

# RESTRICTED BOLTZMANN MACHINES AND DEEP BELIEF NETWORKS ON MULTI-CORE PROCESSORS

Noel Lopes   **Bernardete Ribeiro**   João Gonçalves

University of Coimbra  
Polytechnic Institute of Guarda

June 11, 2012  
**WCCI-IJCNN**

## DEEP BELIEF NETWORKS (DBNs)

“Deep belief nets are probabilistic generative models that are composed of multiple layers of stochastic latent variables. The latent variables typically have binary values and are often called hidden units or feature detectors. [...] The lower layers receive top-down, directed connections from the layers above. The states of the units in the lowest layer represent a data vector.”

Geoffrey E. Hinton [Hinton et al., 2006]

## OUTLINE

- Motivation
- Deep Belief Networks
- Restricted Boltzmann Machines
- GPU implementation
- Results on MNIST Handwritten Database
- Conclusions and Future Work

## MOTIVATION

- The robustness and efficiency by which humans can recognize objects has ever been an intriguing challenge in computational intelligence.

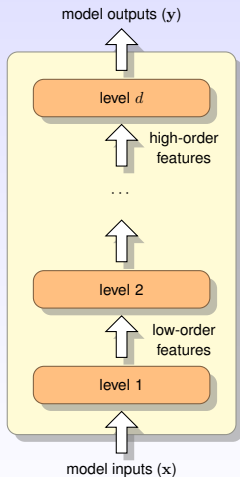
## MOTIVATION

- The robustness and efficiency by which humans can recognize objects has ever been an intriguing challenge in computational intelligence.
- Theoretical results suggest that deep architectures are fundamental to learn complex functions that can represent high-level abstractions (e.g. vision, language) [Bengio, 2009]

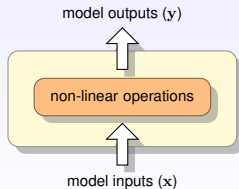
## MOTIVATION

- The robustness and efficiency by which humans can recognize objects has ever been an intriguing challenge in computational intelligence.
- Theoretical results suggest that deep architectures are fundamental to learn complex functions that can represent high-level abstractions (e.g. vision, language) [Bengio, 2009]
- Empirical results show their successful application: classification, regression, dimensionality reduction, object recognition, information retrieval, robotics, and collaborative filtering etc. [Larochelle et al., 2007, Swersky et al., 2010].

# DEEP VERSUS SHALLOW ARCHITECTURES



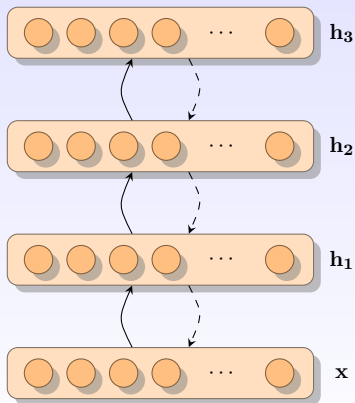
**deep architecture**



**shallow architecture**

# DEEP BELIEF NETWORKS

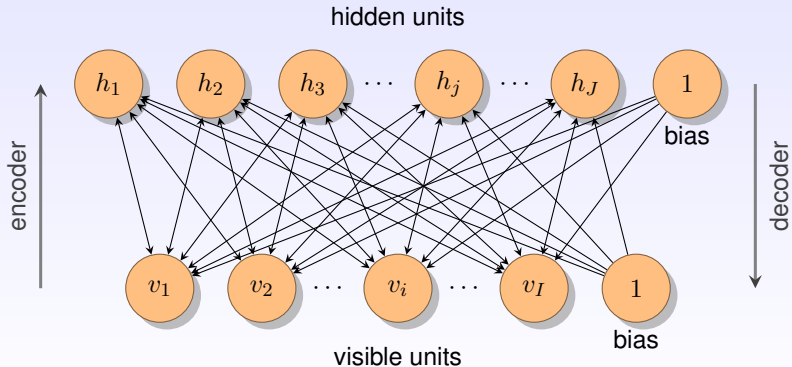
DBNs are composed of several Restricted Boltzmann Machines (RBMs) stacked on top of each other.





