Deep Machine Learning on GPUs

Seminar talk | Daniel Schlegel | 28.01.2015

University of Heidelberg, Computer Engineering Group
Supervisor: JProf. Dr. Holger Fröning

Outline

Introduction

Introduction
What is Machine Learning?

- What is learning?
  - Defined as every active, effort demanding (mental and psychomotorical), confrontation of a human with any objects of experience. In doing so intern representations are created and modified which causes a relative and permanent change of skills and capabilities
- What is Machine Learning
  - Attempt to imitate the human/animal learning process.
  - No explicitly defined functions on how to react to a specific input $\Rightarrow$ System has to "learn" the reaction.
- What is Deep Machine Learning?
  - Like ML but the structure of the system is closer to the human brain.

Source: http://35if8l37rcx617qr9x4es9ybri5.wpengine.netdna-cdn.com/wp-content/uploads/2014/01/Brain1.jpg

Introduction

- Origins are in the area of Artificial Intelligence (AI)
  - Today: Separate field
  - Parts of AI and probability theory
- A pioneer of machine learning once said:

“I discovered how the brain really works.

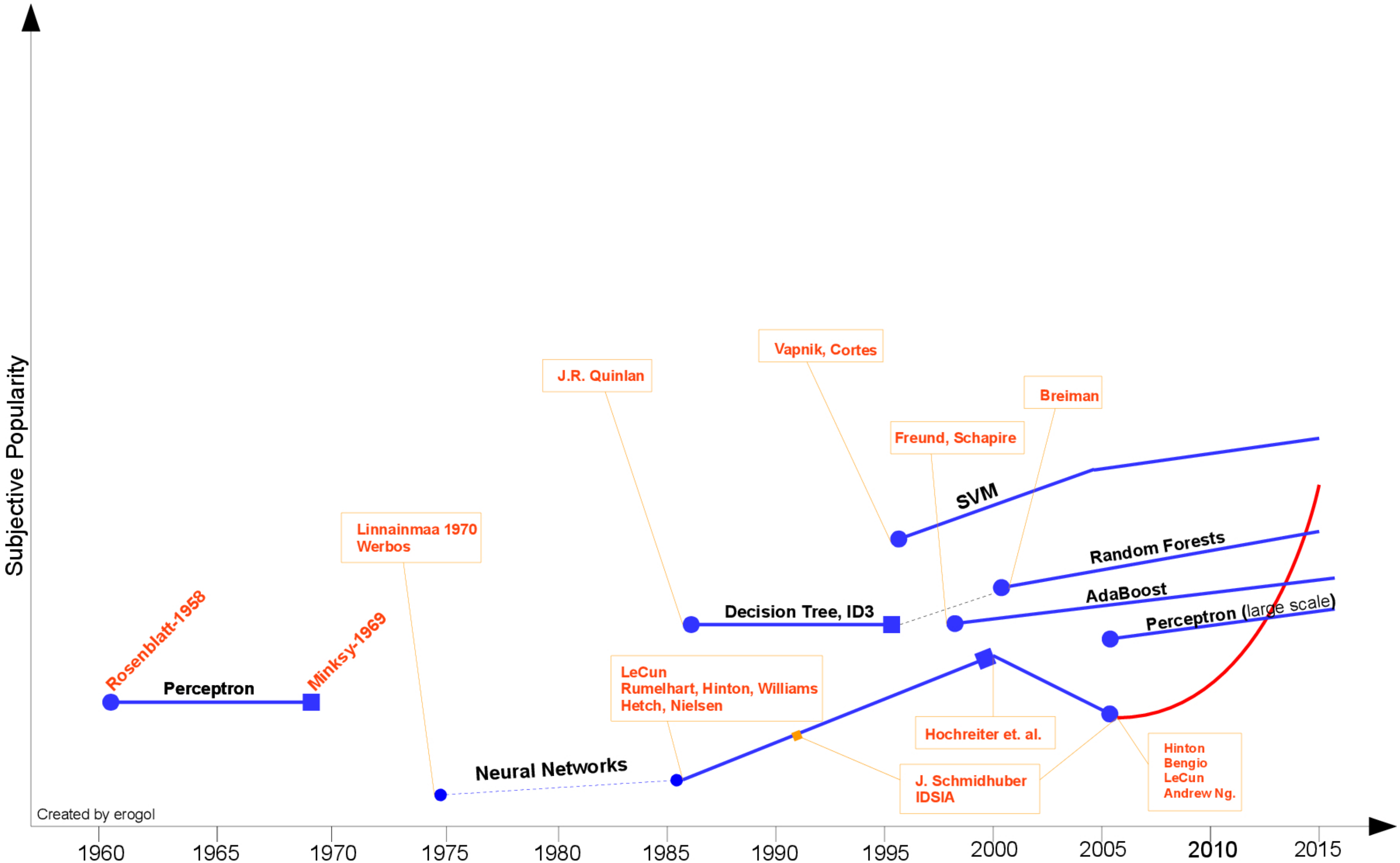
Once a year for the last 25 years.”

Geoffrey Hinton

- We can rebuild the structure of the brain
  - We are able to train it to do what we want.
  - But we don’t really understand it!

