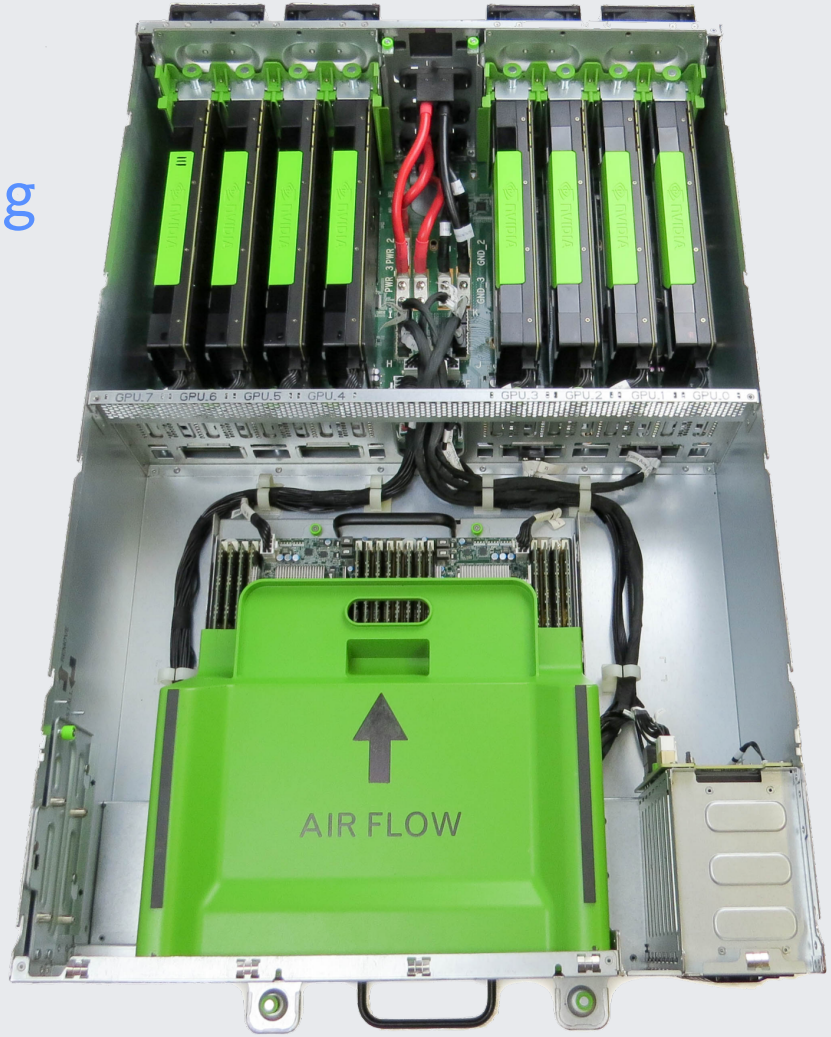
Hardware


GPUs
NVIDIA GPUs
Custom ASIC
CPUs
FPGA
General-purpose processors
Special processors