# RESTRICTED BOLTZMANN MACHINES

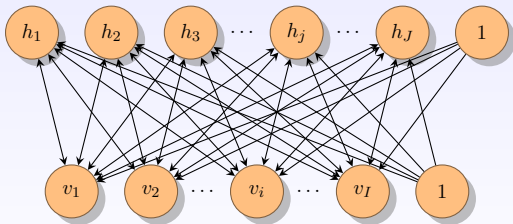
An RBM is an energy-based generative model that consists of a layer of binary visible units,  $\mathbf{v}$ , and a layer of binary hidden units,  $\mathbf{h}$ .



# RESTRICTED BOLTZMANN MACHINES

Given an observed state, the energy of the joint configuration of the visible and hidden units ( $\mathbf{v}, \mathbf{h}$ ) is given by (1):

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{i=1}^I a_i v_i - \sum_{j=1}^J b_j h_j - \sum_{j=1}^J \sum_{i=1}^I W_{ji} v_i h_j, \quad (1)$$



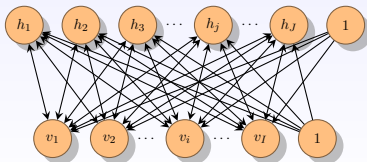
# RESTRICTED BOLTZMANN MACHINES

The RBM defines a joint probability over  $(\mathbf{v}, \mathbf{h})$ :

$$p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v}, \mathbf{h})}}{Z}, \quad (2)$$

where  $Z$  is the partition function, obtained by summing the energy of all possible  $(\mathbf{v}, \mathbf{h})$  configurations:

$$Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}. \quad (3)$$



## RESTRICTED BOLTZMANN MACHINES

Given a random input configuration  $\mathbf{v}$ , the state of the hidden unit  $j$  is set to 1 with probability:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_{i=1}^I v_i W_{ji}) , \quad (4)$$

Similarly, given a random hidden vector,  $\mathbf{h}$ , the state of the visible unit  $i$  can be set to 1 with probability:

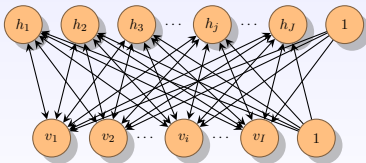
$$p(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_{j=1}^J h_j W_{ji}) . \quad (5)$$

## TRAINING AN RBM

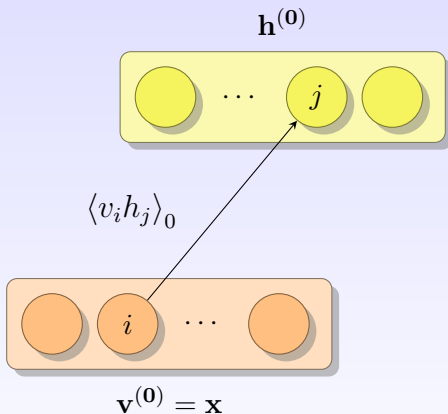
The following learning rule performs stochastic steepest ascent in the log probability of the training data:

$$\frac{\partial \log p(\mathbf{v}, \mathbf{h})}{\partial W_{ji}} = \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty \quad (6)$$

where  $\langle \cdot \rangle_0$  denotes the expectations for the data distribution ( $p_0$ ) and  $\langle \cdot \rangle_\infty$  denotes the expectations under the model distribution

