In-Memory Hamming Similarity Computation in Resistive Arrays

Yuval Cassuto and **Koby Crammer**
Department of Electrical Engineering, Technion – Israel Institute of Technology
ycassuto@ee.technion.ac.il, koby@ee.technion.ac.il

***Abstract*— This paper develops a framework to calculate Hamming similarity between vectors stored in resistive memory. A single-parameter model is proposed for the resistive measurement channel, which is then used to analytically reveal an interesting tradeoff between this parameter and succeeding in the calculation task. We suggest coding techniques that can improve this tradeoff under natural usage assumptions. The proposed constructions work to change the Hamming weight of the stored vectors without corrupting the Hamming distance between pairs of vectors.**

I. Introduction

In recent years, the information-systems world is seeing two significant parallel trends. One is the continued explosion of the data stored and processed by ambitious data-intensive algorithms. The other is the complete breakdown of the traditional memory hierarchy in computers. The former is a challenge rising from transfer bottlenecks between processing and memory units, while the latter is a promise fueled by density and speed advancements of emerging memory technologies. One of the most sought-after objectives of research in computer engineering today is to make data-intensive algorithms more efficient using novel memory architectures. In this paper we explore an idea in that vein: using ultra-dense resistive memories to compute similarity between data vectors stored within then.

Instead of fetching a large number of high-dimension data vectors from memory to the processor for similarity calculations, this operation will be performed *in memory*. As a result, processor-memory bottlenecks will be alleviated. Our key observation is that the physical medium of resistive memories (which are the most promising memory technology density-wise) has natural properties to perform similarity calculations efficiently at a low level. The results of this paper serve as an initial setup of such a framework, highlighting the key challenges and promises of this solution.

From the perspective of information/communication theory, the paper deals with the *physical layer* of a memory medium that is simple enough for clear abstract characterization. More specifically, the medium is an array storing rows of bits, which is readable through low-level resistance measurements between pairs of rows. The value of this readout is regarded as the output of a deterministic *multiple access channel*, and our objective is to use this channel output to infer the similarity between the measured row pair. The measure of similarity we use here is the *Hamming distance* between the corresponding vectors, because it is both fundamental and useful in real machine-learning algorithms [1], [4], [6], [7].

The contents of the paper include the problem formulation and abstraction in Section II, exploring the tradeoff between the array parameter and calculation success in Section III, and the introduction of coding to help improve this tradeoff in Section IV. Throughout the paper we assume no noise is present in the measurement, which leaves models with noise as the principal direction for future work.

A. About resistive memories

In a resistive crossbar array, a resistive element (often called memristor) is positioned on each row-column intersection, and its resistance, low or high, represents two logical states. The state of memristors in the array can be sensed by measuring the current flowing through them when a voltage is applied to array terminals. There is significant literature on the employment of resistive memories for storage applications (e.g. [8], [5], [2]), and also for logic computations (e.g. [3]). Our focus here will be specifically on sensing the resistance (or equivalently conductance) between pairs of rows, for the purpose of finding the Hamming distance between them.

II. In-Memory Hamming-Distance Calculations

This section serves to extract an appropriate abstract model from the physical layer of resistive memories. We define a memory *row* $\boldsymbol{x}$ as a binary vector of a fixed dimension n. Given two rows $\boldsymbol{x}$, $\boldsymbol{y}$, we use the standard definitions of *Hamming weight* $W_H(\boldsymbol{x}) = |\{i : x_i \neq 0\}|$ and *Hamming distance* $D_H(\boldsymbol{x}, \boldsymbol{y}) = |\{i : x_i \neq y_i\}|$.

A. The physical layer of resistive distance measurement

The states of the bits stored in memristive arrays are represented by the resistance values of the corresponding cells. In an *ideal* resistive array, the resistance of 1-state cells is some real value R, and the 0-state cells have infinite resistance. In a *real* resistive array, the resistance of 0-state cells is not set to an infinite value. Instead, a high finite resistance, denoted R_∞, is chosen. The array structure is given in Figure 1, where R-cells are represented by dark dots, and R_∞-cells are light dots. Each row in the array represents one stored vector of length n. For example, the contents of rows 1,2,3 are thus $(0, 0, 1, 1, 1)$, $(0, 0, 1, 0, 1)$, and $(1, 1, 1, 1, 1)$, respectively. Calculating the Hamming distance between two row vectors will be done by applying a voltage between the corresponding row terminals on the left, and measuring the current flowing from the source. The equivalent circuit of this measurement is given in Figure 2 for rows 1 and 2. The resulting circuit has n parallel branches, each with two resistors. It will be most convenient to move from resistance values to the reciprocal

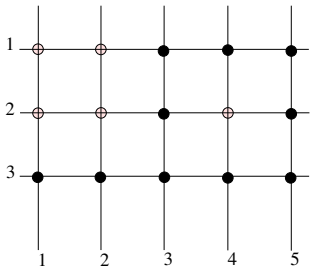

Figure 1. Programmed rows in a real resistive array.