Big Sur

Open Compute for Deep Learning

- Serviceability
- Thermal Efficiency
- Performance

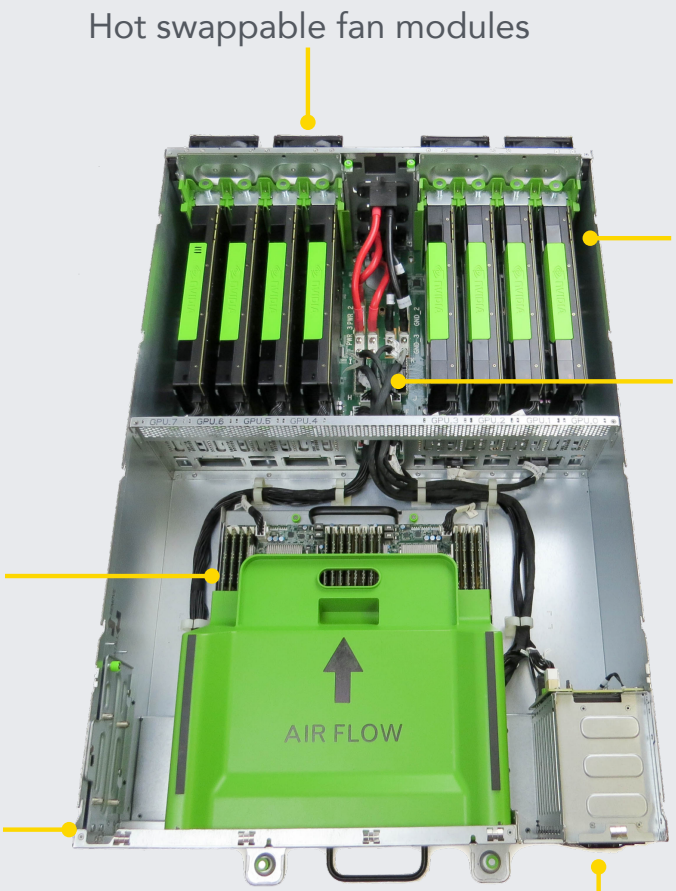


Big Sur

Open Compute for Deep Learning

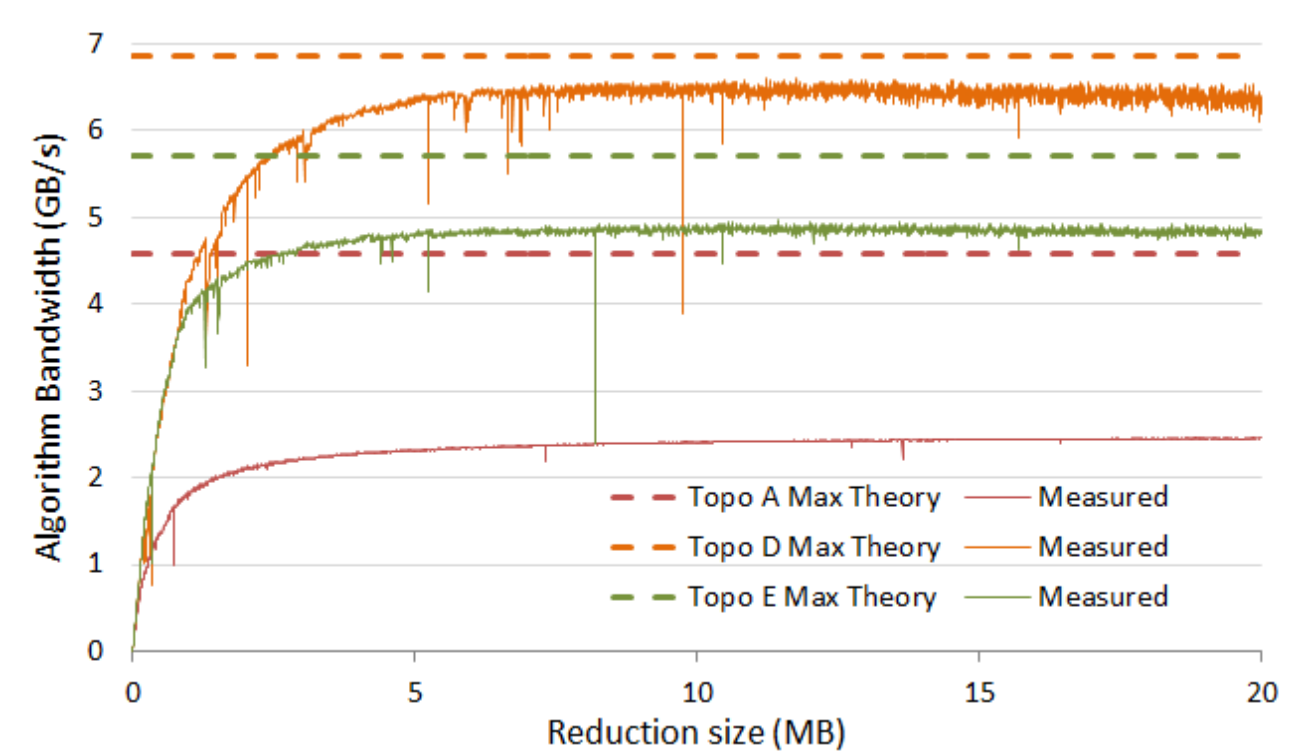
GPU removal using 2 thumb screws

Swap PCI-e Topologies with incredible ease



Big Sur

PCI-e Topologies — Matter!



Big Sur

PCI-e Topologies — Matter!