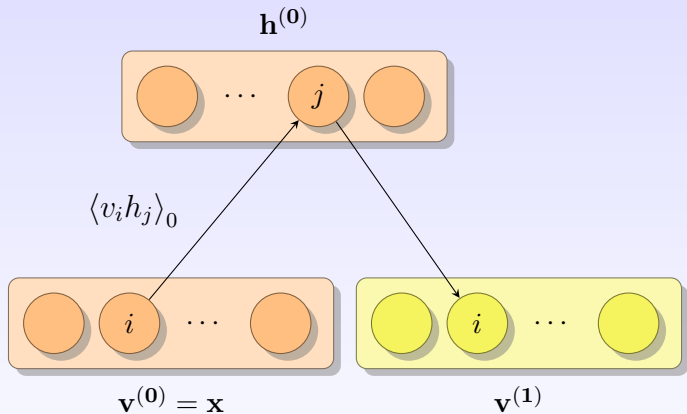


# GIBBS SAMPLING



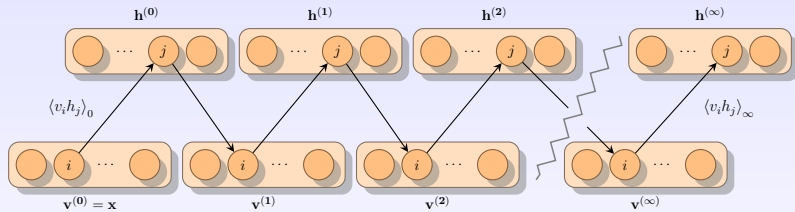
$$p(h_j = 1 | \mathbf{v}) = \sigma(b_j + \sum_{i=1}^I v_i W_{ji})$$

# ALTERNATING GIBBS SAMPLING



$$p(v_i = 1 | \mathbf{h}) = \sigma(a_i + \sum_{j=1}^J h_j W_{ji})$$

# ALTERNATING GIBBS SAMPLING





## CONTRASTIVE DIVERGENCE (CD- $k$ )

- Hinton proposed the Contrastive Divergence (CD) algorithm
- CD- $k$  replaces  $\langle \cdot \rangle_{\infty}$  by  $\langle \cdot \rangle_k$  for small values of  $k$ .

## CONTRASTIVE DIVERGENCE (CD- $k$ )

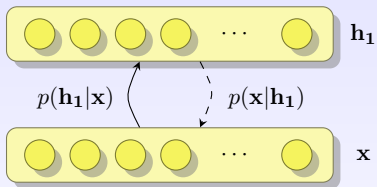
- $\mathbf{v}^{(0)} \leftarrow \mathbf{x}$
- Compute the binary (features) states of the hidden units,  $\mathbf{h}^{(0)}$ , using  $\mathbf{v}^{(0)}$
- **for**  $n \leftarrow 1$  **to**  $k$ 
  - Compute the “reconstruction” states for the visible units,  $\mathbf{v}^{(n)}$ , using  $\mathbf{h}^{(n-1)}$
  - Compute the “reconstruction” states for the hidden units,  $\mathbf{h}^{(n)}$ , using  $\mathbf{v}^{(n)}$
- **end for**
- Update the weights and biases, according to:

$$\Delta W_{ji} = \gamma(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k) \quad (7)$$

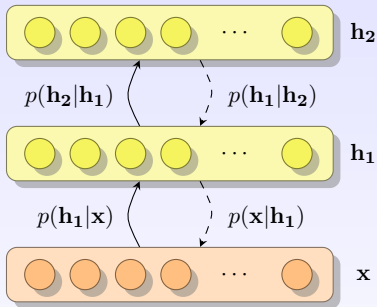
$$\Delta b_j = \gamma(\langle h_j \rangle_0 - \langle h_j \rangle_k) \quad (8)$$

$$\Delta a_i = \gamma(\langle v_i \rangle_0 - \langle v_i \rangle_k) \quad (9)$$

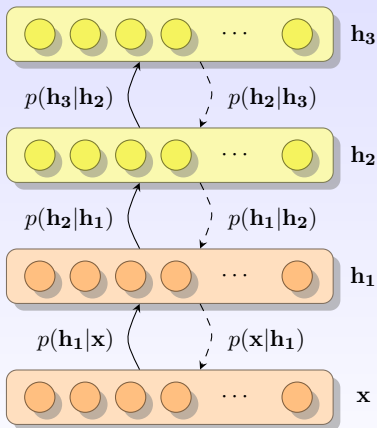
# DEEP BELIEF NETWORKS (DBN)



