Portland State University

PDXScholar

Dissertations and Theses

Dissertations and Theses

7-13-2020

Memristive Architectures and Algorithms for Approximate Graph-based Inference

Mohammad M.A. Taha
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

Part of the [Electrical and Computer Engineering Commons](#)

Let us know how access to this document benefits you.

Recommended Citation

Taha, Mohammad M.A., "Memristive Architectures and Algorithms for Approximate Graph-based Inference" (2020). *Dissertations and Theses*. Paper 5517.
<https://doi.org/10.15760/etd.7391>

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Memristive Architectures and Algorithms for Approximate Graph-based Inference

by

Mohammad M.A. Taha

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Dissertation Committee:
Christof Teuscher, Chair
Dan Hammerstrom
Jun Jiao
Marek Perkowski

Portland State University
2020

© 2020 Mohammad M.A. Taha

ABSTRACT

The goal of this thesis is to design fast, low-power, robust graph-based inference systems. Our approach to this is by using (1) in-memory computing, (2) approximate computing principles, and (3) memristive devices.

This work is motivated by the fact that conventional von Neumann architectures are not efficient for inference applications, mainly due to the data transfer bottleneck. Adding cache memories and using GPUs is a remedy, however, does not eliminate this bottleneck. In-memory computing is an alternative approach that performs all computations inside the memory, thus eliminating the data transfer bottleneck.

The memristor, which is a passive two-terminal device, is capable of providing storage and computation simultaneously. This capability and its compatibility with CMOS devices make it a good candidate to build compact, low power, in-memory computing systems.

To design graph-based inference systems, we rely on both probabilistic and non-probabilistic graph-based models. Bayesian networks and Markov models are the commonly used probabilistic models for inference applications. They use mutual probabilistic information to connect between the attributes of the data. *Naive Bayes* (NB) model is an approximated Bayesian model used to overcome the intractability of the non-approximated Bayesian models. Zhu presented a *Bayesian Memory* (BM) that can perform NB inference using an *Associative Memory* (AM) and showed that the posterior

probabilities can be calculated using a distance metric, such as the *Hamming Distance* (HD). Although her design showed promising results, it lacked the hardware design and the associative memory used suffered from low capacity and high crosstalks.

Current HD systems suffer from either high latency and/or high-power consumption. We designed an in-memory computing, low-power, approximate, and parallel memristive HD circuit that can be used to implement an *Associative Content Addressable Memory* (ACAM). We showed that the operation of our HD circuit under non-ideal fabrication conditions changes only slightly, decreasing the correct classification rates for the MNIST handwritten digits dataset by $< 1\%$. Also, its operation is independent of the memristor model used, as long as the model allows a reverse current. Because we leverage in-memory parallel computing, our circuit is $n\times$ faster than other HD circuits, where n is the number of HDs to be computed, and it consumes on average $\approx 500\times$ less power compared to other memristive and CMOS HD circuits. Another benefit of this circuit is that it can be controlled to trade-off accuracy to power and vice versa. We then used this HD circuit to design a *Hamming Distance Associative Content Addressable Memory* (HDACAM). This HDACAM consumes on average $\approx 25\times$ lower power than state-of-the-art HDACAMs. We then used the HDACAM to build the BM where we proposed a novel and a simple way to incorporate prior probabilities while adding a 4% power overhead only. We tested this BM on the MNIST dataset and showed that using variable prior probabilities can lead to better performance that exceeds the constant prior probabilities system by approximately 3% with a small increase in the consumed power. We also showed that we could conserve approximately 300 $\times$ the power by using the priors as we can lower the input voltage with a 1% performance decrease only. The BM obtained a 77% classification rate on the MNIST dataset, which is comparable to the 79% obtained by using a univariate Gaussian NB system.

Graph Convolutional Neural Networks (GCNNs) can be used to match and infer on graphs. The computational cost of most GCNN models scales quadratically with the number of nodes. Defferrard et al. proposed a GCNN technique that uses localized spectral filters in the convolutional process and has a lower linear complexity. However, their design had a high computational cost when dealing with larger graphs.

We proposed a novel ensemble multi-attribute graph inference technique that reduces the computation cost and energy/power consumption by reducing the graph sizes. For example, when testing on the Yale faces database we were able to decrease the graph sizes by $\approx 70\%$ and reduce the energy/power consumption by $\approx 60\%$, while maintaining comparable accuracy, with a standard deviation of 0.25, to using the single full-size graph. Compared to other face recognition/inference systems, our techniques outperformed them from the classification rate perspective while consuming about $8\times$ less power.

To further reduce the computational time and cost, we introduced another novel multi-attribute graph inference technique called the *Early Exit* (EE) inference scheme that relies on a confidence measure. By using the EE technique, we were able to lower the power consumption by a further 30% while maintaining a comparable accuracy to our ensemble technique. Our schemes outperformed other face recognition techniques by an average of 9% while consuming $\approx 11\times$ less power.

Our proposed approximate designs are useful for embedded inference applications where low power is of great importance and a slight loss in accuracy is tolerable.

ACKNOWLEDGEMENTS

To begin with, I would like to express my genuine gratitude and thanks to Christof, my advisor, for his help, support, motivation, time, knowledge, and patience during my PhD studies. I learned a lot from him in both my academic and non-academic life.

I would like to thank my committee members for their help and support for my dissertation, and to thank the ECE office for their help during my studies.

I would also like to thank my colleagues Walt, Dat, Muayad, Jens, Neil, and Teuscher-lab members for their help, support, and useful discussions.

In addition, I would like to thank my parents, my brother, my sister, and my wife, for their patience, help, support, and encouragement during my studies.

Finally, I would like to thank the sponsors/funders of this work. This work was partially supported by the National Science Foundation under grants #1028120, #1028378, the *Defense Advanced Research Projects Agency* (DARPA) under award #HR0011-13-2-0015, and C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

TABLE OF CONTENTS

Abstract	i
Acknowledgements	iv
List of Tables	viii
List of Figures	ix
List of Abbreviations	xvii
1 Motivation	1
2 Research Overview	5
2.1 Approximate Graph-Based Inference	5
2.2 Memristive Devices and Architectures	7
2.3 Our Work’s Hierarchy Overview	8
3 Background and Related Work	10
3.1 Background	10
3.1.1 Probabilistic Graphs	10
3.1.1.1 Bayesian Networks (BNs)	11
3.1.2 Non-probabilistic (Conventional) Graphs	14
3.1.2.1 Attribute Related Graphs (ARGs)	14
3.1.2.2 Regional Attribute Graphs (RAGs)	15
3.1.3 Memristive Devices and Networks	16
3.2 Related Work	19
3.2.1 Graph Models	19
3.2.1.1 Probabilistic Graph Models	20
3.2.1.1.1 Algorithmic Level	20
3.2.1.1.2 Hardware Implementation	22
3.2.1.2 Non-Probabilistic Graph Models	22

3.2.1.2.1	Graph Convolutional Neural Networks (GCNNs) and Multi-Attribute Graphs	24
3.2.1.2.2	Multi-layered/attribute graphs	25
3.2.2	Graph Matching	27
3.2.3	Associative Memories	28
3.2.3.1	Neural Associative Memory Models	29
3.2.3.2	Associative Content Addressable Memory (ACAM)	32
3.2.3.2.1	Dot Product metric	32
3.2.3.2.2	Hamming Distance (HD) Metric	32
4	Approximate Memristive In-Memory Hamming Distance and Associative Memory Circuits	41
4.1	Hamming Distance Circuit Design	41
4.1.1	Hamming Distance Evaluation	42
4.2	Results and Discussion	44
4.2.1	Scalability	45
4.2.1.1	Increasing the number of bits per pattern	45
4.2.1.2	Increasing the number of patterns	46
4.2.2	Sparsity	46
4.2.3	Robustness to Device Variability	47
4.2.4	Memristor Model Variability	49
4.2.5	Hamming Distance Circuit Approximation	49
4.2.6	Comparison to Related Work	52
4.2.6.1	Datasets and Full System Implementation	52
4.3	Approximate Memristive Associative Content Addressable Memory	54
5	Bayesian Memory	57
5.1	Bayesian Memory Methodology & Implementation	58
5.1.1	Bayesian Memory Spatial Inference	59
5.2	Results and Discussion	61
6	Multi-attribute Graph Inference	65
6.1	Methodology	65
6.1.1	Graph Generation	66
6.2	Implementation and Results	70
6.2.1	OfA vs OfP	70
6.2.2	Memristive Ensemble Multi-attribute Graph Inference for face detection	70
6.2.2.1	Comparison with other face detection techniques	73
6.2.3	Graph Convolutional Neural Network	75

6.2.3.1	Comparison with other multi-graph combination structures	77
6.2.4	Early Exit Inference	78
6.3	Identifying Relevant Relations/Graph edges, and Attributes	83
7	Conclusion	87
7.1	Future Work	90
7.2	List of Contributions	90
7.3	List of Publications	91
	References	93

LIST OF TABLES

Table 3.1	Summarized comparison between different associative memory models	31
Table 4.1	Proposed HD circuit performance when tested with four different memristor models.	49
Table 4.2	Classification rates obtained when testing our proposed HD circuit versus an HD Python software implementation, and other HD circuits from the literature.	54
Table 4.3	ACAMs types and power consumption comparisons. Estimates to the same storage capacities were performed to facilitate the comparisons.	55
Table 5.1	Prior probabilities of each class as per the training dataset.	59
Table 6.1	Classification rates and consumed power of our work compared to related work.	75
Table 6.2	Classification rates and consumed power of our work compared to related work.	82

LIST OF FIGURES

Figure 1.1	An example on how attributes (Ax) from an image are connected to each other in different domains. Solid black lines demonstrate how they are connected in the spatial domain, the dotted green lines represents the connections in the color domain, the dashed blue lines show the ones in the contrast domain, and so on.	2
Figure 2.1	Overview of the hierarchy and the building blocks of this dissertation. For example, showing the systems designed using the memristor as the building block, and the systems that were designed/used using the GPU as the building block. The missing piece, depicted in red, is to design memristive hardware to implement the GCNN model, this will be done as future work.	9
Figure 3.1	Different BNs variations a) the conventional NB network, b) TAN, c) FAN, this is similar to the TAN but with having some threshold to remove unnecessary dependencies for example, in this case, the dependency between A_3 and A_4 is removed, and d) the K-dependence BN, which states that each attribute can have up to K-attributes as parents.	13
Figure 3.2	(a) A graph consisting of three vertices V_1 , V_2 , and V_3 , with E edges connecting between them. It is worth noting that this is a non-directed graph, so, E_{12} is the same as E_{21} . A_i represents the attributes of each vertex while A_{ii+1} represents the attributes of the edges. In this case the attributes of the edges is the absolute difference of the vertices that the edge connects. For example, $A_{ij} = A_i - A_j $. (b) An image consisting of different textures. (c) Shows an ARG of the image in (b), the vertices of this ARG represent different patches of this image. (d) Shows a RAG where each vertex represents a different texture after the textures in the image in (b) were grouped into 3 bins.	15
Figure 3.3	Memristor fundamentals. (a) The memristor as the fourth two-terminal circuit element ; (b) the resulting hysteresis loop in the I-V plane.	17
Figure 3.4	A conventional memristive crossbar network with memristors located at the intersection between the horizontal and vertical nanowires.	19

Figure 3.5	(a) Shows different two-edged letter “A” one denoted by “+” and the other by “o” being matched by the “shape context” technique [6]. Figure from [6]. (b) Two horse skeletons graphs being matched. Figure from [4].	23
Figure 3.6	(a) Euclidean vs (b) Non-euclidean data. The nodes/features and their connections are represented by the blue circles and black lines respectively.	25
Figure 3.7	(a) Mutli-attribute graphs and their connections (redrawn from [89]). (b) Supra-adjacency matrix. “Matrix # n ” represent the adjacency matrix for “Graph # n ”. “Matrix # nm ” represent the relations between “Graph # n ” and “Graph # m ”.	27
Figure 3.8	A conventional crossbar architecture to compute the DP between an input and multiple stored patterns. The $Out(bias)$ is subtracted from each $Out(i)$ where $i = 1, 2, \dots, n$ to produce the precise DP value.	33
Figure 3.9	Block diagram of a minimum HD classifier using the Hamming network and the MAXNET. The number of classes in this network is p . The MAXNET is a recurrent network that has both inhibitory and excitatory connections. It is considered a <i>Winner-Take-All</i> (WTA) system as it suppresses all the Hamming network outputs but the largest one. The output corresponds to the stored pattern that most closely matches the input. For more details on the MAXNET see [45].	35
Figure 3.10	Hamming distance circuit proposed by Zhu [135]. Redrawn from [135].	37
Figure 3.11	RASSA’s word row [60].	38
Figure 3.12	Hamming distance circuit as implemented in an FPGA [49]	39
Figure 3.13	An accuracy versus power comparison of related work when computing the HDs for 10 256-bit patterns. Ideal circuits would be located in the top left corner.	40
Figure 4.1	An RCN where the memristors are located at the intersection of the horizontal and vertical nanowires. RCNs are used for storing patterns and evaluating the correlation between an input and the stored patterns. The patterns are stored in the columns of the RCN, and the binary input pattern is applied to the RCN rows in the form of voltages with V_{read} or 0 V, representing binaries “1” and “0” respectively. The WTA circuit chooses and progresses the highest crossbar output while terminating the other outputs.	42

Figure 4.2	Two-input analog WTA/MAX circuit. Redrawn from [16]. Since this is a two-input max circuit, it consists of two similar sections M_1 to M_5 and M_6 to M_{10} , the M_1 to M_5 section is enclosed in the red box. This section of this circuit consists of a differential amplifier M_1, M_2 , a mirror active load M_3, M_4 and a voltage follower M_5 . The same applies to the M_6 to M_{10} section. During operation, the differential amplifier corresponding to the highest input voltage will keep a balanced gate-to-source voltages to allow the corresponding voltage follower to be ON. Thus, choosing/-passing the highest/maximum voltage. That said, this circuit chooses the maximum voltage but doesn't output its index. In Section 4.3 we will discuss how can use this circuit along with memristive devices to output the index of the maximum voltage.	43
Figure 4.3	This illustration demonstrates how sneak path currents are leveraged to compute the HD. Only a single column's currents are shown, but each column is independent in this circuit, so the same principle applies to the other columns. When the input pattern is presented to the RCN, current flows both into and out of the RCN columns. When there is a voltage drop at the end of the column V_{d1} , currents will flow from the column to the GND nodes provided by "0"s in the input pattern. The currents flowing through a memristor storing a "1" (thick dotted arrow) are larger than currents flowing through a "0" (thin dotted arrow) because a "0" represents high resistance and a "1" represents low resistance. This is beneficial for computing the HD as currents through a "0" device to the ground (which represents a stored "0" with an input "0") will be smaller than currents through a "1" device (stored "1" to an input "0"). The termination resistance R_T can be omitted when "0" inputs are grounded, and as long as the circuit connected to the RCN induces a voltage drop greater than 0V on the columns.	44
Figure 4.4	The HD calculation accuracy when increasing the number of bits per stored pattern while having a fixed number of stored patterns. The number of stored patterns was fixed to 10.	46
Figure 4.5	The HD calculation accuracy when increasing the number of stored patterns while keeping the number of bits fixed to 256.	47
Figure 4.6	(a) HD calculation accuracy obtained when having a single constant sparsity/density for all the patterns. (b) HD calculation accuracy obtained when having a range of sparsity/density levels for the patterns. The number of bits used was 256.	48

Figure 4.7	The minimum HD between two patterns before they are distinguishable from one another in our Hamming distance architecture. This minimum is a function of the number of bits per pattern and the input voltage. As more bits are added, each bit that differs between the input and stored patterns affects the column's voltage less; as the input voltage is increased, each column's voltage becomes more distinct from columns with a similar (but not equal) HD. (a) Showing the minimum Hamming distance when using 0 to 256 bits and 0.01 V to 0.45 V, (b) Zoom-in on the behavior of the system for voltages between 0.01 V to 0.2 V.	50
Figure 4.8	The minimum measurable HD for different read/input voltages as a function of the number of bits per pattern and the input voltage. Increasing the number of bits decreases the HD circuit's accuracy because each added bit means another added resistive device. Since this architecture statically becomes a voltage divider problem, more resistors mean that each mismatch will produce a smaller change in the column's voltage. As the difference between column voltages falls below the 800 μ V threshold, the minimum measurable HD increases. However, increasing the input voltage increases the architecture's accuracy, as the difference between two columns' voltages again increases beyond the 800 μ V offset voltage.	51
Figure 4.9	Obtained classification rate increasing the input voltage. The classification rates increased with the input voltage because larger input voltages equate to larger differences between column voltages, once the difference between two columns surpasses the comparator's threshold of 800 μ V, the architecture could determine which column was a better match for the input pattern. In other words, increasing the input voltage increased the accuracy and the resolution of the system. The classification rate became constant starting from 0.3 V because the problem ceases to be offset voltage related, but rather becomes dataset-specific. The error bars for the classification rate fluctuated from 2 to 4% because this approach is sensitive to which training patterns were selected to be stored in the crossbar.	52
Figure 4.10	An accuracy versus power consumption comparison of related work when computing the HDs for 10 256-bit patterns. Note that the power of the WTA circuit is not included since the other circuits were simulated without one.	53
Figure 4.11	The full ACAM system. The labels of the stored images are stored in the "Label Crossbar" columns "Label #1" to "Label #n", and the final output is written in the "Output Column". The various HD circuits tested were placed in the "Hamming Distance circuit" box to see their effect on the whole system's performance.	55

Figure 5.1	The proposed BM module consists of (1) the CB, (2) the Inference module, and (3) CPT. When $\hat{X}$ is input, the posterior probabilities of the test vector to the stored vectors are calculated and stored using the CB and the CPT. Afterwards, the posterior probabilities are sent to the inference module to determine and provide the output in a Bayesian sense.	58
Figure 5.2	The blocks of the BM module and their equivalents in simulated hardware. In the simulated hardware, the CB and the CPT are combined in a simple $m \times n$ memristive crossbar. The memristors are located at the intersections of the rows m and columns n . The last row of the crossbar before the “Max and Inferring” circuit carries the weights of the <i>prior</i> probabilities. The weight W of the memristor will represent the probabilities. For example, given a set of probabilities, the highest probability will be represented by $W = 1$ and the lowest one will be represented by $W = 0$, while the in-between probabilities will take different weights between 0 and 1 depending on the number of probabilities and the number of bits represented by the memristor. The outputs of the memristive crossbar are sent to the Max and Inferring circuits, which forms the inference module.	60
Figure 5.3	Confusion matrix obtained when inferring without priors. This illustrates which digits cause the confusion for the stored data. The colors, and the annotated numbers represent how many times the labels were confused with each other.	61
Figure 5.4	The correct classification rates with and without using the priors as a function of input voltage/HD accuracy. “cases 1 & 2” are different scenarios of the outcome of having variable priors.	62
Figure 5.5	The consumed power for both systems at a given voltage. One can see that adding the prior using our proposed memristive method add a 4% more power to total system’s power.	63
Figure 5.6	Classification rates on the MNIST dataset when using our NB method versus a univariate Gaussian NB method. One can see that our proposed NB-like memristive architecture achieved comparable performance to the univariate Gaussian NB method. This asserts the NB functionality of our proposed approximate method.	63
Figure 6.1	(a) Examples from the AT&T dataset. (b) Examples from the the YALE dataset.	66
Figure 6.2	Edges graph where the red dotted rectangles represent the patches of the image. This image has 4 patches. The blue graph shows how the vertices and edges are connected. The attributes for each node V_i are the edges location in their corresponding quadrants.	67