Network	#GPUs	batch size	#batches	E to A SpdUp	D to A SpdUp	D to E SpdUp
Alexnet, batch norm	8	1024	20	2%	5%	3%
vgg	8	512	10	-1%	0%	1%
vggstress	8	512	10	-3%	7%	10%
googlenet	8	512	10	0%	-2%	-2%

Scale

Scale

- Multicore CPU
- Single GPU
- Single Machine multi-GPU
- Multi-machine multi-GPU

Basic Optimizations

for ConvNets

- Convolutions, GEMM take ~90% of wall-time
  - Faster Convolutions = faster research
- Parallelize over the mini-batch of input samples
- Asynchronous Hogwild Training
  - Trivially easy to implement and use
  - linear scaling guaranteed (wall-time)

Multicore CPU Optimizations

- Fast convolutions and GEMM
  - SIMD and Threading
  - FFT and Winograd Transforms
- Parallelization over mini-batch
- Asynchronous Hogwild training

Single GPU Optimizations

- Fast convolutions and GEMM
  - FFT and Winograd Transforms
- Parallelization over mini-batch

Single GPU Optimizations

FFT based Convolutions

arxiv.org/abs/1412.7580

Cornell University Library

arXiv.org > cs > arXiv:1412.7580

Computer Science > Learning

Fast Convolutional Nets With fbfft: A GPU Performance Evaluation

Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, Yann LeCun

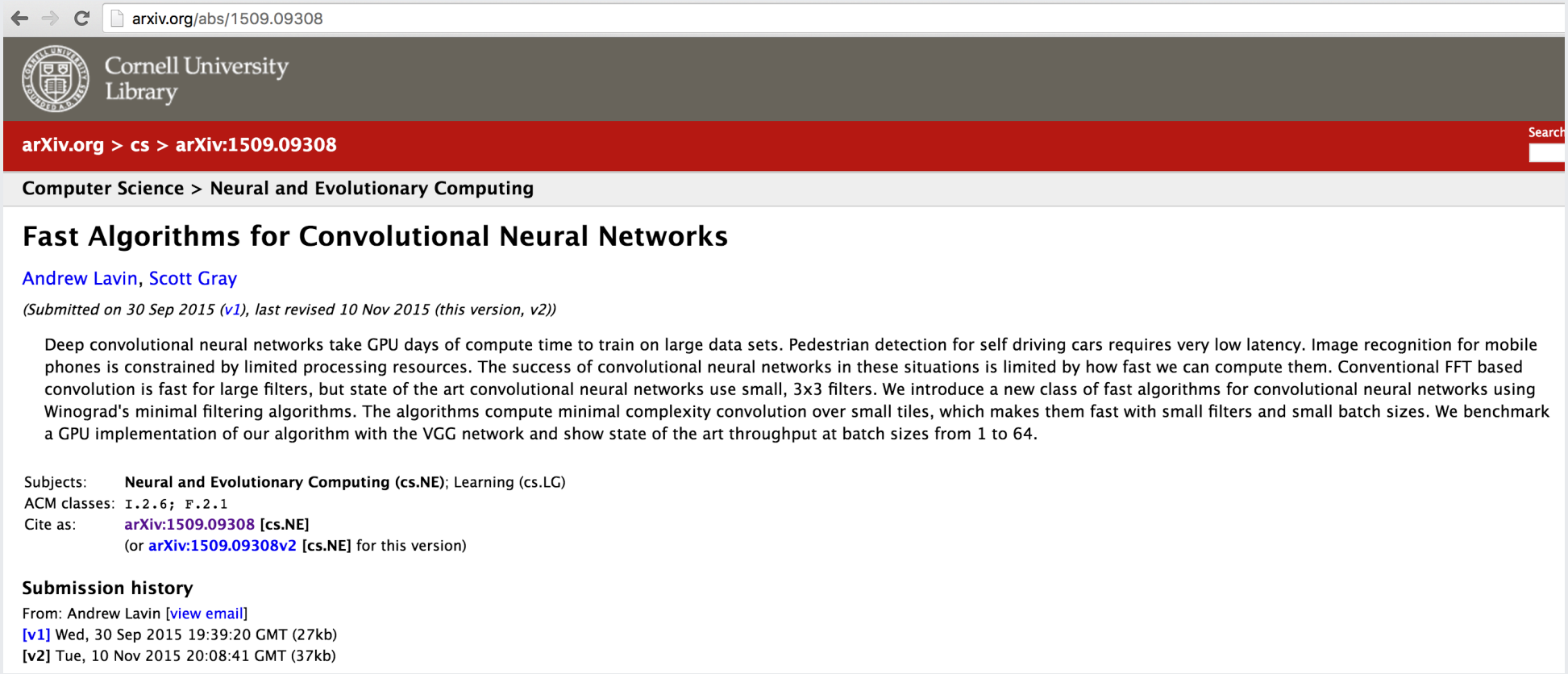
(Submitted on 24 Dec 2014 (v1), last revised 10 Apr 2015 (this version, v3))

