Parallelizations within Neural Networks

James Mnatzaganian

Jim Mazza

Outline

- Artificial Neural Networks (ANNs)
- Types of Parallelism
- Task Partitioning and Grain Size
- Cluster vs. GPU Implementations
- Network Considerations
- Final Remarks

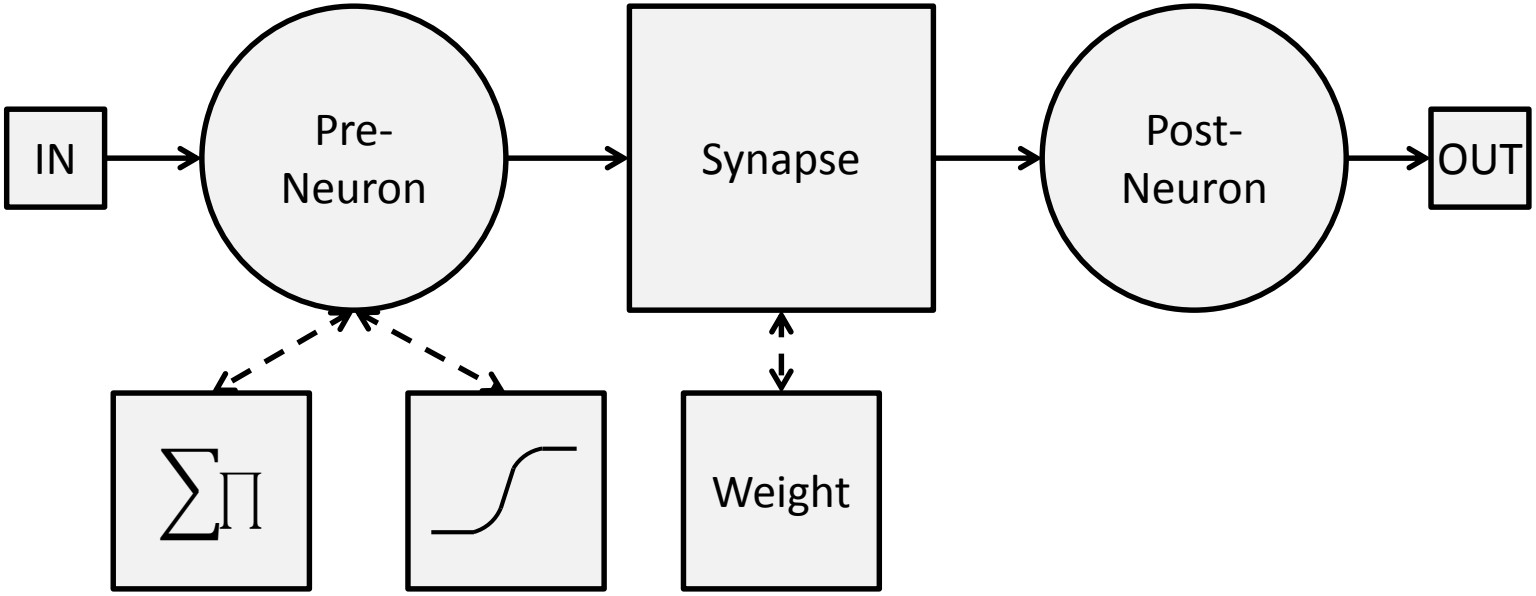
Artificial Neural Networks (ANNs)

Outline

- ANN Overview
- Multilayer Perceptron (MLP)
- Restricted Boltzmann Machine (RBM)
- Deep Belief Network (DBN)

ANN – Overview

- "Biologically-inspired"
  - Neurons (sum of products)
  - Synapses (weight + activation function)
- Non-linear, *universal*, function approximators



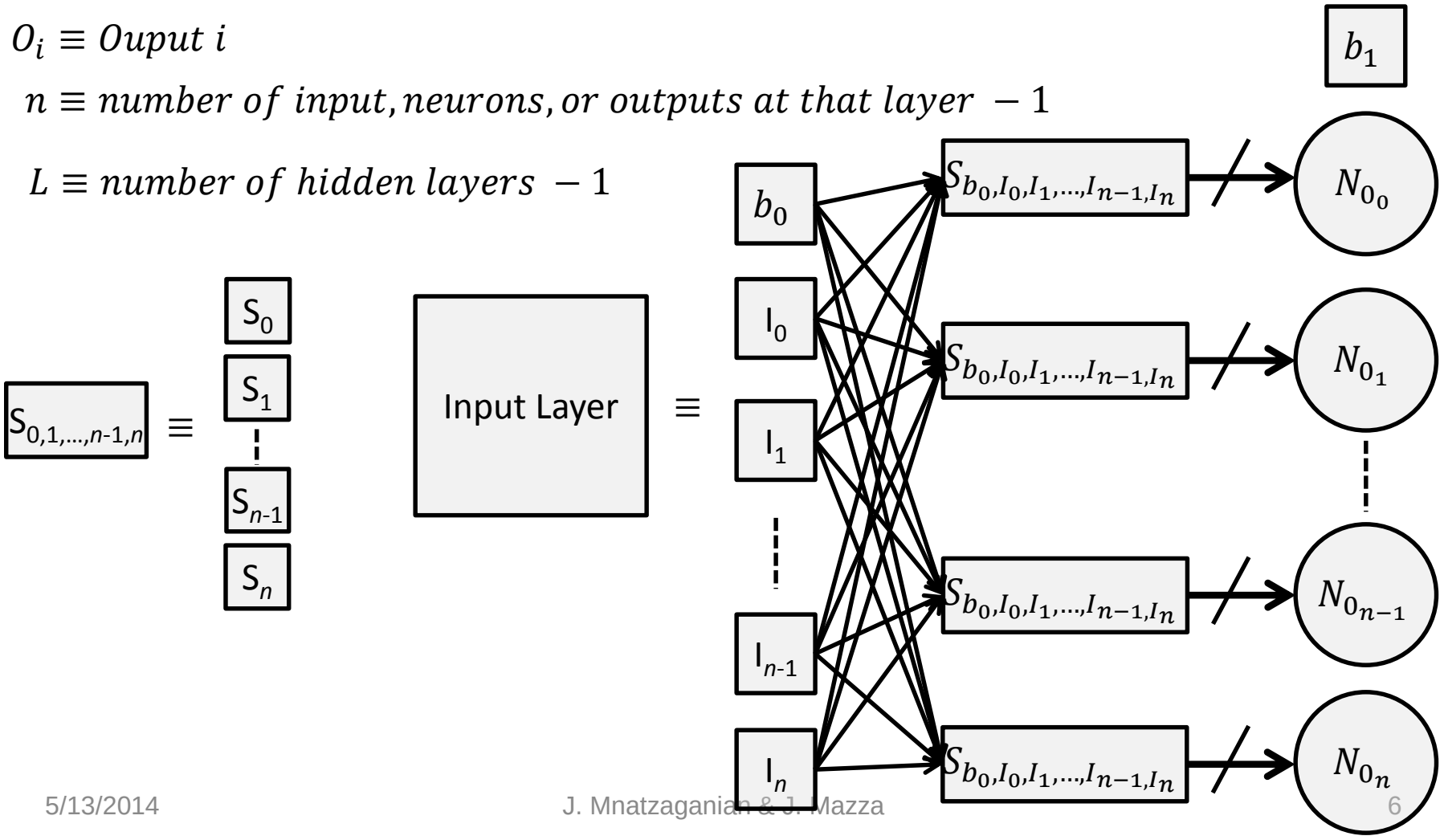
Multilayer Perceptron (1)

