Performance Comparison of Associative Memory Designs for High-Dimensional Computing Classification Applications

EE241B Midterm Paper, Spring 2017, March 26th

Matthew G Anderson (1st Author)
UC Berkeley, EECS Department
BWRC & SWARM Lab
Berkeley, CA, USA.
matthew_g_anderson@berkeley.edu

Miles Rusch (1st Author)
UC Berkeley, EECS Department
BWRC
Berkeley, CA, USA.
miles.rusch@berkeley.edu

Abstract— **In this paper, we compare approaches to reduce the energy and latency challenges associated with performing high dimensional classifications. Specifically, we compare digital and analog Associative Memories with traditional approaches using SRAMs and digital cores. Each of these approaches is compared based on its energy efficiency, time to match, chip area, and susceptibility to noise and process variation.**

High Dimensional (HD) Computing is a desirable alternative to traditional computing for some machine learning tasks because it can achieve higher classification accuracy with less training data and greater robustness to errors [1]. However, as these machine learning applications scale up to allow classification of large number of classes and larger input vectors, the energy and time required to perform searches becomes a bottleneck when using traditional SRAMs and digital cores. This limits the usefulness of HD classifiers in low-power and/or closed-loop applications. Associative Memories (AM) are a potential solution to these challenges but, before this work, there were no comparative studies tailored specifically to HD Computing applications.

Index Terms—Associative Memory, AM, Content Addressable Memory, CAM, Hyperdimensional Computing, Neuromorphic computing,

I. Introduction

Computing in high dimensional (HD) vector spaces shows similarities with computing in the human brain. HD computing can be used for machine learning classification tasks by encoding data samples as binary HD vectors (10,000-bits) and using a distance metric between vectors as a means for classification. This type of computation has been shown to accurately classify text language, EMG data sets, and EEG datasets [1] [2]. High Dimensional (HD) Computing is a desirable alternative to traditional computing for some machine learning tasks because it can achieve higher classification accuracy with less training data [1]. While traditional computing methods use binary numbers, HD Computing algorithms use binary vectors of a very high dimension, D, such as D=10,000 bits. Input data is first encoded into the HD vector space in such a way that two data from the same class will map to vectors that are close together (conversely, two vectors encoded from different classes will be far apart in the space). Classification is then achieved by finding the hamming distance, or the number of matching bits, between two vectors and. is used as a distance metric to classify input vectors. These HD vectors store information holographically, meaning each bit contains the same amount of information. This makes HD computing very robust to errors during computation. Furthermore, the distance between two vectors chosen randomly can be described by a binomial distribution with a very sharp peak at D/2 bits. It is this property that allows for the encoder to easily map unlike data to distant places in the vector space.

In this paper, we address the different methods for performing this classification algorithm. First, we will discuss the performance of a standard 64-bit wide microprocessor and suggest some architecture modifications that would improve the microprocessor's performance. Then we will consider the performance of some associative memory circuits, which are more specialized to compute this nearest-neighbor search. These two approaches are compared on metrics of energy efficiency, time to match, chip area, and susceptibility to noise and process variation.

Note: For this paper, we use the term Associative Memory (AM) to refer to a specific type of Content Addressable Memory (CAM) that searches its contents for the closest match to an input query rather than a perfect match.

II. Background

Calculating and comparing distances between vectors is a fundamental operation for most classifiers, including high dimensional (HD) classifiers. Traditionally, this is done using a large SRAM to store the vectors and a digital core (for instance RISC-V) to sequentially compute the distances, store them, and then select the best match (usually, the smallest distance). This approach has the advantage of minimizing the number of logic elements required for the computation, reducing hardware footprint and leakage currents. But it is also slow, since the operations are all sequential, and it does not take advantage of the robustness to errors inherent in HD classification. The possibility exists to reduce the energy consumed during classifications by parallelizing the process (i.e. increasing the time in low-power modes in between classifications) and utilizing the system robustness to lower constraints on errors (i.e. lower supply voltages, lower SNR, etc.).

In the following sections, we outline the start-of-the-art for both the traditional SRAM / Digital Core and the AM approaches for high dimensional classification.

