GPU Performance Prediction using Representation Learning

Aswin Raghavan, Mohamed Amer, Timothy Shields, David Zhang, Sek Chai FIRSTNAME.LASTNAME@SRI.COM
SRI International, 201 Washington Rd. Princeton, NJ08540

Abstract

GPU performance prediction is an important and complex problem. This is due to the high level of contention among thousands of parallel threads. This problem was mostly addressed using heuristics. We propose a representation learning approach to address this problem. We model any performance metric as a temporal function of the executed instructions with the intuition that the flow of instructions can be identified as distinct activities of the code. Our experiments show high accuracy and non-trivial predictive power of representation learning on public benchmarks.

1. Introduction

The performance of a computing system relies on the sustained operational throughput. Sustained operation is becoming harder to achieve as computation workloads become more complex. At the same time, with the end of Dennard scaling (Esmaeilzadeh & et. al., 2011), and the increasing abundance of Big Data, it is imperative to minimize wasted processor effort in order to achieve processor reliability and scalability (Wulf & McKee, 1995; Patterson, 2006; Kogge & et al., 2008).

The goal of this paper is to demonstrate the efficacy of machine learning to designing computing systems. We anticipate that classical problems such as branch predictions and cache management can be re-evaluated such that heuristically based approaches (Yeh & Patt, 1991; Fung et al., 2007; Li et al., 2015) can be replaced with a machine learning approach (Leng et al., 2015). It is well understood in the computer architecture community that processor behavior is highly complex and data dependent. Processor data is widely available in the form of benchmarks (Che et al., 2010), and algorithms are extensively compared using these benchmarks (Blem et al., 2011).

We choose a well-understood and well-defined problem of predicting GPU Cache Misses (Li et al., 2015; Chen et al., 2014) for this paper. General Purpose GPU (GPGPU) achieve high throughput execution via a high level of parallelism. Predicting GPU Cache Misses is complex due to the high level of contention among thousands of threads. Cache contention is a bottleneck for parallel execution when many threads are waiting for cache operation, causing the addition of more threads (or cores) to be detrimental. Predicting whether a cache miss is about to occur is useful for better cache management such as cache bypassing (Chen et al., 2014), pre-fetching (Lee et al., 2010), prioritized allocation (Li et al., 2015) etc. Further, cache misses indirectly cause increased energy and power usage (Leng et al., 2015) because of second order effects beyond memory latency. In principle, our approach is amenable to predict these higher order events (such as voltage scaling (Leng et al., 2015) and faults (S. Chai, 2014)) either directly (Tiwari et al., 1994) or via hierarchical modeling.

In this paper, we model processor and system dynamics, to predict processor activity. We propose a new model that can predict key processor events that limits processor throughput. We propose a new variant of the Conditional Restricted Boltzmann Machines (CRBMs) (Taylor et al., 2011) to directly address system performance and reliability. CRBMs efficiently model short-term temporal phenomen. We propose an extension to CRBMs to process time-series histograms of processor instructions for processor activity prediction. We extend them with a discriminative component at the output layer, and a Count distribution at the input layer. This new model enables us to process processor instructions and recognize various types of activities. Prior work used a perceptron to predict cache misses (Leng et al., 2013). Unlike their approach, our model accounts for time-series and count data.

Our approach assumes the availability of a simulator for CUDA (Bakhoda et al., 2009) that can generate a dataset for training our model. In principle, this approach can be used in real-time by incrementally augmenting the dataset. Note that the predictor is naturally agnostic to the hardware and architecture as it relies on execution traces. Multiple repeated executions can even lead to increased predictive power because more data is available for machine learning.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

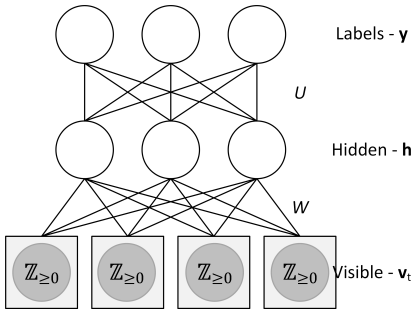

Figure 1. This figure illustrates the CountDCRBM model.

Our contributions:

- Prediction of processor events as temporally-extended activities in a stream of instructions.
- A novel temporal representation learning approach using DCRBM (Discriminative Conditional Restrictive Boltzmann Machines) to learn a representation of processor states.