$b_l \equiv bias\ at\ layer\ l$ $I_i \equiv Input\ i$ $S_i \equiv Synapse\ i$ $N_{l_i} \equiv Neuron\ i\ at\ layer\ l$

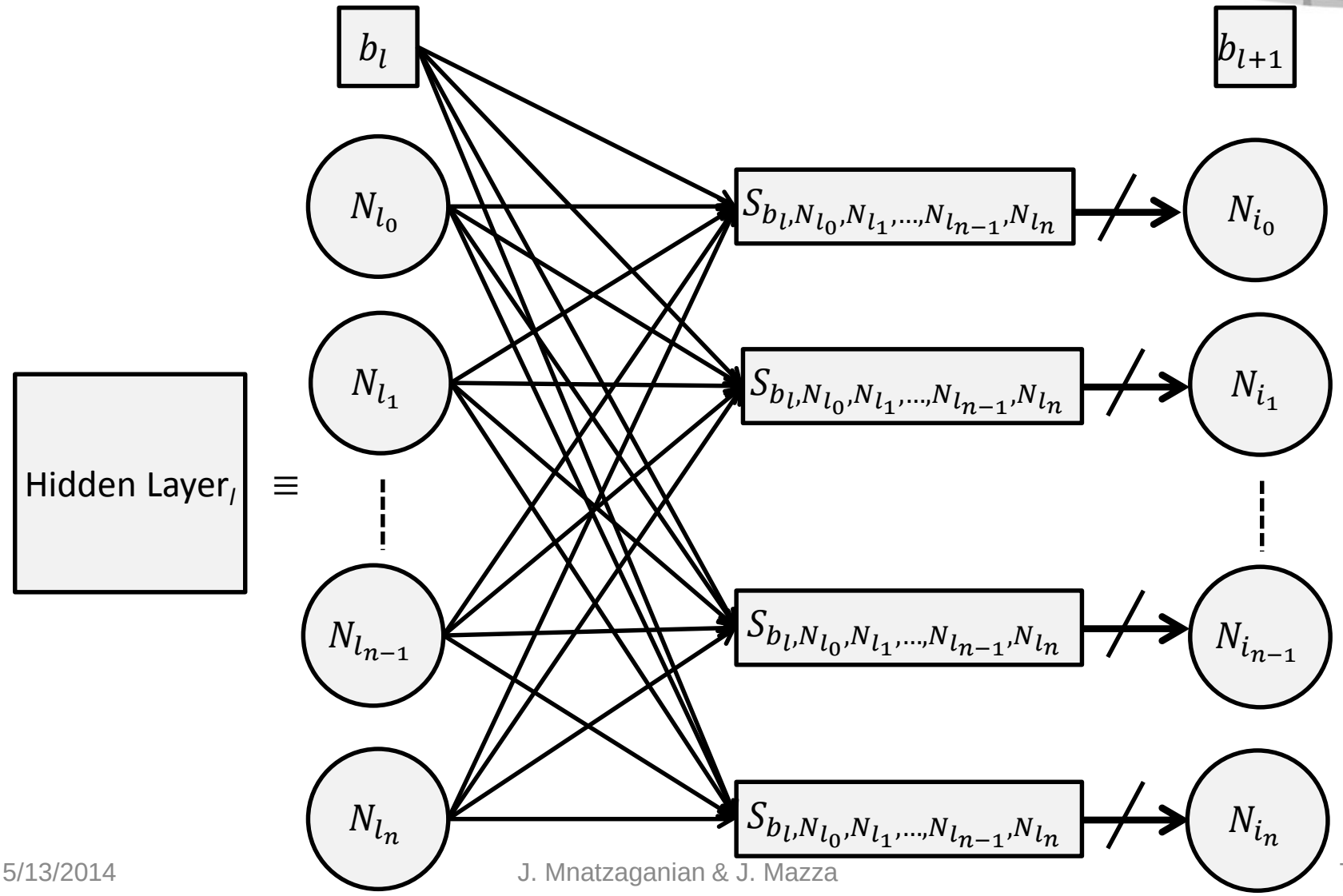