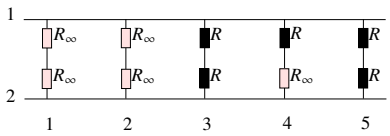

Figure 2. Equivalent circuit for measurement between rows 1 and 2.

conductance values, thereby allowing to sum the contributions of the n branches. We denote the 1-state conductance by G and the 0-state conductance by ϵG, where ϵ is some real-valued constant smaller than 1. Note that the case $\epsilon = 0$ represents the ideal case with infinite 0-state resistance. Now it is clear that the measurement described above yields the equivalent conductance of the circuit induced by the row pair $\boldsymbol{x},\boldsymbol{y}$

$$\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} = \sum_{i=1}^{n} x_i y_i + x_i(1-y_i)\frac{2\epsilon}{1+\epsilon} + (1-x_i)y_i\frac{2\epsilon}{1+\epsilon} + (1-x_i)(1-y_i)\epsilon, \tag{1}$$

where $\tilde{G}$ represents the conductance value normalized by $G/2$. We will use this normalization throughout the paper.

Our objective is to find the Hamming distance between the vectors $\boldsymbol{x},\boldsymbol{y}$ from the value of $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$. The problem structure is presented in Figure 3. The figure describes a problem

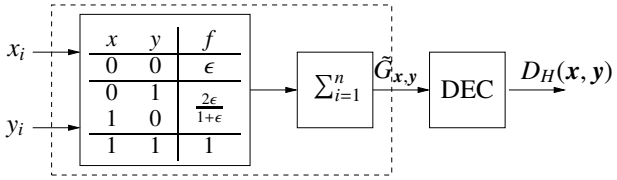

Figure 3. Abstract view of the problem: the measurement channel (in dashed frame) outputs the sum of $f(x, y)$ values representing the branch conductances. Then the objective is to decode the channel output $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ to get the Hamming distance between $\boldsymbol{x},\boldsymbol{y}$.

reminiscent of a deterministic multiple-access channel (MAC), where the channel mixes the inputs using some symmetric function f determined by the physics of the problem. One distinction from traditional MACs is that the channel output is the sum over all the function outputs in the block. As a result, increasing the block length only makes the channel worse, and thus this model will not lend itself to asymptotic characterization.

Given two rows $\boldsymbol{x},\boldsymbol{y}$, define $N_{00}(\boldsymbol{x},\boldsymbol{y})$ to be the number of coordinates $i \in \{1,\ldots,n\}$ such that $x_i = y_i = 0$. Similarly, $N_{10}(\boldsymbol{x},\boldsymbol{y})$ is the number of coordinates with $x_i = 1, y_i = 0$, $N_{01}(\boldsymbol{x},\boldsymbol{y})$ is the number with $x_i = 0, y_i = 1$, and $N_{11}(\boldsymbol{x},\boldsymbol{y})$ with $x_i = y_i = 1$. Clearly we have

$$N_{00}(\boldsymbol{x},\boldsymbol{y}) + N_{10}(\boldsymbol{x},\boldsymbol{y}) + N_{01}(\boldsymbol{x},\boldsymbol{y}) + N_{11}(\boldsymbol{x},\boldsymbol{y}) = n. \tag{2}$$

When $\boldsymbol{x},\boldsymbol{y}$ are clear from the context, we will write N_{00}, N_{10}, N_{01}, N_{11}, without their arguments. Using the above notations, we rewrite (1) as

$$\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} = N_{11} + (N_{01} + N_{10})\frac{2\epsilon}{1+\epsilon} + N_{00}\epsilon. \tag{3}$$

For the special case of $\boldsymbol{y} = \mathbf{1}$, where $\mathbf{1}$ is an all-1 vector, we have $N_{00} = N_{10} = 0$, $N_{11} = W_H(\boldsymbol{x})$, and $N_{01} = n - W_H(\boldsymbol{x})$. From these values it is possible to find the Hamming weight of $\boldsymbol{x}$ by measuring $\tilde{G}_{\boldsymbol{x},\mathbf{1}}$ and taking

$$W_H(\boldsymbol{x}) = \frac{(1+\epsilon)\tilde{G}_{\boldsymbol{x},\mathbf{1}} - 2n\epsilon}{1-\epsilon}, \tag{4}$$

for any $0 \leqslant \epsilon < 1$.