Introduction

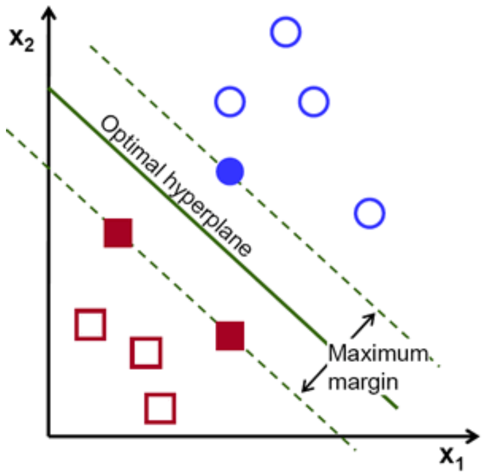
History

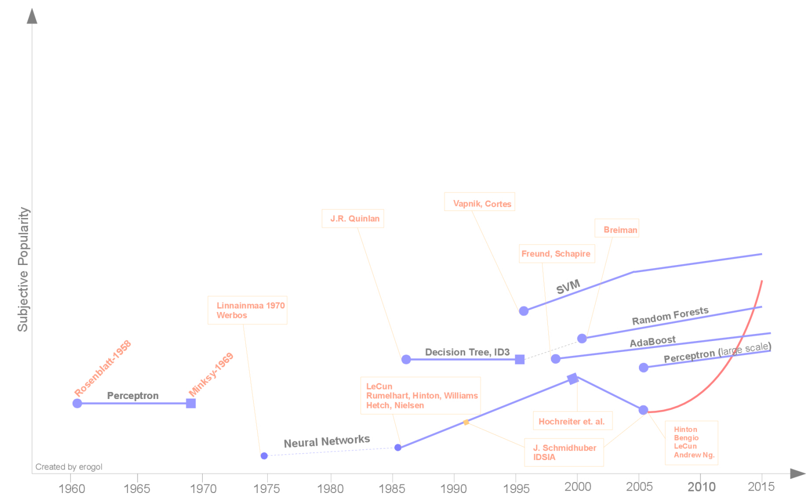


Source: http://www.aboutdm.com/2013/04/history-of-machine-learning.html

Introduction

History

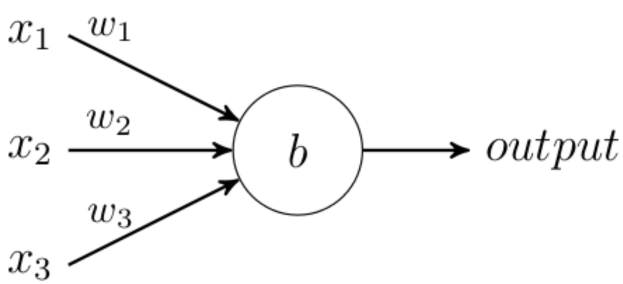


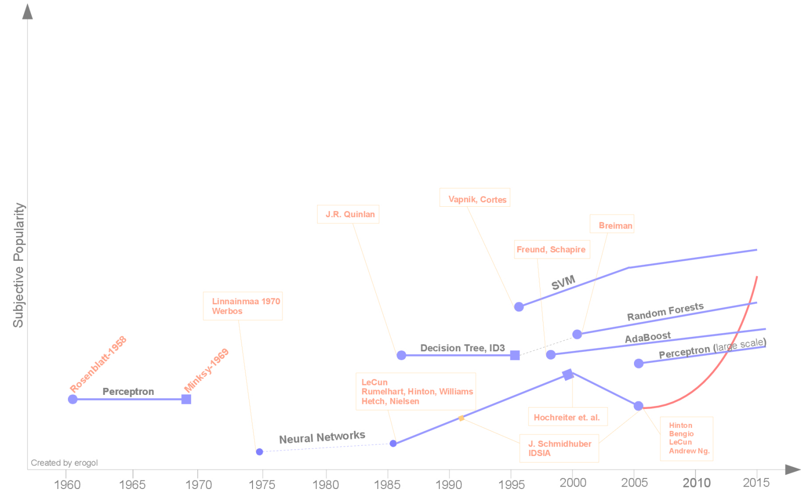


- **Support Vector Machines (SVMs)**
  - SVMs superseded NNs in the 90th
  - They use hyperplanes to separate the classes
  - Only objects close to the hyperplane are important for learning
  - Classes need to be linear separable
    - ★ Or an additional transformation is needed (higher dimension)
    - ★ For image classification ≫ 100k dimensions (RGB image is 3D)

Source: http://www.aboutdm.com/2013/04/history-of-machine-learning.html
Source: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Introduction
History





- **Perceptrons**
  - Predecessor of modern Neural Networks
  - Output either “0” or “1”
  - Only for simple tasks
- **Neural Networks**
  - Emulate the human brain
  - Explained in the next section

Source: http://www.aboutdm.com/2013/04/history-of-machine-learning.html

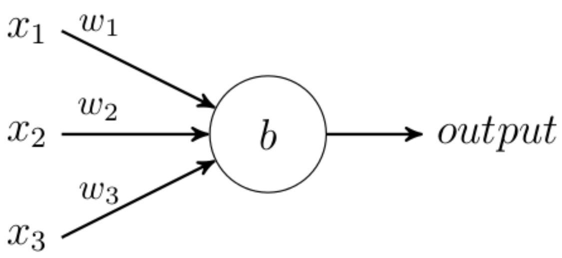
Introduction
Application areas

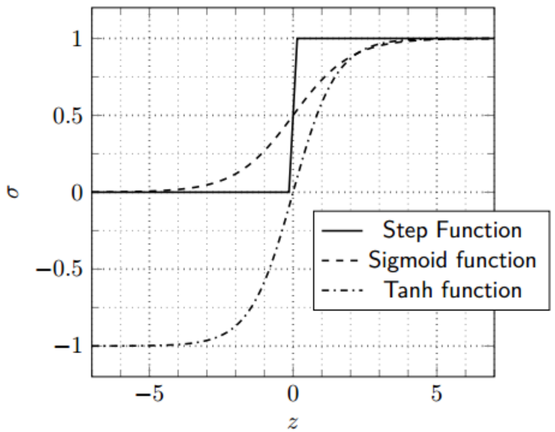
- **Image classification**
  - What does the picture show
- **Natural Language Processing**
  - Speech to text conversion
- **Optical Character Recognition**
  - Convert handwritten text to text document
- Email **Spam filter**
  - Automatically send unwanted emails in Spam folder
- **Google Translate**
  - Translate a text without human intervention
- And of course, **Big Data**
  - Finding structure in unstructured data

Neural Networks

Neural Networks
What are Neural Networks?