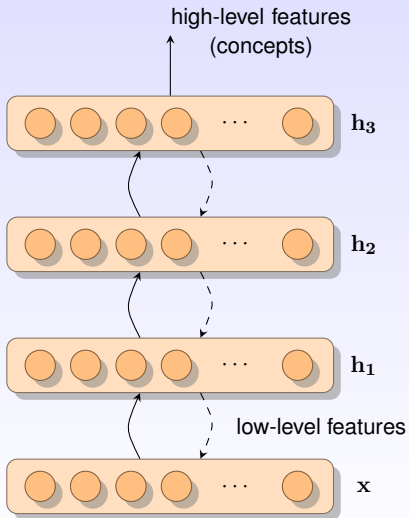
# DEEP BELIEF NETWORKS (DBN)



# DEEP BELIEF NETWORKS (DBN)



# DEEP BELIEF NETWORKS (DBN)



## GPU IMPLEMENTATION

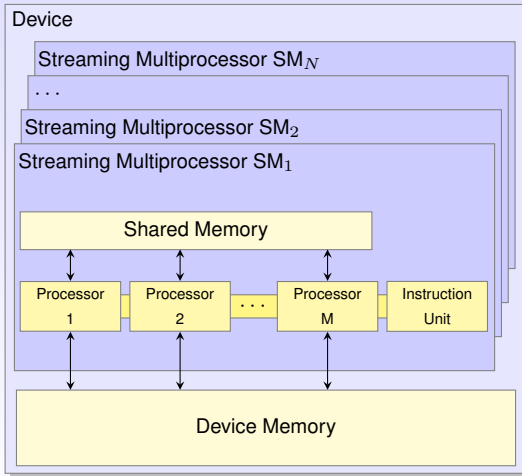
- Training a DBN is a computationally expensive task that involves training several RBMs and may require a considerable amount of time.

# GPU IMPLEMENTATION

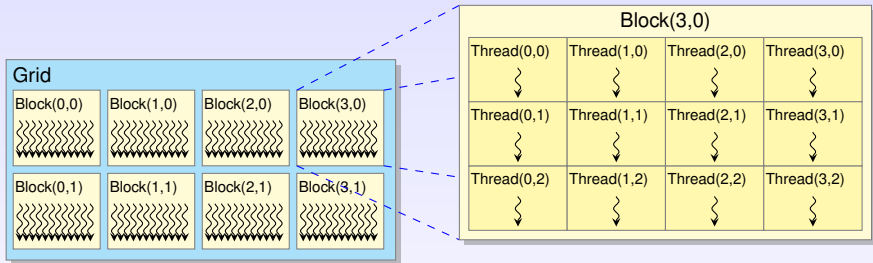
- Training a DBN is a computationally expensive task that involves training several RBMs and may require a considerable amount of time.
- Solution?
  - **GPU Parallel implementation**



# CUDA – DEVICE ARCHITECTURE

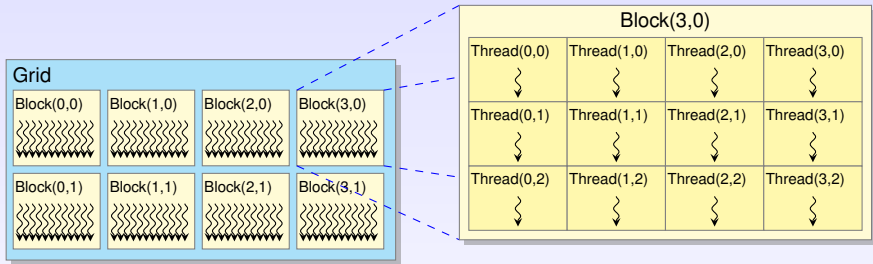


# CUDA – LAUNCHING A KERNEL GRID



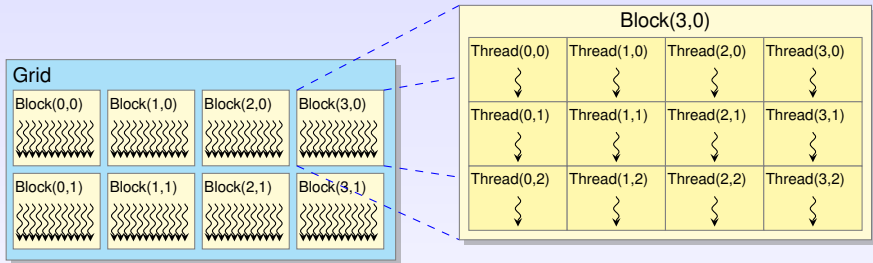
- Threads within a block can share information.