Back to the case of general $\boldsymbol{y}$, we wish to reach an expression for $D_H(\boldsymbol{x},\boldsymbol{y})$ by measuring $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$, and assuming known $W_H(\boldsymbol{x})$, $W_H(\boldsymbol{y})$ from (4). To do so, we observe that

1) $N_{11} = [W_H(\boldsymbol{x}) + W_H(\boldsymbol{y}) - D_H(\boldsymbol{x},\boldsymbol{y})]/2$.
2) $N_{01} + N_{10} = D_H(\boldsymbol{x},\boldsymbol{y})$ by definition.
3) $N_{00} = n - N_{10} - N_{01} - N_{11}$ by (2).

Using the above relations, we substitute in (3) and rearrange to get

$$D_H(\boldsymbol{x},\boldsymbol{y}) = \frac{1+\epsilon}{(1-\epsilon)^2}\left[(1-\epsilon)(W_H(\boldsymbol{x}) + W_H(\boldsymbol{y})) + 2n\epsilon - 2\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}\right], \tag{5}$$

for any $0 \leqslant \epsilon < 1$. The conclusion from (5) and (4) is that the Hamming distance can be calculated by a sequence of three measurements $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}, \tilde{G}_{\boldsymbol{x},\mathbf{1}}, \tilde{G}_{\boldsymbol{y},\mathbf{1}}$, for any $0 \leqslant \epsilon < 1$.

III. Single-Measurement Hamming-Distance Calculation

Requiring three measurements to extract the Hamming distance is too costly in time, and not viable in practice. So our objective in this section is to explore the possibility to obtain the Hamming distance with only *one measurement*. As we are moving to a higher abstraction level, we stop referring to $\boldsymbol{x},\boldsymbol{y}$ as rows, and simply call them vectors.

Theorem 1. *If $0 < \epsilon < 1/(2n-1)$, then the Hamming distance $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ by*

$$D_H(\boldsymbol{x},\boldsymbol{y}) = \frac{\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} - \lfloor\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}\rfloor - \epsilon(n - \lfloor\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}\rfloor)}{\epsilon\frac{1-\epsilon}{1+\epsilon}}.$$

Proof: Since $\epsilon < 1/(2n-1)$, we are guaranteed that the total contribution of $N_{01} + N_{10}$ and N_{00} will sum to at most

$$(N_{01} + N_{10})\frac{2\epsilon}{1+\epsilon} + N_{00}\epsilon \leqslant n\frac{2\epsilon}{1+\epsilon} < 1,$$

and hence will only contribute a fractional number to $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$. That means that we can calculate the Hamming distance in two steps: first find the integer N_{11}, then subtract its contribution from $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ and recover $D_H(\boldsymbol{x},\boldsymbol{y})$ from the residue. N_{11} can be recovered directly from $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ by calculating

$$N_{11} = \left\lfloor\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}\right\rfloor.$$

Once we know N_{11} it is straightforward to rearrange (3) and express the Hamming distance as

$$D_H(\boldsymbol{x},\boldsymbol{y}) = \frac{\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} - N_{11} - \epsilon(n - N_{11})}{\epsilon\frac{1-\epsilon}{1+\epsilon}}. \tag{6}$$

■

Theorem 1 shows that if $0 < \epsilon < 1/(2n-1)$, then $D_H(\boldsymbol{x},\boldsymbol{y})$ can use a single measurement followed by a simple two-step procedure: first finding N_{11} and then obtaining $D_H(\boldsymbol{x},\boldsymbol{y})$ in closed form. It is interesting to note that in the ideal case ($\epsilon = 0$) it is *not* possible to obtain the exact Hamming distance in a single measurement, since the measurement cannot distinguish between the 00 vs. the 01 and 10 combinations. It is shown in the next theorem that the requirement on ϵ can be relaxed by a factor 2 while maintaining the ability to find $D_H(\boldsymbol{x},\boldsymbol{y})$ from $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$.