- Neural Networks are a section of Machine Learning
  - Imitate structure of brain
  - Artificial neuron is basic building block
- Artificial neurons
  - Take *n* inputs $x_1 \dots x_n$ and calculate the *output*
  - Most NNs use Sigmoid or Tanh function
    - ★ **Sigmoid:** *not normalized*; **Tanh:** *normalized*
    - ★ Smooth transition between zero and one
    - ★ Outputs show probability





Neural Networks
How do they work?

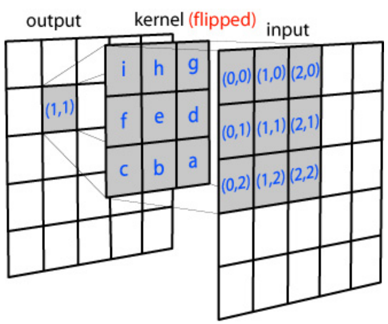
- How do they learn?
  - **Supervised**
    - Network learns from classified data
    - Network adjusts parameters to reduce cost function
    - Used for most tasks, e.g. object classification
  - **Unsupervised**
    - Network learns from unlabeled data
    - Find structure in the data
- Weights and biases are adjusted by Back-propagation
- Basics of **Back-propagation**
  - Process a labeled training object
  - Compare output to desired output (cost function)
  - Calculate the share of each parameter to the error
  - Adjust the weights and biases to minimize error

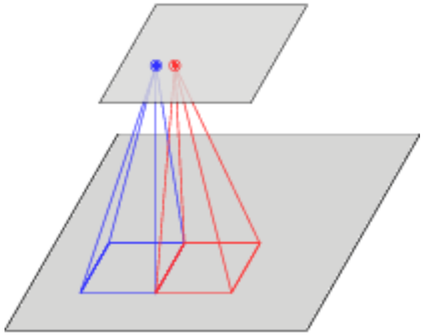
Neural Networks
How do they work?

- **Neural Networks (NNs)**
  - Simplest implementation
  - No hierarchical feature extraction
- **Deep Neural Networks (DNNs)**
  - Based on the structure of the human brain
  - All-to-all connection between layers
  - Millions of weights and biases
    - ★ Nearly impossible to train with more than 3 layers
- **Convolutional Neural Networks (CNNs)**
  - Based on the human visual recognition system
  - No all-to-all connection
  - Shift invariance during feature extraction
  - Reduced amount of weights and biases
    - ★ Can be trained with many layers (common are 7 layers)

Neural Networks
How do they work? | Basic operations

- **Convolution**
  - Used for feature extraction
  - Reduces amount of weights and biases
  - Reduces feature map size when used with stride
- **Pooling**
  - Used to reduce the size of feature maps
  - Several different forms
    - ★ MaxPooling (most common)
    - ★ MedianPooling
    - ★ AveragePooling
- **SoftMax**
  - Used at the output to scale the probabilities
    - ★ All outputs sum up to “1”
    - ★ All outputs lie between “0” and “1”

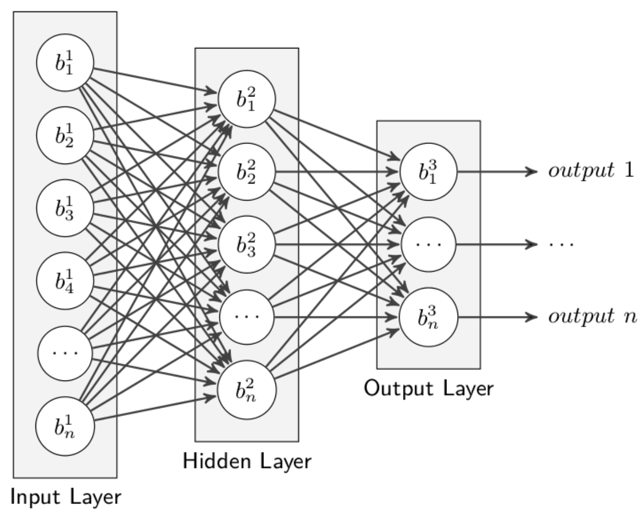




Source: http://wiki.ldv.ei.tum.de/show_image.php?id=259
Source: http://www.songho.ca/dsp/convolution/files/conv2d_matrix.jpg

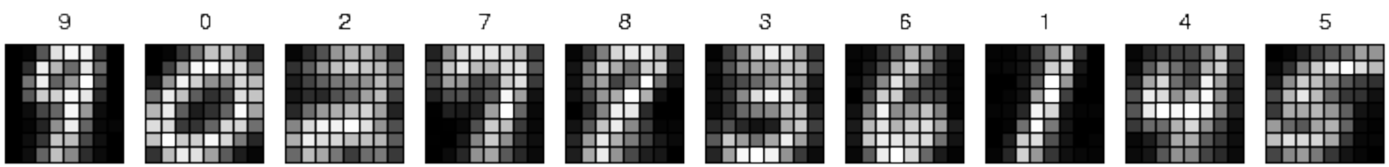
Neural Networks
Example (simple version)

- **Simple Neural Network for handwritten digit recognition**
  - Shallow NN (only one hidden layer)
  - Number of neurons: 810
  - Input images are all the same size and centered (MNIST dataset)
  - Error rate at ~ 5 %