We examine the performance profile of Convolutional Neural Network training on the current generation of NVIDIA Graphics Processing Units. We introduce two new Fast Fourier Transform convolution implementations: one based on NVIDIA's cuFFT library, and another based on a Facebook authored FFT implementation, fbfft, that provides significant speedups over cuFFT (ove 1.5x) for whole CNNs. Both of these convolution implementations are available in open source, and are faster than NVIDIA's cuDNN implementation for many common convolutional layers (u 23.5x for some synthetic kernel configurations). We discuss different performance regimes of convolutions, comparing areas where straightforward time domain convolutions outperform Fo frequency domain convolutions. Details on algorithmic applications of NVIDIA GPU hardware specifics in the implementation of fbfft are also provided.

Comments: Camera ready for ICLR2015
Subjects: **Learning (cs.LG)**; Distributed, Parallel, and Cluster Computing (cs.DC); Neural and Evolutionary Computing (cs.NE)
Cite as: **arXiv:1412.7580 [cs.LG]**
(or **arXiv:1412.7580v3 [cs.LG]** for this version)

Single GPU Optimizations

Winograd transform based Convolutions



arxiv.org/abs/1509.09308

Cornell University Library

arXiv.org > cs > arXiv:1509.09308

Search

Computer Science > Neural and Evolutionary Computing

Fast Algorithms for Convolutional Neural Networks

Andrew Lavin, Scott Gray

(Submitted on 30 Sep 2015 (v1), last revised 10 Nov 2015 (this version, v2))

Deep convolutional neural networks take GPU days of compute time to train on large data sets. Pedestrian detection for self driving cars requires very low latency. Image recognition for mobile phones is constrained by limited processing resources. The success of convolutional neural networks in these situations is limited by how fast we can compute them. Conventional FFT based convolution is fast for large filters, but state of the art convolutional neural networks use small, 3x3 filters. We introduce a new class of fast algorithms for convolutional neural networks using Winograd's minimal filtering algorithms. The algorithms compute minimal complexity convolution over small tiles, which makes them fast with small filters and small batch sizes. We benchmark a GPU implementation of our algorithm with the VGG network and show state of the art throughput at batch sizes from 1 to 64.

Subjects: **Neural and Evolutionary Computing (cs.NE)**; Learning (cs.LG)
ACM classes: I.2.6; F.2.1
Cite as: arXiv:1509.09308 **[cs.NE]**
(or arXiv:1509.09308v2 **[cs.NE]** for this version)

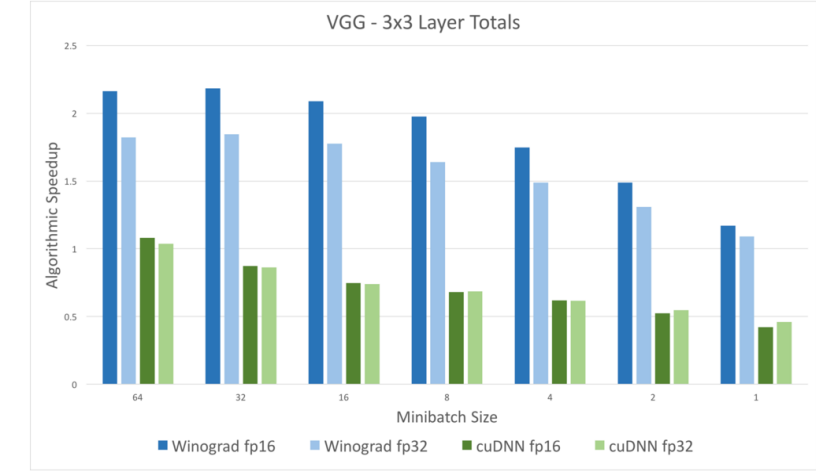
Submission history

From: Andrew Lavin [view email]
[v1] Wed, 30 Sep 2015 19:39:20 GMT (27kb)
[v2] Tue, 10 Nov 2015 20:08:41 GMT (37kb)

Single GPU Optimizations

Winograd transform based Convolutions

nervana



Single GPU Optimizations

- The standard in deep learning:

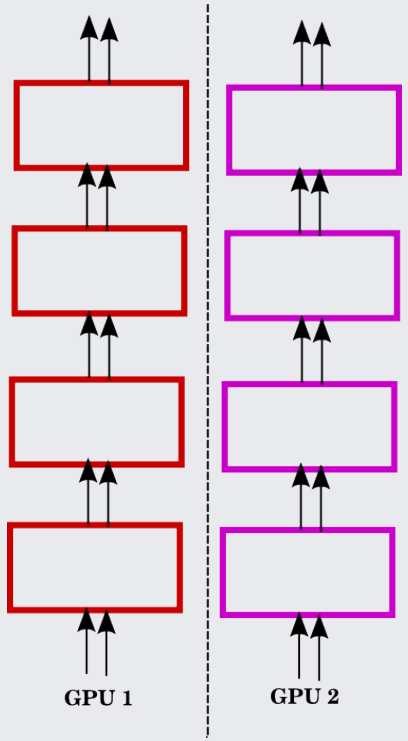
NVIDIA GPUs + CUDA + CuDNN

Multi-GPU optimizations

- Multiple GPUs on single machine

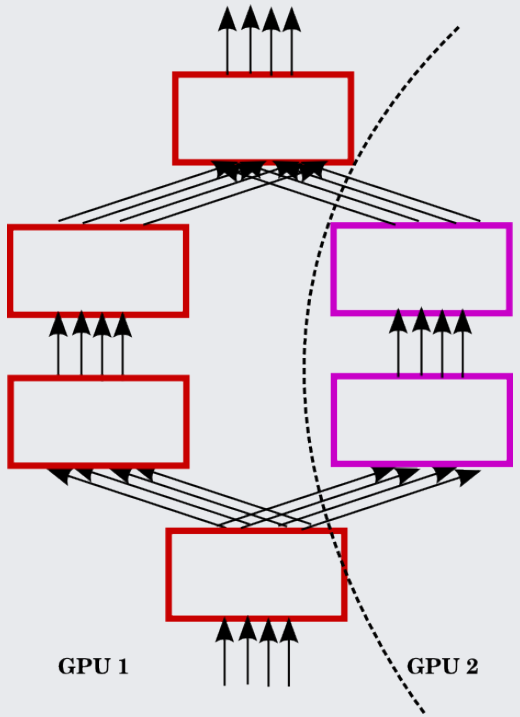
Multi-GPU optimizations

- Data parallel



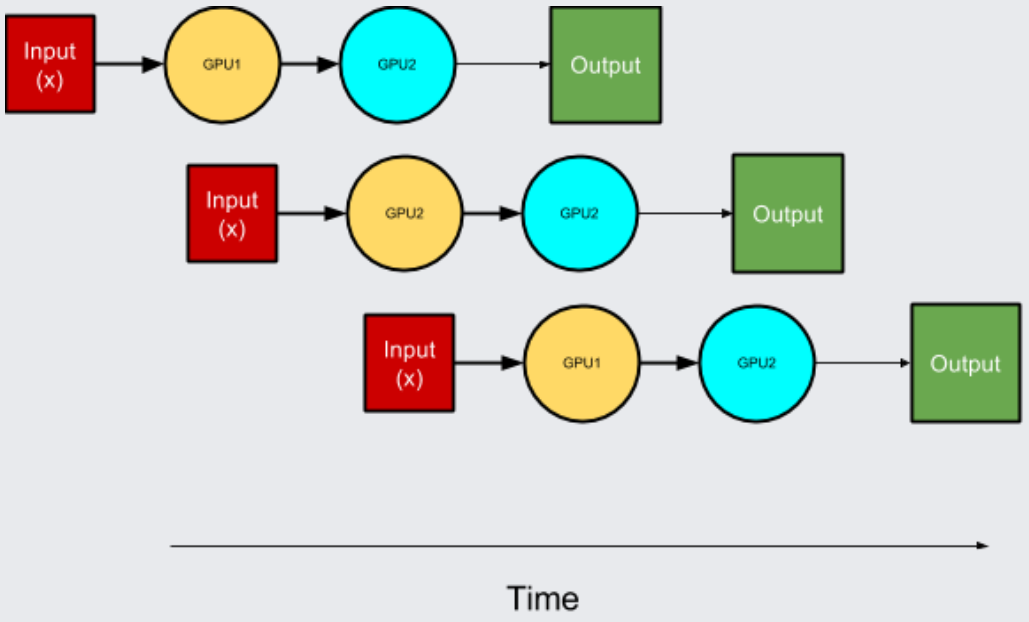
Multi-GPU optimizations

- Model parallel



Multi-GPU optimizations

- Pipeline-parallel

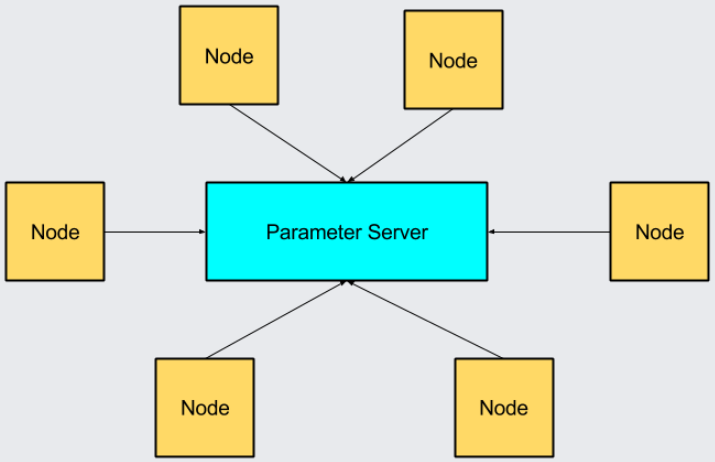


Multi-GPU optimizations

- Bottleneck: interconnects
  - reduce transfer precision
  - faster interconnects
  - different PCI-e topologies