$O_i \equiv Ouput\ i$

$n \equiv number\ of\ input, neurons, or\ outputs\ at\ that\ layer\ -1$

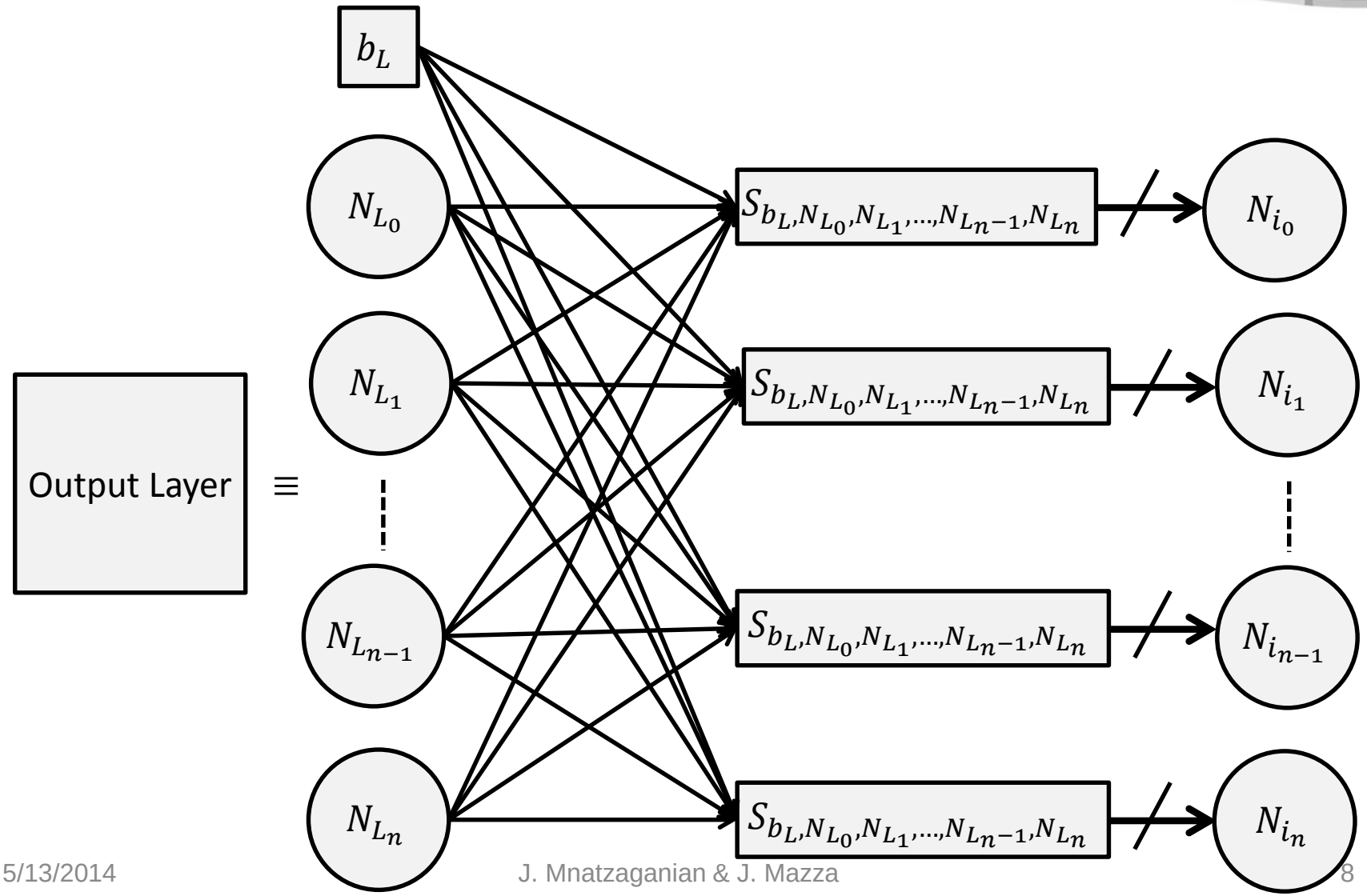
$L \equiv number\ of\ hidden\ layers\ -1$