Neural Networks
Example (simple version)

- **Simple Neural Network for handwritten digit recognition**
  - Shallow NN (only one hidden layer)
  - Number of neurons: 810
  - Input images are all the same size and centered (MNIST dataset)
  - Error rate at ~ 5 %

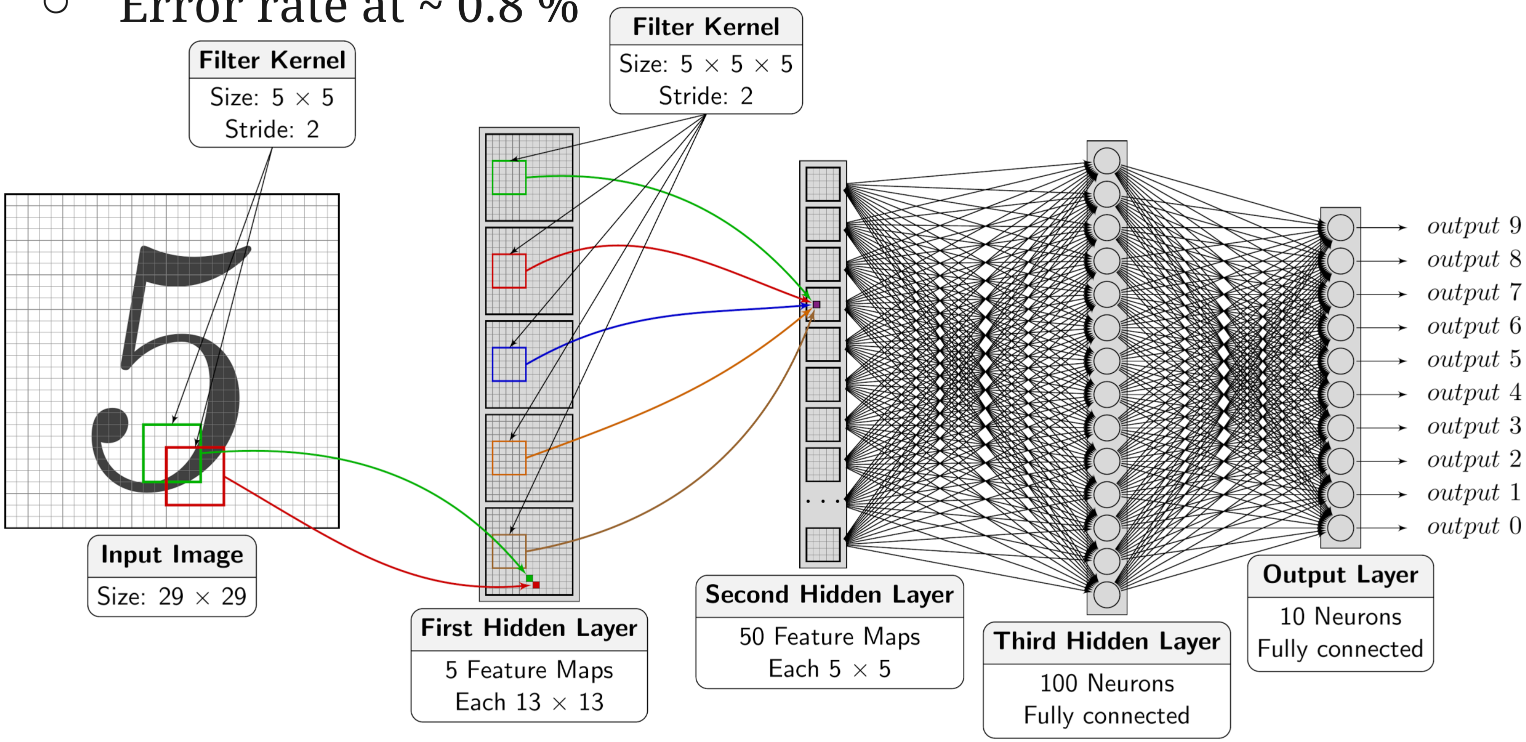


- Shallow architecture
  - Easy to implement and train
  - “Human understandable” weights and biases
  - Not accurate enough for most tasks

Source: http://nn.cs.utexas.edu/demos/digit-recognition/

Neural Networks
Example (advanced version)

- **Convolutional Neural Net for handwritten digit recognition**
  - Number of neurons: 2989
  - Same input as in the first example (one pixel for padding)
  - Error rate at ~ 0.8 %



Tools for Neural Networks

Tools for Neural Networks
Available tools

- Lots of frameworks and libraries are available
  - Caffe
    - ★ Universal framework with good performance
    - ★ CPU and GPU implementation
  - cuDNN
    - ★ Highly optimized functions for NVidia GPUs
  - cuda-convnet2
    - ★ Python library written in C++/CUDA-C
    - ★ Multi GPU support
  - THEANO
    - ★ Full Python implementation (CPU and GPU)
  - Microsoft Azure Machine Learning
    - ★ Cloud based Neural Networks
  - MATLAB
    - ★ Text based or graphical

Tools for Neural Networks
Caffe

- Open Source Project: BVLC
  - https://github.com/BVLC/Caffe
- No “real” programming needed
  - Structure defined by configuration files
  - Edit paths is predefined scripts
- Can run on CPU and GPU
  - determined by parameter
- Lots of examples included
  - Character recognition
  - Object classification
- Currently only single GPU support