# CUDA – LAUNCHING A KERNEL GRID



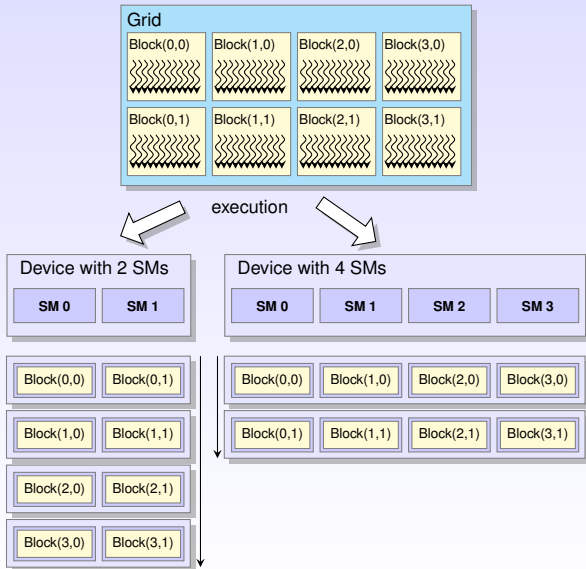
- Threads within a block can share information.
- However blocks are required to run independently.

# CUDA – LAUNCHING A KERNEL GRID

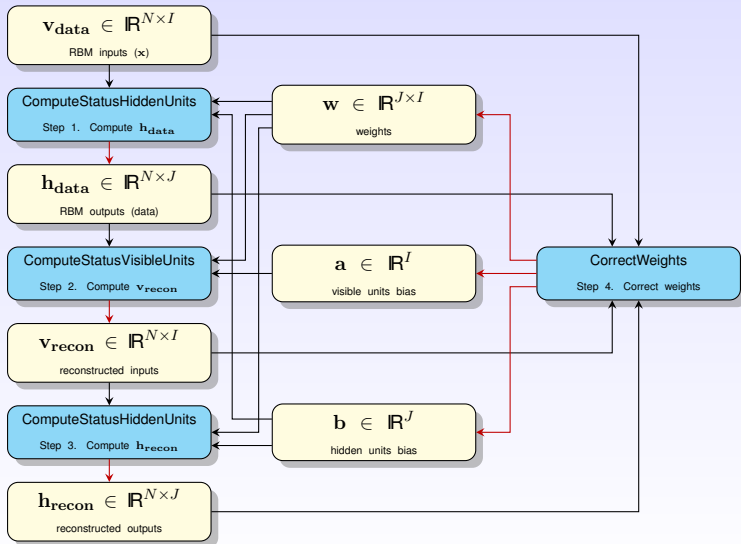


- Threads within a block can share information.
- However blocks are required to run independently.
- To address scalability the tasks should be partitioned.

# CUDA – SCALABILITY

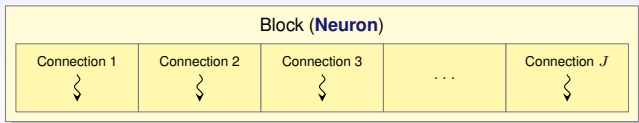


# KERNELS



# COMPUTESTATUSHIDDENUNITS AND COMPUTESTATUSVISIBLEUNITS KERNELS

- Each thread represents a connection
  - Multiplies the clamped input by the weight
  - Stores the weight in the shared memory
- Each block represents a neuron
  - Uses fast shared memory to sum up the values computed by each thread



# STORING THE CONNECTION WEIGHTS

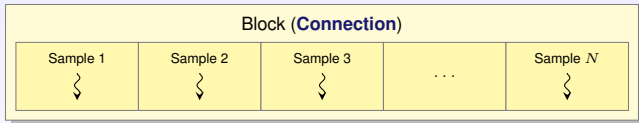
- ComputeStatusHiddenUnits - Coalesced access
- ComputeStatusVisibleUnits - Uncoalesced access