2. Literature Review

Cache Miss Prediction: There is a large body of research on branch prediction to improve cache performance. Simple static solutions can achieve 80% correct prediction by analyzing control-flow and static heuristics (Ball & Larus., 1993). Dynamic solutions (Yeh & Patt, 1991) are more complicated as they are implemented with counters and tables to store branch history based on branch memory address. Other approaches that are data-driven use perceptrons (Jimnez & Lin, 2001) and feed-forward neural networks (Calder & et al., 1997).

Representation Learning: Restricted Boltzmann Machines (RBMs) form the building blocks in energy based deep networks (Hinton et al., 2006; Salakhutdinov & Hinton, 2006). Recently, temporal models based on deep networks have been proposed, capable of modeling a more temporally rich set of problems. These include Conditional RBMs (CRBMs) (Taylor et al., 2011) and Temporal RBMs (TRBMs) (Sutskever & Hinton, 2007). CRBMs have been used in both visual (Taylor et al., 2011) and audio (Mohamed & Hinton, 2009). In addition to efficiently modeling time-series data, RBMs were formulated be trained discriminativly for classification (Larochelle & Bengio, 2008), and model word-count vectors from a large set of documents (Salakhutdinov & Hinton, 2009).

3. Model

We discuss a sequence of models, gradually increasing in complexity, so that the different components of our model can be understood in isolation. We start with the basic CRBM model, then we extend to the discriminative DCRBM, and finally CountDCRBM. The input to our model (called visible units) is an instruction mix per time step, ie. the histogram of counts of instructions being excuted, obtained from the GPU simulator. The labels are any chosen performance metric also output by the simulator.

Conditional Restricted Boltzmann Machines: CRBMs (Taylor et al., 2011), are a natural extension of RBMs for modeling short term temporal dependencies. A CRBM is an RBM which takes into account history from the previous time instances $t-N,\ldots,t-1$ at time t. This is done by treating the previous time instances as additional inputs. Doing so does not complicate inference. $\mathbf{v}$ is a vector of visible nodes, $\mathbf{h}$ is a vector of hidden nodes, and $\mathbf{v}_{<t}$ is the visible vectors from the previous N time instances, which influences the current visible and hidden vectors. E_{C} is the energy function, and Z is the partition function. The parameters $\boldsymbol{\theta}$ to be learned are $\mathbf{a}$ and $\mathbf{b}$ the biases for $\mathbf{v}$ and $\mathbf{h}$ respectively and the weights W. A and B are matrices of concatenated vectors of previous time instances of $\mathbf{a}$ and $\mathbf{b}$. The CRBM is fully connected between layers, with no lateral connections. This architecture implies that $\mathbf{v}$ and $\mathbf{h}$ are factorial given one of the two vectors. This allows for the exact computation of $p_{\text{C}}(\mathbf{v}|\mathbf{h},\mathbf{v}_{<t})$ and $p_{\text{R}}(\mathbf{h}|\mathbf{v},\mathbf{v}_{<t})$. Some approximations have been made to facilitate efficient training and inference, more details are available in (Taylor et al., 2011). A CRBM defines a probability distribution p_{C} as a Gibbs distribution (1).

$$p_{\text{C}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = \exp[-E_{\text{C}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t})]/Z(\boldsymbol{\theta}). \quad (1)$$

The energy function $E_{\text{C}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t})$ in (2) is defined in a manner similar to that of the RBM.

$$\begin{aligned}
&E_{\text{C-Real}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = -\textstyle\sum_i (c_i - v_{i,t})^2/2 - \sum_j d_j h_{j,t} \\
&- \textstyle\sum_{i,j} v_{i,t} w_{i,j} h_{j,t}, \\
&E_{\text{C-Binary}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = -\textstyle\sum_i c_i v_{i,t} - \sum_j d_j h_{j,t} \\
&- \textstyle\sum_{i,j} v_{i,t} w_{i,j} h_{j,t}, \\
&E_{\text{C-Count}}(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = -\textstyle\sum_i (c_i v_{i,t} - \log(v_{i,t}!)) \\
&- \textstyle\sum_j d_j h_{j,t} - \sum_{i,j} v_{i,t} w_{i,j} h_{j,t},
\end{aligned} \quad (2)$$