Figure 6.3	(a), (b) Extracted textures with two different set of parameters from the same image. To extract the textures features, we used the <i>Local Binary Patterns</i> (LBP) function that uses modules from Python’s scikit-image. (c), (d) Number of graph nodes, and the value of their attributes for (a) & (b) respectively. From (a) & (b) one can see that more textures were detected for the OfA (b) which caused the OfA to have more (double in this case) number of nodes compared to the OfP. This more information will increase the classification accuracy but will also increase the power consumption because it will increase the graph size. Section 6.2.2 shows how the textures histograms are processed before storing/using them in our proposed systems.	69
Figure 6.4	Classifications rates and power consumption for OfA versus OfP graphs tested on the Yale faces dataset on our memristive design.	70
Figure 6.5	In the training phase, each graph from each type, i.e., edges, colors, and textures, is written, offline, to a column in the crossbar corresponding to it, as mentioned before, see Section 6.2.2. In the inference phase, after the input graph is presented to the crossbar as input voltages, the winning labels are read as the “Final labels”, as explained in Section 4.3. The final winning label is the most occurring one from the final labels according to the majority ensemble, see Fig. 6.6.	72
Figure 6.6	Ensembling example. If the Edges, Colors, and Textures output labels are 2, 3, and 2, respectively, then, the winning label according to the majority ensembling, will be 2.	73
Figure 6.7	Classification rates and consumed power by different graph inference techniques. Single relation inference optimized for either accuracy or power, and multiple relation graph inference. One can see that when using our proposed technique, we were able to obtain comparable classification rates to the OfA one while lowering the overall power consumption by 41%. For the experiment performed here, the flattened edges graphs were 2048 bits long, the textures ones were 196 bits, and the colors graphs were 1530 bits for the proposed technique, for each image in the Yale faces dataset. These sizes were obtained according to the parameters from the PHS algorithm.	74
Figure 6.8	Block diagram of our GCNN ensemble technique. The “Ensembling” box performs ensembling as shown in Fig. 6.6, or Example 6.2.2. The output labels are the labels chosen from each GCNN module. Two ensembling techniques can be used here either by class majority from the output probabilities, as shown in Fig. 6.6 or by averaging the probabilities as in Example 6.2.2. This GCNN ensembling was performed on the RTX 2080 Ti GPU.	76

Figure 6.9 Our ensemble technique classification rates versus consumed energy on the Extended Yale faces dataset B, ran on the RTX 2080 Ti GPU. The relation types used here are color, edge, and <i>Scale Invariant Feature Transform</i> (SIFT) [74] relations. One can see that our proposed techniques achieved comparable classification rates to the OfA while lowering the energy consumption by about 89%.	77
Figure 6.10 Classification rates vs number of nodes for graph technique using the Extended Yale faces dataset B. Our proposed “Ensemble 1” & “Ensemble 2” have the same number of nodes since the only difference is at the final ensembling stage, see Fig. 6.8. One can see that our proposed techniques achieved comparable classification rates to the OfA while decreasing the number of graph nodes by about 86%. N.B. the combined sizes of our proposed techniques mean the total number of nodes of all used graphs. That said, in our techniques, graph inferences are done in parallel and separately as shown in Fig. 6.8, thus the complexity (computation time) remains the same as in a single graph inference.	78
Figure 6.11 Number of graph nodes, classification rates, and energy consumption. The sizes of the bubbles depicts the nodes ratio. One can see that our proposed techniques achieved similar classification accuracy to the OfA while lowering both the number of nodes and energy consumption by 89% and 86%, respectively.	79
Figure 6.12 The energy consumption and the classification rates for our technique versus the “Hetero Technique.” One can see although the “Hetero Technique” outperformed our proposed techniques, ours still consumed 70% less energy. It is worth mentioning that the cumulative number of nodes of our method is equal to the number of nodes of the “Hetero Technique.” That said since our methods work on each relation separately and in parallel, unlike the “Hetero Technique,” this helped in reducing the computation time and cost.	79
Figure 6.13 Domain-wise or early exit algorithm block diagram.	80
Figure 6.14 The CM used here is the Hamming distance. One can see that choosing a low confidence measure helps in reducing the power consumption, by about 40%, but it also lowers the classification rate because it tells the system to exit early even if the mismatch value is high. However, the high confidence measure forces the system to use more information, other graph types, before exiting which helps in increasing the classification rates by 13%, in this case.	81

Figure 6.15	An illustration of the results mentioned in Table 6.2. One can see that our work has outperformed the other works by 7% while consuming $\approx 10\times$, on the Yale faces dataset. The desired system's location here would be at the top left corner, i.e., highest classification accuracy and lowest power. Not all related systems are depicted here due to missing information.	82
Figure 6.16	Our ensemble and EE techniques classification rates versus the consumed energy on the Extended Yale faces dataset B. One can see that our proposed techniques obtained comparable classification rates to the OfA, within 2%, while lowering the energy consumption by $\approx 89\%$. . .	83
Figure 6.17	(a) & (b) HoC for images from the MNIST dataset 7 and 6 classes, respectively. (c) & (d) HoC for images from the Yale faces dataset s_4 and s_6 classes, respectively. One can see that in the case of the MNIST dataset images ((a) & (b)), the HoC didn't provide any useful information to differentiate between the images. On the other hand, for the Yale faces dataset the HoC did. So, one can conclude that the HoC is redundant and irrelevant to be used as an attribute in the MNIST dataset.	84
Figure 6.18	The classification rate vs power consumption when using the HoC as the third relation for both the MNIST and the Yale faces dataset. One can see that for the MNIST dataset adding the HoC not only increased the power consumption but decreased the classification rate as well because the HoC, in this case, didn't add useful information but rather caused confusion in the system. However, it helped in increasing the classification rates for the Yale face dataset by 16% because it helped in differentiating between the patterns.	85
Figure 6.19	Classification-power-ratio for both the MNIST and Yale faces dataset when using 2 and 3 relations. The colors in this figure matches the colors in Fig. 6.18	86

LIST OF ABBREVIATIONS

ADC	Analog Digital Converter
AM	Associative Memory
ANN	Artificial Neural Network
ACAM	Associative Content Addressable Memory
ARG	Attribute Related Graph
BM	Bayesian Memory
BN	Bayesian Network
CAM	Content Addressable Memory
CM	Confidence Measure
CNN	Convolutional Neural Network
DAG	Direct Acyclic Graph
DOM	Degree Of Match
DP	Dot Product
EE	Early Exit
ETAN	Extended Tree Augmented Bayes Network
FAN	Forest Augmented Bayes Network
GCNN	Graph Convolutional Neural Network
GFAN	Global Forest Augmented Bayes Network
GTAN	Global Tree Augmented Bayes Network
HD	Hamming Distance
HDACAM	Hamming Distance Associative Content Addressable Memory
HMM	Hidden Markov Models
HoC	Histogram of Colors

HTM	Hierarchical Temporal Memory
LBP	Local Binary Patterns
MAP	Maximum A Posteriori
NB	Naive Bayes
NBCP	Naive Bayes Constant Prior
NBNN	Naive Bayes Nearest-Neighbour
NBVP	Naive Bayes Variable Prior
OfA	Optimized for Accuracy
OfP	Optimized for Power
PHS	Particle Hill Swarm
RAG	Regional Attribute Graph
RCN	Resistive Crossbar Network
SBN	Simple Bayesian Network
SIFT	Scale Invariant Feature Transform
TAN	Tree Augmented Bayes Network
VGND	Virtual Ground
WTA	Winner-Takes-All

MOTIVATION

Vision, motion control, reasoning, and decision making demand low attention from humans; however, they require extremely high computational resources from conventional computers. The reasons for this are manifold: the mismatch between the algorithm and the underlying hardware, understanding and inferring the input, and the data transfer bottleneck. This bottleneck arises from all the computation being performed in the CPU while the data is stored in the memory. As a result, the data has to be transferred to and from the memory for each computation.

Graphs are powerful and flexible tools for representing structures; they have been shown to benefit image and object recognition applications [29, 109]. However, recognizing objects by understanding the relations between their features is a complex and compute-intensive task [28].

Approximate computing is a promising approach to designing systems with energy and computational constraints that do not necessarily require 100% accuracy. Probabilistic computing is a non-deterministic approach to computation that helps when reasoning and inferring under uncertainty. We argue that combining graphs with approximate and probabilistic computing can be the key to designing graph-based inference models that are hardware-and computational-friendly.

The goals of this work are to propose:

- Novel, low-power, fast, in-memory computing memristive architectures to perform inference tasks.
- Graph-based inference algorithms and models that utilize and take into consideration different types of information and relations, for example probabilistic, spatial, colors, edges, and shapes (see Fig. 1.1) to perform inference.

Our systems will also be capable of trading-off computational requirements and power with performance, and vice versa. These graph-based models will also be easy to map to hardware. Memristors are our chosen type of technology because they are compatible with CMOS technology [18, 125, 128], have zero standby power, can be integrated densely, and can be used as both storage and computation units simultaneously, thus eliminating the data transfer bottleneck [18, 48, 125, 128].

Figure 1.1: An example on how attributes (Ax) from an image are connected to each other in different domains. Solid black lines demonstrate how they are connected in the spatial domain, the dotted green lines represents the connections in the color domain, the dashed blue lines show the ones in the contrast domain, and so on.

Probabilistic graphs have been proposed before for inference applications. The most commonly known probabilistic graphs are Bayesian networks [55] and Markov Models [98]. These types of graphs use mutual probabilistic information to connect between

the attributes of the data [42]. Calculating the mutual information between all the attributes can be intractable due to the exponential explosion of the number of variables being calculated. To overcome this limitation, the *Naive Bayes* (NB) assumes that all attributes are independent, this assumption led to good results. Other variants were also investigated, *Chow and Liu* multinets [22], *Tree Augmented Bayes Network* (TAN) [34], *Forest Augmented Bayes Network* (FAN) [54,75], *Global Tree Augmented Bayes Network* (GTAN) [53], *Global Forest Augmented Bayes Network* (GFAN) [53], *Extended Tree Augmented Bayes Network* (ETAN) [26,27], and *Simple Bayesian Network* (SBN) [93].

Graph Convolutional Neural Network (GCNN) are models that were proposed [29,64] to match and classify graphs. These models are a modification of the *Convolutional Neural Network* (CNN) techniques to deal with graphs.

The models mentioned above suffer from high computational costs and require high energy/power resources due to the large graph sizes and the lack of low-power, fast hardware designs.

Our research:

- (a) introduces novel approximate graph-based inference algorithms that model different relations between the attributes of images,
- (b) introduces ways to lower the memory and computation requirements,
- (c) introduces different ways to implement these models in low-power memristive hardware,
- (d) introduces a novel, simple, fast hardware design to perform NB-like inference,
- (e) introduces an approximate, in-memory, scalable, low power Hamming distance design,

-
- (f) introduces a fast, in-memory associative content addressable memory design, and
 - (g) investigates the effect of device-to-device variations on these designs.

Our work provides new methods to perform approximate and probabilistic inference using graph-based models that overcome some of the issues that are faced by current systems. Compared to other approaches, our systems will best suit image recognition applications with constrained power budgets, such as in embedded systems. Also, they can be used in robots and drones applications where image recognition is required.

RESEARCH OVERVIEW

In this chapter, we will give a brief overview of the proposed research in order to provide the reader with an idea about the problems we are addressing, and the questions to answer.

2.1 Approximate Graph-Based Inference

Graphs are used in image and video recognition; however, they suffer from some issues, for example, learning the perfect graph structure given the entire data, i.e. learning the graph that best represent the entire data, has been proven to be an NP-hard problem [20]. Also, they are compute-intensive, which produces power-hungry hardware systems [108]. There are two main types of graph-based models probabilistic, e.g. Bayesian networks, and non-probabilistic ones.

A broad class of probabilistic inference approaches is based on Bayesian inference. However, Bayesian approaches have a high computational cost [94, 108]. As a result, researchers had to come up with assumptions to lower the computation needed. One of the primary assumptions is called the NB, which assumes that the features/attributes of the input are independent.

Work has been done to remove the NB assumption and to take more dependencies between the attributes. Some tried to add more dependencies in the Bayesian network and

2.1. APPROXIMATE GRAPH-BASED INFERENCE

calculate those dependencies using the joint conditional probability mutual information rule [53, 75]. Graph models have also been used in a non-probabilistic way in image recognition applications. These models used the spatial relations between the features to perform the recognition task [6, 110].

The first part of this dissertation will explore and answer the following questions:

- Q1** Will modeling several and different types of relations between the features/attributes improve the inference process/accuracy and reduce the complexity/computational cost of the system?
- Q2** How can we know what the relevant types of relations are given the input data? Furthermore, how can we formalize them to be data type dependent? The answer to this question can lead to an NP-hard problem. If that is the case, then we will try to approximate the algorithm from a mathematical perspective and/or start with using a small number of attributes, for example.
- Q3** How can we identify weak and redundant relations that negatively affect or do not enhance the inference process?
- Q4** What effect will removing these weak and redundant relations have on the computational complexity/cost and the performance of the model?
- Q5** What type of thresholding or weighing techniques can be used to remove these weak dependencies/relations?
- Q6** How can the associative memory, since it already can perform inexact inference, be adapted to perform inexact whole and sub-graph inference/matching? Note: The sub-graph matching was not investigated because it was found to be out of scope of this dissertation.

2.2 Memristive Devices and Architectures

Traditional von Neumann architectures have separate computing and memory units, which requires the data to be transferred between them to perform the required computations. This leads to a bottle-neck in data-heavy applications, such as image and object recognition, because they require large amounts of data to be transferred back and forth to be processed. This limitation not only slows down the system but also wastes power [120]. Cache memories and GPUs were used to overcome this limitation; however, they could not completely resolve it [51].

Memristive architectures, e.g. memristive crossbars, were proposed to solve this limitation because memristors are passive two-terminal nano-scale devices that are capable of both storage and computation [48, 96, 105]. Memristive crossbars are widely used in the neuromorphic engineering and machine learning fields because they allow dense storage with in-memory computing capabilities.

The second part of this dissertation will focus on designing memristive architectures for the proposed graph-based algorithms and answering the following questions.

Q7 How can we utilize/use memristive crossbars to perform probabilistic inference, with low power consumption?

Q8 How do these probabilistic systems scale?

Q9 How can memristive crossbars be utilized to model the dependencies/relations between the attributes?

Q10 Are there other architectures that can be utilized to build such systems?

Q11 How can we adapt these memristive architectures to perform inexact graph infer-

ence/matching?

Q12 What is the effect of device variations on the performance of such systems?

2.3 Our Work's Hierarchy Overview

Fig. 2.1 depicts the hierarchy of this dissertation showing the building blocks used to design low power and approximate graph-based inference systems. For example, one can see that we used the memristor as the building block to design the HD circuit. We then used this HD circuit to design the HDACAM, which was in turn used to design the memristive graph-based systems and the probabilistic graph-based inference system, i.e. the “NB System.” As for designing and simulating the GCNN systems, we used the GPU as its building block. The part depicted in red, at the right side of the memristor's box, means the usage of the memristor to build the GCNN models. However, this part is still missing and will be done as future work.

2.3. OUR WORK'S HIERARCHY OVERVIEW

Inference				
Memristive Graph-based Ensemble System	Memristive Graph-based <i>Early Exit</i> (EE) System	Memristive <i>Naive Bayes</i> (NB) System	GCNN Ensemble	GCNN <i>Early Exit</i> (EE)
<i>Hamming Distance Associative Content Addressable Memory</i> (HDACAM)				
<i>Hamming Distance</i> (HD) Circuit			<i>Graph Convolutional Neural Network</i> (GCNN)	
Memristor			<i>Graphical Processing Unit</i> (GPU)	

Figure 2.1: Overview of the hierarchy and the building blocks of this dissertation. For example, showing the systems designed using the memristor as the building block, and the systems that were designed/used using the GPU as the building block. The missing piece, depicted in red, is to design memristive hardware to implement the GCNN model, this will be done as future work.

BACKGROUND AND RELATED WORK

In this chapter, we will discuss the probabilistic and non-probabilistic (conventional) graph-based models and the related work. We will also present and discuss memristive crossbars, and Hamming distance, and *Associative Content Addressable Memories* (ACAMs) related work.

3.1 Background

3.1.1 Probabilistic Graphs

Probabilistic graphs are types of graphs that use probabilistic dependencies to represent their relations/edges. There are different types of probabilistic graphs, such as *Bayesian Networks* (BNs), and *Hidden Markov Models* (HMMs). Here we will focus on the BNs because they are more commonly used in image/spatial pattern recognition than the HMMs.

To better understand the upcoming sections, one needs to learn about joint and conditional probabilities. The joint probability is the likelihood of more than one event occurring at the same time. The joint probability for two events A and B is expressed mathematically as $P(A, B)$. As for the conditional probability, it is a measure of the event B occurring given that event A occurred, and it is expressed as $P(B|A)$. It can be calculated using Eq. (3.1).

$$P(B|A) = \frac{P(B \cap A)}{P(A)} \quad (3.1)$$

In probability theory, the chain rule allows the calculation of the joint probability distribution using conditional probabilities, see example in Eq. (3.2).

$$P(A, B, C, D) = P(A|B, C, D) * P(B|C, D) * P(C|D) * P(D) \quad (3.2)$$

3.1.1.1 Bayesian Networks (BNs)

BN is a probabilistic graph model that perform Bayesian inference, using Bayes' theorem see Eq. (3.3) [91]. A BN consists of a set of conditional probabilities expressed in a *Directed Acyclic Graph* (DAG) where the graph nodes represent variable attributes and the edges are the dependence or independence between these attributes [8, 9]. The objective of the BN classifier is to predict the class C of a test instance X with attributes $\{a_1, \dots, a_n\}$, given some trained classes C , by using Bayes' theorem as shown in Eq. (3.4).

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{\int_y P(Y) \times P(Y|X) dY} \quad (3.3)$$

$$\operatorname{argmax}_C P(C|a_1, \dots, a_n) = \operatorname{argmax}_C \frac{P(a_1, \dots, a_n, C)}{P(a_1, \dots, a_n)} \propto \operatorname{argmax}_C P(a_1, \dots, a_n, C) \quad (3.4)$$

$P(a_1, \dots, a_n, C)$ can be rewritten using the joint probability chain rule as in Eq. (3.5).

$$P(a_1, \dots, a_n, C) = P(C)P(a_1|C)P(a_2|a_1, C) \dots P(a_n|a_1, \dots, a_{n-1}, C) \quad (3.5)$$

This classifier is of a generative type, meaning that it learns the model of the joint

probability $P(X, Y)$ that generates the data and then makes its prediction by using the Bayes' rule Eq. (3.3) [34, 85]. For example, in the MNIST dataset, one would fit each class separately with a probability distribution, then to classify a new pattern, one would find the class that the input pattern is more probable to come from.