Theorem 2. *If $0 < \epsilon < 1/(n-1)$, then the Hamming distance $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ by*

$$D_H(\boldsymbol{x},\boldsymbol{y}) = \frac{\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} - N - \epsilon(n - N)}{\epsilon\frac{1-\epsilon}{1+\epsilon}}, \tag{7}$$

where N is the unique integer that gives $0 \leqslant D_H(\boldsymbol{x},\boldsymbol{y}) \leqslant n$.

Proof: If $N = N_{11}$, we already saw in (6) that the Hamming distance is obtained from $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ and N_{11} according to (7). We now prove that when $0 < \epsilon < 1/(n-1)$, there is a unique integer N that gives a Hamming distance between 0 and n. Assume to the contrary that there are two integers N and $N' = N + 1$ that give a valid Hamming distance in (7). Then we have

$$\frac{\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} - N - \epsilon(n - N)}{\epsilon\frac{1-\epsilon}{1+\epsilon}} - \frac{\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} - N' - \epsilon(n - N')}{\epsilon\frac{1-\epsilon}{1+\epsilon}} = \frac{1+\epsilon}{\epsilon} > n,$$

where the inequality follows from the condition $\epsilon < 1/(n-1)$. Because $D_H(\boldsymbol{x},\boldsymbol{y})$ is an integer in $\{0,\ldots,n\}$, only one of the hypotheses N and N' gives a legal distance value. The same applies if the difference between N' and N is > 1. ■

The result of Theorem 2 is essentially tight in the sense that if we set $\epsilon = 1/(n-1)$, measuring $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ can leave us with the maximal ambiguity of n in determining $D_H(\boldsymbol{x},\boldsymbol{y})$. For example, let $\boldsymbol{x} = 100\cdots0$ and $\boldsymbol{y} = 011\cdots1$. Then we have $D_H(\boldsymbol{x},\boldsymbol{x}) = 0$ and $D_H(\boldsymbol{x},\boldsymbol{y}) = n$, while

$$\tilde{G}_{\boldsymbol{x},\boldsymbol{x}} = 1 + (n-1)\epsilon = 2 = n\frac{2\epsilon}{1+\epsilon} = \tilde{G}_{\boldsymbol{x},\boldsymbol{y}},$$

where the first equality is from $N_{11} = 1$, $N_{00} = n - 1$ between $\boldsymbol{x}$ and itself and from (3), the second and third equalities from substituting $\epsilon = 1/(n-1)$ and rearrangement, and the fourth equality from $N_{10} + N_{01} = n$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ and from (3). So in the example of $\boldsymbol{x}$ and $\boldsymbol{y}$ above we have the same readout $\tilde{G}$ for the two extreme cases of identical and inverted vectors.

Remark: one may be tempted to guess that the $1/\epsilon > n - 1$ requirement in Theorem 2 comes from the fact that $f(0,0) = \epsilon f(1,1)$. But in fact this relation between n and the maximal ϵ stems from the deeper fact that $f(0,1) - f(0,0) = \epsilon[f(1,1) - f(1,0)]$. The latter expression is a result of the underlying resistive medium, and may not apply for alternative physical realizations of the memory (for example, when adding non-linearities to the cell elements).

A. Improved measurement with partial knowledge on vectors

To loosen up the single-measurement upper bounds on ϵ from Theorems 1 and 2, we now take a closer look at a likely usage of similarity measurements in information-retrieval applications. A common use of similarity calculations is to compare between pairs of binary vectors, in which each entry represents some feature present or not present in a corresponding object.

The first case we consider is when we have some lower bound on the weight of one of the compared vectors. The reason to consider this case is because comparing vectors to a test vector with very low weight may not be interesting for the application, as a test vector with few features is not very informative.

With a weight lower bound on at least one input vector, we can now relax the requirement on ϵ in Theorem 2. We prove this in the following assuming a bound on $W_H(\boldsymbol{x})$, but from symmetry the same result applies if $W_H(\boldsymbol{y})$ (or both) is bounded. In the sequel, $\underline{w}$ and $\bar{w}$ represent scalar lower and upper bounds on vector weights.