The probability distributions for the visible nodes are defined in (3),

$$\begin{aligned}
&p_{\text{C-Real}}(v_{i,t}|\mathbf{h}_t,\mathbf{v}_{<t}) = \mathcal{N}(c_i + \textstyle\sum_j h_{j,t} w_{i,j}, 1), \\
&p_{\text{C-Binary}}(v_{i,t}=1|\mathbf{h}_t,\mathbf{v}_{<t}) = \sigma(c_i + \textstyle\sum_j h_{j,t} w_{i,j}), \\
&p_{\text{C-Count}}(v_{i,t}|\mathbf{h}_t,\mathbf{v}_{<t}) = \mathcal{P}(m, \tfrac{\exp(c_i + \sum_j h_j w_{ij})}{\sum_q \exp(c_q + \sum_j h_j w_{qj})}),
\end{aligned} \quad (3)$$

where, $\mathcal{N}$ is a normal distribution, σ is a sigmoid distribution, and $\mathcal{P}$ is a Poisson distribution. The hidden nodes is defined in (4),

$$p_{\text{C}}(h_{j,t} = 1|\mathbf{v}_t, \mathbf{v}_{<t}) = \sigma(d_j + \sum_i v_{i,t} w_{i,j}). \quad (4)$$

where,

$$c_i = a_i + \sum_p A_{p,i} v_{p,<t}, \quad d_j = b_j + \sum_p B_{p,j} v_{p,<t}. \quad (5)$$

Discriminative CRBMs: DCRBMs are based on the model in (Larochelle & Bengio, 2008), generalized to account for temporal phenomenon using CRBMs. DCRBMs are a simpler version of the Factored Conditional Restricted Boltzmann Machines (Taylor et al., 2011) and Gated Restricted Boltzmann Machines (Memisevic & Hinton, 2007). Both these models incorporate labels in learning representations, however, they use a more complicated potential which involves three way connections into factors. DCRBMs define the probability distribution p_{DC} as a Gibbs distribution (6).

$$p_{\text{DC}}(\mathbf{y}_t, \mathbf{v}_t, \mathbf{h}_t|\mathbf{v}_{<t}; \boldsymbol{\theta}) = \exp[-E_{\text{DC}}(\mathbf{y}_t, \mathbf{v}_t, \mathbf{h}_t|\mathbf{v}_{<t})]/Z(\boldsymbol{\theta}). \quad (6)$$

The hidden layer $\mathbf{h}$ is defined as a function of the labels y and the visible nodes $\mathbf{v}$. A new probability distribution for the classifier is defined to relate the label y to the hidden nodes $\mathbf{h}$ as in (7),

$$p_{\text{DC}}(h_{j,t} = 1|y_t, \mathbf{v}_t, \mathbf{v}_{<t}) = \sigma(d_j + u_{j,k} + \sum_i v_{i,t} w_{ij}), \quad (7)$$

as well as relate $\mathbf{h}$ to y as in (8),

$$p_{\text{DC}}(y_{l,t}|\mathbf{h}_t) = \frac{\exp[s_l + \sum_j u_{j,l} h_{j,t}]}{\sum_{l^*} \exp[s_{l^*} + \sum_j u_{j,l^*} h_{j,t}]}. \quad (8)$$

The new energy function E_{DC} as in (9).

$$E_{\text{DC}}(\mathbf{y}_t, \mathbf{v}_t, \mathbf{h}_t|\mathbf{v}_{<t}) = \underbrace{E_{\text{C}}(\mathbf{v}_t, \mathbf{h}_t|\mathbf{v}_{<t})}_{\text{Generative}} \underbrace{- \sum_{j,l} h_{j,t} u_{jl} y_{l,t} - \sum_l s_l y_{l,t}}_{\text{Discriminative}} \quad (9)$$

Count-DCRBMs: We extend the DCRBM to CountDCRBM Figure 1. Count-DCRBMs are based on the model in (Salakhutdinov & Hinton, 2009), generalized to account for temporal phenomenon using CRBMs, and discriminative classification. Count-DCRBMs are used to model time varying histograms of counts. The probability distribution over the visible layer will follow a constrained Poisson distribution, $p_{\text{C-Count}}(v_{i,t}|\mathbf{h}_t, \mathbf{v}_{<t})$ defined in (3), the hidden layer follows (7) and the label layer follows (8) and the energy function $E_{\text{C-Count}}(\mathbf{v}_t, \mathbf{h}_t|\mathbf{v}_{<t})$ defined in (9).