If the Bayesian classifier implements Eq. (3.5), then the model will be optimal because all the conditional dependencies between all the attributes were taken into account. However, without some assumptions, this has been proven to be an NP-hard problem [20]. Moreover, the integral part in the denominator of Eq. (3.3) can be intractable. The simplest form of the BN is the NB, where each attribute a_i , with $i = 1, \dots, n$, has only one parent, class C (see Fig. 3.1(a)). This leads to an approximate Bayes rule, see Eq. (3.6). Although the NB assumption simplifies the calculation of the conditional likelihood probability and reduces the computational complexity of the system, it is an unrealistic assumption [34, 53], but it gave good results. There have been several variations of assumptions over the BNs which take some conditional dependencies into consideration when learning. For example the *Chow and Liu* (CL) multinets [22], *Tree Augmented Bayes Network* (TAN) [34], *Forest Augmented Bayes Network* (FAN) [54, 75], *Global Tree Augmented Bayes Network* (GTAN) [53], *Global Forest Augmented Bayes Network* (GFAN) [53], *Extended Tree Augmented Bayes Network* (ETAN) [26, 27], and *Simple Bayesian Network* (SBN) [93]. All of the TAN variations of the BN classifiers are assuming that each attribute has not more than two parents where one of them must be the class, as shown in Fig. 3.1(b) and 3.1(c). Some researchers have investigated the use of multiple dependencies (K-dependencies) for a BN and devised learning algorithms for these kind of networks [124], see Fig. 3.1(d).

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)} \quad (3.6)$$

Figure 3.1: Different BNs variations a) the conventional NB network, b) TAN, c) FAN, this is similar to the TAN but with having some threshold to remove unnecessary dependencies for example, in this case, the dependency between A_3 and A_4 is removed, and d) the K-dependence BN, which states that each attribute can have up to K-attributes as parents.

The *Maximum A Posteriori* (MAP) is commonly used algorithms with NB networks [90]. In MAP, the NB network tries to maximize the posterior probability $P(Y|X)$ in Eq. (3.6) in the inference process. Let Y be the stored vector, and X be the test vector. The MAP algorithm then consists of performing the following operations:

$$MAP = \underset{Y}{\operatorname{argmax}} P(Y|X). \quad (3.7)$$

argmax is an operation that finds the points/arguments that maximizes a given function.

Substituting Eq. (3.7) in Eq. (3.6) leads to

$$MAP = \underset{Y}{\operatorname{argmax}} \frac{P(X|Y) \times P(Y)}{P(X)}. \quad (3.8)$$

$P(X)$ can be neglected because it is used as a normalizing factor. This leads to

$$MAP = \underset{Y}{\operatorname{argmax}} P(X|Y) \times P(Y), \quad (3.9)$$

where $P(X|Y)$ and $P(Y)$ are called the likelihood and prior probabilities, respectively. $P(Y)$ can be calculated using a specific belief or according to the number of occurrences of the patterns per a specific class. Zhu [134] calculated the $P(X|Y)$ using a the Hamming distance metric in her proposed Bayesian associative memory.

3.1.2 Non-probabilistic (Conventional) Graphs

A graph $G = (V, E, A)$ consists of a set of vertices “ V ,” edges “ E ,” and attributes “ A ” as shown in Figs. 3.2(a) and 3.2(c). For example, in an image, the vertices, the edges, and the attributes may represent its features, the relation between the features, and the values associated with the vertices and edges, respectively. Two common graph types are 1) *Attribute Related Graphs* (ARGs), and 2) *Regional Attribute Graphs* (RAGs).

3.1.2.1 Attribute Related Graphs (ARGs)

A graph is considered an ARG when the attributes represent the vertices and the edges. The attributes for vertex V_i can be denoted as a vector $A_i = [A_i^{(m)}]$, where m is the number of attributes for vertex V_i . The same can be applied for the attributes of the edges. In Fig. 3.2(a), each vertex and edge has a single scalar attribute.

Figure 3.2: (a) A graph consisting of three vertices V_1 , V_2 , and V_3 , with E edges connecting between them. It is worth noting that this is a non-directed graph, so, E_{12} is the same as E_{21} . A_i represents the attributes of each vertex while A_{ii+1} represents the attributes of the edges. In this case the attributes of the edges is the absolute difference of the vertices that the edge connects. For example, $A_{ij} = |A_i - A_j|$. (b) An image consisting of different textures. (c) Shows an ARG of the image in (b), the vertices of this ARG represent different patches of this image. (d) Shows a RAG where each vertex represents a different texture after the textures in the image in (b) were grouped into 3 bins.

3.1.2.2 Regional Attribute Graphs (RAGs)

A RAG is an ARG where the vertices represent regions, and the edges represent the connections/relations between them, see Fig. 3.2(d). The vertices attributes are set according to the properties or characteristics of the region, such as colors or textures. In Fig. 3.2(d), the vertices attributes would be set according to the textures.

3.1.3 Memristive Devices and Networks

The realization of the memristor in hardware aided pattern matching and inference applications by combining the needed storage and computation elements into a single device [96, 105]. Memristors are generally considered good candidates for these kinds of applications because their dynamic resistance can be exploited to perform analog operations. Using memristors to build our systems will help in designing fast and low power systems. Chua [23] first postulated the memristor in 1971 as the fourth fundamental electronic circuit element in addition to the resistor, capacitor, and inductor. In 2008, HP successfully fabricated a memristor [114]. The memristor is characterized by the relationship between charge and flux Fig. 3.3 (a) [114].

$$M(q) = \frac{d\phi}{dq} \quad (3.10)$$

Memristors can be categorized in several ways. For example, there are binary [69], analog [81], volatile [130], or non-volatile [57, 81] devices, to cite a few. A main difference between binary and analog memristors is determined by how fast the memristor can switch from lowest resistance (R_{ON}) and highest resistance (R_{OFF}) and vice versa, and what in-between states can be achieved by the voltage applied. Binary memristors have only two states, representing 0 and 1, while analog memristors can have several states. An example of a binary memristor is presented in [70], and an analog one in [81, 130]. Memristor reading and writing times are crucial parameters to consider when choosing a memristor for a specific application. A comparison between 14 memristor models can be found in [127]. There is another type of binary memristor known as a “self-rectifying memristor.” This memristor acts as a resistive switch with a diode-like behavior, thus eliminating the sneak path current by suppressing the reverse current flow to 0.1 pA [61].

(a) Fundamental circuit elements. Figure from [114]. (b) Typical I-V plot for an analog memristor. Created from the model presented in [99].

Figure 3.3: Memristor fundamentals. (a) The memristor as the fourth two-terminal circuit element ; (b) the resulting hysteresis loop in the I-V plane.

In other words, in a rectifying memristor a positive voltage program it into a more conductive state, while a negative voltage program it into a more resistive state while suppressing the reverse current to a negligible value [62, 70].

Most memristors are based on a metal-oxide material such as TiO_2 [127], and their weight is called “memristance,” which is defined by the distribution of the oxygen vacancies. The memristance of the memristor is changed by the amplitude of the applied voltage and its application duration. Thus, the memristor equation can be written in terms of voltages and currents in time, as shown in Eq. (3.11). The memristance has two boundaries, namely R_{ON} and R_{OFF} , representing the lowest and highest memristance, respectively. The range between these boundaries changes between different memristor models.

$$M(q) = \frac{d\phi/dt}{dq/dt} = \frac{V(t)}{I(t)} \quad (3.11)$$

Several research groups focused on creating memristor models with improved computational speed and power efficiency, many of which proposed their own models of device behavior depending on physical models or purely theoretical ones. For example *Berdan* [7], *Merrikh-Bayat* [81], *Yang* [130], *Jo* [57], *Miao* [82], *Miller* [83], *Oblea* [86], and *Jo & Lu* [58] published models with parameters fit to physical models. While *Batas* [5], *Bi-olek* [11], *Eshraghian* [32], *Lehtonen* [69], *Pershin* [92], and *TEAM* [67] published purely theoretical or curve-fitted to models from other research groups. We published a detailed survey comparing all of these models in [127].

A conventional topology for memristive circuits is a nanowire crossbar structure, where the memristors are located at the intersection between the horizontal and vertical nanowires, as shown in Fig. 3.4. The compatibility of memristors with state-of-the-art CMOS devices allows for dense memristive crossbar networks to be used alongside traditional circuitry providing circuit blocks that both store information and compute several different functions based on that information. Such networks are beneficial for pattern matching applications, where a correlation between input and previously stored data needs to be computed [48, 105, 113]. This correlation is called *Degree of Match* (DOM). Such networks are also known as *Resistive Crossbar Networks* (RCNs).

In a memristive crossbar, where the input patterns are presented as input voltages to the crossbar rows, the current flows through the memristor with conductance g_{ij} is $V_i \times g_{ij}$, where i and j denote the row and column number respectively. As a result, the total current flowing through one column is $\sum_i V_i \times g_{ij}$. For example, $Out(1)$ is equal $\sum_{i=1}^m V_i \times g_{i1}$ to The total output current from each crossbar column represents the correlation between the

Figure 3.4: A conventional memristive crossbar network with memristors located at the intersection between the horizontal and vertical nanowires.

input and each of the stored patterns. Hence, the memristive crossbar can perform a direct evaluation of the correlation between the analog input and the stored patterns, with the best matching pattern having the highest $\sum_i V_i \times g_{ij}$. There is an undesired phenomenon that memristive crossbars suffer from undesired currents, known as the “sneak path current,” where current flows in unwanted paths thus altering the crossbar’s output. Several solutions have been proposed to solve this issue by adding additional devices, such as diodes [76], transistors [63], complimentary memristors [59], and AC sensing [97].

3.2 Related Work

3.2.1 Graph Models

In this section, we will present the work that has been done in this area, so we can know how our work compares to it. Graph-based models can be divided into two main types.

1) Probabilistic ones, e.g., BNs, and HMM, and 2) non-probabilistic ones that use spatial or other relations instead of probabilistic ones.

3.2.1.1 Probabilistic Graph Models

3.2.1.1.1 Algorithmic Level

HMMs are commonly used to model time-series data and in speech recognition applications. BNs, however, are used in modelling or representing conditional dependencies between variables or attributes [42]. Since we are working with dependencies and spatial data in this work, we will focus more on BNs. Knowing the related work that has been in this area will give us an idea of what problems are there and how they can be solved. The main idea of the BNs is that the nodes in the network represent the attributes/features of the learned data, and the connections between them represent the probabilistic conditional and mutual information. NB is the most commonly used variant of the BNs.

Zhu [134] presented a Palm associative memory model that can perform NB inference and called it the *Bayesian Memory* (BM), and showed that the “likelihood probability” $P(X|Y)$ from Eq. (3.6) could be calculated using a distance metric. This BM was constructed hierarchically and represented the probability distribution of the input space. This probability contained information that was defined by Pearl in 1988 [91]. *Hierarchical Temporal Memory* is another hierarchy model that represents the probability, in the same way, [39, 40]. Our work, shown in Chapter 5, will use the idea of using the associative memory to perform NB inference. However, we will use a different type of associative memory namely the ACAM because Zhu’s BM did not scale well as the Palm associative memory model suffers from high errors due to high cross-talks when more data is stored.

Jayech and Mahjoub [53,54] proposed a system that divides the image into blocks and clusters them according to the color and texture features, the blocks are then clustered using k-means. Afterwards, they performed Bayesian structural learning to create the BN using the “Structure Learning Package” from [72]. For the TANs, they calculate the mutual information (attribute dependencies) using Eq. (3.12) from [35].

$$I_p(A_i, A_j|C) = \sum_{\substack{x \in A_i \\ y \in A_j \\ z \in C}} P(x, y, z) \log \frac{P(x, y|z)}{(P(x|y)P(y|z))}, \quad (3.12)$$

where A_i and A_j are sets of feature variables and C is the set of class variables. And x, y and z are the values of variables X, Y , and Z respectively. The I_p function measures the mutual information provided by A_i and A_j for each other when C is given. They also introduced another variant called *Forest Augmented Naive Bayes Network* (FAN), where edges of the TAN that are lower than a certain threshold are removed. They calculated the threshold by averaging the mutual information I_p . They tested their TAN on the ORL face database [3]. Their results showed that the TAN performs worse than the NB, 42% compared to 66% classification rates, and when using the FAN they obtained 57% [54]. The FAN performs better because they remove some edges that were added by the TAN thus moving the whole network towards the NB.

In [53], Jayech and Mahjoub used a technique, called “*Tangent Distance*” (TD), in their network development phase. The TD is a mathematical tool that compares two images with taking into consideration small transformations such as rotations, etc. When they used this TD in their network development, they were able to improve the classification accuracy to about 81% compared to previous networks without using the TD.

The work in [53, 54] shows that not only the NB approximation helps in reducing

the system's requirements but can still outperform other more complicated algorithms. It also proves that using other relations along with the probabilistic ones can improve the performance of the system.

3.2.1.1.2 Hardware Implementation

Deshpande [30] derived a simple model from HTM and implemented it on an FPGA. Their Bayesian module depended on using a combination between *Principle Component Analysis* (PCA) [46] and *Vector Quantization* (VQ). Although the FPGA implementation showed a promising speed-up compared to a software simulation, no power evaluations were provided, and the design was also FPGA-specific.

Bill [10] showed that a compound memristive synapse model could be used to perform inherited Bayesian-like inference by using *Spike-Timing Dependent Plasticity* (STDP) and Softmax. Fan *et al.* [33] implemented HTM's inference part by using spin-neurons. Their system is more energy-efficient compared to its equivalent 45nm CMOS ASIC design, yet, they did not provide any performance evaluation of the design.

To the best of our knowledge, none have shown a low-power memristive architecture that performs NB inference while considering the prior probabilities.

3.2.1.2 Non-Probabilistic Graph Models

These types of graph models do not use probabilistic relations while connecting between the attributes; instead, they use spatial relations. They are commonly known as shape matching models.

There are two approaches for graphical shape matching 1) feature-based, and 2) brightness based. Featured-based approaches rely on using the spatial information between the extracted features such as edges or junctions [6, 122]. Some approaches used silhou-

Figure 3.5: (a) Shows different two-edged letter “A” one denoted by “+” and the other by “o” being matched by the “shape context” technique [6]. Figure from [6]. (b) Two horse skeletons graphs being matched. Figure from [4].

ettes [37, 106, 132]. Since silhouettes are limited, as they ignore internal features and are sometimes difficult to extract from the real image, other approaches were proposed that treat shapes as a set of points in the 2D image [36, 50]. For example, finding the key points in an image and recognizing it using the spatial arrangement of these points [2]. Nevertheless, using the key points, such as edges, alone wastes the information from the smooth parts of the objects, for example, a curve.

Belongie *et. al* [6] presented an approach to measure the similarity between the objects’ shapes and use this similarity for object recognition, they call it the “shape context” approach, as shown in Fig. 3.5(a).

Bai *et. al* [4] proposed an approach for shape retrieval and classification based on geodesic paths in skeleton graphs, where the skeleton graphs were formed from the spatial relation between the nodes, edges, and features on the image, as shown in Fig. 3.5(b). Other approaches, brightness-based, used the brightness values of the pixels to perform shape matching [24]. However, these approaches do not perform well under distortion from illumination changes [6].

Mathuria and Hammerstrom [77] presented an approximate graph matching tech-

nique that uses hierarchical graph constructions and *Sparse Distributed Representation* (SDR). By using their technique, they were able to perform approximate graph matching in $O(1)$ time compared to the non-polynomial time that classic techniques take. Their work showed that combining biological and neuromorphic techniques with graphs can lead to improvements, in this case, speedups.

3.2.1.2.1 Graph Convolutional Neural Networks (GCNNs) and Multi-Attribute Graphs

Deep learning is currently the state-of-the-art when dealing with Euclidean data, such as an image or a video frame where the pixels or the features form a grid, see Fig. 3.6(a). However, when it comes to non-Euclidean data, like graphs, see Fig. 3.6(b), deep learning techniques are not the most efficient. For example, in e-commerce or social media, graph-based learning is used to be able to exploit the interactions between users and products or users and other users to make the best recommendations. The irregular graph structures imposed some notable challenges on current machine learning algorithms. GCNN generalizes the image/data convolution operation to graph data where we need to learn a function f to generate a node v by its features and neighbors. There are multiple types of GCNNs such as 1) GCNN with pooling where the GCNN layer is followed by a pooling layer to coarsen a graph into sub-graphs to have higher-level graph representations [29], 2) graph auto-encoder with GCNNs [64]. In the auto-encoder type, an encoder uses the GCNN layers to obtain suppressed representations for each node. A pair-wise distance between these nodes is then computed by a decoder which reconstructs the graph adjacency matrix after applying a non-linear activation function.

Defferrard *et al.* [29] proposed a GCNN technique that uses localized spectral filters in the convolutional process. These filters are convolutional filters that are defined in

Figure 3.6: (a) Euclidean vs (b) Non-euclidean data. The nodes/features and their connections are represented by the blue circles and black lines respectively.

the spectral domain. Using spectral filters is costly due to $O(n^2)$ multiplication with the graph Fourier basis. However, they overcame this issue by parameterizing their filters as a polynomial function, Chebyshev expansion, that can be computed from the graph Laplacian, the Laplacian matrix of the graph. By doing so, they were able to achieve a lower linear complexity of $O(K|E|)$ instead of $O(n^2)$ where K , E , and n are the spectral filter size, number of graph edges, and number of graph nodes respectively. Still, the computation complexity can be high when dealing with larger graphs.

We hypothesize that by using multi-attributed and multi-relational graphs, we can decrease the graph size, thus decreasing the overall complexity.

3.2.1.2.2 Multi-layered/attribute graphs

These graph algorithms integrate several attributes to construct a single combined attribute. Most of these algorithms integrate the attributes in two different ways, namely 1) attribute-level integration, and 2) affinity-level integration. The attribute-level integration combines multiple attributes at the attributes level step to establish a new single attribute that contains all the information from the others. The Histogram-Attributed Re-

lational Graph (HARG) by Cho *et al.* [21] adopts this approach. This approach uses a histogram-based approach for edges and nodes, where the edges are defined using the log-polar histogram that is constructed by concatenating the log-distance and polar-angle histograms. The other type, namely the affinity-level integration, is performed at the affinity computation step. To perform this integration, the affinity value for each attribute type is computed first, and then all the computed affinity values are integrated to construct a single affinity value. For example, Yan *et al.* [129] and Zhou, and De la Torre [133] compute two affinity values based on the differences in the distances and angles between the vertices. Then, they linearly integrate the values to compute the unified affinity.

Park *et al.* [89] claim that these integration methods suffer from some problems. Firstly, the attribute-level integration is performed in a simple manner, so the information from these several attributes are often not completely utilized. Secondly, the integration method needs to be customized according to the target applications as the applications that are formulated as graph matching problems have diverged characteristics. Thirdly, when using multiple attributes, not all attributes are helpful, and erroneous or improper attributes should be determined and excluded. However, selecting the unhelpful attributes depends on the target applications. For example, the attributes that are unhelpful in one application might be important for another. As a result, Park *et al.* [89] proposed a multi-attribute graph structure matching technique that uses random walks.

In their proposed structure, they form a supra-adjacency matrix, which consists of two block types.