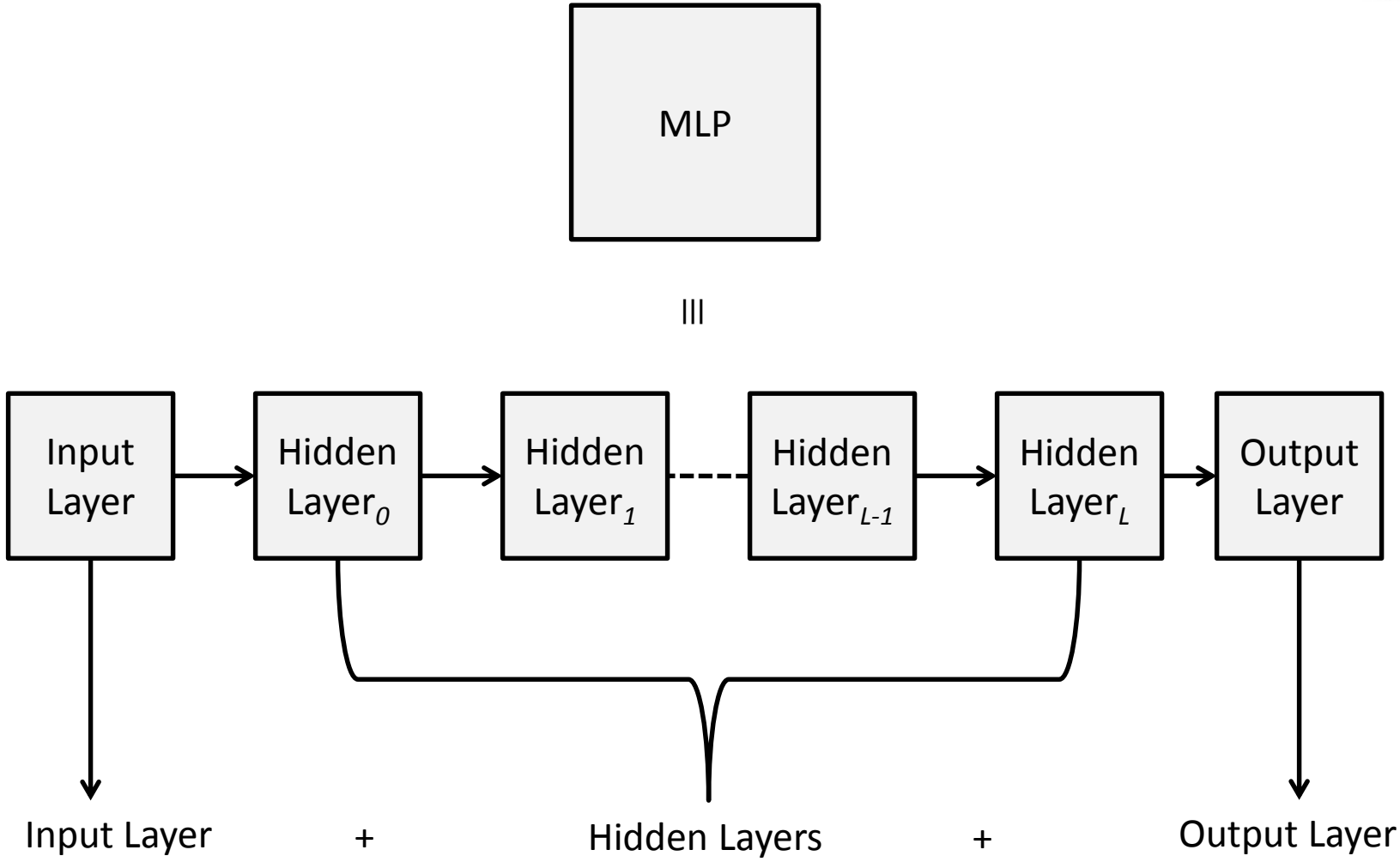
Multilayer Perceptron (2)



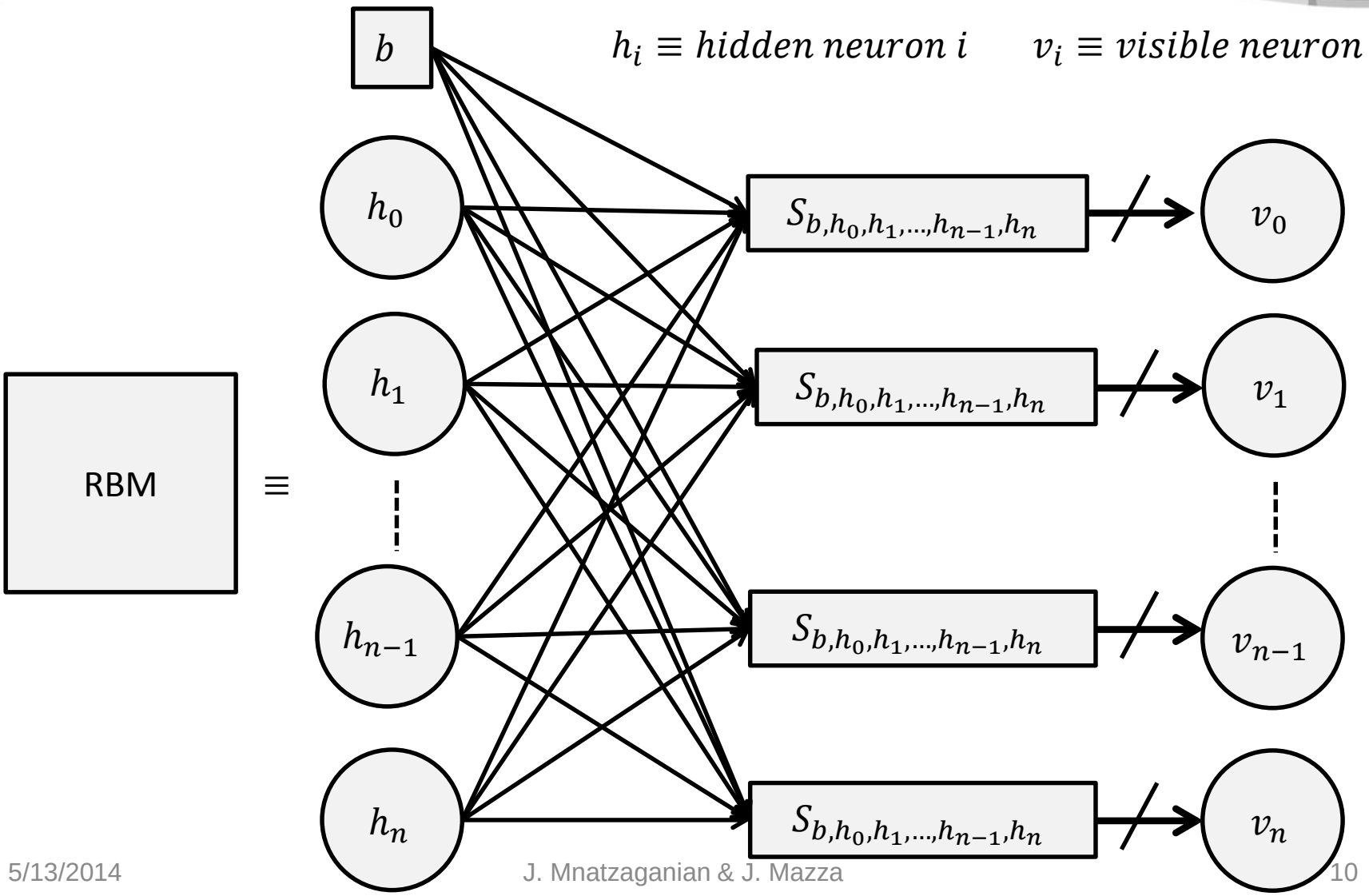
Multilayer Perceptron (3)