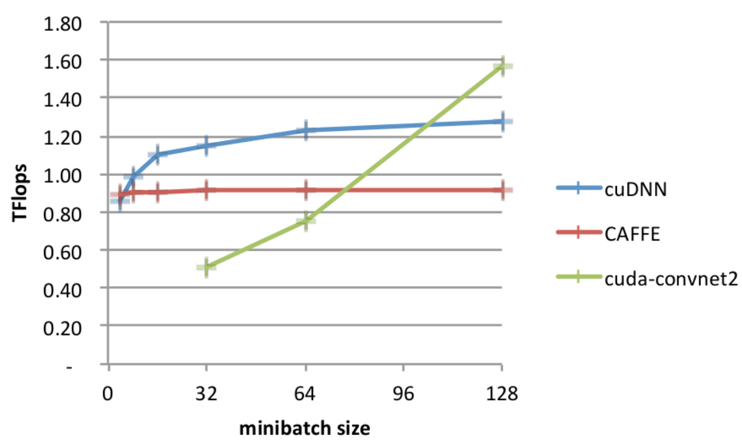
```
# Simple convolutional layer
layers {
    name: "conv1"
    type: CONVOLUTION
    bottom: "data"
    top: "conv1"
    convolution_param {
        num_output: 96
        kernel_size: 11
        weight_filler {
            type: "gaussian"
            std: 0.01
        }
        bias_filler {
            type: "constant"
            value: 0
        }
    }
}
```

Tools for Neural Networks
Caffe | Implementation

- How does Caffe work internally?
- Each function is implemented for CPU and GPU
- Uses cuBLAS library internally for most tasks
- Between each layer is a “*blob*” for the communication
    - Include forward and backward pass
    - Multi dimensional array (num, channels, height & width)
    - Syncs CPU and GPU memory automatically if needed
- Neuron Layer on GPU
    - Performed in two steps
        - ★ Sum up all inputs with weights and biases (SAXPY + all-reduce)
        - ★ Calculate output with corresponding activation function
- Convolutional Layer on GPU
    - Performed in four steps
        - ★ Rearrange data (`im2col()`)
        - ★ Perform convolution (`cublasSgemm()`)
        - ★ Add bias to results
        - ★ Calculate final value with activation function

Tools for Neural Networks
cuDNN

- Library for CUDA capable GPUs from NVidia
    - GPU optimized functions for DNNs
    - Including forward and backward operations
    - Not open source, but freely available at NVidia https://developer.nvidia.com/cuDNN
- Will be included in Caffe 1.0 (not yet released)
    - Speedup of ~ 13 % compared to normal implementation
        - ★ 7 days training ⇒ 6 days training
- Measurements done with cuDNN RC1
    - CUDA 7 brings new version with improved performance



Tools for Neural Networks

cuda-convnet2

- Open source project hosted at https://code.google.com/p/cuda-convnet2/
- Python library written in C++ and CUDA-C
- Fastest implementation so far
- Supports multiple GPUs with different parallelism approaches[1]
- Network is defined by configuration file (like Caffe)
- Written for ILSVRC-2012
  - One node with two GPUs
  - Winning system with 17 % error rate (second best: 27 %)
- 6.25x Speedup on 8 GPUs

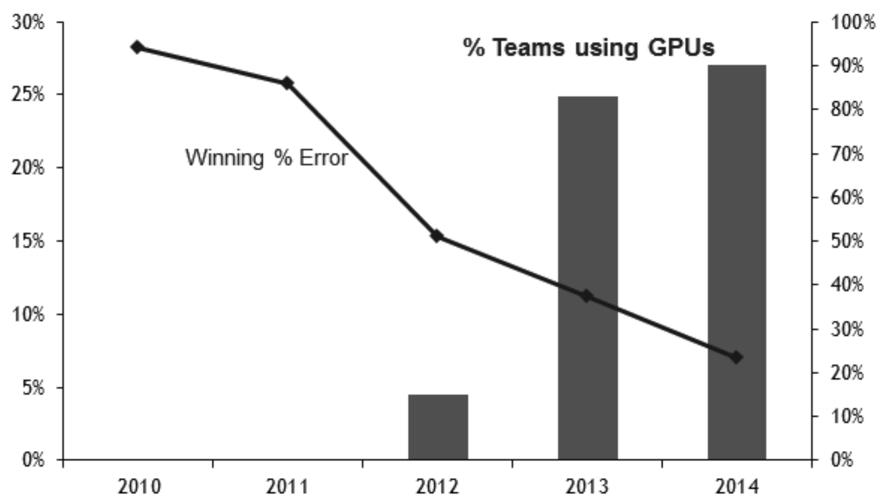
```
# Simple convolutional layer
[conv32]
  type = conv
  inputs = data
  channels = 3
  filters = 32
  padding = 4
  stride = 1
  filterSize = 9
  neuron = logistic
  initW = 0.00001
  initB = 0.5
  sharedBiases = true
  sumWidth = 4
```