Multi-machine Training

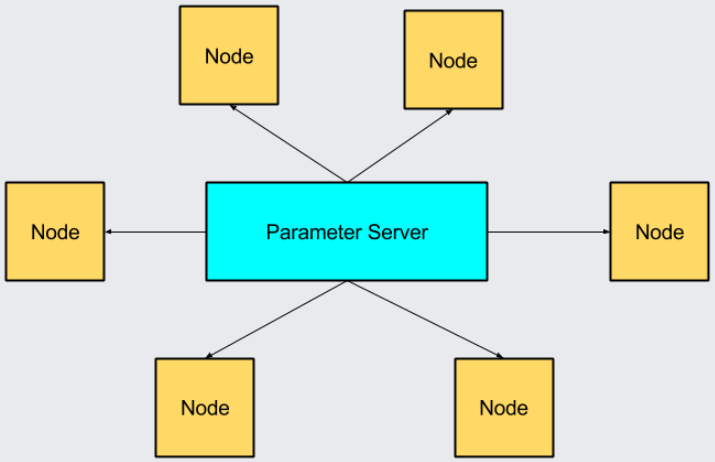
- Multi-machine SGD



Send gradients

Multi-machine Training

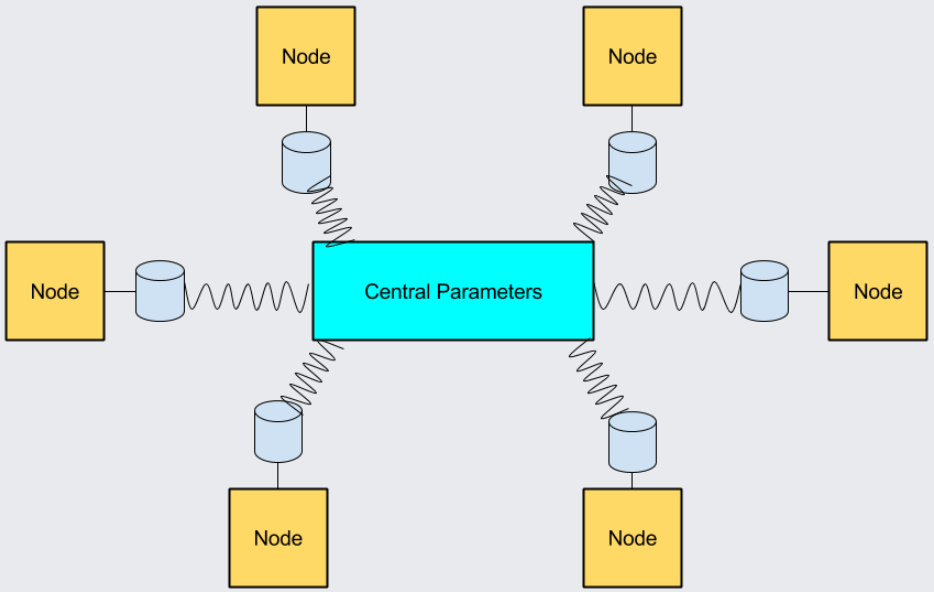
- Multi-machine SGD



Send Weights

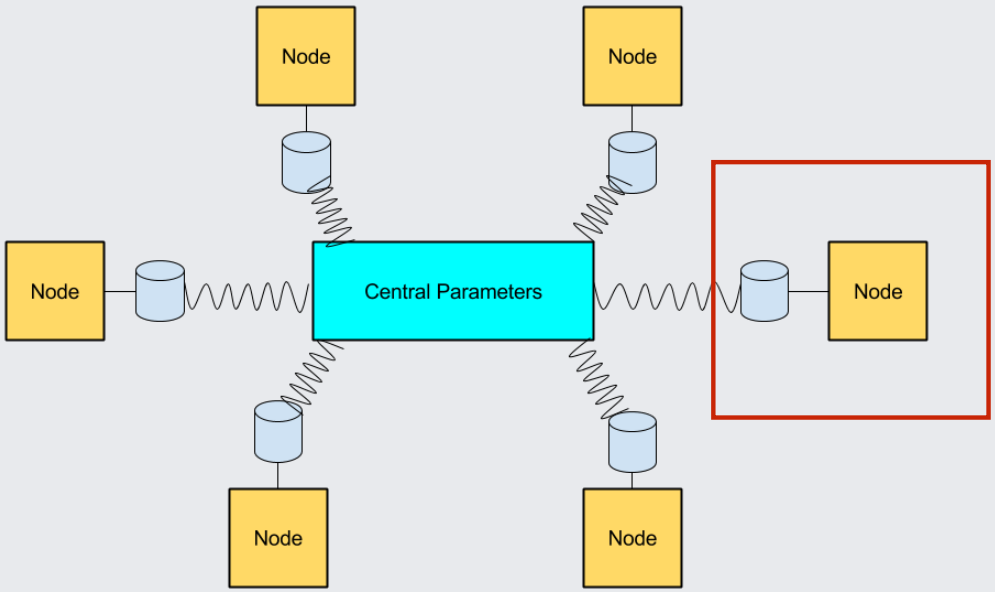
Multi-machine Training

- Elastic Averaging SGD! (Sixin Zhang, Anna Choromanska, Yann LeCun)



Multi-machine Training

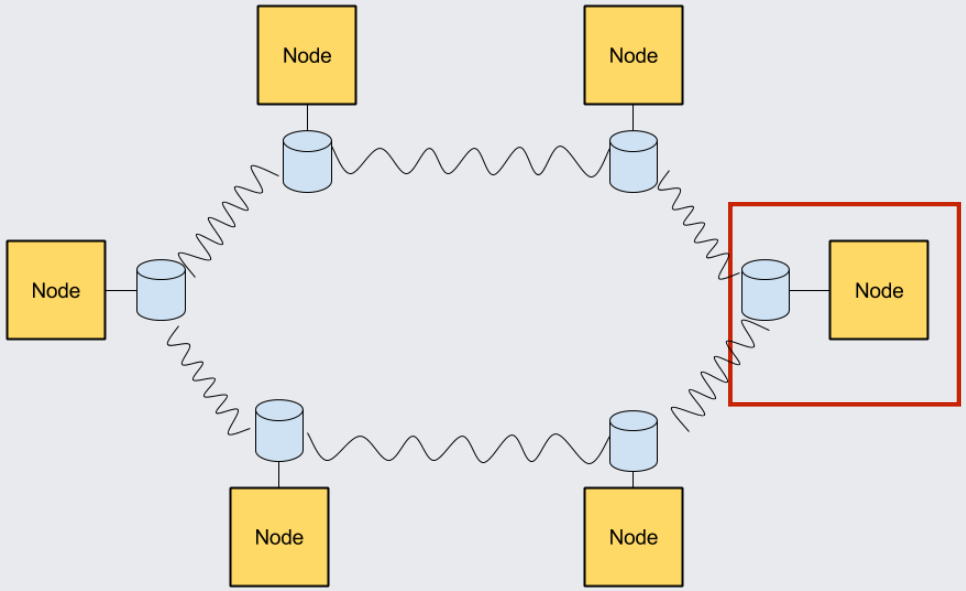
- Elastic Averaging SGD!



Train synchronously
Occasionally, check with master
Dont go too far from everyone else

Multi-machine Training

- Elastic Averaging SGD!



Train synchronously
Occasionally, check with neighbors
Dont go too far from everyone else

Multi-machine Training

- Elastic Averaging SGD!
  - Empirical speedup of SquareRoot(N)
    - N = number of nodes
  - No communication overhead with pre-fetching
    - 128 GPUs (32 clients * 4 GPUs)
    - Sharded parameters over 64 CPU servers
    - Tau = 10, prefetch = 5
    - zero overhead

Multi-machine Training

- Elastic Averaging SGD!
  - Fun fact: Trained AlexNet in 5 epochs of Imagenet data
  - Good success in training Vision and Text networks

Computing Paradigms

Computing Paradigms

- Collectives / MPI
  - Torch, Caffe
- Graph Compilation
  - Theano, MXNet, TensorFlow,
  - Caffe2, Purine