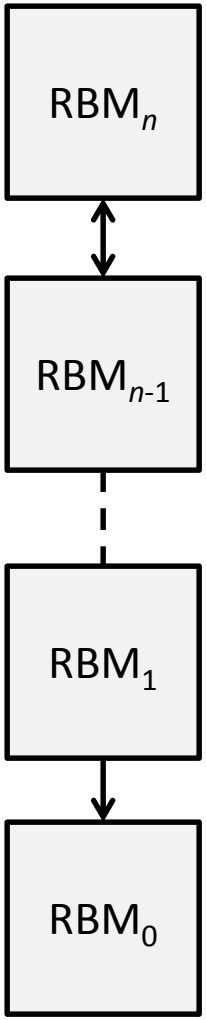
Multilayer Perceptron (4)



Restricted Boltzmann Machine



Deep Belief Network



RBM$_1$ – RBM$_n$ train on either the mean activations or the samples:

$$Mean\ Activations(i) = p(h_{i+1} = 1|h_i)$$

$$Samples(i) = p(h_{i+1}|h_i)$$

Trains on input

- DBN Training
  - Unsupervised, pre-training with stacked RBMs
  - Fine-tune, supervised, training
    - Initialize weights that RBMs learned to weights of deep MLP

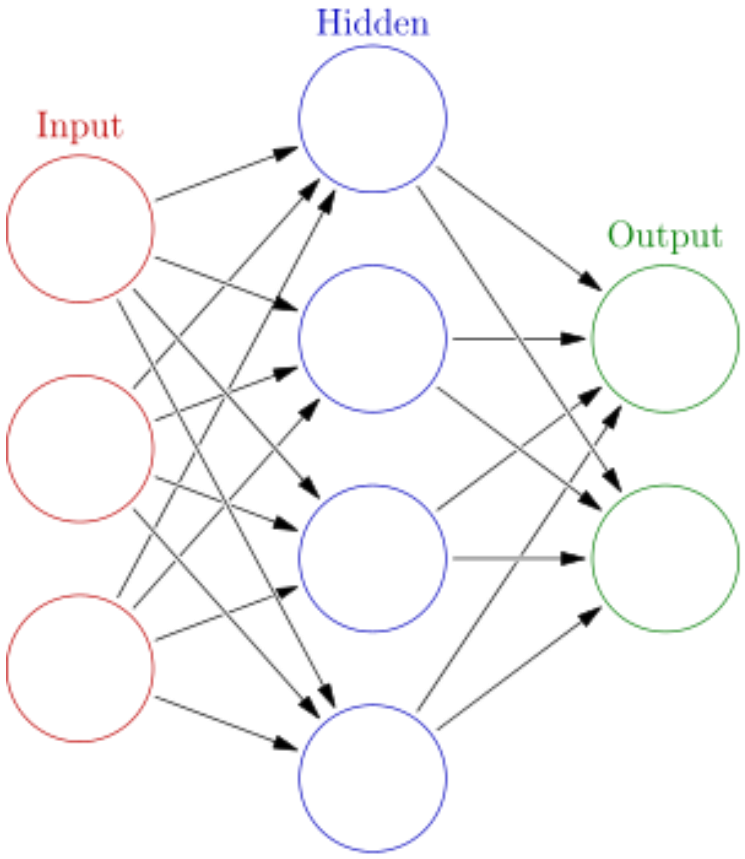
Types of Parallelism

Outline

- Data Parallelism
- Functional Parallelism

Data Parallelism

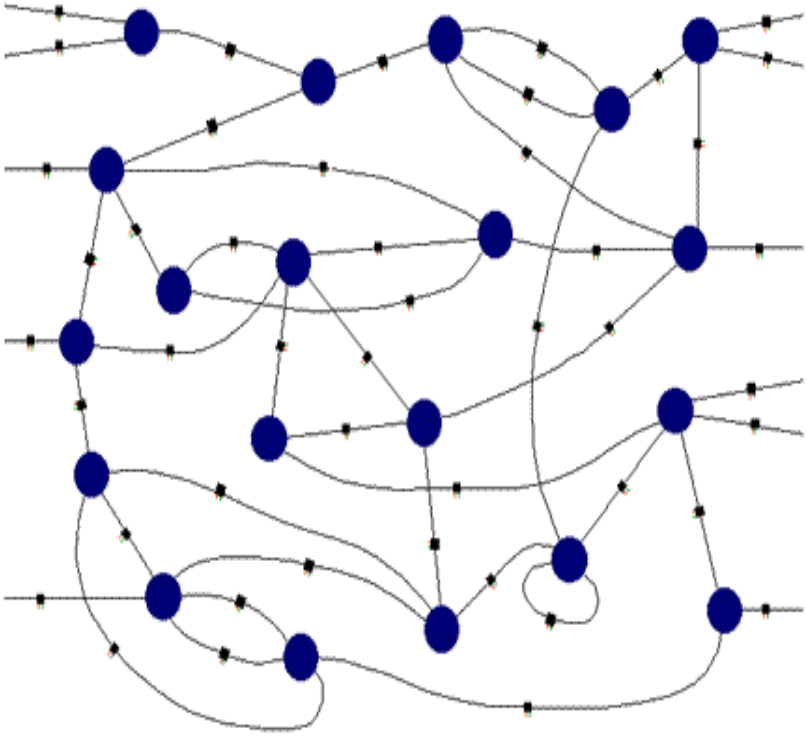
- First type of parallelism present lies in update synapse step
- Data parallelism typically scales with problem size
- Each synapse not dependent on others
- Can all update in parallel
- Easy to load balance
- Computation is the same among each neuron