1. See Alex Krizhevsky, *One weird trick for parallelizing convolutional neural networks*, eprint arXiv:1404.5997, 2014

DNN on GPUs

DNN on GPUs

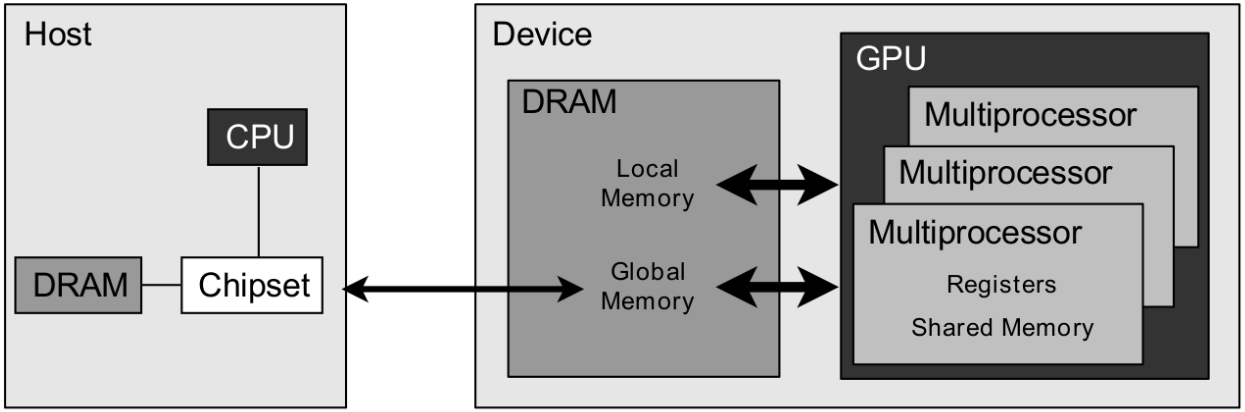
- Number of competitors in the ImageNet challenge.
  - 2012 ⇒ One system, won with 10% lead (mostly CPU-based SVMs)
  - 2014 ⇒ 90 % use GPUs
- Networks can get more complex due to high computational power
  - Only limited by GPU memory



Source: http://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library/

DNN on GPUs
GPU | What's that?

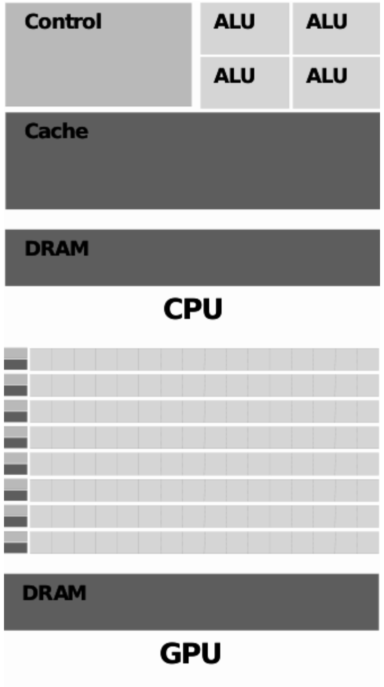
- Extension card for PCs
  - Optimized for graphics processing
  - Recent GPUs capable of general purpose computations (GPGPU)
  - Special GPUs without video output
- Used as an accelerator
  - Can increase the performance of special workloads
  - Different architecture and execution model than a CPU



Source: NVidia

DNN on GPUs
GPU | Basic architecture

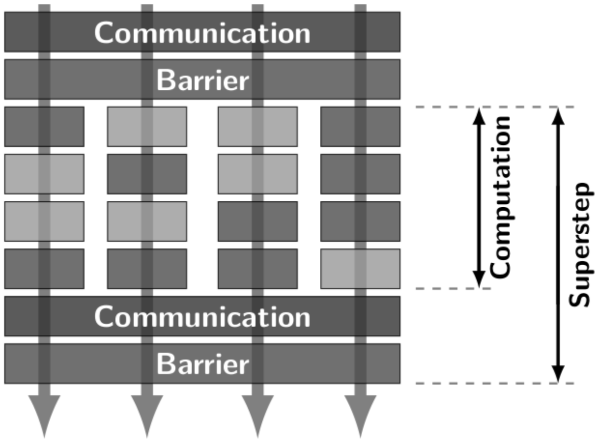
- CPU: Multicore ⇔ GPU: Manycore
  - CPU has few complex cores
  - GPU has many simple cores
- Basic building block **Streaming Multiprocessor (SM)**
  - SMs contain many ALUs for calculation
  - Each ALU in an SM performs same operation on different memory ⇒ **SIMT**
- Context switch every clock cycle
  - Lots of outstanding loads
    ⇒ Memory latency can be tolerated
- High memory bandwidth
- User controlled cache (SharedMemory)



Source: NVidia

DNN on GPUs

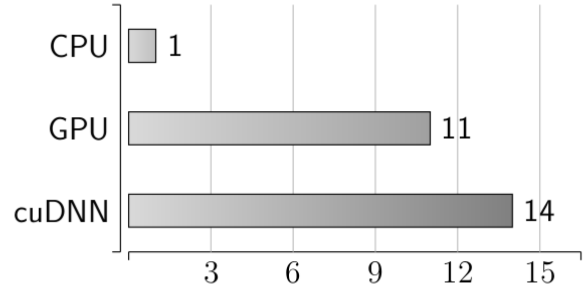
GPU | Execution model



- Execution described by “Bulk Synchronous Parallel” model
  - Execution is done in supersteps
    - ★ Computation
    - ★ Communication
    - ★ Barrier
  - More tasks than resources to overcome parallel slackness
- Memory loads and stores should be coalesced
- Task is split into several blocks
  - Block indices have three dimensional ID
  - On block runs on one SM
  - No safe synchronization between blocks possible

DNN on GPUs
Performance

- Execution model optimal for NNs
  - Compute one layer
  - Perform memory operations
  - Synchronize
- High computational power of GPUs can be utilized
  - Caffe on GPU is 11x faster than on CPU (14x with cuDNN)
  - cuDNN achieves 2.5 TFLOPS on a GTX 980 (51 % of peak perf.)
- No data dependencies in layers
  - Relatively easy to implement