1. The intra-layer adjacency matrices, which describe the single graphs (depicted in black in Fig. 3.7).
2. The inter-layer matrices, which are all-ones diagonal matrices that describe the

relations between the different single graphs (depicted in red in Fig. 3.7).

Figure 3.7: (a) Mutli-attribute graphs and their connections (redrawn from [89]). (b) Supra-adjacency matrix. “Matrix # n ” represent the adjacency matrix for “Graph # n ”. “Matrix # nm ” represent the relations between “Graph # n ” and “Graph # m ”.

Their technique showed promising results compared to others. However, their method still suffered from scalability issues due to the need for adding the inter-layer matrices. Using these inter-layer matrices in addition to the intra-layer ones increases the size of the supra-adjacency matrix exponentially, which increases the memory requirements and computation time.

3.2.2 Graph Matching

After extracting and learning the graph, one needs to perform inference. In graph-based models, this process is commonly known as “graph matching,” a technique to compare two graphs, for example, an input and a reference graph. Graph matching algorithms can be divided into the following categories:

- **Exact graph matching algorithms:** The mapping of nodes and edges of the input graph has to match a reference graph exactly. This can be a tough constraint to deal with [25, 123].

- **Inexact graph matching algorithms:** This type of graph matching technique allows some deformations to be allowed when comparing graphs [25, 103].
- **Graph embeddings and graph kernels:** This technique works by converting the graph into a low dimension space so that it can be computed efficiently [15].

3.2.3 Associative Memories

An *Associative Memory* (AM) is a storage circuit that mimics the low-power parallel capabilities of the human brain in storing and retrieving data [71]. The operation of the AM depends on associating incoming data with already stored data [65]. This stands in contrast to a conventional *Random Access Memory* (RAM) where the location of the stored data must be known in order to retrieve the desired data. In an AM, the extra processing of finding the closest matching data is performed in the memory itself, which increases retrieval speed and makes the system more energy efficient [88]. The operation of the AM helps when the input vector is incomplete or noisy, as it will still be able to retrieve the vector that corresponds to the original input vector [52].

AMs have been introduced to the research in two ways, 1) The neural AM, which requires neural models such as Hopfield, Willshaw and Palm models, which will be discussed later. These types of AMs require matrix multiplication in their retrieval process. 2) the *Associative Content Addressable Memory* (ACAM) which is a generalization of the conventional *Content Addressable Memory* (CAM) functionality, where it output data instead of the data's location. The retrieval of the stored vectors in these AM can be done in several ways. For example one can use the *Euclidean Distance* (ED) or the *Hamming Distance* (HD) [1]. These distance-measuring methods help in the “best-match” procedure where the associative memory retrieves and outputs the vector that most closely matches

with the input vector.

3.2.3.1 Neural Associative Memory Models

In the 1960s the first neural associative memory model called *Die Lernanmatrix* was introduced by Steinbuch [112]. This type of AM stores its vectors in a weight matrix. The input and output vectors of the *Die Lernanmatrix* are both binary vectors, and the Hebbian learning rule adjusts their weights. This rule is applied to store all the mappings in the network by creating the weight matrix W . This binary weight matrix can be implemented by:

$$W = \bigvee_{\mu=1}^M [x^{\mu} \cdot (y^{\mu})^T] \quad (3.13)$$

Where $\mu = 1, 2, \dots, M$ is the number of mappings, in other words, the number of training patterns. x^{μ} is the μ^{th} training pattern and y^{μ} is its mapping. The output pattern is retrieved in the following manner:

$$\hat{y} = f(W \cdot \hat{x} - \theta) \quad (3.14)$$

Where $\hat{y}$ and $\hat{x}$ are the output and testing pattern, respectively. And θ is a threshold which is determined by the *K-Winner Takes All* K-WTA rule. Where K is the number of active nodes in an output vector. Therefore, the threshold θ is set, so that only those nodes that have the K maximum sums are set to “1”, and the remaining nodes are set to “0”.

The *Die Lernanmatrix* model performance was further studied by Willshaw [126] and Palm [87]. The Willshaw group [14, 44, 126] developed their associative network from modifying the *Die Lernanmatrix*. Their associative net is mostly the *Die Lernanmatrix* but with higher information storage capacity. However, the problem with this model is that its performance is lower if the number of non-orthogonal training patterns increases because

the crosstalk increases [101]. The Palm and Willshaw models follow the Hebbian learning rule and retrieving rule (Equations Eq. (3.13), Eq. (3.14), respectively), and are examples of the AMs that use matrix multiplication in their retrieval process.

Zhu [134] argued that the Palm net is simpler and more reliable with higher information storage capacity than the Willshaw net. Palm [87] modelled the AM network as a communication channel and the information stored in the network is defined by the knowledge gained from the retrieved results in the network, that is, it has the iterative updating feature. Zhu also claimed that the Palm model is more noise-robust, fault-tolerant, and return the best match value given the input than the Willshaw model.

John Hopfield invented the *Hopfield Network* model in 1982 [47]. He devised this model on his understanding of the human brain. This network is a form of a recurrent artificial neural network and an auto-associative memory. He named the processing devices/units as neurons where each neuron has two states, binary units, following what was mentioned by McCullough and Pitts [80]. The states of the units were determined by whether the unit's input exceeds or falls behind a certain threshold. In the *Hopfield Network* an energy function is proposed to compute the symmetrical synaptic connections/weights of the recurrent network. However, there are some restrictions in this network:

- No unit has a connection with itself.
- Connections/Weights are symmetrical.

The *Hopfield Network* suffers from some problems such as firstly, its capacity is relatively low as compared to other models, namely Palm and Willshaw. Secondly, it has and vulnerable to spurious attractors. Finally, it has high computational complexity.

In 1988 Kosko [65] extended the Hopfield model by adding another layer of neurons. As a result, the new network was able to perform auto-associations as well as hetero-

3.2. RELATED WORK

associations. Also, this network had the bi-directionality feature, and this is why it was given the name *Bidirectional Associative Memory* (BAM). This feature means that the information flows in both directions, forward and backwards, inside the memory and the association can happen in any direction. So, it was introduced to in neural nets to give a two-way search for stored associations.

The Hopfield and the BAM models are both forms of *Artificial Neural Network* (ANN) models used as associative memories. However, there are some differences between them, such as the Hopfield model can only operate as an auto-associative memory and is unidirectional. In contrast, the BAM can operate as both auto and hetero-associative memories and is bidirectional.

A summarized comparison between the four neural AM models described earlier, Palm, Willshaw, Hopfield and BAM is presented in Table 3.1.

Table 3.1: Summarized comparison between different associative memory models

Model	Performance	Capacity	Latency	Time vs Space
Willshaw [126]	Low if there are errors in the input.	High when using sparsely encoded binary vectors.	One time step.	Space.
Palm [87]	Noise robust, fault tolerant.	Higher than that of Willshaw.	One time step.	Space
Hopfield [47]	Low as it requires extensive computation.	Relatively low compared to Palm and Willshaw.	Several time steps till convergence.	Space
BAM [65]	Requires extensive computation, however, better than the Hopfield.	Lower than that of Palm and Willshaw.	Several time steps till convergence.	Space

3.2.3.2 Associative Content Addressable Memory (ACAM)

An ACAM is a type of memory that chooses/outputs the best(closest) match according to some metric [68]. Due to the neural associative memory problems mentioned earlier, our *Bayesian Memory* (BM), which will represent our NB network, will rely on an *Associative Content Addressable Memory* (ACAM) with a distance metric to calculate the likelihood probability. Here we are going to present previous distance metric circuits and related ACAMs.

3.2.3.2.1 Dot Product metric

Since we will be using memristive crossbar for our BM design, the *Dot Product* (DP) metric should be considered because it can be computed using memristive crossbars [33, 104, 105, 127]. To compute the DP using a crossbar, one needs to terminate its columns with *Virtual Ground* (VGND) modules using inverting amplifiers to eliminate sneak path currents, as these currents can skew the DP computation, see Fig. 3.8. A bias column, produced by passing the inputs through a column of maximally-resistive memristive devices, is then subtracted off of the measured currents to force the final calculation to produce the DP evaluation exactly [111, 127]. Note that, the VGND modules are required to calculate the DP: without the VGND modules, sneak paths (traditionally undesired currents) would develop between columns in the memristive crossbar, making it impossible to calculate the DP exactly.

3.2.3.2.2 Hamming Distance (HD) Metric

Hamming Distance Associative Content Addressable Memories (HDACAMs) circuits

Figure 3.8: A conventional crossbar architecture to compute the DP between an input and multiple stored patterns. The $Out(bias)$ is subtracted from each $Out(i)$ where $i = 1, 2, \dots, n$ to produce the precise DP value.

Architectures that calculate the HD between binary patterns have been used before to build ACAMs. For example, Yusuke *et al.* [131] designed a hierarchical multi-chip architecture of fully parallel HD associative memory units. Although their multi-chip structure achieved good capacity and scalability, they implemented the HD module as a separate block, which increased the latency and resulted in power consumption of 51.3 mW for a 64×32 (word size $\times$ number of words) associative memory.

Mattausch *et al.* [78] designed an associative memory architecture that has low area requirements. However, their design still suffered from high latency and high power consumption of 260 mW. Mattausch *et al.* [79] designed another associative memory for nearest HD search based on frequency domain. This design consumed 36.5 mW, and 307 μ W at 1.8 V and 0.7 V, respectively, for a 64-word 256-bit system. Although their design achieved parallel HD computation with low power, its computation speed increased with the increase in the number of bits.

Rahimi *et al.* [100] proposed an *Approximate Associative Memristive Memory* (A^2M^2) for energy-efficient GPUs which depended on modifying the *Ternary Content Addressable Memory* (TCAM) cell design of Li *et al.* [73] by replacing some of the CMOS transistors with memristive devices. Although this lowered the power consumption of the original TCAM, the A^2M^2 design still featured some CMOS transistors, which limited the area-saving potential of using memristive devices. Their design also had a higher operating voltage than the approach in this work; our lower operating voltage should result in additional power savings. Unfortunately, a direct power comparison is impossible as Rahimi *et al.* only provide energy data for the entire system, and not separately for the A^2M^2 .

So, designing an ACAM that performs low power, parallel, and fast HD calculation will be a contribution and enhancement to the field of ACAMs. In order to do so, one needs to design a parallel, fast, and low-power HD circuit.

Hamming Distance Circuits

The Hamming network and MAXNET is a two-layer *Artificial Neural Network* (ANN) classifier of bipolar vectors [45]. A bipolar vector is similar to a binary vector but each 0 is replaced by -1 . This network is capable of identifying the pattern, from its stored ones, that most closely matches the input according to the HD metric. The number of input a and output C neurons is equal to the number of bits per pattern and the number of classes, respectively, see Fig. 3.9.

As shown in Fig. 3.9, the vector/pattern a consisting of i bits is presented to the Hamming network, and the neuron firing the most at the MAXNET corresponds to the pattern that has the lowest HD to the input, labeled as “Output.” The weight matrix of the Hamming network is created by encoding the class vectors. Let $s^{(C_1)}$ and $s^{(C_2)}$ be vectors/patterns relating to classes 1 and 2 respectively. The HD can then be computed by:

Figure 3.9: Block diagram of a minimum HD classifier using the Hamming network and the MAXNET. The number of classes in this network is p . The MAXNET is a recurrent network that has both inhibitory and excitatory connections. It is considered a *Winner-Take-All* (WTA) system as it suppresses all the Hamming network outputs but the largest one. The output corresponds to the stored pattern that most closely matches the input. For more details on the MAXNET see [45].

$$a^t s^{(x)} = \left(i - HD(a, s^{(x)}) \right) - HD(a, s^{(x)}), \quad (3.15)$$

which can be written as:

$$\frac{1}{2} a^t s^{(x)} = \frac{i}{2} - HD(a, s^{(x)}), \quad (3.16)$$

where $x = \{0, 1, 2, \dots, p\}$, and a , s , and i are the input pattern, stored pattern, and the number of bits of each pattern respectively. The network's weight matrix Wm can be written as:

$$Wm = \frac{1}{2} \begin{bmatrix} s_0^{(C_0)} & s_1^{(C_0)} & s_2^{(C_0)} & \dots & s_i^{(C_0)} \\ s_0^{(C_1)} & s_1^{(C_1)} & s_2^{(C_1)} & \dots & s_i^{(C_1)} \\ s_0^{(C_2)} & s_1^{(C_2)} & s_2^{(C_2)} & \dots & s_i^{(C_2)} \\ \vdots & \vdots & \vdots & \vdots & \\ s_0^{(C_p)} & s_1^{(C_p)} & s_2^{(C_p)} & \dots & s_i^{(C_p)} \end{bmatrix},$$

where the $\frac{1}{2}$ is a scaling factor. The output of each neuron of the Hamming network is:

$$f(output) = \frac{1}{i} \left(\frac{1}{2} a^t s^{(x)} + \frac{i}{2} \right), \text{ for } x = 1, 2, \dots, p \quad (3.17)$$

where $\frac{i}{2}$ is the network's bias, and $a^t s^{(x)}$ is the DP between a^t and all the bipolar stored patterns in W_m [45]. Zhu *et al.* [135] proposed a hybrid Hamming network circuit based on CMOS and memristors. They used two memristors along with NOT gates at the inputs to represent one bit, as shown in Fig. 3.10 to produce high and low currents in the cases of a match or a mismatch, respectively. For example, when using R_{ON} & R_{OFF} to represent logic 1, R_{OFF} & R_{ON} for logic 0, and having the inputs V_H and V_L representing 1 and 0 respectively. If the j^{th} in the input pattern matches the j^{th} bit in the i^{th} stored pattern, then the current flowing through the memristor pair is given by:

$$I_{ij} = \frac{V_H}{R_{ON}} + \frac{V_L}{R_{OFF}} = I_{ON} \quad (3.18)$$

As for the mismatch:

$$I_{ij} = \frac{V_L}{R_{ON}} + \frac{V_H}{R_{OFF}} = I_{OFF} \quad (3.19)$$

They also use VGND modules to eliminate sneak path currents.

$$\therefore Out_i = -R_F \sum_{j=1}^m I_{ij} \quad (3.20)$$

As a result of Eqs. (3.18) to (3.20) the pattern that closely matches the input will corresponds to the highest Out_i .

The usage of two memristors and a NOT gate to represent one bit along with the

VGND modules increased both their design size and power consumption.

Figure 3.10: Hamming distance circuit proposed by Zhu [135]. Redrawn from [135].

Cassuto and Crammer [17] developed a framework to calculate the Hamming similarity between vectors stored in a resistive memory. Their approach is not designed to calculate the HD between an input and the stored patterns but only between the stored patterns themselves, which makes it inefficient for pattern/image recognition applications.

Ge *et al.* [38] proposed an HD comparator using a unipolar memristor array, where they used the ON/OFF switching characteristics of the memristive array to compute the HD between two vectors presented as voltages. Their design is also inefficient in pattern matching applications because (1) the patterns have to be brought from a different memory module, which increases the latency and power consumption, and (2) their design can only compute the HD between two vectors at a time. In order to compute the HD between an input and several patterns, they either can do it serially, which is time-consuming or have multiple circuits/memristive arrays, which in turn increases the power consumption.

Kaplan *et al.* [60] proposed a resistive pre-alignment accelerator for approximate

DNA long read mapping “RASSA” that computes the HD between DNA strands. Their design featured several word rows, each row consisting of multiple 2T1R bitcells connected by a *Match Line* (ML) to an *Analog to Digital Converter* (ADC), as shown in Fig. 3.11. Since their HD calculation depends on the ML discharge amount, they used a single 4-bit ADC per up to 60 bitcells to sense the ML accurately and reduce the power consumption. Their design consumed 1.8 mW for a single comparison cycle for a 960 bits pattern/strand. An issue with their design is that it can only detect a mismatch between an input “1” and a stored “0,” but not the opposite. This is acceptable when using one-hot encoding for DNA base-pair strands to detect a single mismatch. However, it will be inefficient in other applications that require the detection of a mismatch between an input “0” and a stored “1.”

Figure 3.11: RASSA’s word row [60].

Zokaee *et al.* [136] adopted a similar design to Ge *et al.* [38] to accelerate the FM-Index in DNA short read alignment. Although their design improves the read’s throughput per Watt per mm^2 by $13\times$ compared to the state-of-the-art, it requires a 4×4 ReRAM array of 16 memristors to compute the HD between 2-base pairs (4 bits), which increases the area and power consumption.

Huang *et al.* [49] proposed an FPGA architecture for on-board corner detection and

matching. One of their sub-systems computes the HD between two binary patterns. Their HD circuit consists of a multi-input XOR gate and an adder tree, as shown in Fig. 3.12, where the XOR finds the difference between two patterns, and the adder tree counts the number of differences. Since this design computes the HD between two patterns only, they used n circuits to parallelize the HD computation between an input and n stored patterns. They did not report any power numbers, so we implemented their HD circuit on the Xilinx XC7K325T FPGA. Our results showed that 0.425 W of power is required to compute the HD between one input pattern and 10 stored 256-bit patterns.

Figure 3.12: Hamming distance circuit as implemented in an FPGA [49]

Fig. 3.13 shows an accuracy and power consumption comparison of the previously mentioned HD circuits and the direction of improvement. One can see that although these circuit achieve 100% on the HD calculation they consume high power, more than 10 mW for computing the HDs for 10 256-bit patterns.

To summarize, we presented the background and related work to our proposed research and showed that the current algorithms and architectures either suffer from high computational cost or high power consumption. In the next chapters, we will explain our

3.2. RELATED WORK

Figure 3.13: An accuracy versus power comparison of related work when computing the HDs for 10 256-bit patterns. Ideal circuits would be located in the top left corner.

proposed architectures and algorithms and compare them to this related work.

APPROXIMATE MEMRISTIVE IN-MEMORY HAMMING DISTANCE AND ASSOCIATIVE MEMORY CIRCUITS

In this chapter, we will present a novel, parallel, and low-power approximate memristive HD computing circuit design. This circuit can be used to compute the HD between an input and multiple stored binary patterns. We will then use this HD circuit to build a low-power *Approximate Memristive Associative Content Addressable Memory* AMACAM.

4.1 Hamming Distance Circuit Design

Fig. 4.1 shows a basic *Resistive Crossbar Network* (RCN) consisting of horizontal and vertical nanowires with memristors located at the wires' intersections. The binary patterns are stored in the columns, one pattern per column, with the memristor set to R_{OFF} (high resistance) to represent binary "0," and R_{ON} (low resistance) to represent binary "1." The input/test pattern is introduced to the crossbar through V_1 to V_m , where a read voltage V_{read} represents binary "1" and 0 V represents binary "0."

When the input pattern is presented as voltages to the crossbar rows, the current flowing through the memristor with conductance g_{ij} , where i and j denote the row and column number respectively, is $V_i \times g_{ij}$. Therefore, the total current through column j is $\sum_i V_i \times g_{ij}$ when the crossbar columns are grounded. The total current of each column represents the

4.1. HAMMING DISTANCE CIRCUIT DESIGN

Figure 4.1: An RCN where the memristors are located at the intersection of the horizontal and vertical nanowires. RCNs are used for storing patterns and evaluating the correlation between an input and the stored patterns. The patterns are stored in the columns of the RCN, and the binary input pattern is applied to the RCN rows in the form of voltages with V_{read} or 0 V, representing binaries “1” and “0” respectively. The WTA circuit chooses and progresses the highest crossbar output while terminating the other outputs.

correlation or DOM between the input and the pattern stored in that column; thus, the best matching pattern to the input is the pattern corresponding to the highest correlation magnitude $\sum_i V_i \times g_{ij}$ [105, 127]. Choosing the highest output between two crossbar columns can be done using the WTA circuit shown in Fig. 4.2. This WTA can be extended to have multiple inputs depending on the number of crossbar columns, 1 WTA input per crossbar column.