$w_{11}$	$w_{12}$	$w_{13}$	$w_{14}$	$w_{15}$	...	$w_{1I}$
$w_{21}$	$w_{22}$	$w_{23}$	$w_{24}$	$w_{25}$	...	$w_{2I}$
$w_{31}$	$w_{32}$	$w_{33}$	$w_{34}$	$w_{35}$	...	$w_{3I}$
$w_{41}$	$w_{42}$	$w_{43}$	$w_{44}$	$w_{45}$	...	$w_{4I}$
$w_{51}$	$w_{52}$	$w_{53}$	$w_{54}$	$w_{55}$	...	$w_{5I}$
...	...	...	...	...	...	...
$w_{J1}$	$w_{J2}$	$w_{J3}$	$w_{J4}$	$w_{J5}$	...	$w_{JI}$

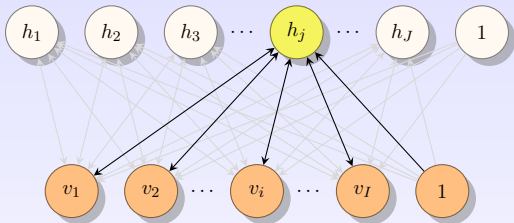


# CORRECTWEIGHTS KERNEL FIRST APPROACH

- Each thread gathers and sums up the values for one or more samples
- Each block corrects the weight of a connection



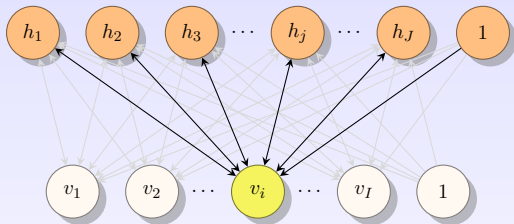
# PROBLEMS?



$$\Delta W_{ji} = \gamma(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k)$$

$$\Delta b_j = \gamma(\langle h_j \rangle_0 - \langle h_j \rangle_k)$$

## PROBLEMS?

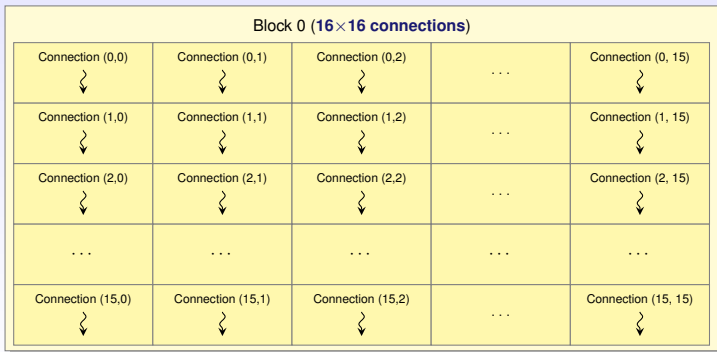


$$\Delta W_{ji} = \gamma(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_k)$$

$$\Delta a_i = \gamma(\langle v_i \rangle_0 - \langle v_i \rangle_k)$$

# CORRECTWEIGHTS KERNEL IMPROVED APPROACH

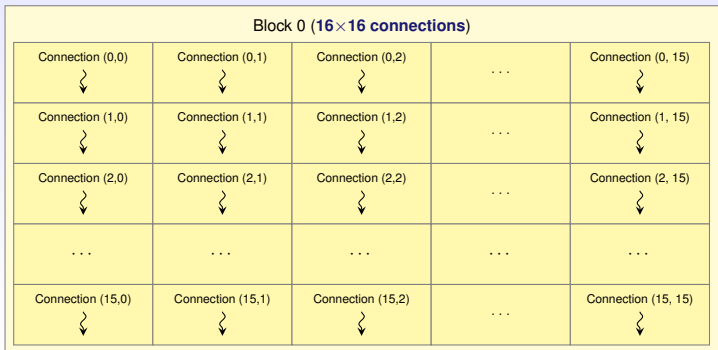
- Each block has  $16 \times 16$  threads.
- Each thread within a block must now process all the samples.