Theorem 3. *Suppose $W_H(\boldsymbol{x}) \geqslant \underline{w}$. Then if $0 < \epsilon < 1/(n - \underline{w})$, the Hamming distance $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$.*

Proof: We prove by contradiction. Suppose there exist two pairs of vectors $\boldsymbol{x}$,$\boldsymbol{y}$ and $\boldsymbol{x}'$,$\boldsymbol{y}'$ such that $D_H(\boldsymbol{x},\boldsymbol{y}) \neq D_H(\boldsymbol{x}',\boldsymbol{y}')$, and which give $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} = \tilde{G}_{\boldsymbol{x}',\boldsymbol{y}'}$. Denote $W_H(\boldsymbol{x}) = w_x$ and $W_H(\boldsymbol{x}') = w_{x'}$, which satisfy $w_x, w_{x'} \geqslant \underline{w}$. From $D_H(\boldsymbol{x},\boldsymbol{y}) \neq D_H(\boldsymbol{x}',\boldsymbol{y}')$ we know that $N_{11} \neq N'_{11}$ (recall from (6) that N_{11} and $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$ determine $D_H(\boldsymbol{x},\boldsymbol{y})$ unambiguously when $\epsilon > 0$), and denote $\Delta_{11} = N_{11} - N'_{11}$. Assume without loss of generality that $N_{11} > N'_{11}$, and thus $\Delta_{11} \geqslant 1$. For $\boldsymbol{x}$,$\boldsymbol{y}$ we write the relations $N_{10} = w_x - N_{11}$ and $N_{00} = n - w_x - N_{01}$, and similar relations hold for $\boldsymbol{x}'$,$\boldsymbol{y}'$. Now substituting for each pair in (3) we get

$$\tilde{G}_{\boldsymbol{x},\boldsymbol{y}} = N_{11} + (w_x - N_{11})\frac{2\epsilon}{1+\epsilon} + N_{01}\frac{2\epsilon}{1+\epsilon} + (n - w_x - N_{01})\epsilon,$$

$$\tilde{G}_{\boldsymbol{x}',\boldsymbol{y}'} = N'_{11} + (w_{x'} - N'_{11})\frac{2\epsilon}{1+\epsilon} + N'_{01}\frac{2\epsilon}{1+\epsilon} + (n - w_{x'} - N'_{01})\epsilon.$$

Subtracting the two equations we get

$$1 \leqslant \Delta_{11} = (N'_{01} + w_{x'} - N_{01} - w_x)\epsilon, \tag{8}$$

and from the relations $N'_{01} + w_{x'} \leqslant n$ and $N_{01} + w_x \geqslant \underline{w}$ we get $1 \leqslant (n - \underline{w})\epsilon$, which is a contradiction. ■

The second case we consider is when we know that the weights of the input vectors are bounded in some interval of size Δw. That is

$$\underline{w} \leqslant W_H(\boldsymbol{x}), W_H(\boldsymbol{y}) \leqslant \bar{w}, \quad \bar{w} - \underline{w} \triangleq \Delta w.$$

In this case we want to get a relaxation of the requirement on ϵ that will improve as Δw gets smaller. The motivation for this case comes from applications where the vectors stored in the database have similar numbers of present features, which is often set as a prior for the algorithm that generates these vectors. For this case we have the following theorem.

Theorem 4. *Suppose $W_H(\boldsymbol{x})$ and $W_H(\boldsymbol{y})$ are in some known interval of size Δw. Then if $0 < \epsilon < 1/(1 + 2\Delta w)$, the Hamming*

distance $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{\boldsymbol{x},\boldsymbol{y}}$.

Proof: We follow an argument similar to Theorem 3 and observe in addition that $N_{01} = w_y - N_{11}$, and similarly for the pair $\boldsymbol{x}'$,$\boldsymbol{y}'$. Therefore

$$N'_{01} - N_{01} = w_{y'} - N'_{11} - (w_y - N_{11}) = w_{y'} - w_y + \Delta_{11} \leqslant \Delta w + \Delta_{11}. \tag{9}$$

In addition, from the weight bounds on $\boldsymbol{x}$ we have

$$w_{x'} - w_x \leqslant \Delta w. \tag{10}$$

Substituting (9), (10) into (8) we get after simple manipulation

$$\epsilon \geqslant \frac{\Delta_{11}}{\Delta_{11} + 2\Delta w} \geqslant \frac{1}{1 + 2\Delta w},$$

where the last inequality follows because the function $a/(a+b)$ is increasing in a, and Δ_{11} is at least 1. We now reached a contradiction. ■

Note that when $\Delta w = 0$, i.e., when the weights of the inputs are known exactly, Theorem 4 implies that $D_H(\boldsymbol{x},\boldsymbol{y})$ can be found exactly with any $\epsilon < 1$. This fact was already stated in (5), so Theorem 4 gives a generalization for partially known weights.