http://en.wikipedia.org/wiki/Artificial_neural_network

Functional Parallelism

- Functional parallelism in multiply accumulate stages of the network
- Refers to sets of tasks that can be done in parallel
- Not perfectly load balanced
- In networks with cyclic connections, number of synapses summed in each stage may not be constant
- Could be used to pipeline synapse updating or neuron multiply accumulate functions



http://www.alanturing.net/

Task Partitioning and Grain Size

Outline

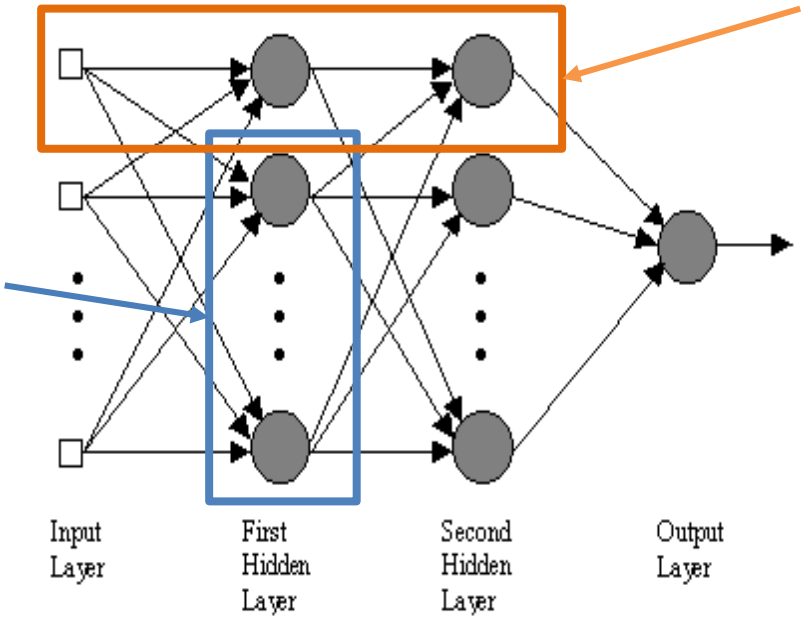
- Task Partitioning
- Task Grain Size

Task Partitioning

- Size of network is known at time of execution
- Network can be statically partitioned into different sections
- Execution time can be reasonably estimated ahead of time
- Static assignment determines number of synapses and nodes per task
- Dynamic partitioning not necessary and could incur additional overheads without justification

Task Grain Size

- Many nodes could be grouped together
- Increases c-to-c ratio if task size too small
- Load balance could become more uneven if size too large



http://www.neurosolutions.com/

- Layers could be pipelined to increase performance
- Exploits inherent function parallelism
- Grain size determines size of pipeline stage and number of nodes included in each stage

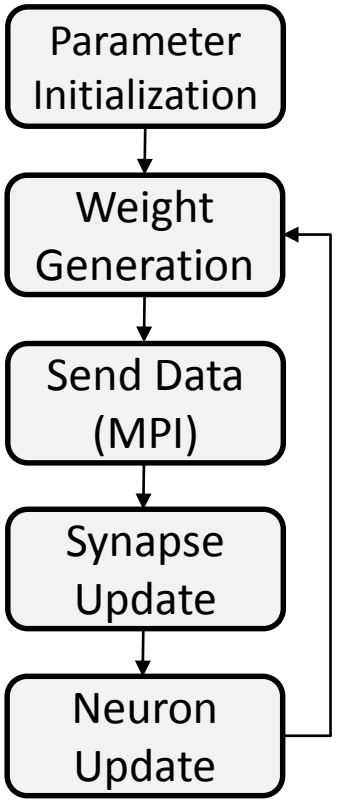
Cluster vs. GPU Implementations

Outline

- Cluster Implementation
- GPU Utilization
- Cluster vs. GPU
- Cluster Speedup
- GPU Speedup

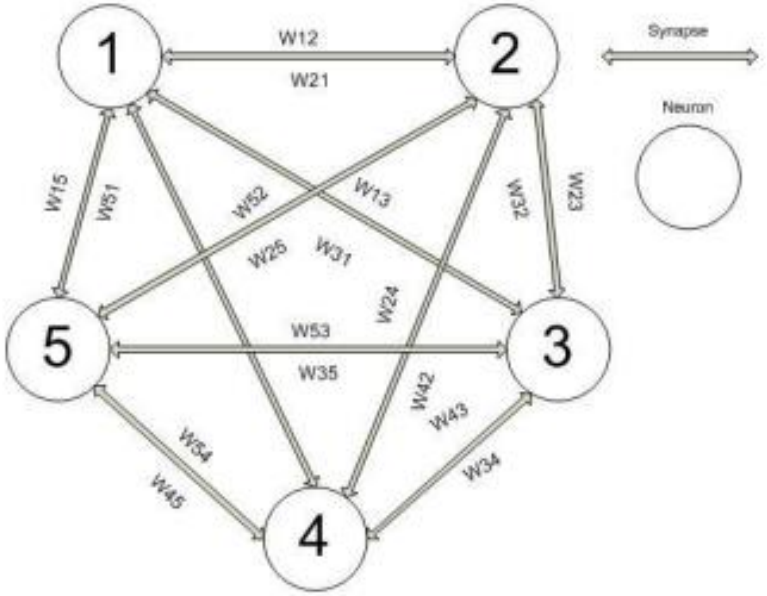
Cluster Implementation

- Neural network can be implemented on clusters using OpenMP and MPI programming models
- Master-slave model is used
- Master takes care of seed generation, updating the weights, and determining execution
- Slaves perform highly parallel computational steps