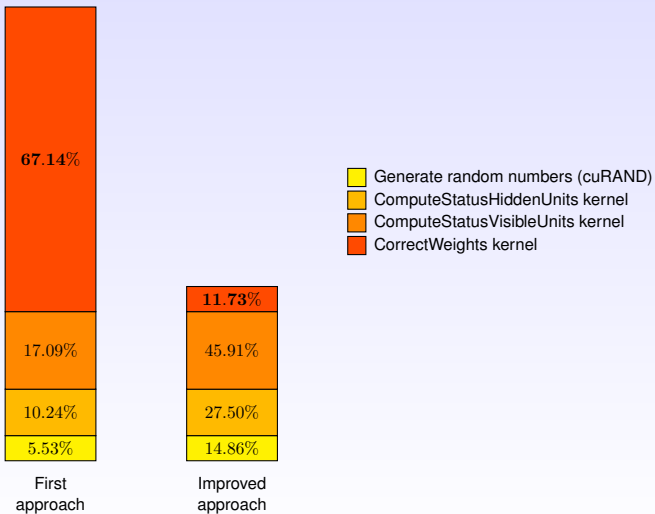
# CORRECTWEIGHTS KERNEL

## IMPROVED APPROACH

- But we can access  $v_i$  and  $h_j$  variables in a coalesced way and store them in shared memory for faster accesses.
- Although, this new approach has a smaller number of blocks, it performs much better than our first approach ( $\approx 15\times$  faster).



# TIME SPENT IN EACH TASK FIRST AND SECOND APPROACHES



# EXPERIMENTAL SETUP

- We tested our approach with the MNIST database
- Each sample has  $28 \times 28$  pixel image of a hand-written digit (**784 inputs**)
- Hardware
  - CPU: Intel dual-core i5-2410M (8GB Memory)
  - GPU: NVIDIA GeForce 460 GTX

## NVIDIA GeForce 460 GTX

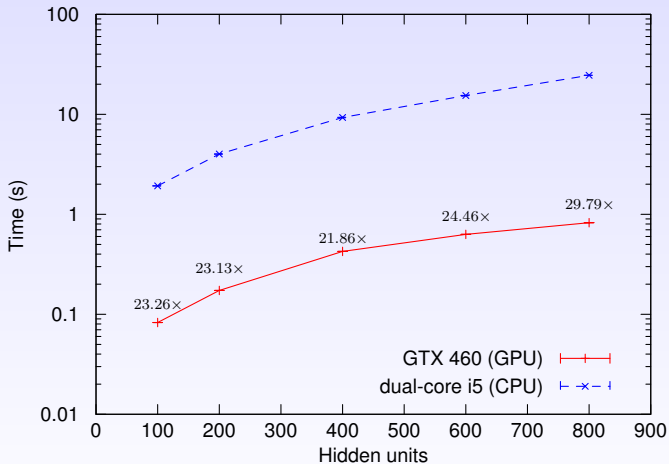
---

Number of Streaming Multiprocessors	7
Number of cores	336
Peak performance (GFLOPS)	940.8
Device Memory (GB)	1
Memory bandwidth (GB/sec)	112.5
Shading clock speed (GHz)	1.4