4. Inference and Learning

Inference: to perform classification at time t in the CountDCRBM given $\mathbf{v}_{<t}$ and $\mathbf{v}_t$ we use a bottom-up approach, computing a cost for each possible label $\mathbf{y}_t$ then choosing the label with least cost. We compute the cost for label $\mathbf{y}_t$ to be the free energy $-\log p_{\text{DC}}(\mathbf{y}_t, \mathbf{v}_t|\mathbf{v}_{<t})$ computed by marginalizing over $\mathbf{h}_{<t}$ and $\mathbf{h}_t$. Then, the cost associated with the candidate label is the free energy in the CountDCRBM, namely $-\log p_{\text{DC}}(\mathbf{y}_t, \mathbf{h}_t|\mathbf{h}_{<t})$ is tractable, because the sum over exponentially many terms can be algebraically eliminated.

Learning: the parameters our model could be learned using Contrastive Divergence (CD) (Hinton, 2002), where $\langle\cdot\rangle_{data}$ is the expectation with respect to the data distribution and $\langle\cdot\rangle_{recon}$ is the expectation with respect to the reconstructed data. The learning is done using two steps a bottom-up pass and a top-down pass using sampling equations from (3), (7), and (8). *Bottom-up:* the reconstruction is generated by first sampling the hidden layer $p(h_{t,j} = 1|\mathbf{v}_t, \mathbf{v}_{<t}, y_l)$ for all the hidden nodes in parallel. *Top-down:* This is followed by sampling the visible nodes $p(v_{i,t}|\mathbf{h}_t, \mathbf{v}_{<t})$ and $p(y_{l,t}|\mathbf{h}_t, \mathbf{h}_{<t})$ for all the visible nodes in parallel.

5. Experiments

We use the open-source simulator cycle-level GPGPU-Sim (Bakhoda et al., 2009) to generate data to validate our approach. The simulator has been verified rigorously for accuracy against on a suite of 80 microbenchmarks (Leng & et al., 2013). We used the BACKProp problem from the RODINIA benchmark (Che et al., 2010), and simulate a NVIDIA GTX480 GPU with the default configurations for GPGPU-Sim. This benchmark CUDA program trains a feedforward neural network with one hidden layer consisting of 4096 units [1]. We tested our idea on three different caches (Instruction (IC), Data Read (D_R), Data Write (D_W)) localized within one core of the GPU. For each cache, GPGPU-Sim outputs a list of time-indexed binary labels. To complete our dataset, we modified GPGPU-Sim to retrieve the time-indexed list of instruction mix (in PTX format) for each time cycle the number of different instruction types (based on opcode) executed.

The Count-DCRBM was trained on a Tesla K20C GPU using Contrastive Divergence and a constant learning rate of 10^{-5}. Table 1 shows the final accuracies of the trained model with varying temporal history available for DCRBM. The second and third columns are metrics that describe predictive power, taking into account false positives and negatives. We observe high accuracy and pre-

[1] We only profile the forward propagation part of the benchmark.

Cache	History	MCC	F1	Accuracy
DC_R	1	0.32	0.59	0.67
	5	0.35	0.63	0.68
	10	**0.39**	**0.66**	**0.69**
DC_W	1	**0.36**	0.58	**0.70**
	5	**0.36**	**0.59**	0.65
	10	0.32	0.58	0.65
IC	1	0.32	0.40	0.86
	5	0.32	0.39	**0.87**
	10	**0.37**	**0.44**	0.85

Table 1. Scores vs History for different types of cache. The best model in each case was selected using MCC. The larger the history, the higher the complexity and training difficulty of the model. According to the table, larger history is better except in the case of Data Write Cache.

dictive power of the model for all three caches. We also observe that increased history generally leads to better performance despite the increased model complexity.

Model accuracy can be misleading because cache miss events are rare (e.g. about 10% for IC). Figure 2(Top) shows these metrics over training epochs for data write cache. Note that the initial model accuracy is already about 70% where the model predicts that cache miss never occurs, with a corresponding metric F1 and Mathew Correlation Coefficient (MCC) value of zero. As training epochs increase, we note a sharp increase in predictive power around 5000 epochs. We also show the reconstruction error in Figure 2(Middle), the objective value for training, over epochs for the data write cache. We observe that the reconstruction error significantly drops in the first 20k epochs. Figure 2(Bottom) shows a measure of the classification error, measured in terms of the binary cross entropy between the true and predicted labels. We also observe that the classification error continues to drop steadily even though the reconstruction error has converged, showing that the model accounts for label information. Finally, Figure 3 shows the prediction using a history of 10 cycles, in comparison with the ground truth.