GPU Utilization

- Neural Networks have also been implemented using Cuda for Nvidia GPUs
- The neuron update step is essentially a vector dot product with a column of the weight matrix
- Each step is extremely data parallel
- Vector operations are well suited to GPU architectures



$$\begin{pmatrix} W12 & W21 & W31 & W41 & W51 \\ W13 & W23 & W32 & W42 & W52 \\ W14 & W24 & W34 & W43 & W53 \\ W15 & W25 & W45 & W45 & W54 \end{pmatrix}$$

Casas [3]

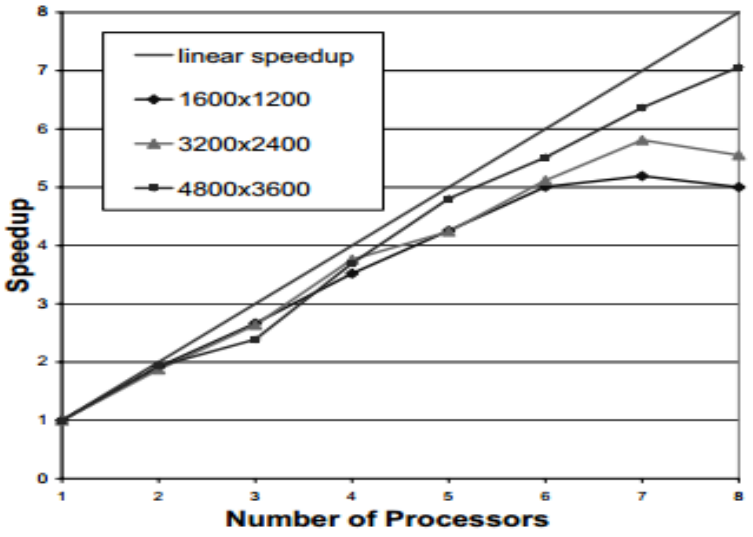
Cluster vs. GPU

- Cluster Advantages
  - Master-Slave architecture works well for learning section of algorithm
  - Slaves can compute synapse values and neuron function in parallel
- Cluster Disadvantages
  - May not scale well depending on network
  - Extreme contention at master node if grain size is too small

- GPU Advantages
  - Vector operations are extremely well suited for GPU's
  - Scales well with increasing problem size
- GPU Disadvantages
  - Some portions of algorithm are very inefficient on GPU architecture
  - Power consumption of GPU is very high compared to CPU approach

Cluster Speedup

- Cellular Neural Network speedup
- 16 nodes w/ Pentium 4 @3.06 GHz



Gonzalez [4]

- ANN used in time forecasting applications
- Intel Xeon E5504 @2.0 GHz

Standard		Cores 2	3	4	5	6
MPI	acceleration	24%	46%	52%	**60%**	55%
	efficiency	**65%**	62%	52%	52%	38%
	M	44.7%	54.0%	51.8%	**56%**	46.7%
	ranking	10°	2°	3°	1°	8°
OpenMP	acceleration	30%	44%	48%	**56%**	54%
	efficiency	**73%**	58%	49%	45%	36%
	M	51.5%	51.2%	48.6%	**50.5%**	44.8%
	ranking	4°	5°	7°	6°	9°

Weishaupl [5]

GPU Speedup

Input size [a]	Network properties				Execution time (in seconds)			Speedup	
	Input area	Neurons	Trainable parameters	Connections	$CPU_{triv.}$	$CPU_{opt.}$	GPU	$\frac{CPU_{triv.}}{GPU}$	$\frac{CPU_{opt.}}{GPU}$
32×32	1,024	8,010	51,046	331,114	3.6567	2.0317	0.9345	3.9132	2.1742
40×40	1,600	13,770	60,646	956,842	7.2532	3.6693	1.1103	6.5326	3.3048
48×48	2,304	21,130	79,846	2,047,210	13.0610	5.9001	1.3388	9.7554	4.4068
56×56	3,136	30,090	108,646	3,602,218	21.1530	8.7290	1.6000	13.2206	5.4556
64×64	4,096	40,650	147,046	5,621,866	32.5300	12.1500	1.9901	16.3463	6.1054
72×72	5,184	52,810	195,046	8,106,154	44.5400	16.5210	2.3532	18.9274	7.0207
80×80	6,400	66,570	252,646	11,055,082	59.4970	21.3960	2.9072	20.4650	7.3595
88×88	7,744	81,930	319,846	14,468,650	76.3710	26.5900	3.5705	21.3894	7.4471
96×96	9,216	98,890	396,646	18,346,858	95.1780	33.1480	4.1635	22.8601	7.9616
104×104	10,816	117,450	483,046	22,689,706	116.4700	40.8270	4.9055	23.7427	8.3227

	Core i7 860	GeForce GTX 275
Processsor core clock	2800 MHz	633 MHz
ALU clock	5600 MHz	1404 MHz
Memory size	4096 MB	896 MB
Bandwidth core ↔ memory	21.3 GB/s	127.0 GB/s
Number of processor cores	4	30
Local cache per core	64 KB L1 + 512 KB L2 + 2048 KB L3	16 KB (shared) + 8 KB (texture) + 8 KB (constant)
SP FLOPS / core and clock cycle	4 MUL or ADD	8 MUL and 8 MAD
Total SP FLOPS peak performance	89.6 GFLOPS/s	1010.8 GFLOPS/s
Termal Design Power	95 Watt	216 Watt