III. *Traditional SRAM / Digital Core Approach*

Distance based classifications performed using on-chip SRAM and a digital core follow a standard procedure. The procedure outlined below would be carried out on a B-bit RISC-V core to classify an input vector of dimension D, into C classes.

1. **Compute the distance from the input vector to each of the C classes**: This could be done using the RISC-V ALU and would require ~(C x D / B) instructions to compute the bit-matches from the input to each class, B-bits at a time. This can be performed using bit-wise XORs. Then another ~(C x D / B) instructions to accumulate each bit-match into C distances, B-bits at a time, assuming that the RISC-V ALU is equipped to perform bit-wise accumulations (an optimistic assumption). If using a standard ALU without this function, accumulating bit-matches will take ~(C x D) instructions, which would be the largest bottleneck in the datapath of the design (however accessing the C stored class vectors of D bits each from memory would likely be the most significant bottleneck of the design)

2. **Identify the best match or smallest distance**: This would require iterating through the C classes while storing and updating the smallest distance encountered, which would take on the order of ~(C) instructions.

Overall, assuming a design which uses require C x (D/B + D) = C*D*(1+1/B) instructions to classify an input. It would also require C x D bits read through the memory hierarchy of SRAM DRAM, and FLASH.

As an example, consider using readily available off-the-shelf components, such an algorithm would 64-bit RISC-V processor, 10,000 bit vectors, and 256 classes (B=64, D=10000, C=256). One associative search would take approximately C*D = 2.56e6 cycles in the datapath, ignoring cache misses.

Considering a modified processor which is able to accumulate B bits at a time, the algorithm requires ~(2*C*D/B) cycles, or 80e3 cycles, ignoring cache misses. From this result, we see that a simple addition to the ALU can significantly reduce the latency of classification. Since each of the C class vectors must be loaded from memory in a predictable way, cache misses can be greatly reduced by pre-fetching class vectors from the slower physical memories.

IV. *Digital AM Approach*

Much of this algorithm can be parallelized by using a custom digital Associative Memory (AM) design [4]. This custom hardware approach computes the number of bit-matches for each class in parallel using C rows of D XOR gates, sends the bit-matches to C binary counters which iteratively accumulates its D input bits. The distances from the binary counters are then passed through a binary comparator tree. Assuming the comparator tree finds the max distance in one cycle, which is possible at great area cost, the digital AM would take D cycles, or 10e3 in our previous example. When considering the large area reported in Table 1 of [4] (15mm^2), the speedup is a possibly unworthy improvement over the 80e3 cycles required by the slightly modified processor design.

Table I
ENERGY AND AREA PARTITIONING FOR D-HAM.

D-HAM	Modules	Area (mm^2)	Energy (pJ)
D=10,000	CAM Array	15.2	4976.9
	Counters and comparators	10.9	1178.2
d=9,000	CAM Array	13.7	4479.2
	Counters and comparators	10.2	1131.1
d=7,000	CAM Array	10.6	3483.8
	Counter and comparator	8.3	883.6

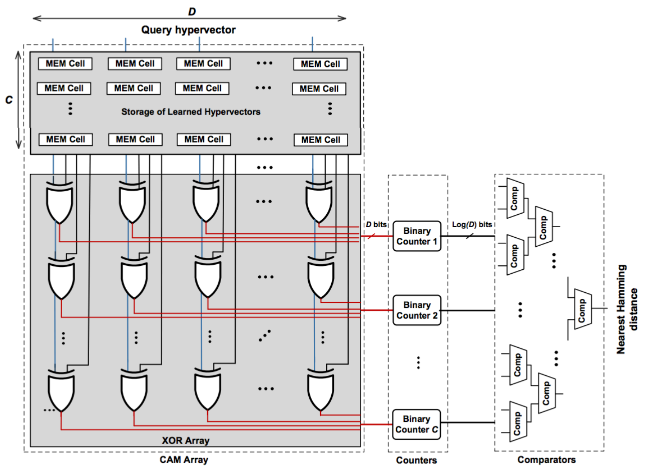


Figure 2. Overview of D-HAM.