6. Conclusions

Our approach have significant implications for the GPU revolution of computing. The data driven approach can potentially identify mix of instructions that cause performance bottlenecks. Although we focused on cache misses, any statistic of interest to the computer architecture community such as power consumption and voltage can potentially be predicted. The extension to an online embedded setting is straightforward and could potentially save computation time. Prediction of performance bottlenecks is a step towards a cognitive processor architecture.

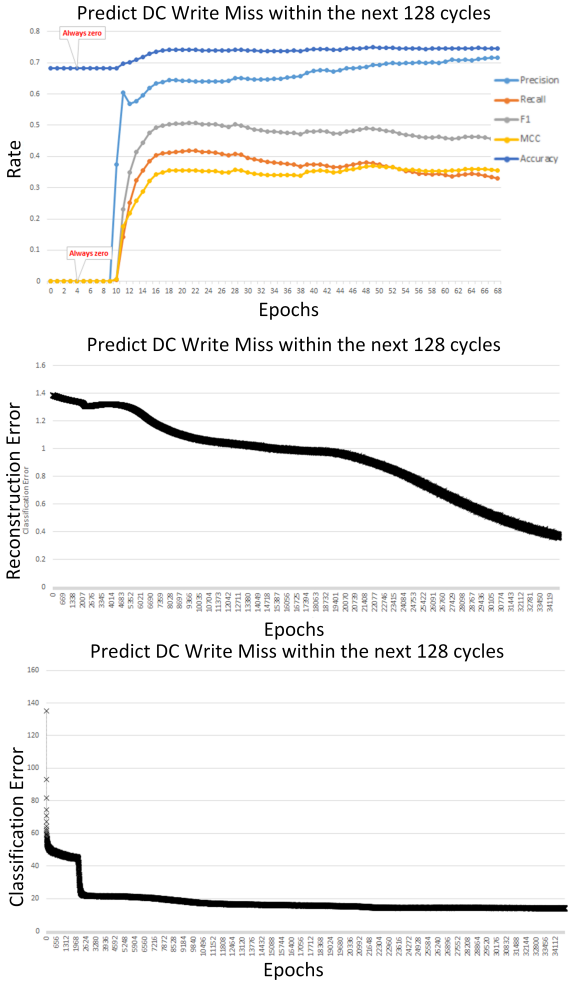

Figure 2. (Top) Accuracy dark blue, precision light in light blue, recall in red, MCC in yellow, F1 in grey, (Middle) Reconstruction Error, (Bottom) Classification Error

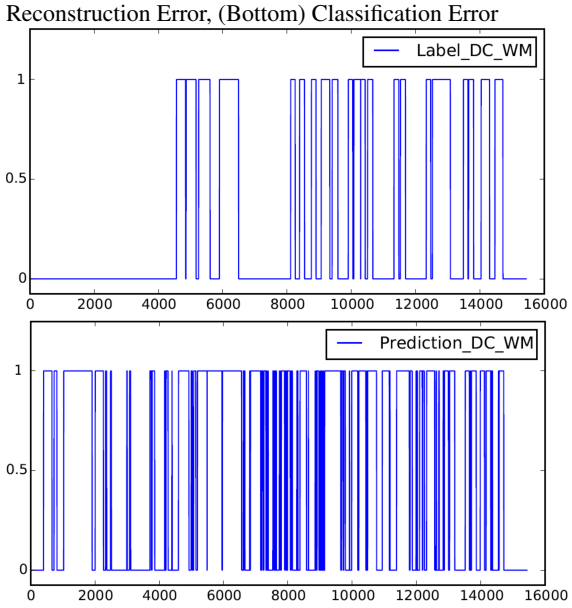

Figure 3. (Top) Ground truth labels, (Bottom) Prediction

References

Bakhoda, Ali, Yuan, George L, Fung, Wilson WL, Wong, Henry, and Aamodt, Tor M. Analyzing cuda workloads using a detailed gpu simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 2009.

Ball, Thomas and Larus., James R. Branch prediction for free. In *ACM*, 1993.