4.1.1 Hamming Distance Evaluation

A common architecture to compute the DP in-memory is the RCN with VGND modules terminating its columns [127]. When computing the DP, the mismatch between a stored “1” and an input “0” does not affect the output. In the HD computation, we require this

4.1. HAMMING DISTANCE CIRCUIT DESIGN

Figure 4.2: Two-input analog WTA/MAX circuit. Redrawn from [16]. Since this is a two-input max circuit, it consists of two similar sections M_1 to M_5 and M_6 to M_{10} , the M_1 to M_5 section is enclosed in the red box. This section of this circuit consists of a differential amplifier M_1, M_2 , a mirror active load M_3, M_4 and a voltage follower M_5 . The same applies to the M_6 to M_{10} section. During operation, the differential amplifier corresponding to the highest input voltage will keep a balanced gate-to-source voltages to allow the corresponding voltage follower to be ON. Thus, choosing/passing the highest/maximum voltage. That said, this circuit chooses the maximum voltage but doesn't output its index. In Section 4.3 we will discuss how can use this circuit along with memristive devices to output the index of the maximum voltage.

mismatch to influence the total RCN's column current. Practically this can be achieved by inducing a voltage drop at the end of each RCN column rather than using a VGND module as in the DP evaluation. When there is a positive voltage drop on each column, that voltage produces a current back through the memristive devices corresponding to rows where a "0" is input. If these rows were storing a "1," then more current would leave through the device than if the row was storing a "0," (because a stored "1" is represented by low resistance R_{ON} while "0" is R_{OFF} , which is high resistance), this is illustrated in Fig. 4.3. The result is that the voltage of a column will be lower if there are more mismatches between input "0" and stored "1." Thus, the column with the highest output voltage will be the column corresponding to the best matching stored pattern according to an approximate HD calculation. This approximation is due to the difference between

the currents in the different mismatches cases. That is, a mismatch between an input “0” to a stored “1” does not have the same weight as a mismatch between an input “1” to a stored “0.” Besides, there is still some lost current, sunk to the GND node, when there is a match between a “0” and a “0.” However, because of the memristive range, this current is negligible when compared to the sunk current due to mismatches.

Figure 4.3: This illustration demonstrates how sneak path currents are leveraged to compute the HD. Only a single column’s currents are shown, but each column is independent in this circuit, so the same principle applies to the other columns. When the input pattern is presented to the RCN, current flows both into and out of the RCN columns. When there is a voltage drop at the end of the column V_{d1} , currents will flow from the column to the GND nodes provided by “0”s in the input pattern. The currents flowing through a memristor storing a “1” (thick dotted arrow) are larger than currents flowing through a “0” (thin dotted arrow) because a “0” represents high resistance and a “1” represents low resistance. This is beneficial for computing the HD as currents through a “0” device to the ground (which represents a stored “0” with an input “0”) will be smaller than currents through a “1” device (stored “1” to an input “0”). The termination resistance R_T can be omitted when “0” inputs are grounded, and as long as the circuit connected to the RCN induces a voltage drop greater than 0V on the columns.

4.2 Results and Discussion

The experiments here were simulated using the Xyce open-source, SPICE-compatible, high-performance analog circuit simulator [102]. They were performed on a binary

dataset that we generated using Python’s NumPy random generator. The patterns used had 40% to 55% density levels unless otherwise noted. We chose to generate our dataset to be able to scale and change it to meet our experiments’ test criteria. It is worth noting that the resistance of the nanowires was taken into consideration when performing these experiments. This resistance was taken from [56], and was set to 30 k Ω for a 1kb crossbar array.

4.2.1 Scalability

We tested our circuit by increasing both the number of bits per pattern and the number of patterns per crossbar. This is important to know how our HD circuit scale.

4.2.1.1 Increasing the number of bits per pattern

As a benchmark, we performed the same experiment, using the same patterns, with a software HD computing function. Fig. 4.4 shows our HD circuit’s accuracy when increasing the number of bits.

From Fig. 4.4, one can see that the average HD calculation accuracy is $\approx 95\%$. However, for the patterns with 6 to 10 bits, the accuracy is $\approx 79\%$. This is due to the weight of the mismatch between an input “0” and a stored “1” being higher than the mismatch of “1” and “0.” For example, take the case when a stored pattern has an HD of 2, to the input, with two mismatches of input “1” to a stored “0,” and another pattern having an HD of 1 with a mismatch of “0” to “1.” The HD circuit will output a higher voltage for the first pattern, although the closest should be the second one. Given the fast, parallel, and low power computation of our circuit, this decrease in performance can still suit applications that can sacrifice some accuracy to gain speed and save power.

Figure 4.4: The HD calculation accuracy when increasing the number of bits per stored pattern while having a fixed number of stored patterns. The number of stored patterns was fixed to 10.

4.2.1.2 Increasing the number of patterns

In order to see the effect of increasing the stored patterns on our circuit’s performance, we stored and retained a constant 20 patterns while introducing new patterns. When computing the performance, we only compared the outputs from the constant 20 patterns to see whether their outputs would be affected by the increase of the number of stored patterns. From Fig. 4.5, one can see that increasing the number of patterns did not affect our circuit’s performance; it remained constant at $\approx 96\%$.

4.2.2 Sparsity

The goal of the following experiments is to test the performance of our HD circuit under different dataset sparsity levels. We generated binary datasets using Python’s NumPy random generator with different “1” density percentages. Section 4.2.2(a) shows the HD calculation accuracy at each density level. Section 4.2.2(b) shows the average HD calculation accuracy when the dataset has a range of “1” density, e.g., 10% range means that

Figure 4.5: The HD calculation accuracy when increasing the number of stored patterns while keeping the number of bits fixed to 256.

the dataset patterns have different densities within 10% of each other, and so on. One can see that our circuit’s performance decreases when the range of densities increases. This is because, as mentioned before, the mismatch between “0” to “1” has a higher weight than the mismatch between “1” to “0,” and a higher density range increases the probability of having more “0” to “1” mismatches than lower density ranges. That said, our HD circuit achieved $\approx 96\%$ on single densities, and as can be seen in Table 4.2, it achieved classification rates within 1% of the rates obtained by other HD circuits on the MNIST dataset.

4.2.3 Robustness to Device Variability

Memristors are prone to variations because the migration of oxygen vacancies determines their resistance change [107]. We performed experiments to test the operation of our HD circuit under device variations, i.e. random fluctuations in the devices characteristics and parameters. We used the Yang model [130] in this experiment as its device-to-device variations were reported in [107]. The Yang model is a metal oxide WO_x memristor

Figure 4.6: (a) HD calculation accuracy obtained when having a single constant sparsity/density for all the patterns. (b) HD calculation accuracy obtained when having a range of sparsity/density levels for the patterns. The number of bits used was 256.

in which the electric field causes the oxygen vacancies to move and thus modulates the resistance and results in the characteristic conductance hysteresis. This memristor can be modeled by:

$$I = w(\gamma \sinh(\delta V)) + (1 - w)(\alpha(1 - e^{-\beta V})) \quad (4.1)$$

$$\frac{dw}{dt} = \eta_1 \sinh(\eta_2 V) F(w, V), \quad (4.2)$$

where $\gamma, \delta, \alpha, \beta, \eta_1$, and η_2 are the material parameters, w is the state variable between 0 and 1, and $F(w, V)$ is window function to account for the oxygen vacancies nonlinear change as described in [130].

Having device variations means that the resistances and current flow through each memristor will be different. We tested our circuit with variations according to the numbers given in [107]. Fig. 4.5 shows the HD calculation accuracy with 100% device-to-device variation when increasing the number of stored patterns. The results were averaged over

10 iterations. One can see that the HD calculation accuracy is lowered to an average of 92%.

4.2.4 Memristor Model Variability

In this experiment, we tested our HD circuit operation when using different memristor models. Table 4.1 shows that the performance of the circuit changed by $< 1\%$. The characteristics of the memristor models tested were as follows:

- have R_{OFF} to R_{ON} ratios ranging from $3\times$ to $18\times$,
- are built from different materials such as Titanium Ti , Copper Cu , Silver Ag , and Silver Chalcogenide Ge_2Se_3 and Ag , which gives them different I-V characteristics (symmetrical and asymmetrical), and
- allow reverse current to flow normally through the memristor without suppressing it. So, for example, the Lehtonen model [70] will not be suitable for our circuit because it suppresses the reverse current. This suppression will affect the detection of a “0” to “1” mismatch and only detect a “1” to “0” mismatch.

Table 4.1: Proposed HD circuit performance when tested with four different memristor models.

Memristor model	Yang	Berdan	Merrikh-Bayat	Jo
HD Classification accuracy	94.3%	93.8%	93.9%	93.2%

4.2.5 Hamming Distance Circuit Approximation

What makes this circuit approximate is that the value of the input voltage affects the accuracy of the HD calculation. This occurs because a higher input voltage also produces a

Figure 4.7: The minimum HD between two patterns before they are distinguishable from one another in our Hamming distance architecture. This minimum is a function of the number of bits per pattern and the input voltage. As more bits are added, each bit that differs between the input and stored patterns affects the column's voltage less; as the input voltage is increased, each column's voltage becomes more distinct from columns with a similar (but not equal) HD. (a) Showing the minimum Hamming distance when using 0 to 256 bits and 0.01V to 0.45V, (b) Zoom-in on the behavior of the system for voltages between 0.01V to 0.2V.

larger differential voltage between different columns. When that differential surpasses a specific threshold voltage, for example, $800\mu\text{V}$ as required by the TSV621 Op-amp, the HD circuit becomes capable of discerning which column's pattern is a better match to the input patterns. Once that threshold is surpassed for a hamming distance of 1, a further increase in the input voltage would not be beneficial. Therefore, higher input voltages produced higher classification rates because the resolution at which two HDs could also be distinguished increased. That is, the minimum measurable Hamming distance between two patterns decreased. Figs. 4.7 and 4.8 shows the minimum HD between two stored patterns as a function of the number of bits per pattern (the number of rows in the RCN) and the input voltage. The minimum HD increases as the number of bits per pattern increases, or when the read voltage is decreased. This increase is because the difference between the highest and second-highest column output is less than the offset voltage ($800\mu\text{V}$) at low voltages.

However, increasing the input voltages decreases the minimum measurable HD, increasing the accuracy of the system. The downside of increasing the input voltage is

Figure 4.8: The minimum measurable HD for different read/input voltages as a function of the number of bits per pattern and the input voltage. Increasing the number of bits decreases the HD circuit’s accuracy because each added bit means another added resistive device. Since this architecture statically becomes a voltage divider problem, more resistors mean that each mismatch will produce a smaller change in the column’s voltage. As the difference between column voltages falls below the $800\text{ }\mu\text{V}$ threshold, the minimum measurable HD increases. However, increasing the input voltage increases the architecture’s accuracy, as the difference between two columns’ voltages again increases beyond the $800\text{ }\mu\text{V}$ offset voltage.

that this will also increase power consumption. The system was tested using the MNIST dataset to measure the classification rate and power consumption at different input voltages. Fig. 4.9 shows the obtained results, which confirms that increasing the input voltage increased the classification rate, but also increased the power consumption quadratically. That is using 0.1 V with a 256×100 RCN consumes 0.4 mW , while using 0.5 V for the same RCN consumes 11 mW . Choosing an appropriate input voltage depends on the application: if we choose a lower voltage, accuracy will be diminished, but power consumption will be decreased as well. A larger value will produce higher accuracy: increasing the accuracy from 72.5% accuracy to 74% when 0.5 V is used instead of 0.1 V , as shown in Fig. 4.9.

Figure 4.9: Obtained classification rate increasing the input voltage. The classification rates increased with the input voltage because larger input voltages equate to larger differences between column voltages, once the difference between two columns surpasses the comparator’s threshold of $800\mu\text{V}$, the architecture could determine which column was a better match for the input pattern. In other words, increasing the input voltage increased the accuracy and the resolution of the system. The classification rate became constant starting from 0.3 V because the problem ceases to be offset voltage related, but rather becomes dataset-specific. The error bars for the classification rate fluctuated from 2 to 4% because this approach is sensitive to which training patterns were selected to be stored in the crossbar.

4.2.6 Comparison to Related Work

Fig. 4.10 shows the accuracy and power consumption of our and other HD circuits. Although our circuit achieves $\approx 96\%$ accuracy only, it consumes less power. This makes our circuit beneficial for low power applications where one can sacrifice some accuracy. The work of Kaplan *et al.* is not depicted in Fig. 4.10 because their reported power number includes other circuitry, which we were not able to reproduce. However, it is worth noting that their HD circuit bitcell consists of 2T1R while ours is only 1R. Furthermore, our circuit, unlike theirs, can detect a mismatch between an input “0” and a stored “1.”

4.2.6.1 Datasets and Full System Implementation

In this section, we tested our circuit on the MNIST handwritten digits and benchmarked it against the other HD computing circuits/schemes, and implemented it in a full system

Figure 4.10: An accuracy versus power consumption comparison of related work when computing the HDs for 10 256-bit patterns. Note that the power of the WTA circuit is not included since the other circuits were simulated without one.

to assess its impact on the system. Table 4.2 shows the correct classification rates obtained over 100 stored patterns. One can see that our circuit's classification rates with and without device variations are within 1% of the results obtained by the other HD circuit-s/schemes. This confirms the operation of our circuit and proves that our HD circuit can tolerate large device-to-device variation.

To determine the impact of each circuit on the full system, we also tested our HD circuit versus the other HD circuits in a full HD ACAM, which will be explained later in Section 4.3. Fig. 4.11 shows our *Approximate Memristive Associative Content Addressable Memory* (AMACAM) system, which takes an input pattern and outputs a label. Table 4.2 shows the power percentage occupied by each HD circuit in relation to the full system. One can see that our circuit had the smallest power percentage, which helps in lowering the full system's power consumption while leaving room for further decreasing the power consumption by improving the rest of the system.

Table 4.2 shows the power percentage of each HD circuit to the whole system. One can see that our circuit consumed the least power compared to the rest of the system. This means that our circuit not only helped in lowering the full system's power consumption,

4.3. APPROXIMATE MEMRISTIVE ASSOCIATIVE CONTENT ADDRESSABLE MEMORY

but there is still room for decreasing the power consumption more by improving the rest of the system.

Table 4.2: Classification rates obtained when testing our proposed HD circuit versus an HD Python software implementation, and other HD circuits from the literature.

HD circuit type	Python	Proposed 0% device variation	Proposed 100% device variation	Zhu et al. [135]	Zokae et al. [136]	Ge et al. [38]	FPGA [49]
Classification rates	72.7%	72.5%	71.2%	72.7%	72.7%	72.7%	72.7%
Power ratio [§]	N/A	2.2%	2.4%	77%	88%	88%	99.5% [†]

[†] This percentage is calculated, not simulated, because we could not integrate the FPGA with the rest of simulated full AMACAM system design.

[§] Power percentage occupied by the HD circuit in relation to the full AMACAM system.

4.3 Approximate Memristive Associative Content Addressable Memory

Fig. 4.11 shows our AMACAM system, which takes an input image and outputs an image label. The ACAM consists of an in-memory metric computing circuit, a WTA circuit, and a label crossbar to output the winning label. In our case, we used our previously designed HD circuit. The metric circuit is then connected to a *Winner Takes All* (WTA) circuit. Since, this WTA circuit only outputs the maximum voltage and not its index we designed an indexing circuit that takes the WTA's output and convert it to a one-hot code where the location of the one corresponding the index of the maximum output voltage. 1.7 V and 0 V are used to represent 1 and 0 in the one-hot code, respectively. This one-hot code is then presented to the "Label crossbar", and the 1 is used to perform a NOT operation, as proposed by Borghetti [13], on its corresponding column in the label crossbar. Since a NOT operation is performed on the label crossbar columns, the inverse of the labels are stored in the label crossbar. The Lehtonen SPICE memristor model is used because of its

4.3. APPROXIMATE MEMRISTIVE ASSOCIATIVE CONTENT ADDRESSABLE MEMORY

rectifying capabilities, and its rapid ON & OFF switching process, which makes it a good candidate to be used in logic circuits.

Figure 4.11: The full ACAM system. The labels of the stored images are stored in the “Label Crossbar” columns “Label #1” to “Label #n”, and the final output is written in the “Output Column”. The various HD circuits tested were placed in the “Hamming Distance circuit” box to see their effect on the whole system’s performance.

Table 4.3: ACAMs types and power consumption comparisons. Estimates to the same storage capacities were performed to facilitate the comparisons.

	Computation Type	Power Consumption
Yuskue [131]	Parallel out-of-memory	64 mW
Mattausch [79]	Parallel patterns in-memory bit-wise	Estimate 50 μ W @0.7 V Estimate 5.7 mW @1.8 V
This work	Parallel patterns and bits in-memory	1.4 mW

Table 4.3 shows the type and power consumption comparison between our ACAM design and related ones. One can see that our design consumed 45 $\times$ and 4 $\times$ lower power

4.3. APPROXIMATE MEMRISTIVE ASSOCIATIVE CONTENT ADDRESSABLE MEMORY

than Yusuke's and Mattausch. Mattausch design can consume lower power at low supply voltages; however, it will suffer from slow search speed. Also, our AMACAM computation speed, unlike theirs, is independent of the number of stored bits and patterns.

To summarize, we designed a low-power, fast, approximate, in-memory computing HD circuit and used it to design an AMACAM. We will then use this HD circuit and the AMACAM to build our approximate *Bayesian Memory* (BM) system, presented in the next chapter.

BAYESIAN MEMORY

A type of graph-based model is the probabilistic model. These models include *Bayesian Networks* (BN) and *Hidden Markov Models* (HMM). In this chapter, we will focus on the low-power implementation of the BNs. We chose to work with BNs instead of HMMs because the BNs are more commonly used in modelling or representing conditional dependencies between attributes or variables. While the HMMs are mainly used to model time series data [42].

NB is an approximation of the full BNs approach to build more scalable systems. Here we use a Bayesian inference approach that depends on the *Naive-Bayes-Nearest-Neighbour* (NBNN). The NBNN is a feature-wise nearest neighbor algorithm that was introduced by Boiman *et al.* [12]. It retains the reference descriptors in their original form and does not quantize them [12]. Its task is to determine the most probable class “Y” given an input “X”, namely the *likelihood* in Eq. (3.6). The BM will be our basic module to perform the inference. We are going to build a *Bayesian Memory* (BM) that relies on an ACAM. This BM will represent an NB network.

5.1 Bayesian Memory Methodology & Implementation

The algorithm that we want our BM to implement is the *Maximum A Posteriori* (MAP), which is described in Section 3.1.1.1. Our BM consists of three main parts, as shown in Fig. 5.1.