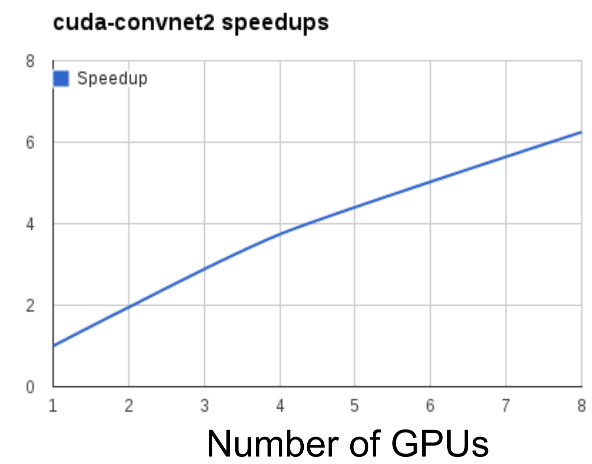


GPU: NVidia K40
CPU: 24-core Intel E5-2697v2 CPU @ 2.4GHz running Caffe with Intel MKL 11.1.3

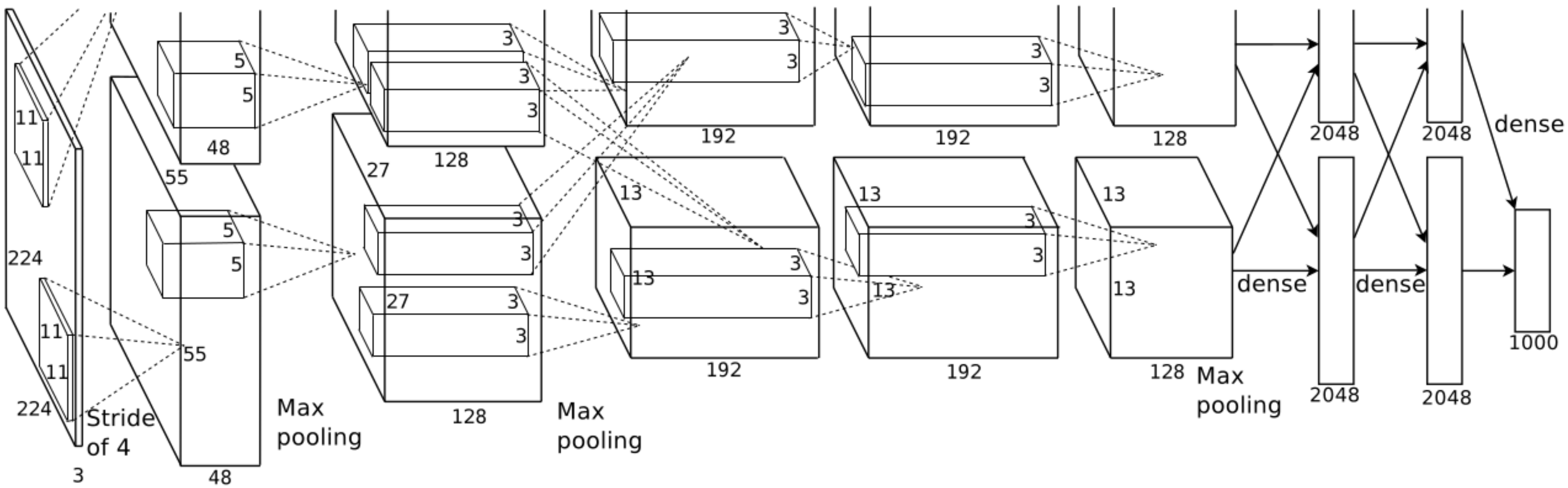
DNN on GPUs
Scalability evaluation

- Important Factor: *Does it scale?*
  - Several multi GPU implementations
  - All have good linear speedup
- Only the training has to be split
  - Each node broadcasts changes in weights and biases
- GPU memory is very limited
  - Limits size of networks
  - Limits mini-batch size

    ⇒ Multiple GPUs increase the possible size
- CNNs reduce amount of communication dramatically
  - CNNs can be designed to fit network topology
- Only tested and documented with 8 GPUs in one node and 16 nodes with 4 GPUs each



DNN on GPUs
Example

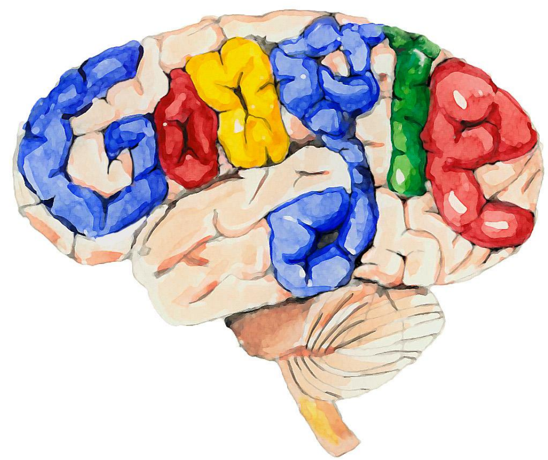


- Winning system of the ILSVRC-2012
  - 1,000 classes and 1,400,000 images
- Specs:
  - Network split to two GPUs (NVidia GTX 580)
  - 650.000 neurons
  - 60.000.000 weights and biases
- P2P access limited to two cards
  - Have to be connected to same PCIe root complex

Source: A. Krizhevsky, I. Sutskever and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks.* NIPS. 2012.

DNN on GPUs
Example | System level

- Google build a “Brain” to find two most common images in the internet (2006 - 2011)
  - 1,000 nodes (2,000 CPUs, 16,000 cores)
  - ~ 600 kW energy consumption (IDLE)
  - $5,000,000 system costs
  - 10,000,000,000 connections
    - ★ Complexity comparable to a bee
- First GPU-based challenger (2014)
  - 3 nodes with (3 Tesla K20 each)
  - 4 kW energy consumption
  - $33,000 system costs
- Second GPU-based challenger (2014)
  - 1 node (3 GeForce Titan Z / 6 GPUs)
  - 2 kW energy consumption
  - $12,000 system costs



Source: http://hilaryschenker.files.wordpress.com/2011/08/googlebrain2.jpg

What can we expect?