Strigl [6]

Network Considerations

Outline

- Mapping Neural Networks
- Mapping Scheme Comparison
- Mapping Conclusions

Mapping Neural Networks

- Neural Networks can be mapped to any number of networks but hyper cubes and meshes are typically used
- Forward learning stages include large amounts of communication between nodes
- For cluster implementations large amounts of contention exist at the master node

Mapping Scheme Comparisons

Mapping Scheme	Architecture	Communications	Multiplications	Additions
Kung	Systolic Ring	O(N)	O(N)	O(N)
Shams	SIMD 2D Mesh	O(N)	O(N)	O(N)
Lin	SIMD 2D Mesh	O(N)	O(1)	O(N)
Shams	SIMD 2D Mesh	O(N)	O(N)	O(N)
Kim	SIMD MPAA	O(1)	O(N)	O(N)
Malluhi	SIMD Hypercube	O(logN)	O(1)	O(logN)
Ayoubi	SIMD 2D Mesh	O(logN)	O(1)	O(logN)

Ayoubi [7]

Final Remarks

Outline

- Parameter Optimizations
- Applications
- Conclusion
- References

Parameter Optimizations

- DBN consists of 10 + *L* parameters
- Model for ideal parameters is non-linear
- Perform “search” in parallel
  - Master utilizes genetic algorithm (or similar)
  - Dynamic queue
  - Slaves compute specific parameter set
    - Sequential operation OR
    - Parallelizations within each slave’s parameter set

Applications

- Computer Vision
  - Optical character recognition
  - Image recognition
- Natural Language Processing
  - Document recognition
  - Sentiment analysis
- Automatic Speech Recognition
  - Call routing
  - Speech to text

Conclusion

- ANNs provide a powerful modeling framework
- ANNs are *highly* parallelizable
  - Both data and functional parallelism
- Cluster and GPU can be targeted for implementation
- ANNs are best suited for solving non-traditional problems

References

[1] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. “Theano: A CPU and GPU Math Expression Compiler”. Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX

[2] G.E. Hinton and R.R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science, 28 July 2006, Vol. 313. no. 5786, pp. 504 - 507

[3] Casas, C.A., "Parallelization of artificial neural network training algorithms: A financial forecasting application," *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on* , vol., no., pp.1,6, 29-30 March 2012

[4] Gonzalez, B.P.; Donate, J.P.; Cortez, P.; Sanchez, G.G.; de Miguel, A.S., "Parallelization of an evolving Artificial Neural Networks system to Forecast Time Series using OPENMP and MPI," *Evolving and Adaptive Intelligent Systems (EAIS), 2012 IEEE Conference on* , vol., no., pp.186,191, 17-18 May 2012

[5] Weishaupl, T.; Schikuta, E., "Parallelization of cellular neural networks for image processing on cluster architectures,"*Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on* , vol., no., pp.191,196, 6-9 Oct. 2003

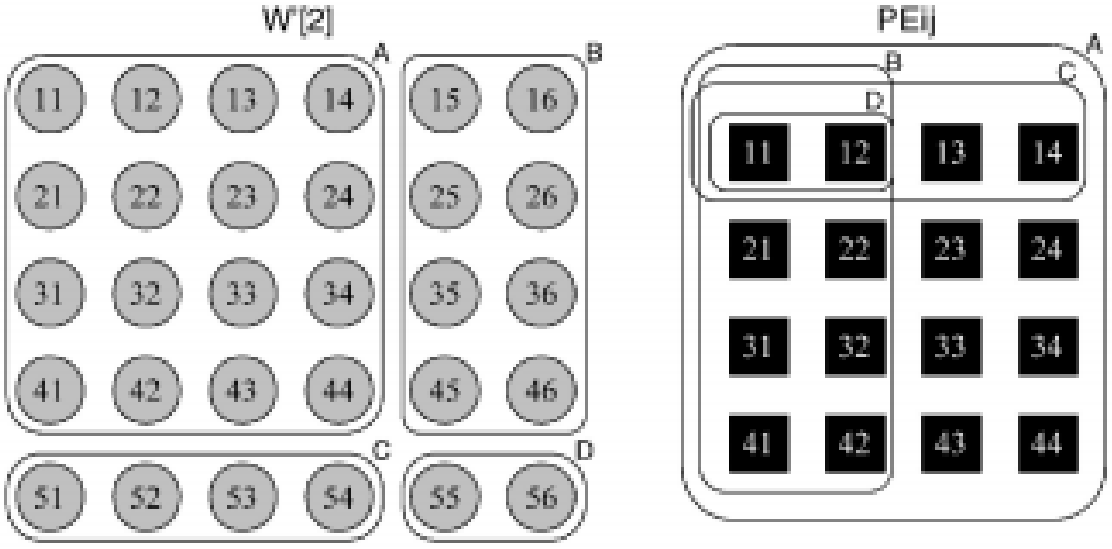
[6] Strigl, D.; Kofler, K.; Podlipnig, S., "Performance and Scalability of GPU-Based Convolutional Neural Networks," *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on* , vol., no., pp.317,324, 17-19 Feb. 2010

[7] Ayoubi, R.A.; Bayoumi, M.A., "Efficient mapping algorithm of multilayer neural network on torus architecture,"*Parallel and Distributed Systems, IEEE Transactions on* , vol.14, no.9, pp.932,943, Sept. 2003

Questions?

Backup

Virtual Layers in Mapping



Ayoubi [7]

- Example of Mapping 5x6 weight matrix onto 4x4 processor array
- Needs 4 virtual layers (A,B,C,D)