1. the *Code Book* (CB);
2. the Inference module; and
3. the *Conditional Probability Table* (CPT)

Figure 5.1: The proposed BM module consists of (1) the CB, (2) the Inference module, and (3) CPT. When $\hat{X}$ is input, the posterior probabilities of the test vector to the stored vectors are calculated and stored using the CB and the CPT. Afterwards, the posterior probabilities are sent to the inference module to determine and provide the output in a Bayesian sense.

The CB and the inference module combined contain all of the stored patterns, while the CPT calculates and stores the posterior probabilities of the test vector to the stored vectors to be used in the Bayesian inference operation.

We will try two NB schemes one with constant priors and the other with variable ones we call them *Naive-Bayes-Constant-Prior* (NBCP) Eq. (5.1) and *Naive-Bayes-Variable-Prior* (NBVP) Eq. (5.2) respectively.

5.1. BAYESIAN MEMORY METHODOLOGY & IMPLEMENTATION

$$P(Y|X) = P(X|Y) \times k, \quad (5.1)$$

where k is a constant.

$$P(Y|X) = P(X|Y) \times P(Y). \quad (5.2)$$

5.1.1 Bayesian Memory Spatial Inference

The spatial inference is a type of inference that is performed on static images, in space only. In this inference, an input image is presented to the BM in one-time step, and the BM infers the most probable class that the input image belongs to by choosing the label of the closest matching stored image, according to NB-like inference. We are going to use the MNIST handwritten digits dataset to test our BM spatially. This will be done using the NBCP and the NBVP schemes. As mentioned in Section 3.1.1.1 the likelihood probability can be calculated using a distance metric, and that priors can be calculated using either a user's belief or the number of occurrence of patterns per class. In our case, the prior probability is calculated using the number of occurrences of patterns per class. Since we are using the MNIST dataset to test our BM with priors, we had to choose random patterns from the training the testing dataset to end up with variable priors. This is due to initially, the MNIST dataset has approximately the same number of patterns per class, thus giving us equal priors, see Table 5.1.

Class	0	1	2	3	4	5	6	7	8	9
$P(X)$	9.87	11.24	9.93	10.22	9.74	9.03	9.86	10.44	9.75	9.92

Table 5.1: Prior probabilities of each class as per the training dataset.

Since the likelihood probability $P(X|Y)$ can be calculated using a distance metric,

5.1. BAYESIAN MEMORY METHODOLOGY & IMPLEMENTATION

either the *Hamming Distance* (HD) or the *Dot Product* (DP). We used our previously designed HD circuit due to its low-power, fast, and in-memory computation. Fig. 5.2 shows the implementation of the BM node using the previously designed HD. The last row in the memristive crossbar shown in Fig. 5.2 stores the prior probability weights, W , of each corresponding column.

Figure 5.2: The blocks of the BM module and their equivalents in simulated hardware. In the simulated hardware, the CB and the CPT are combined in a simple $m \times n$ memristive crossbar. The memristors are located at the intersections of the rows m and columns n . The last row of the crossbar before the “Max and Inferring” circuit carries the weights of the *prior* probabilities. The weight W of the memristor will represent the probabilities. For example, given a set of probabilities, the highest probability will be represented by $W = 1$ and the lowest one will be represented by $W = 0$, while the in-between probabilities will take different weights between 0 and 1 depending on the number of probabilities and the number of bits represented by the memristor. The outputs of the memristive crossbar are sent to the Max and Inferring circuits, which forms the inference module.

The Merrikh-Bayat SPICE memristor model [81] was used for the prior weight memristors instead of the Yang model because it has a wider memristive range. A wider memristive range means that the prior probabilities can be distributed over this wider range to increase the differences between them. Storing the prior probabilities in this way spares

the need for a multiplier circuit, thus further saving area and power. The “Max and Inferring” circuit consists of an indexing circuit and a memristive crossbar that contains the labels of the stored patterns, as mentioned before and as shown in Fig. 4.11.

5.2 Results and Discussion

Whether priors are beneficial or not depends on the confusion that occurs in the inference process.

Figure 5.3: Confusion matrix obtained when inferring without priors. This illustrates which digits cause the confusion for the stored data. The colors, and the annotated numbers represent how many times the labels were confused with each other.

Fig. 5.3 shows that, given the stored data, different digits have different confusion rates. Fig. 5.4 shows the classification rates obtained in cases with and without priors. One can see that in “case 2” the classification rates were similar to the without priors

case. This is because the priors used in “case 2” aided in recognizing digits that already had low confusion rates, for example, digits 7, 1, and/or 0, thus didn’t increase the classification rates. However, in “case 1” the priors set aided in recognizing digits with high confusion rates, for example, 5, 9, and 8, thus had higher classification rates compared to the “without priors” case.

Adding the priors increased the consumed power by $\approx 20 \mu\text{W}$, 4%, see Fig. 5.5. However, this extra consumed power is acceptable because some cases will be similar to “case 1,” where the setting of the priors helped in lowering the high confusion rates that the system suffered from when inferring or classifying digits such as 5, 9, 8, etc. This is because the system now depends not only on the distance measure but its knowledge as well. The correct classification rate was therefore increased by about 3%, depending on the patterns used. Moreover, we can conserve approximately 300 $\times$ the power by using the priors as we can lower the input voltage with sacrificing negligible accuracy rate in our case the accuracy decreased from 77% to 76%. Fig. 5.6 shows that our NB method obtained comparable accuracy to an NB method using univariate Gaussians.

Figure 5.4: The correct classification rates with and without using the priors as a function of input voltage/HD accuracy. “cases 1 & 2” are different scenarios of the outcome of having variable priors.

To summarize, in this chapter, we proposed an approximate, in-memory circuit based

Figure 5.5: The consumed power for both systems at a given voltage. One can see that adding the prior using our proposed memristive method add a 4% more power to total system's power.

Figure 5.6: Classification rates on the MNIST dataset when using our NB method versus a univariate Gaussian NB method. One can see that our proposed NB-like memristive architecture achieved comparable performance to the univariate Gaussian NB method. This asserts the NB functionality of our proposed approximate method.

on our AMACAM circuit to perform NB-like inference, including priors with adding around 4% power overhead. We showed that by using our method to add the prior probabilities, we only add 20 μ W at 0.1 V. We also showed that our method obtained comparable classification rates to the univariate Gaussian NB method. In the next chapter,

we will demonstrate our proposed algorithms and architectures for graph inference using non-probabilistic relations.

MULTI-ATTRIBUTE GRAPH INFERENCE

Graphs have been widely used and proved to be useful for shape matching [84], object detection and tracking [43], and object categorization [31]. Graphs are also useful in capturing and representing the structure of data, which can help in decreasing the data size, thus lowering the memory and power requirements. In this chapter, we introduce a novel ensemble multi-attribute graph inference technique, which shows that using multiple types of relations helps in increasing the classification accuracy while decreasing the computational intensity, and power consumption of the system. We also introduce a novel multi-attribute graph inference *Early Exit* (EE) technique with a confidence measure to further lower the power consumption without sacrificing the system’s accuracy.

6.1 Methodology

In our ensemble inference technique, we propose to use multiple relation graphs instead of a single full graph. With this technique, we will be able to use smaller graphs, 50% or lower graph sizes, compared to the single full graph method. This will, as mentioned earlier, reduce the computational intensity and power consumption of the system while maintaining comparable classification rates. We will test our technique on:

1. The Yale faces dataset [121], which contains 165 images of 15 subjects with differ-

ent lighting conditions, and facial expressions.

2. The AT&T faces dataset [3] which contains 400 images of 40 subjects with different facial expressions, occlusions, 20 degree rotation and 10% scale variations, and compare our results to other systems. Examples from both datasets are shown in Fig. 6.1.
3. The Extended Yale faces dataset B [41] which contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions.

Figure 6.1: (a) Examples from the AT&T dataset. (b) Examples from the the YALE dataset.

We will test our ensemble technique on our memristive system design, and on the spectral domain *Graph Convolutional Neural Network* (GCNN) model proposed by Deferrard *et al.* [29], which will be explained later.

6.1.1 Graph Generation

We first need to generate the graphs from the original images. Here we generated three types of relation graphs (1) edges graph, (2) color graph and (3) texture graph. We used

Figure 6.2: Edges graph where the red dotted rectangles represent the patches of the image. This image has 4 patches. The blue graph shows how the vertices and edges are connected. The attributes for each node V_i are the edges location in their corresponding quadrants.

the Canny edge detector, from OpenCV, to create the edges graph where the vertices of the graph are patches of the detected edges, and the graph edges are the relation between these vertices. Fig. 6.2 shows an example of how the vertices and edges are connected. For generating the colors and textures graphs, we used the *Histogram of Colors* (HoC), from OpenCV, and the *Local Binary Patterns* (LBP) function which uses modules from Python's scikit-image, respectively. The textures and colors graphs are considered to be *Regional Attribute Graphs* (RAGs). The parameters for the Canny edge detector, the HoC, and the LBP functions were obtained via a *Particle Hill Swarm* (PHS) optimization technique.

The PHS is a robust optimization technique capable of efficiently addressing a variety of large-scale engineering and science optimization problems [19]. It is a variant of a swarm intelligence approach that describes the collective behavior of natural and biolog-

ical systems such as bird flocking, or bacterial growth, that finds a global maximum as a point or surface in an n -dimensional space [95]. In our experiments, a global maximum can be the point where the systems achieve the highest accuracy regardless of any other parameters; we call it *Optimized for Accuracy* (OfA). Alternatively, when the systems reach the highest classification accuracy given a restricted power budget, *Optimized for Power* (OfP). In other words:

- **Optimized for accuracy:** This means setting the parameters for the Canny edge detector, HoC, or LBP to grab the maximum amount of information from the images and generate a graph from them without restricting the graph size or a power budget. For example, in the textures graph, the LBP function has parameters to control the number of textures obtained from an image. That is, if an image has 120 different textures, then this function will return a graph of 120 nodes.
- **Optimized for power:** Setting the parameters for the Canny edge detector, HoC, or LBP to grab the maximum amount of information given a restricted power budget or graph size. For example, in the textures graph setting the graph size to 60 instead of 120 nodes, will reduce the graph size by 50% thus lowering the consumed power. However, doing so will truncate some useful information.

Fig. 6.3(a) & (b) shows the extracted textures for the same image under different parameters. Fig. 6.3(c) & (d) shows the number of graph nodes and values for Fig. 6.3(a) and Fig. 6.3(b), respectively. One can see that the graph size in Fig. 6.3(c) (OfP) is half that of Fig. 6.3(d) (OfA). This means that when optimizing for power, we lose information, which will affect the classification rate negatively, and vice versa when optimizing for accuracy.

Figure 6.3: (a), (b) Extracted textures with two different set of parameters from the same image. To extract the textures features, we used the *Local Binary Patterns* (LBP) function that uses modules from Python’s scikit-image. (c), (d) Number of graph nodes, and the value of their attributes for (a) & (b) respectively. From (a) & (b) one can see that more textures were detected for the OfA (b) which caused the OfA to have more (double in this case) number of nodes compared to the OfP. This more information will increase the classification accuracy but will also increase the power consumption because it will increase the graph size. Section 6.2.2 shows how the textures histograms are processed before storing/using them in our proposed systems.

We hypothesize that if we used OfP graphs or even smaller graphs with our ensemble technique, we could achieve similar classification accuracies to the OfA while maintaining low power consumption.

6.2 Implementation and Results

To obtain the classification rates for the Yale faces dataset, we used $\frac{2}{3}$ as training patterns and $\frac{1}{3}$ for testing. After running the system, if the winning label matches the testing label, we then set the classification to 1, otherwise 0. We then sum these ones and divide them by the number of testing patterns to obtain the final classification rate.

6.2.1 OfA vs OfP

Fig. 6.4 shows the results obtained when using single relations (optimized for both accuracy, and power). One can see that the classification rate for the OfP single relation is 22% lower than that OfA this is due to the information loss mentioned earlier, and the power consumption is lower for the OfP this is because the graph size was reduced.

Figure 6.4: Classifications rates and power consumption for OfA versus OfP graphs tested on the Yale faces dataset on our memristive design.

6.2.2 Memristive Ensemble Multi-attribute Graph Inference for face detection

We chose to use our previously designed AMACAM circuit, (see Section 4.3), to perform the inference task to benefit from its low-power, parallel, and in-memory computing capabilities. Since our AMACAM deals with binary data, we had to binarize the generated

graphs' attributes. The output of the Canny edge detector is already binary, so we only had to binarize the colors and textures graphs.

We obtained the optimum bit precision for these graphs using the particle swarm technique. This was done by selecting a range of bit precision and refining them till a global maximum, of classification accuracy, is found. The optimal bit precisions were 4 and 6-bit for the colors and textures graphs, respectively.

We then used these precisions to binarize the graphs; this was done by dividing the values of the graph into multiple ranges and assigning binary values to them, see Example 6.2.1.

Example 6.2.1. For a 4-bit precision we have 16 binary values from 0000 to 1111. So, If the maximum and minimum graph values were 0 and 16, we assigned 0000 for 0 and 1111 for 16. As for the values in between we divided the values between 0 and 16 into 16 ranges and assigned them their binary equivalent, e.g. range#1 was assigned 0001, range#2 was assigned 0010, and so on.

To implement our ensemble technique, we stored each type of graph offline in its crossbar, as shown in Fig. 6.5. In order to store these binary graphs, we unravelled/flattened each graph and wrote it 1-bit per memristor column-wise.

We will also use this technique to perform an *Early Exit* (EE) method, which will help in further reducing the power consumption and the computational cost of the system. The EE method will be explained later.

In order to infer the label that closely matches the input graph from this ensemble technique, we read the highest output from each crossbar. We then select the class that had the highest repetition from the chosen outputs, see Fig. 6.6.

The experiments in this section were performed using the Xyce SPICE simulator [102]

Figure 6.6: Ensembling example. If the Edges, Colors, and Textures output labels are 2, 3, and 2, respectively, then, the winning label according to the majority ensembling, will be 2.

Fig. 6.7 shows that by using our proposed ensemble technique, we were able to obtain similar classification rate to the OfA graph while reducing the overall power consumption by 41%. These rates were obtained, as explained in Section 6.2.1.

6.2.2.1 Comparison with other face detection techniques

Jayech *et al.* [54] proposed a Bayesian network approach that uses different attributes, such as textures and colors, and captures the relations and dependencies between them. They obtained classification rates on the AT&T dataset of about 74%.

Suri *et al.* [115] proposed a multimodal authentication system using a novel bio-inspired architecture that uses the CM1K chip. They obtained about 86%, and about 98% classification on the Yale and AT&T face datasets, respectively, while consuming 275 mW for their recognition phase.

Krestinskaya *et al.* [66] developed a modified *Hierarchical Temporal Memory* (HTM)

Figure 6.7: Classification rates and consumed power by different graph inference techniques. Single relation inference optimized for either accuracy or power, and multiple relation graph inference. One can see that when using our proposed technique, we were able to obtain comparable classification rates to the OfA one while lowering the overall power consumption by 41%. For the experiment performed here, the flattened edges graphs were 2048 bits long, the textures ones were 196 bits, and the colors graphs were 1530 bits for the proposed technique, for each image in the Yale faces dataset. These sizes were obtained according to the parameters from the PHS algorithm.

spatial pooler and temporal memory memristive circuit. They obtained around $\approx 83\%$ classification accuracy on the AT&T faces datasets. Their inference/pattern matching phase consumed ≈ 570 mW. They also had to transfer the data to the pattern matching module, which increased the latency of the system.

Zyarah and Kudithipudi [137] proposed a neuromemristive crossbar architecture for the HTM's spatial pooler and a sparsely distributed representation classifier. They obtained $\approx 87\%$ classification accuracy on the Yale faces dataset; however, they didn't report any power numbers.

From Table 6.1, one can see that our combined relations system outperformed the

6.2. IMPLEMENTATION AND RESULTS

other systems from the classification rate perspective while consuming 5 $\times$, and 11 $\times$ less power compared to the recognition sub-systems of Suri *et al.*, and Krestinaskaya *et al.* on the Yale faces dataset respectively. Our system is also n times faster than Krestinaskaya *et al.* recognition sub-system, where n is the number of stored patterns due to our leverage of in-memory parallel computing. As for the AT&T dataset, although our system obtained lower classification rates, it consumed 5 $\times$, and 11 $\times$ less power compared to the other systems.

	Yale dataset	AT&T faces datasets	Consumed power on Yale faces dataset
This Work (Combined Relations)	95.5%	83%	50 mW
Jayech <i>et al.</i> [54]	N/A	74%	N/A
Suri <i>et al.</i> [115]	86%	96%	275 mW
Zyarah and Kudithipudi [137]	87%	N/A	N/A
Krestinaskaya <i>et al.</i> [66]	87%	87%	570 mW

Table 6.1: Classification rates and consumed power of our work compared to related work.

6.2.3 Graph Convolutional Neural Network

Defferrard [29] *et al.* proposed a GCNN with pooling modules based on spectral filtering that uses Chebyshev expansion as the filter. The usage of the Chebyshev expansion helped in lowering the complexity of the system from $O(n^2)$ to $O(K|E|)$ where n , K , and E are the number of graph nodes, size of the filter, and the number of graph edges respectively.

Although their model had lower complexity and had comparable results to a conventional convolutional neural network on the MNIST dataset, its computation intensity can still be further reduced by decreasing the graph sizes. As we mentioned before, we hypothesize that we can lower this intensity by using our ensemble multi-attribute graph inference technique because this will decrease the graph sizes, as seen and explained

earlier. Fig. 6.8 shows the block diagram of our ensemble technique using the GCNN models.

Figure 6.8: Block diagram of our GCNN ensemble technique. The “Ensembling” box performs ensembling as shown in Fig. 6.6, or Example 6.2.2. The output labels are the labels chosen from each GCNN module. Two ensembling techniques can be used here either by class majority from the output probabilities, as shown in Fig. 6.6 or by averaging the probabilities as in Example 6.2.2. This GCNN ensembling was performed on the RTX 2080 Ti GPU.

Since the outputs of these GCNNs are softmax probabilities, we can use two ensemble techniques to determine the output. (1) By classes majority, see Fig. 6.6, or (2) by averaging the output probabilities and taking the highest mean probability, see Example 6.2.2.

Example 6.2.2.

If we have 10 classes (ranging from 0 – 9), 3 GCNNs, and the output probabilities are:

- #1: [0.0737, 0.0982, 0.0984, 0.0678, 0.141, 0.0519, 0.2537, 0.0713, 0.0991, 0.045].
- #2: [0.065, 0.2444, 0.0033, 0.0054, 0.0037, 0.0021, 0.0041, 0.0028, 0.311, 0.3582].
- #3: [0.1193, 0.0616, 0.0211, 0.041, 0.0227, 0.0323, 0.0119, 0.0392, 0.4803, 0.1704].

then:

6.2. IMPLEMENTATION AND RESULTS

- Average probabilities: [0.086, 0.1347, 0.0409, 0.0381, 0.0558, 0.0288, 0.0899, 0.0378, **0.2968**, 0.1912].
- Final chosen output class: 8, the corresponding probability is highlighted in bold in the “Average probabilities”.