IV. Coding for Hamming-Distance Measurement

So far in the paper we analyzed the calculation of Hamming distances between vectors stored in the memory in their raw form. The main conclusion from the results of Section III is that exact Hamming-distance calculation for raw vectors casts constraints on ϵ that may not be favorable in practice. Now we seek to explore *coding* techniques that allow milder restrictions on the memory hardware. By coding we mean that the input vectors are modified before they are stored in the memory, such that measurement of the modified vectors allows calculating the Hamming distance between the *original* vectors. We denote the length of the original vectors by n and the length of the modified vectors by $N > n$. The *overhead* of the code is defined as $(N - n)/n$.

A. Doubling construction

We first present a simple code construction that allows exact calculation of the Hamming distance for any $\epsilon < 1$. Later we give a much more efficient construction based on the same ideas.

For a vector $\boldsymbol{s}$ we define by $\neg\boldsymbol{s}$ the bit-wise inversion of $\boldsymbol{s}$.

Construction 1. *Given a vector $\boldsymbol{y}$ in raw form, we define the encoding of the vector as $c(\boldsymbol{y}) = [\boldsymbol{y}|\neg\boldsymbol{y}]$, where | represents concatenation.*

Proposition 5. *For any pair of vectors $\boldsymbol{x}$, $\boldsymbol{y}$ encoded according to Construction 1, $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single measurement for any $\epsilon < 1$.*

Proof: Immediate from the facts $W_H(c(\boldsymbol{x})) = n$ and $D_H(c(\boldsymbol{x}), c(\boldsymbol{y})) = 2D_H(\boldsymbol{x},\boldsymbol{y})$. ■

The overhead of Construction 1 is 1, which is rather large.

B. Efficient construction with one known weight

To obtain more efficient codes, we assume a more specific operation of the memory array. A common use of similarity calculations is when a large database of vectors $\{\boldsymbol{y}\}$ is stored in memory, and we want to measure the similarity between the database vectors and *a given test vector* $\boldsymbol{x}$ provided as input to the array. This usage is more favorable than the general pair-wise comparisons in two ways. One is that we are able to measure the Hamming weight of $\boldsymbol{x}$ without adding significant time overhead (one such weight measurement suffices for an entire sequence of distance measurements). Another is that we are able to encode $\boldsymbol{x}$ differently than the coding used for the vectors $\{\boldsymbol{y}\}$ in the memory. Under this use case, we now show how we can use coding to relax the requirement on ϵ with little redundancy. We again suppose that the input vectors from $\{\boldsymbol{y}\}$ have their Hamming weights in the interval $[\underline{w}, \bar{w}]$ of size $\Delta w + 1$. We assume that Δw is even. For the purpose of the next construction we define $\sigma_w(k)$ to be a vector of length w with k ones followed by $w - k$ zeros.

Construction 2. *Given a pair of vectors $\boldsymbol{x}$, $\boldsymbol{y}$, we apply the following encodings:*

$$c_1(\boldsymbol{x}) = \left[\boldsymbol{x} \mid \sigma_{\frac{\Delta w}{2}}\left(\frac{\Delta w}{2}\right) \mid \sigma_{\frac{\Delta w}{2}}(0)\right]$$

to the vector $\boldsymbol{x}$ and

$$c_2(\boldsymbol{y}) = \left[\boldsymbol{y} \mid \sigma_{\frac{\Delta w}{2}}\left(\left\lceil\frac{\bar{w} - w_y}{2}\right\rceil\right) \mid \sigma_{\frac{\Delta w}{2}}\left(\left\lceil\frac{\bar{w} - w_y}{2}\right\rceil\right)\right]$$

to the vector $\boldsymbol{y}$.

Each of the encodings c_1 and c_2 appends Δw redundant bits to the input vector. c_1 appends one string of $\Delta w/2$ ones and one string of $\Delta w/2$ zeros. c_2 appends two identical strings of $\Delta w/2$ bits that together complete the weight of $c_2(\boldsymbol{y})$ to either $\bar{w}$ when $\bar{w} - w_y$ is even, or to $\bar{w} + 1$ when $\bar{w} - w_y$ is odd. The objective of the redundant bits is to add certainty about the weights of the coded vectors, while allowing to recover the original (uncoded) Hamming distance following a single measurement. The following theorem shows how Construction 2 achieves this objective.