Blem, Emily, Sinclair, Matthew, and Sankaralingam, Karthikeyan. Challenge benchmarks that must be conquered to sustain the gpu revolution. *CELL*, 2011.

Calder, Brad and et al. Evidence-based static branch prediction using machine learning. In *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1997.

Che, Shuai, Sheaffer, Jeremy W, Boyer, Michael, Szafaryn, Lukasz G, Wang, Liang, and Skadron, Kevin. A characterization of the rodinia benchmark suite with comparison to contemporary cmp workloads. In *IISWC*, 2010.

Chen, Xuhao, Chang, Li-Wen, Rodrigues, Christopher I, Lv, Jie, Wang, Zhiying, and Hwu, Wen-Mei. Adaptive cache management for energy-efficient gpu computing. In *Microarchitecture*, 2014.

Esmaeilzadeh, Hadi and et. al. Dark silicon and the end of multicore scaling. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2011.

Fung, Wilson WL, Sham, Ivan, Yuan, George, and Aamodt, Tor M. Dynamic warp formation and scheduling for efficient gpu control flow. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, 2007.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. In *NC*, 2002.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. In *NC*, 2006.

Jimnez, Daniel A. and Lin, Calvin. Dynamic branch prediction with perceptrons. In *High-Performance Computer Architecture HPCA*, 2001.

Kogge, Peter and et al. Exascale computing study: Technology challenges in achieving exascale systems. In *Tech Report*, 2008.

Larochelle, H. and Bengio, Y. Classification using discriminative restricted boltzmann machines. In *ICML*, 2008.

Lee, Jaekyu, Lakshminarayana, Nagesh B, Kim, Hyesoon, and Vuduc, Richard. Many-thread aware prefetching mechanisms for gpgpu applications. In *Microarchitecture*, 2010.

Leng, Jingwen and et al. Gpuwattch: enabling energy optimizations in gpgpus. In *ACM SIGARCH Computer Architecture News*, 2013.

Leng, Jingwen, Hetherington, Tayler, ElTantawy, Ahmed, Gilani, Syed, Kim, Nam Sung, Aamodt, Tor M, and Reddi, Vijay Janapa. Gpuwattch: enabling energy optimizations in gpgpus. *ACM SIGARCH*, 2013.

Leng, Jingwen, Buyuktosunoglu, Alper, Bertran, Ramon, Bose, Pradip, and Reddi, Vijay Janapa. Safe limits on voltage reduction efficiency in gpus: a direct measurement approach. In *Microarchitecture*, pp. 294–307. ACM, 2015.

Li, Dong, Rhu, Minsoo, Johnson, Daniel R, O'Connor, Mike, Erez, Mattan, Burger, Doug, Fussell, Donald S, and Redder, Stephen W. Priority-based cache allocation in throughput processors. In *HPCA*, 2015.

Memisevic, R. and Hinton, G. E. Unsupervised learning of image transformations. In *CVPR*, 2007.

Mohamed, A. R. and Hinton, G. E. Phone recognition using restricted boltzmann machines. In *ICASSP*, 2009.

Patterson, David. Future of computer architecture. In *Berkeley EECS Annual Research Symposium (BEARS)*, 2006.

S. Chai, et al. Lightweight detection and recovery mechanisms to extend algorithm resiliency in noisy computation. In *WNTC*, 2014.

Salakhutdinov, R. and Hinton, G. E. Reducing the dimensionality of data with neural networks. In *Science*, 2006.

Salakhutdinov, Ruslan and Hinton, Geoffrey. Semantic hashing. In *International Journal of Approximate Reasoning*, 2009.

Sutskever, I. and Hinton, G. E. Learning multilevel distributed representations for high-dimensional sequences. In *AISTATS*, 2007.

Taylor, Graham W., Hinton, Geoffrey E., and Roweis, Sam T. Two distributed-state models for generating high-dimensional time series. In *Journal of Machine Learning Research*, 2011.

Tiwari, Vivek, Malik, Sharad, and Wolfe, Andrew. Power analysis of embedded software: a first step towards software power minimization. *VLSI Systems*, 1994.

Wulf, Wm A. and McKee, Sally A. Hitting the memory wall: implications of the obvious. In *ACM SIGARCH computer architecture news*, 1995.

Yeh, Tse-Yu and Patt, Yale N. Two-level adaptive training branch prediction. In *ACM ISM*, 1991.