V. *Analog AM Approach*

To avoid the expensive area requirements of the digital implementation of the distance computation and comparison, we consider computing these values using analog signals.

First, we consider computing each distance in an analog fashion. The digital implementation uses an XOR gate to represent a bit-match as a digital voltage. By instead representing a bit-match as a current, currents from each cell that matches the input data can be added on a match line. This essentially replaces a digital counter with D=10,000 inputs with a single match line. Furthermore, the XOR function can be achieved by adding only three NMOS transistors to a standard 6T SRAM cell [3]. Due to the analog nature of adding currents, the consequence is reduced accuracy of the distance computation. However, from Figure 1 of [4], we see that the classification accuracy of the overall algorithm can tolerate up to 1,000 bits out of D=10,000 before showing any degradation. This means that the current sources can have noise and mismatch errors of up to 10% without significantly affecting the accuracy of the classification.

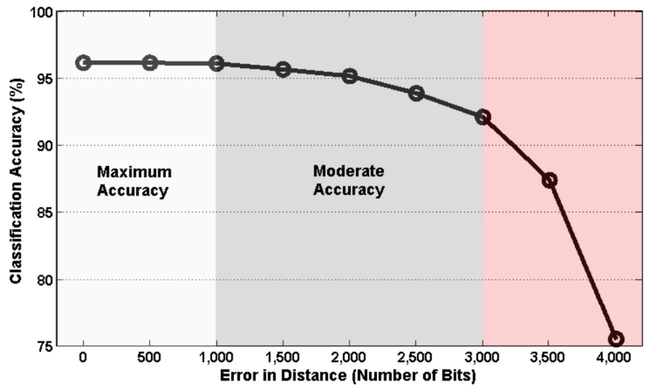


Figure 1. Language classification accuracy with wide range of errors in Hamming distance using $D = 10,000$.

The analog currents from each match line in the associative memory must then be compared. One way to do this is to use a Loser-Take-All (LTA) circuit [5]. [5] reports "a precision of 99.5% at data rate of 2.5 MSamples/s and energy consumption of 0.3-1 pJ per input" using simulation results from a 0.18μm process. The precision is likely an optimistic result. A similar LTA circuit is presented in [6], which reports a circuit with 99% precision using experimental data on a MIETEC 2.4-um CMOS process. It is difficult to compare timing and power results between different processes, however [6] uses less transistors per cell than [5] while achieving similar precision.

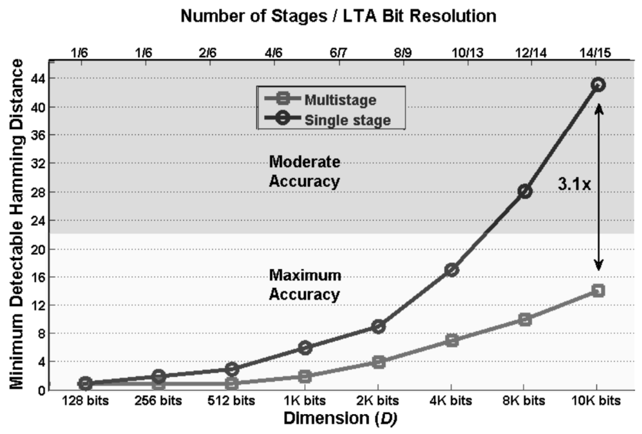
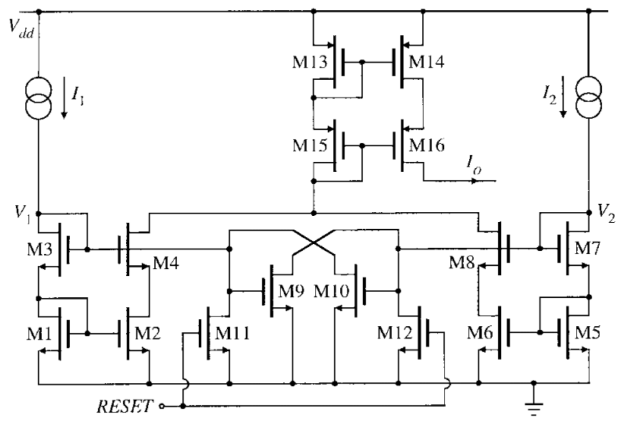


Figure 7. Minimum detectable distance in A-HAM.