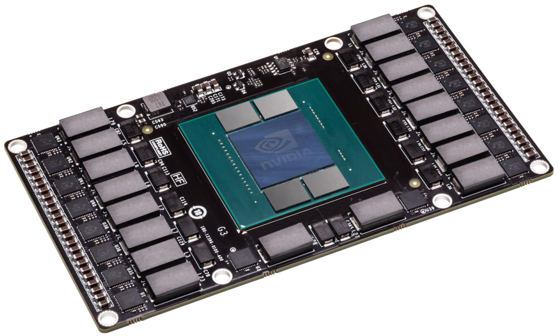
- Robot learning how to cook by watching YouTube videos
- Two CNNs for:
  - Object recognition
    - ★ Which ingredient is next
  - Grasping type
    - ★ Which tool and which operation
  - Precisions:
    Object 79 %; Grasping type 91 %; Action 83 %
- Predefined set of tools and ingredients
  - Can not learn new tools or ingredients

	Grasp_PoS(LH, Brush) Grasp_PrS(RH, Corn) Action_Spread(Brush, Corn)	Grasp_PoS(LH, Brush) Grasp_PrS(RH, Corn) Action_Spread(Brush, Corn)
	Grasp_PoS(LH, Spreader) Grasp_PrL(RH, Bread) Action_Spread(Spreader, Bread)	Grasp_PoS(LH, Spreader) Grasp_PrL(RH, Bowl) Action_Spread(Spreader, Bowl)

Source: http://images.gizmag.com/hero/youtube-robot-7.jpg
Source: Y. Yang et. al.. *Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web.* AAAI-15. 2015.

Outlook

- GPU memory and performance increased over the last years
  ⇒ Stacked Memory
  - Bigger networks
  - Less copy operations
- Faster Host-GPU connection
  ⇒ NVlink
  - Biggest bottleneck at the moment
- Focus of most NN architects/researchers lies on GPUs
  ⇒ A lot of research at the moment
  - Better accuracy of DNNs
  - Better performance on GPUs
  - Better communication strategies for clusters



Source: http://cdn.wccftech.com/wp-content/uploads/2014/03/NVIDIA-Pascal-GPU-Chip-Module.jpg

Conclusion

- DNNs offer an unified way to realise ML systems
  - A lot of frameworks available
  - Basic functions are the same for different tasks
- High amount of parallelism with few data dependencies
  - Fits the BSP model
  - Optimal task for GPUs
- CNNs reduce amount of communication
  - Can be trained with lots of layers
    - ★ Complex networks can be realized
    - ★ High accuracy if trained well
  - Can be designed to match a network topology
    - ★ Increased performance on cluster level

The End

Questions?

References

[1] E. Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning Series, MIT Press. 2014.

[2] Lotter, Hempel. *Lernen, Lernschwierigkeiten – Diagnostik der Lernvoraussetzungen*. Regierung Oberbayern. 2008.

[3] S. Chetlur. *cuDNN: Efficient Primitives for Deep Learning*. arXiv preprint arXiv:1410.0759c2. 2014

[4] T. Brants et. al.. *Large Language Models in Machine Translation*. EMNLP-CoNLL. 2007.

[5] W. Ding, et. al.. *Theano-based Large-Scale Visual Recognition with Multiple GPUs*. ICLR. 2015.

[6] P. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press. 2012.

[7] H. Fröning. *GPU Computing Slides*. University of Heidelberg, ZITI. 2014.

[8] Y. Jia et. al.. *Caffe: Convolutional Architecture for Fast Feature Embedding*. arXiv preprint arXiv:1408.5093v1, 2014.

[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*. NIPS. 2012.

[10] A. Krizhevsky. *One weird trick for parallelizing convolutional neural networks*. arXiv:1404.5997 [cs.NE]. 2014.

[11] Y. LeCun, et. al.. *Backpropagation applied to handwritten zip code recognition.* AT&T Bell Laboratories. 1989.

[12] Y. LeCun, et. al.. *Efficient BackProp*. Neural Networks: Tricks of the trade, Springer. 1998.

[13] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press. 2012.

[14] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press. 2015.

[15] NVidia. *User Guide – cuDNN Library*. NVidia. DU-06702-001_v6.5. 2014.

[16] T. Paine et al.. *Gpu asynchronous stochastic gradient descent to speed up neural network training*. arXiv preprint arXiv:1312.6186. 2013.

[17] O. Russakovsky el. al.. *ImageNet Large Scale Visual Recognition Challenge*. arXiv preprint arXiv:1409.0575. 2014.

[18] S. Shalev-Shwartz, S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press. 2014.

References

[19] N. Srivastava, et. al.. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. University of Toronto, Department of Computer Science. 2014.

[20] C. Stergiou, D. Siganos. *Neural Networks*. Imperial College London. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#WhatisaNeuralNetwork, last visited 28.12.14.

[21] X. Tang. *Introduction to General Purpose GPU Computing*. University of Rochester. 2011.

[22] L. G. Valiant. *A bridging model for parallel computation*. Communications of the ACM, Volume 33 Issue 8. 1990.

[23] J. Wart, et. al.. *Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster*. Eindhoven University of Technology, The Netherlands. 2011.

[24] O. Yadan et. al.. *Multi-gpu training of convnets*. arXiv preprint arXiv:1312.5853. 2013.

[25] Y. Yang et. al.. *Robot Learning Manipulation Action Plans by “Watching” Unconstrained Videos from the World Wide Web*. AAAI-15. 2015.

[26] Y. Zou et. al.. *Deep learning platform and its applications*. Proceedings of the VLDB Endowment. 2014.