We tested our GCNN ensemble technique on the Extended Yale faces dataset B on the RTX 2080 Ti GPU. Fig. 6.9 shows that we were able to obtain comparable classification rates to the OfA graph while decreasing the consumed energy by $\approx 89\%$. This decrease in energy is due to the reduction of the number of nodes by $\approx 86\%$, see Fig. 6.10. Fig. 6.11 shows the number of nodes, classification rates, and energy consumption together.

Figure 6.9: Our ensemble technique classification rates versus consumed energy on the Extended Yale faces dataset B, ran on the RTX 2080 Ti GPU. The relation types used here are color, edge, and *Scale Invariant Feature Transform* (SIFT) [74] relations. One can see that our proposed techniques achieved comparable classification rates to the OfA while lowering the energy consumption by about 89%.

6.2.3.1 Comparison with other multi-graph combination structures

As mentioned in Section 3.2.1.2, Park *et al.* proposed a multi-graph combination structure using inter and intra-layers. Their method suffered from scalability issues; for example, it increases the graph/adjacency matrix sizes exponentially. We adopted a similar technique

Figure 6.10: Classification rates vs number of nodes for graph technique using the Extended Yale faces dataset B. Our proposed “Ensemble 1” & “Ensemble 2” have the same number of nodes since the only difference is at the final ensembling stage, see Fig. 6.8. One can see that our proposed techniques achieved comparable classification rates to the OfA while decreasing the number of graph nodes by about 86%. N.B. the combined sizes of our proposed techniques mean the total number of nodes of all used graphs. That said, in our techniques, graph inferences are done in parallel and separately as shown in Fig. 6.8, thus the complexity (computation time) remains the same as in a single graph inference.

to Park *et al.*’s, named it “Hetero Technique,” and used it in the GCNN model to compare it with our ensemble GCNN model. Fig. 6.12 shows a comparison between our proposed ensembling techniques and the “Hetero Technique” in the GCNN model. One can see that although the “Hetero Technique” outperformed ours by 4% in terms of the classification rate, ours consumed $3\times$ less energy.

6.2.4 Early Exit Inference

In this section, we propose a new inference algorithm using our ensemble multi-attributes graph inference method, which performs one single graph inference at a time, and exits or continues depending on a *Confidence Measure* (CM). If this measure is met then the algorithm exits, otherwise the algorithm moves on to the other graph, etc., as shown in Example 6.2.3, and Fig. 6.13.

6.2. IMPLEMENTATION AND RESULTS

Figure 6.11: Number of graph nodes, classification rates, and energy consumption. The sizes of the bubbles depicts the nodes ratio. One can see that our proposed techniques achieved similar classification accuracy to the OfA while lowering both the number of nodes and energy consumption by 89% and 86%, respectively.

Figure 6.12: The energy consumption and the classification rates for our technique versus the “Hetero Technique.” One can see although the “Hetero Technique” outperformed our proposed techniques, ours still consumed 70% less energy. It is worth mentioning that the cumulative number of nodes of our method is equal to the number of nodes of the “Hetero Technique.” That said since our methods work on each relation separately and in parallel, unlike the “Hetero Technique,” this helped in reducing the computation time and cost.

Example 6.2.3.

If our CM is based on a similarity percentage of 75% or the class majority, then:

- Case #1: If the similarity measure we obtained from inferring the first graph is 82%, then our EE algorithm will exit and the winning class label from the first graph will be our winner. However,
- Case #2: if the similarity measure we obtained from inferring the first graph is 67%, then our EE algorithm will go to the next domain (another relation), and perform inference on it. If the winning labels from these two stages are the same, for example, class 2 for both, then our EE algorithm will exit and label 2 will be our winning label. If not, then our EE algorithm will continue until the CM is met.

Figure 6.13: Domain-wise or early exit algorithm block diagram.

The CM is an arbitrary measure and can be set according to the user or what fits the graphs. For example, when dealing with binary attribute graphs, one can use the Hamming distance as the measure, but if one is using the Softmax, then the probability value would be a better measure choice. In this case, we are going to use both the Hamming

6.2. IMPLEMENTATION AND RESULTS

distance measure for our implementation shown in Section 6.2.2, and the Softmax for the *Graph Convolutional Neural Network* (GCNN).

Using this algorithm will help in reducing the power/energy consumption as parts of the system will not be used during the inference phase, depending on the CM.

Figure 6.14: The CM used here is the Hamming distance. One can see that choosing a low confidence measure helps in reducing the power consumption, by about 40%, but it also lowers the classification rate because it tells the system to exit early even if the mismatch value is high. However, the high confidence measure forces the system to use more information, other graph types, before exiting which helps in increasing the classification rates by 13%, in this case.

Fig. 6.14 shows the effect of the CM value on the early exit algorithm accuracy and power when inferring on the Yale faces dataset. One can see that when setting a low CM, e.g., 55%, the classification rates becomes approximately equal to that of using one single relation. This is because a low CM means to trust the output even if the mismatch value is high. However, choosing a high CM, e.g. 85%, forces the system to go to the other graphs to gain more confidence. As a result, the classification rates increases, and the

6.2. IMPLEMENTATION AND RESULTS

power increases because of the usage of more sub-systems.

	Yale dataset	AT&T faces datasets	Consumed power on Yale faces dataset
This Work (Memristive combined relations)	95.5%	83%	50 mW
This Work (Memristive Early Exit)	92%	81%	24 mW
Jayech <i>et al.</i> [54]	N/A	74%	N/A
Suri <i>et al.</i> [115]	86%	96%	275 mW
Zyarah and Kudithipudi [137]	87%	N/A	N/A
Krestinaskaya <i>et al.</i> [66]	87%	87%	570 mW

Table 6.2: Classification rates and consumed power of our work compared to related work.

Figure 6.15: An illustration of the results mentioned in Table 6.2. One can see that our work has outperformed the other works by 7% while consuming $\approx 10\times$, on the Yale faces dataset. The desired system's location here would be at the top left corner, i.e., highest classification accuracy and lowest power. Not all related systems are depicted here due to missing information.

From Table 6.2 and Fig. 6.15, one can see that using the EE inference algorithm helped in lowering the total power consumption by $\approx 50\%$ while achieving an accuracy of 92% compared to using the whole combined relation system. This is because using a lower number of relations is sufficient to obtain a correct inference, in some cases.

Fig. 6.16 shows that using our EE algorithm on the GCNN helped in reducing the consumed energy by $\approx 28\%$ while obtaining a classification rate of $\approx 93\%$ compared to our full ensemble technique's 97%, as well.

6.3. IDENTIFYING RELEVANT RELATIONS/GRAPH EDGES, AND ATTRIBUTES

Figure 6.16: Our ensemble and EE techniques classification rates versus the consumed energy on the Extended Yale faces dataset B. One can see that our proposed techniques obtained comparable classification rates to the OfA, within 2%, while lowering the energy consumption by $\approx 89\%$

Choosing the CM: The CM offers a trade-off between accuracy and power consumption. So, choosing a low CM similarity measure, will suite applications that require low power, and can sacrifice some accuracy. However, if the accuracy is of high importance, then one can choose a high CM. The low and high CM numbers are application, and dataset-specific. In the case of our memristive ensemble system, we used a 55% and an 85% similarity measures as our low and high CMs, respectively.

6.3 Identifying Relevant Relations/Graph edges, and Attributes

Identifying relevant relations/graph edges and attributes is a difficult task as that depends on the dataset and application. In this work, we use the PHS technique to identify them. This is done by adding the relation and parameters to the PHS to exclude the irrelevant relations and edges.

That said, for example, in the black and white MNIST dataset, the HoC or the LBP textures function will be irrelevant/redundant attributes because of the dataset's binary nature. Fig. 6.17 shows the HoC for two images from two different classes in both the

6.3. IDENTIFYING RELEVANT RELATIONS/GRAPH EDGES, AND ATTRIBUTES

MNIST and the Yale dataset. One can see that in Fig. 6.17(a) & (b), the HoC didn't provide useful information to separate the digits, however, for the Yale faces dataset it did, see Fig. 6.17(c) & (d).

Figure 6.17: (a) & (b) HoC for images from the MNIST dataset 7 and 6 classes, respectively. (c) & (d) HoC for images from the Yale faces dataset *s4* and *s6* classes, respectively. One can see that in the case of the MNIST dataset images ((a) & (b)), the HoC didn't provide any useful information to differentiate between the images. On the other hand, for the Yale faces dataset the HoC did. So, one can conclude that the HoC is redundant and irrelevant to be used as an attribute in the MNIST dataset.

Fig. 6.18, shows the effect of using different (relevant and irrelevant) attributes, on the MNIST and Yale faces dataset, on the classification rates and power consumption of the system. Fig. 6.19, shows the classification-power-ratio between using 2, and 3 relations for the MNIST and Yale faces datasets. In conjunction with Fig. 6.18, one can see that when using the third relation (HoC) in the MNIST dataset not only the classification rate

6.3. IDENTIFYING RELEVANT RELATIONS/GRAPH EDGES, AND ATTRIBUTES

was reduced by about 10%, the classification-power-ratio was decreased by $\approx 43\%$. However, for the Yale faces, although the classification-power-ratio was reduced by 25%, the classification rate was increased by 16%. Thus, this decrease in the classification-power-ratio can be overseen in an application that can trade-off power to gain more accuracy. The classification-power-ratio can also be a key to determine the number of useful attributes used. So, if this ratio keeps on decreasing when adding more attributes, one should start to either use different attributes or stop adding them.

Figure 6.18: The classification rate vs power consumption when using the HoC as the third relation for both the MNIST and the Yale faces dataset. One can see that for the MNIST dataset adding the HoC not only increased the power consumption but decreased the classification rate as well because the HoC, in this case, didn't add useful information but rather caused confusion in the system. However, it helped in increasing the classification rates for the Yale face dataset by 16% because it helped in differentiating between the patterns.

To summarize, we:

1. Proposed an ensemble multi-attribute graph inference technique that helps in re-

6.3. IDENTIFYING RELEVANT RELATIONS/GRAPH EDGES, AND ATTRIBUTES

Figure 6.19: Classification-power-ratio for both the MNIST and Yale faces dataset when using 2 and 3 relations. The colors in this figure matches the colors in Fig. 6.18

ducing the consumed power/energy, by about 85%, and graph size by $\approx 86\%$ while maintaining a comparable classification accuracy to the OfA full single graph.

2. Proposed an EE technique using our ensemble method to reduce the computation intensity and consumed energy/power further by about 30%.

CONCLUSION

The motivation for this research emerged from the advancements of emerging nano-scale devices, memristors, and their possibilities in designing and building low-power, fast, in-memory, and approximate computing systems. The memristor is two-terminal device capable of performing storage and computation simultaneously. This made it a good candidate for building such computing systems. In this research, we:

- (a) introduced an approximate, in-memory, scalable, low power Hamming distance design,
- (b) introduced a fast, in-memory associative content addressable memory design, and
- (c) investigated the effect of device-to-device variations on these designs.
- (d) introduced a novel, simple, fast hardware design to perform Naive Bayes-like inference,
- (e) introduced novel approximate graph-based inference algorithms that model different relations between the attributes of images,
- (f) introduced ways to lower the memory and computation requirements,

Hamming Distance and Associative Content Addressable Memory

We designed a low-power, approximate, fast, and parallel memristive *Hamming Distance* (HD) circuit that relied on the crossbar’s sneak path currents to compute the HD between an input and several stored patterns in parallel, see Chapter 4. We showed that the operation of our HD circuit under non-ideal fabrication conditions changes only slightly, decreasing the correct classification rates for the MNIST handwritten digits dataset by $< 1\%$. Also, its operation is independent of the memristor model used, as long as the model allows a reverse current. Because we leverage in-memory parallel computing, our circuit is $n\times$ faster than other HD circuits, where n is the number of HDs to be computed, and it consumes on average $\approx 500\times$ less power compared to other memristive and CMOS HD circuits. Another benefit of this circuit is that it can be controlled to trade-off accuracy to power and vice versa.

We then used this HD circuit to design the ACAM. This ACAM consumes on average $\approx 25\times$ lower power than state-of-the-art HD ACAMs and is faster due to its leveraging of parallel in-memory patterns and bits search. We compared our HD circuit against the other HD circuits in a full ACAM system to determine its benefits in a complete system. We found that our HD circuits helped in reducing the overall power consumption by 95%.

Bayesian Memory

We used our designed ACAM to build the BM, an NB inference system, where we proposed a novel and a simple way to incorporate prior probabilities, which is a type of probability used in Bayes theorem while adding a 4% power overhead. We tested this BM on the MNIST dataset and showed that using variable priors can lead to better performance that exceeds the constant priors system by approximately 3% with a small increase

in the consumed power. We also showed that we could conserve approximately $300\times$ the power by using the priors as we can lower the input voltage without significantly decreasing the performance, 1% in this case. It is worth noting that this NB obtained a 77% classification rate on the MNIST dataset, which is comparable to the 79% obtained by using a univariate Gaussian NB system.

Non-probabilistic Graph Models

We proposed a novel ensemble multi-attribute graph inference technique that reduces the computation cost and energy/power consumption by reducing the graph sizes. For example, when testing on the Yale faces database we were able to decrease the graph sizes by $\approx 70\%$ and reduce the energy/power consumption by $\approx 60\%$, while maintaining comparable accuracy, with a standard deviation of 0.25, to using the single full-size graph. Compared to other face recognition/inference techniques our techniques outperformed them from the classification rate perspective while consuming about $8\times$ lower power.

We also introduced an *Early Exit* (EE) inference scheme using our ensemble multi-attribute inference method, using a confidence measure. Using this scheme helped in lowering the power consumption by a further 30% while maintaining a comparable accuracy, with a standard deviation of 2. Compared to other face recognition techniques, our schemes outperformed them from the classification accuracy perspective by obtaining an average of 94% to 86% while consuming $\approx 11\times$ lower power.

Our proposed approximate designs and techniques are useful for inference applications where speed and low power are of great importance, and a slight loss in accuracy is tolerable. Our work is also particularly relevant for image recognition applications with constrained power budgets, such as embedded applications, and portable devices.

7.1 Future Work

For the proposed Hamming distance circuit, we will investigate in more detail how our circuit scales and performs under other fabrication parameters such as faulty devices.

As for the graph classification, we will look into 1) implementing/building the proposed memristive systems, 2) implementing the GCNN models in memristive hardware, and 3) looking in more details into identifying relevant types of relations given the dataset.

7.2 List of Contributions

- I designed a novel, in-memory, low power, and fast memristive Hamming distance circuit [119, 120], which is suitable for DNA sequencing, and pattern matching applications, (see Section 4.1).
 - It consumes $100 - 1,000\times$ less power than other CMOS and memristive Hamming distance circuits,
 - is independent of the memristor model used,
 - can be approximated to trade accuracy for power consumption,
 - and is tolerant to 100% device variations.
- I designed a fast, efficient, and in-memory memristive Hamming distance AMACAM circuit using my Hamming distance circuit that can be used in inference applications [116]. I showed that my AMACAM design is faster and consumes $4\times - 45\times$ less power than state-of-the-art ACAMs (see Section 4.3).
- I introduced a novel and simple way for incorporating the prior probabilities of the Naive Bayes theory in memristive hardware. I represented the prior probability

value by using a single analog memristive weight [116, 118] (see Chapter 5).

- I proposed a novel ensemble technique for multi-attributes in graph inference. This technique reduces the power/energy consumption by 41% on my proposed memristive hardware compared to using full size graphs. I showed that it reduces the energy consumption by 89% on the RTX 2080 Ti GPU, and the number of graph nodes by 86% when using the GCNN model, while maintaining a comparable classification accuracy of 97% on the Extended-YaleB faces database to using a single full relation (see Section 6.1).
- I also introduced a novel early exit technique based on a confidence measure, which helped in lowering the overall power/energy consumption even more, by 30%, while maintaining a comparable accuracy (see Section 6.2.4).
- I devised new arithmetic operators based on frequency signal representation [117].

7.3 List of Publications

- **M. M. A. Taha**, C. Teuscher, *Approximate Memristive In-memory Hamming Distance Circuit*, ACM Journal on Emerging Technologies in Computing Systems (JETC), 16, 2, Article 18, March 2020, 14 pages. DOI: <https://doi.org/10.1145/3371391>
- **M. M. A. Taha**, M. Perkowski, *Realization of Arithmetic Operators based on Stochastic Number Frequency Signal Representation*, In IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL), Linz, 2018, pp. 215-220. DOI: 10.1109/ISMVL.2018.00045

- **M. M. A. Taha**, W. Chavez, and C. Teuscher, *Spatial and Temporal Probabilistic Inference Using a Memristive Associative Memory*, International Journal of Unconventional Computing (IJUC), 13(2):117-137, 2017.
- **M. M. A. Taha** and C. Teuscher, *Naïve Bayesian Inference of Hand-Written Digits using a Memristive Associative Memory*, In IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2017, pages 139-140. DOI: 10.1109/NANOARCH.2017.8053732
- **M. M. A. Taha**, W. Woods, and C. Teuscher, *Approximate In-Memory Hamming Distance Calculation With A Memristive Associative Memory*, In IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2016, pages 159-164.
- W. Woods, **M. M. A. Taha**, D. Tran SJ, J. Burger, and C. Teuscher *Memristor Panic - A Survey of Different Device Models in Crossbar Architectures*, In IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), 2015, pages 106-111. DOI: <https://doi.org/10.1109/NANOARCH.2015.7180595>

REFERENCES

- [1] M. Abedin, Y. Tanaka, A. Ahmadi, T. Koide, and H. Mattausch. Fully parallel associative memory architecture with mixed digital-analog match circuit for nearest euclidean distance search. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1309–1312, Dec 2006.
- [2] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997.
- [3] AT&T Laboratories Cambridge. ORL Face Database. <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.
- [4] X. Bai, C. Li, X. Yang, and L. J. Latecki. Shape Retrieval and Classification Based on Geodesic Paths in Skeleton Graphs. *Graph-Based Methods in Computer Vision: Developments and Applications: Developments and Applications*, page 190, 2012.
- [5] D. Batas and H. Fiedler. A Memristor SPICE Implementation and a New Approach for Magnetic Flux-Controlled Memristor Modeling. *IEEE Transactions on Nanotechnology*, 10(2):250–255, Mar. 2011.

- [6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr 2002.
- [7] R. Berdan and C. Lim. A Memristor SPICE Model Accounting for Volatile Characteristics of Practical ReRAM. *Electron Device Letters, IEEE*, 35(1):2013–2015, 2014.
- [8] C. Bielza and P. Larrañaga. Discrete Bayesian Network Classifiers: A Survey. *ACM Computing Surveys*, 47(1):5:1–5:43, July 2014.
- [9] C. Bielza and P. Larranaga. Bayesian networks in neuroscience: a survey. *Frontiers in Computational Neuroscience*, 8:131, 2014.
- [10] J. Bill and R. Legenstein. A compound memristive synapse model for statistical learning through STDP in spiking neural networks. *Frontiers in Neuroscience*, 8:412, 2014.
- [11] Z. Biolek, D. Biolek, and V. Biolkova. SPICE model of memristor with nonlinear dopant drift. *Radioengineering*, 18(1):210–214, 2009.
- [12] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
- [13] J. Borghetti, G. Snider, P. Kuekes, J. Yang, D. Stewart, and R. Williams. Memristive switches enable Stateful logic operations via material implication. *Nature*, 464(7290):873–876, 2010.