Computing Paradigms

Efficient Collectives + Imperative Programs

- Data / Model / Pipeline parallel seems sufficient
  - allreduce and other MPI collectives
- Torch (nn / autograd / distlearn)
- Caffe

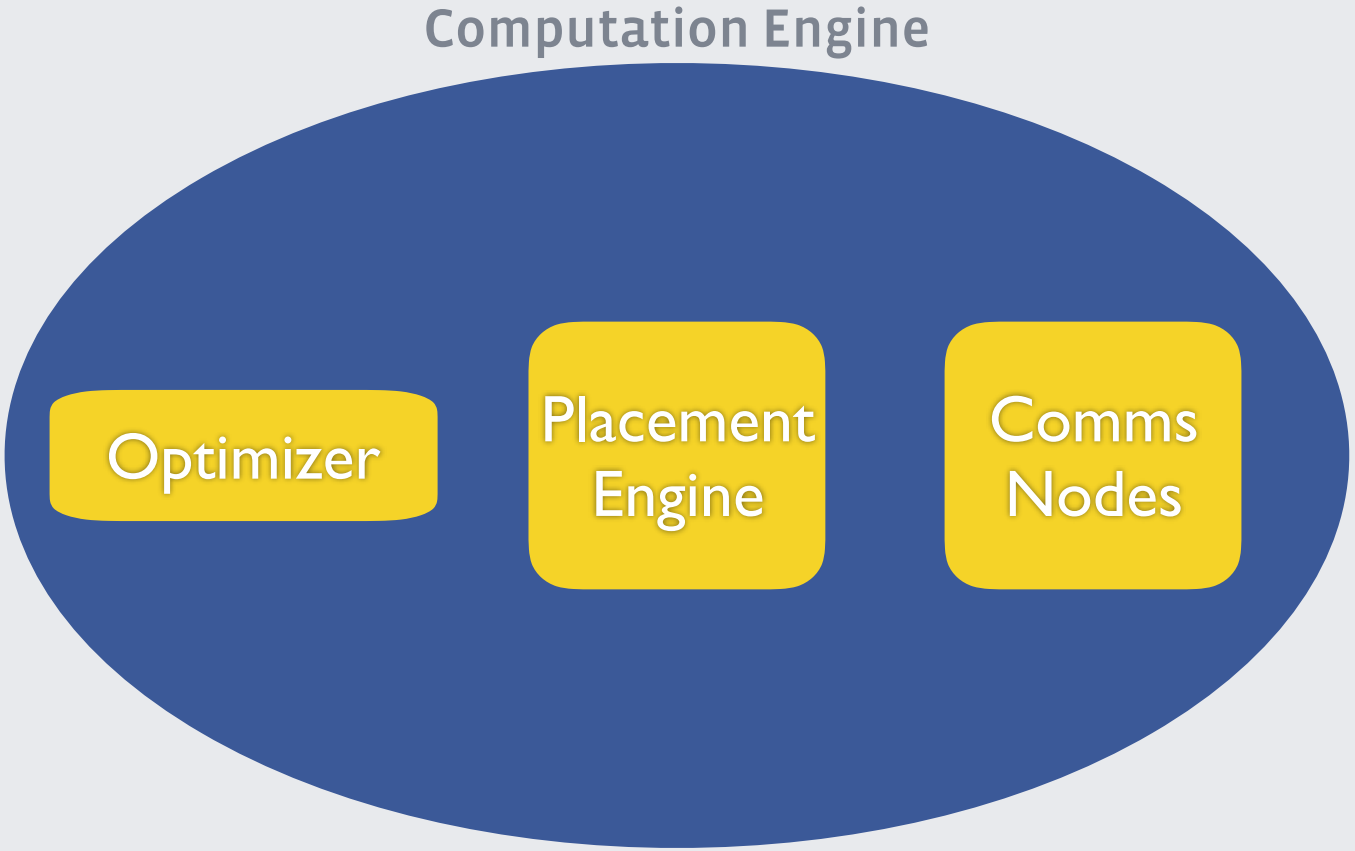
Computing Paradigms

Computational Graph Toolkits

- Intel CnC, Caffe2, TensorFlow, MXNet, Theano
- Graph placement hints + execution
- DSLs to write the computation graphs

Computing Paradigms

Computational Graph Toolkits



Silver Bullet

Imperative Language + Graph Compiler

- Best of both worlds
- Hard problem of automatic graph placement
- Limited heuristic-driven success