Theorem 6. *For any pair of vectors $\boldsymbol{x}$, $\boldsymbol{y}$, satisfying $W_H(\boldsymbol{x}) = w_x$ and $\bar{w} - \Delta w \leqslant W_H(\boldsymbol{y}) \leqslant \bar{w}$, if $0 < \epsilon < 1/2$, then the Hamming distance $D_H(\boldsymbol{x},\boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{c_1(\boldsymbol{x}),c_2(\boldsymbol{y})}$.*

Proof: Construction 2 implies the property that for any vectors $\boldsymbol{x}$,$\boldsymbol{y}$ we have

$$D_H(c_1(\boldsymbol{x}), c_2(\boldsymbol{y})) = D_H(\boldsymbol{x},\boldsymbol{y}) + \Delta w/2.$$

Hence finding $D_H(c_1(\boldsymbol{x}), c_2(\boldsymbol{y}))$ will give $D_H(\boldsymbol{x},\boldsymbol{y})$ unequivocally. We also know that

$$W_H(c_2(\boldsymbol{y})) \in \{\bar{w}, \bar{w} + 1\}. \tag{11}$$

In the terminology of Theorem 4, $W_H(c_2(\boldsymbol{y}))$ has $\Delta w' = 1$ (from (11)), and $W_H(c_1(\boldsymbol{x}))$ has $\Delta w'' = 0$ ($W_H(c_1(\boldsymbol{x}))$ is simply $w_x + \Delta w/2$, where w_x is known). By following a similar argument to the proof of Theorem 4, we can conclude that $\epsilon < 1/(1 + \Delta w' + \Delta w'') = 1/2$ suffices. ■

When $\Delta w \ll n$ (a typical case) the $\Delta w/n$ overhead of Construction 2 is much better than Construction 1's overhead of 1, and it similarly allows margins for ϵ that are independent of Δw.

C. Efficient construction with no known weights

We now show that efficient codes exist even without requiring to know the weight of $\boldsymbol{x}$. The setup now is that we have two large databases of feature vectors: $\{\boldsymbol{x}\}$ and $\{\boldsymbol{y}\}$. Their weights are *not* known upon their joint measurement, but as usual their weights are in a known interval $[\underline{w}, \bar{w}]$. Our objective now is to be able to calculate the Hamming distance of any pair of vectors where one is from $\{\boldsymbol{x}\}$ and one is from $\{\boldsymbol{y}\}$. To accomplish that, we define the following length 4 vectors

$$X_1 = (0,0,0,1),\ X_3 = (1,1,1,0)$$
$$Y_1 = (1,0,0,0),\ Y_3 = (0,1,1,1)$$

These vectors have the property that $D_H(X_i, Y_j) = 2$, for every $i, j \in \{1,3\}$. Also notice that the index of each vector specifies its weight.

Construction 3. *Given a pair of vectors $\boldsymbol{x}$, $\boldsymbol{y}$, we apply the following encodings:*

$$c_1(\boldsymbol{x}) = \left[\boldsymbol{x} \mid \left\lceil \frac{\bar{w} - w_x}{2} \right\rceil \times X_3 \mid \left\lfloor \frac{w_x - \underline{w}}{2} \right\rfloor \times X_1 \right]$$

to the vector $\boldsymbol{x}$ and

$$c_2(\boldsymbol{y}) = \left[\boldsymbol{y} \mid \left\lceil \frac{\bar{w} - w_y}{2} \right\rceil \times Y_3 \mid \left\lfloor \frac{w_y - \underline{w}}{2} \right\rfloor \times Y_1 \right]$$

to the vector $\boldsymbol{y}$. The left argument of $\times$ is the number of vector copies to concatenate.

The encodings of $\boldsymbol{x}$ and $\boldsymbol{y}$ are identical, except that for $\boldsymbol{x}$ we use the X_1,X_3 vectors and for $\boldsymbol{y}$ we use Y_1,Y_3. Each codeword has $\Delta w/2$ vectors concatenated to the information vector, total of $2\Delta w$ redundant bits. Assuming that $\underline{w}$,$\bar{w}$ and Δw are all even, it is clear that the weight of every encoded vector is $\bar{w} + \Delta w/2$ if its uncoded weight is even, and 1 larger if it is odd. The distance properties of the vectors X,Y guarantee that $D_H(c_1(\boldsymbol{x}), c_2(\boldsymbol{y})) = D_H(\boldsymbol{x}, \boldsymbol{y}) + \Delta w$. Therefore, we have the following.