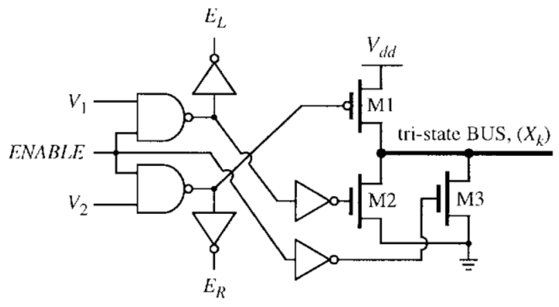
According to Figure 7 in [4], the LTA's must resolve a minimum of 24 bits in the Hamming Distance before the accuracy of the algorithm begins to degrade. While 24 bits out of 10,000 is a fraction far below 1%, [4] claims the LTA design of [5] has the required resolution.

Below in Figure 3 and Figure 4 of [6] are diagrams of the circuit that passes the max current and the circuit that computes the partial address of the class corresponding to the max current.



Transistor	M1 - M10	M11, M12	M13 - M16
W (μm)	20	4.8	40
L (μm)	2.4	2.4	2.4

Fig. 3. FPC and transistor dimensions.



V_1	V_2	$ENABLE$	E_L	E_R	X_k
1	0	0	0	0	0
0	1	0	0	0	0
1	0	1	1	0	0
0	1	1	0	1	1

Fig. 4. Logic schematic and truth tables.

VI. *Method Of Comparison*

The key Figures Of Merit (FOM) that we will use to compare the approaches outlined are listed below.

- Energy Efficiency [pJ/search] versus D & C
- Time to Match [ns] versus D & C
- Energy Efficiency x Time to Match [pJ ns] versus D & C
- Chip Area [um^2] versus D & C
- Resolution of Distance Computation [bits] versus D & C
- Resolution of Distance Comparison [bits] versus D & C

VII. Next Steps

Using MATLAB models, Spectre simulation, and paper references we hope to compare the energy efficiency, time to match, chip area, and susceptibility to noise / process variation of the approaches list above for different input vector sizes (D) and number of classes (C).

We expect the traditional SRAM and digital core approach to perform best at small values of ~(D x C), outperforming the Analog and Digital AM approaches in almost all metrics. However, at some larger ~(D x C), we expect the Analog and Digital AM approaches to begin to excel, especially in terms of time to match and possibly energy efficiency. We will need to be quantify whether this potential increase in time to match and energy efficiency outweigh the increased chip area for the digital AM solutions and potential errors in distance measurements for the analog AM solutions.

Acknowledgment

Special thanks to Professor Borivoje Nikolic of UC Berkeley (bora@eecs.berkeley.edu) for his help preparing this report.

References

[1] Abbas Rahimi et al. Hyperdimensional Computing for Noninvasive Brain–Computer Interfaces: Blind and One-Shot Classification of EEG Error-Related Potentials. ACM 2017.

[2] Abbas Rahimi et al. A Robust and Energy Efficient Classifier Using Brain-Inspired Hyperdimensional Computing. ISLPED 2016.

[3] Pagiamtzis, Kostas, and Ali Sheikholeslami. "A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme." IEEE Journal of Solid-State Circuits 39.9 (2004): 1512-1519.

[4] M. Imani, **A. Rahimi**, D. Kong, T. Rosing, J. M. Rabaey "Exploring Hyperdimensional Associative Memory," In *IEEE Symposium on High Performance Computer Architecture (HPCA)*, February 2017.

[5] R. Dlugosz, A. Rydlewski, and T. Talaska, "Low power nonlinear min/max filters implemented in the cmos technology," in Microelectronics Proceedings-MIEL 2014, 2014 29th International Conference on, pp. 397–400, IEEE, 2014

[6] A. Demosthenous, S. Smedley, J. Taylor, "A CMOS analog Winner-Takes-All network for large-scale applications", IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 45, No. 3, 1998, pp.300-304.