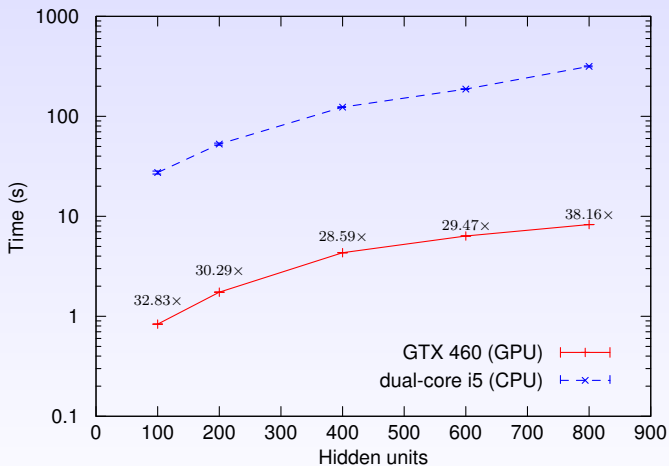
---



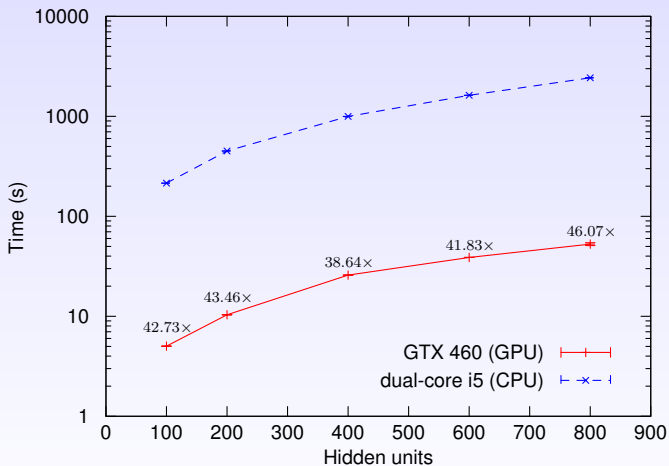
# RESULTS (1,000 SAMPLES)



# RESULTS (10,000 SAMPLES)




















































# RESULTS (60,000 SAMPLES)



















































# ADAPTIVE STEP SIZES

Adaptive Step Sizes

Training images							
Reconstruction after 10 epochs							
Reconstruction after 100 epochs							
Reconstruction after 250 epochs							
Reconstruction after 500 epochs							
Reconstruction after 750 epochs							
Reconstruction after 1000 epochs							

Fixed (optimized) learning rate 0.1

Training images							
Reconstruction after 10 epochs							
Reconstruction after 100 epochs							
Reconstruction after 250 epochs							
Reconstruction after 500 epochs							
Reconstruction after 750 epochs							
Reconstruction after 1000 epochs							

## CONCLUSIONS AND FUTURE WORK

- Creating a Deep Belief Network (DBN) model is a time consuming and computationally expensive task that involves training several Restricted Boltzmann Machines (RBMs) upholding considerable efforts.

## CONCLUSIONS AND FUTURE WORK

- Creating a Deep Belief Network (DBN) model is a time consuming and computationally expensive task that involves training several Restricted Boltzmann Machines (RBMs) upholding considerable efforts.
- Our work has demonstrated that by taking a non-trivial approach of GPU implementation we attained significant speedups.

## CONCLUSIONS AND FUTURE WORK

- Creating a Deep Belief Network (DBN) model is a time consuming and computationally expensive task that involves training several Restricted Boltzmann Machines (RBMs) upholding considerable efforts.
- Our work has demonstrated that by taking a non-trivial approach of GPU implementation we attained significant speedups.
- The adaptive step-size procedure for tuning the learning rate has been incorporated in the learning model with excellent results.

## CONCLUSIONS AND FUTURE WORK

- Creating a Deep Belief Network (DBN) model is a time consuming and computationally expensive task that involves training several Restricted Boltzmann Machines (RBMs) upholding considerable efforts.
- Our work has demonstrated that by taking a non-trivial approach of GPU implementation we attained significant speedups.
- The adaptive step-size procedure for tuning the learning rate has been incorporated in the learning model with excellent results.
- Future work will test this approach with other databases in particular in real-world problems.



# RESTRICTED BOLTZMANN MACHINES AND DEEP BELIEF NETWORKS ON MULTI-CORE PROCESSORS

Noel Lopes and **Bernardete Ribeiro** and João Gonçalves

University of Coimbra  
Polytechnic Institute of Guarda

June 11, 2012  
**WCCI-IJCNN**



Bengio, Y. (2009).

Learning deep architectures for AI.

*Foundations and Trends in Machine Learning*, 2(1):1–127.



Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006).

A fast learning algorithm for deep belief nets.

*Neural Computation*, 18(7):1527–1554.



Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007).

An empirical evaluation of deep architectures on problems with many factors of variation.

*In Proceedings of the 24th international conference on Machine learning (ICML 2007)*, pages 473–480. ACM.



Swersky, K., Chen, B., Marlin, B., and de Freitas, N. (2010).

A tutorial on stochastic approximation algorithms for training restricted boltzmann machines and deep belief nets.

*In Information Theory and Applications Workshop*.