Theorem 7. *For any pair of vectors $\boldsymbol{x} \in \{\boldsymbol{x}\}$, $\boldsymbol{y} \in \{\boldsymbol{y}\}$, satisfying $\bar{w} - \Delta w \leqslant W_H(\boldsymbol{x}), W_H(\boldsymbol{y}) \leqslant \bar{w}$, if $0 < \epsilon < 1/3$, then the Hamming distance $D_H(\boldsymbol{x}, \boldsymbol{y})$ can be calculated exactly from a single array measurement $\tilde{G}_{c_1(\boldsymbol{x}), c_2(\boldsymbol{y})}$.*

The factor $1/3$ follows from the $+1$ uncertainty in the weights of both $c_1(\boldsymbol{x})$,$c_2(\boldsymbol{y})$. It turns out that Construction 3 has optimal redundancy.

Theorem 8. *Any code allowing Hamming-distance calculation for vector pairs with weight span Δw (and ϵ independent of Δw) requires at least $2\Delta w$ redundant bits.*

Proof: Denote by r the number of redundancy bits added by the encoder. For the lower bound we assume that the encoder for $\boldsymbol{x}$ does not know the specific vectors in $\{\boldsymbol{y}\}$, and vice versa. Knowing the vectors does not make the problem particularly easy, but some pathologies may arise. We refer to the redundancy bits output from the $\boldsymbol{x}$ and $\boldsymbol{y}$ encoders as redundancy vectors, and denote them by $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, respectively. First we observe that any encoder will only use the input vector's *weight* to determine the output redundancy vector. Denote by w the lowest weight of a redundancy vector. To have a constant encoded weight there must be redundancy vectors with weight $w + \Delta w$. Moreover, every pair of redundancy vectors $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$ needs to satisfy $D_H(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = D$, for some fixed integer D. If a vector $\hat{\boldsymbol{x}}$ has different distances from $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{y}}'$, then the measurement cannot be precise because it is not possible to distinguish between a different distance due to the redundancy and due to the information vector itself. So from pairs with $w_{\hat{x}} = w$ and $w_{\hat{y}} = w + \Delta w$ we know that $D \geqslant \Delta w$. Now we consider $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$, each with weight $w + \Delta w$. The number of coordinates where both $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are one equals $w + \Delta w - D/2$. Which means that at least $D/2$ ones of $\hat{\boldsymbol{y}}$ need to fall on zeros of $\hat{\boldsymbol{x}}$. This adds the constraint $D/2 \leqslant r - w - \Delta w$. Finally, from a pair each with weight w we get $D \leqslant 2w$. Combining the three inequalities we obtain $r \geqslant w + \Delta w + \frac{D}{2} \geqslant \Delta w + D \geqslant 2\Delta w$. ■

V. Conclusion

This paper attempts at presenting a first-order analysis and code construction for low-level computation of Hamming similarity. The fact that we assumed noiseless measurements clearly limits the applicability of the results, but we believe that the current view of the problem is conducive to future research that does consider noise. Another interesting direction for future work is to relax the precision of the distance measurement and see the effect of this on machine learning algorithms. In addition, it is desirable to extend the analysis and constructions to resistive memories with non-linearities.

VI. Acknowledgement

This work was supported by the Intel ICRI-CI center, and by research grants from the ISF and the EU Marie Curie.

References

[1] A. Bergamo, L. Torresani and A.W. Fitzgibbon, "PiCoDes: learning a compact code for novel-category recognition," in *'NIPS' 2011*, pp. 2088-2096.

[2] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *Proc. IEEE International Symposium on Information Theory*, pp. 156–160, 2013.

[3] S. Kvatinsky, N. Wald, G. Satat, E.G. Friedman, A. Kolodny and U.C. Weiser, "MRL - Memristor ratioed Logic," *Proceedings of the International Cellular Nanoscale Networks and their Applications*, pp. 1-6, August 2012.

[4] M. Norouzi, A. Punjani and D.J. Fleet, "Fast search in Hamming space with multi-index hashing," in *'CVPR' 2012, IEEE*, pp. 3108-3115.

[5] E. Ordentlich and R.M. Roth, "Low complexity two-dimensional weight-constrained codes", *IEEE Transactions on Information Theory*, 58(6):3892–3899, 2012.

[6] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *'NIPS' 2009*, pp. 1509-1517.

[7] A. Torralba, R. Fergus and Y. Weiss, "Small codes and large image databases for recognition," in *'CVPR' 2008, IEEE*.

[8] P.O. Vontobel, W. Robinett, P.J. Kuekes, D.R. Stewart, J. Straznicky, and R.S. Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, October 2009.