- [14] J. Buckingham and D. Willshaw. Performance characteristics of the associative net. *Network: Computation in Neural Systems*, 3(4):407–414, 1992.
- [15] H. Cai, V. W. Zheng, and K. Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, PP(99):1–1, 2018.
- [16] R. Carvajal, J. Ramirez-Angula, and J. Tombs. High-speed high-precision voltage-mode MIN/MAX circuits in CMOS technology. *IEEE International Symposium on Circuits and Systems (ISCAS). Emerging Technologies for the 21st Century. Proceedings*, 5:13–16, 2000.
- [17] Y. Cassuto and K. Crammer. In-memory hamming similarity computation in resistive arrays. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 819–823, 2015.
- [18] K.-T. T. Cheng and D. B. Strukov. 3D CMOS-memristor Hybrid Circuits: Devices, Integration, Architecture, and Applications. In *Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design, ISPD '12*, pages 33–40, New York, NY, USA, 2012. ACM.
- [19] R. Cheng and Y. Jin. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 291:43–60, 2015.
- [20] D. M. Chickering, D. Heckerman, and C. Meek. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, Dec. 2004.

- [21] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *European conference on Computer vision*, pages 492–505. Springer, 2010.
- [22] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.
- [23] L. Chua. Memristor-the missing circuit element. *Circuit Theory, IEEE Transactions on* 18, 18(5):507–519, 1971.
- [24] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.
- [25] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Subgraph Transformations for the Inexact Matching of Attributed Relational Graphs. In *Graph Based Representations In Pattern Recognition*, pages 43–52. Springer, 1998.
- [26] C. P. de Campos, G. Corani, M. Scanagatta, M. Cuccu, and M. Zaffalon. Learning extended tree augmented naive structures. *International Journal of Approximate Reasoning*, 68(Supplement C):153–163, 2016.
- [27] C. P. de Campos, M. Cuccu, G. Corani, and M. Zaffalon. *Extended Tree Augmented Naive Classifier*, pages 176–189. Springer International Publishing, Cham, 2014.
- [28] D. de Santos-Sierra, M. F. Arriaga-Gomez, G. Bailador, and C. Sanchez-Avila. Low computational cost multilayer graph-based segmentation algorithms for hand recognition on mobile phones. In *2014 International Carnahan Conference on Security Technology (ICCST)*, pages 1–5, 2014.

- [29] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.
- [30] M. Deshpande. FPGA Implementation & Acceleration of Building Blocks for Biologically Inspired Computational Models. Master’s thesis, Portland State University, 2011.
- [31] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2383–2395, 2011.
- [32] K. Eshraghian, O. Kavehei, K. Cho, J. M. Chappell, A. Iqbal, S. F. Al-Sarawi, and D. Abbott. Memristive Device Fundamentals and Modeling: Applications to Circuits and Systems Simulation. *Proceedings of the IEEE*, 100(6):1991–2007, 2012.
- [33] D. Fan, M. Sharad, A. Sengupta, and K. Roy. Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing. *IEEE Transactions on Neural Networks and Learning Systems*, 27(9):1907–1919, 2016.
- [34] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. *Mach. Learn.*, 29(2-3):131–163, 1997.
- [35] N. Friedman and M. Goldszmidt. Building Classifiers using Bayesian Networks. In *In Proceedings of the thirteenth national conference on artificial intelligence*, pages 1277–1284. AAAI Press, 1996.

-
- [36] D. M. Gavrila and V. Philomin. Real-time object detection for “smart” vehicles. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 87–93, 1999.
- [37] Y. Gdalyahu and D. Weinshall. Flexible Syntactic Matching of Curves and Its Application to Automatic Hierarchical Classification of Silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999.
- [38] N. Ge, J. Yoon, M. Hu, E. Merced-Grafals, N. Davila, J. Strachan, Z. Li, H. Holder, Q. Xia, X. Z. Williams, R. S., and J. Yang. An efficient analog hamming distance comparator realized with a unipolar memristor array: a showcase of physical computing. *Scientific reports*, 7:40135, 2017.
- [39] D. George and J. Hawkins. A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex. *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 3:1812–1817, 2005.
- [40] D. George and J. Hawkins. Towards a mathematical theory of cortical microcircuits. *PLoS Computational Biology*, 5(10):e1000532, 2009.
- [41] A. Georgiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.
- [42] Z. Ghahramani. An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42, 2001.

- [43] C. Gomila and F. Meyer. Graph-based object tracking. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, volume 2, pages II–41. IEEE, 2003.
- [44] B. Graham and D. Willshaw. Capacity and information efficiency of the associative net. *Network: Computation in Neural Systems*, 8(1):35–54, 1997.
- [45] A. Gupta and Y. Singh. Analysis of Hamming network and MAXNET of neural network method in the string recognition. In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pages 38–42. IEEE, 2011.
- [46] S. Haykin. *Neural networks: a comprehensive foundation*, volume 13. Pearson Education, Second edition, 1994.
- [47] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554–2558, 1982.
- [48] M. Hu, H. Li, Q. Wu, and G. Rose. Hardware Realization of BSB Recall Function Using Memristor Crossbar Arrays. In *Design Automation Conference (DAC), 49th ACM/EDAC/IEEE*, pages 498–503, 2012.
- [49] J. Huang, G. Zhou, X. Zhou, and R. Zhang. A new FPGA architecture of fast and brief algorithm for on-board corner detection and matching. *Sensors*, 18(4):1014, 2018.

- [50] D. P. Huttenlocher, R. H. Lilien, and C. F. Olson. View-based recognition using an eigenspace approximation to the Hausdorff measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):951–955, 1999.
- [51] G. Indiveri and S. C. Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015.
- [52] L. Ionescu, A. Mazare, G. Serban, V. Barbu, and A. Constantin. Fpga implementation of an associative content addressable memory. In *Applied Electronics (AE), 2011 International Conference on*, pages 1–4, Sept 2011.
- [53] K. Jayech. Clustering and Bayesian network for image of faces classification. *International Journal of Advanced Computer Science and Applications (IJACSA)*, (Special Issue: Image Processing):239–274, 2011.
- [54] K. Jayech and M. A. Mahjoub. New approach using Bayesian Network to improve content based image classification systems. *IJCSI International Journal of Computer Science Issues*, 7(6):53–62, 2010.
- [55] F. V. Jensen et al. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.
- [56] S. Jo, K. Kim, and W. Lu. High-density crossbar arrays based on a si memristive system. *Nano letters*, 9(2):870–874, 2009.
- [57] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Letters*, 10(4):1297–1301, 2010.

- [58] S. H. Jo and W. Lu. CMOS compatible nanoscale nonvolatile resistance switching memory. *Nano Letters*, 8(2):392–397, 2008.
- [59] C. M. Jung, J. M. Choi, and K. S. Min. Two-step write scheme for reducing sneak-path leakage in complementary memristor array. *IEEE Transactions on Nanotechnology*, 11(3):611–618, May 2012.
- [60] R. Kaplan, L. Yavits, and R. Ginosar. Rassa: Resistive pre-alignment accelerator for approximate dna long read mapping. *IEEE Micro*, 2018.
- [61] K.-H. Kim, S. Hyun Jo, S. Gaba, and W. Lu. Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Applied Physics Letters*, 96(5):053106, 2010.
- [62] K. M. Kim, J. Zhang, C. Graves, J. J. Yang, B. J. Choi, C. S. Hwang, Z. Li, and R. S. Williams. Low-power, self-rectifying, and forming-free memristor with an asymmetric programming voltage for a high-density crossbar application. *Nano letters*, 16(11):6724–6732, 2016.
- [63] S. Kim, H. Y. Jeong, S. K. Kim, S.-Y. Choi, and K. J. Lee. Flexible memristive memory array on plastic substrates. *Nano Letters*, 11(12):5438–5442, 2011. PMID: 22026616.
- [64] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.
- [65] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):49–60, 1988.

- [66] O. Krestinskaya, T. Ibrayev, and A. P. James. Hierarchical temporal memory features with memristor logic circuits for pattern recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(6):1143–1156, June 2018.
- [67] S. Kvatinsky and E. G. Friedman. TEAM: threshold adaptive memristor model. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 60(1):211–221, 2013.
- [68] M. Laiho, J. K. Poikonen, E. Lehtonen, M. Pankaala, J. H. Poikonen, and P. Kanerva. A 512-Cell Associative CAM/Willshaw Memory with Vector Arithmetic. In *IEEE International Symposium on Circuits and Systems (ISCAS)*,, pages 1350–1353, 2015.
- [69] E. Lehtonen and M. Laiho. CNN using memristors for neighborhood connections. *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pages 1–4, Feb. 2010.
- [70] E. Lehtonen, J. Tissari, J. Poikonen, M. Laiho, and L. Koskinen. A cellular computing architecture for parallel memristive stateful logic. *Microelectronics Journal*, 45(11):1438–1449, 2014.
- [71] B. Leiner, V. Lorena, T. Cesar, and M. Lorenzo. Hardware architecture for fpga implementation of a neural network and its application in images processing. In *Electronics, Robotics and Automotive Mechanics Conference, 2008. CERMA '08*, pages 405–410, Sept 2008.
- [72] P. Leray and O. C. H. Francois. BNT structure learning package: Documentation and experiments. Technical report, Laboratoire PSI, INSA de Rouen, 2004.

-
- [73] J. Li, R. K. Montoye, M. Ishii, and L. Chang. 1 Mb $0.41 \mu\text{m}^2$ 2T-2R Cell Non-volatile TCAM with Two-Bit Encoding and Clocked Self-Referenced Sensing. *IEEE Journal of Solid-State Circuits*, 49(4):896–907, 2014.
- [74] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
- [75] M. A. Mahjoub, N. Ghanmy, I. Miled, et al. Multiple models of Bayesian networks applied to offline recognition of Arabic handwritten city names. *arXiv preprint arXiv:1301.4377*, 2013.
- [76] H. Manem, G. S. Rose, X. He, and W. Wang. Design considerations for variation tolerant multilevel CMOS/Nano memristor memory. In *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI, GLSVLSI '10*, pages 287–292, New York, NY, USA, 2010. ACM.
- [77] A. Mathuria and D. W. Hammerstrom. Approximate pattern matching using hierarchical graph construction and sparse distributed representation. In *Proceedings of the International Conference on Neuromorphic Systems*, pages 1–10, 2019.
- [78] H. J. Mattausch, T. Gyohten, Y. Soda, and T. Koide. Compact Associative-Memory Architecture with Fully Parallel Search Capability for the Minimum Hamming Distance. *IEEE Journal of Solid-State Circuits*, 37(2):218–227, 2002.
- [79] H. J. Mattausch, W. Imafuku, A. Kawabata, T. Ansari, M. Yasuda, and T. Koide. Associative memory for nearest-Hamming-distance search based on frequency mapping. *IEEE Journal of Solid-State Circuits*, 47(6):1448–1459, 2012.

-
- [80] W. S. McCulloch and W. Pitts. A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY. *Bulletin of Mathematical Biology*, 52(1):99–115, 1943.
- [81] F. Merrikh Bayat, B. Hoskins, and D. B. Strukov. Phenomenological modeling of memristive devices. *Applied Physics A*, 118(3):779–786, 2015.
- [82] F. Miao, J. P. Strachan, J. J. Yang, M. Zhang, I. Goldfarb, A. C. Torrezan, P. Eschbach, R. D. Kelley, G. Medeiros-Ribeiro, and R. S. Williams. Anatomy of a Nanoscale Conduction Channel Reveals the Mechanism of a High-Performance Memristor. *Advanced Materials*, 23(47):5633–5640, 2011.
- [83] K. Miller, K. S. Nalwa, A. Bergerud, N. M. Neihart, and S. Chaudhary. Memristive Behavior in Thin Anodic Titania. *IEEE Electron Device Letters*, 31(7):737–739, 2010.
- [84] P. Morrison and J. J. Zou. Inexact graph matching using a hierarchy of matching processes. *Computational Visual Media*, 1(4):291–307, 2015.
- [85] A. Y. Ng and M. I. Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS’01, pages 841–848, Cambridge, MA, USA, 2001. MIT Press.
- [86] A. S. Oblea, A. Timilsina, D. Moore, and K. A. Campbell. Silver chalcogenide based memristor devices. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–3, 2010.
- [87] G. Palm. On associative memory. *Biological Cybernetics*, 36(1):19–31, 1980.

- [88] B. Parhami. Associative memories and processors: An overview and selected bibliography. *Proceedings of the IEEE*, 61(6):722–730, 1973.
- [89] H.-M. Park and K.-J. Yoon. Multi-attributed graph matching with multi-layer graph structure and multi-layer random walks. *IEEE Transactions on Image Processing*, 27(5):2314–2325, 2017.
- [90] J. D. Park and A. Darwiche. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research*, 21:101–133, 2004.
- [91] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
- [92] Y. V. Pershin and M. D. Ventra. SPICE model of memristive devices with threshold. *Radioengineering*, 22(2):485–489, 2012.
- [93] D. T. Pham and G. A. Ruz. Unsupervised training of Bayesian networks for data clustering. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 465(2109):2927–2948, 2009.
- [94] T. Ploetz. *Deterministic Approximation Methods in Bayesian Inference*. PhD thesis, Technical University of Darmstadt, 2012.
- [95] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.
- [96] D. Querlio, W. S. Zhao, P. Dollfus, J.-O. Klein, O. Bichler, and C. Gamrat. Bioinspired Networks with Nanoscale Memristive Devices that Combine the Unsupervised and Supervised Learning Approaches. *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 203–210, 2012.

- [97] M. S. Qureshi, W. Yi, G. Medeiros-Ribeiro, and R. S. Williams. Ac sense technique for memristor crossbar. *Electronics Letters*, 48(13):757–758, June 2012.
- [98] L. Rabiner and B. Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.
- [99] A. Radwan, M. Zidan, and K. Salama. On the Mathematical Modeling of Memristors. In *International Conference on Microelectronics (ICM)*, pages 284–287. IEEE, 2010.
- [100] A. Rahimi, A. Ghofrani, K. T. Cheng, L. Benini, and R. K. Gupta. Approximate Associative Memristive Memory for Energy-Efficient GPUs. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1497–1502. IEEE, 2015.
- [101] U. Rückert and H. Surmann. Tolerance of a binary associative memory towards stuck-at-faults. In T. Kohonen, editor, *Artificial Neural Networks*, volume 2, pages 1195–1198, 1991.
- [102] Sandia National Laboratories. Xyce Simulator. <https://xyce.sandia.gov>.
- [103] L. G. Shapiro and R. M. Haralick. Structural Descriptions and Inexact Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):504–519, 1981.
- [104] M. Sharad, D. Fan, K. Aitken, and K. Roy. Energy-Efficient Non-Boolean Computing with Spin Neurons and Resistive Memory. *IEEE Transactions on Nanotechnology*, 13(1):23–34, 2014.

- [105] M. Sharad, D. Fan, and K. Roy. Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory. In *Design Automation Conference (DAC), 50th ACM/EDAC/IEEE*, pages 1–6, 2013.
- [106] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. In *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries (Cat. No.98EX173)*, pages 56–62, 1998.
- [107] P. Sheridan, C. Du, and W. Lu. Feature extraction using memristor networks. *IEEE transactions on neural networks and learning systems*, 27(11):2327–2336, 2016.
- [108] L. Shi and T. L. Griffiths. Neural implementation of hierarchical Bayesian inference by importance sampling. In *Advances in Neural Information Processing Systems*, pages 1669–1677, 2009.
- [109] A. Shokoufandeh, Y. Keselman, M. F. Demirci, D. Macrini, and S. Dickinson. Many-to-many feature matching in object recognition: a review of three approaches. *IET Computer Vision*, 6(6):500–513, November 2012.
- [110] K. Siddiqi, A. Shokoufandeh, S. J. Dickenson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.
- [111] G. S. Snider. Spike-Timing-Dependent Learning in Memristive Nanodevices. In *IEEE International Symposium on Nanoscale Architectures, (NANOARCH)*, pages 85–92, 2008.
- [112] K. Steinbuch and U. Piske. Learning matrices and their applications. *Electronic Computers, IEEE Transactions on*, EC-12(6):846–862, Dec 1963.

- [113] D. Strukov and K. Likharev. Reconfigurable Hybrid CMOS/Nanodevice Circuits for Image Processing. *IEEE Transactions on Nanotechnology*, 6(6):696–710, 2007.
- [114] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453(7191):80–83, May 2008.
- [115] M. Suri, V. Parmar, A. Singla, R. Malviya, and S. Nair. Neuromorphic hardware accelerated adaptive authentication system. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1206–1213, Dec 2015.
- [116] M. M. A. Taha, W. Chavez, and C. Teuscher. Spatial and Temporal Probabilistic Inference Using a Memristive Associative Memory. *International Journal of Unconventional Computing*, 13(2):117–137, 2017.
- [117] M. M. A. Taha and M. Perkowski. Realization of arithmetic operators based on stochastic number frequency signal representation. In *2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 215–220, May 2018.
- [118] M. M. A. Taha and C. Teuscher. Naive Bayesian Inference of Handwritten Digits Using a Memristive Associative Memory. In *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 139–140, July 2017.
- [119] M. M. A. Taha and C. Teuscher. Approximate Memristive In-Memory Hamming Distance Circuit. *J. Emerg. Technol. Comput. Syst.*, 16(2), Mar. 2020.
- [120] M. M. A. Taha, W. Woods, and C. Teuscher. Approximate In-Memory Hamming Distance Calculation With A Memristive Associative Memory. In *Nanoscale Ar-*

- chitectures (NANOARCH), 2016 IEEE/ACM International Symposium on*, pages 159–164, 2016.
- [121] USCD Computer Vision. Yale Face Database. <http://vision.ucsd.edu/content/yale-face-database>.
- [122] R. C. Veltkamp and M. Hagedoorn. State of the art in shape matching. In M. S. Lew, editor, *Principles of Visual Information Retrieval*, pages 87–119. Springer-Verlag, London, UK, 2001.
- [123] M. Vento and P. Foggia. Graph matching techniques for computer vision. *Graph-Based Methods in Computer Vision: Developments and Applications: Developments and Applications*, page 1, 2012.
- [124] L. Wang and H. Zhao. Learning a Flexible K-Dependence Bayesian Classifier from the Chain Rule of Joint Probability Distribution. *Entropy*, 17(6):3766–3786, 2015.
- [125] W. Wang, T. T. Jing, and B. Butcher. FPGA based on integration of memristors and CMOS devices. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1963–1966, May 2010.
- [126] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197)(1):960–962, 1969.
- [127] W. Woods, M. M. A. Taha, D. Tran, J. Bürger, and C. Teuscher. Memristor Panic - A Survey of Different Device Models in Crossbar Architectures. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 106–111, 2015.

- [128] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov, and Others. Memristor- CMOS Hybrid Integrated Circuits for Reconfigurable Logic. *Nano Letters*, 9(10):3640–3645, 2009.
- [129] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu. Consistency-driven alternating optimization for multigraph matching: A unified approach. *IEEE Transactions on Image Processing*, 24(3):994–1009, 2015.
- [130] Y. Yang and W. Lu. Nanoscale resistive switching devices: mechanisms and modeling. *Nanoscale*, 5(21):10076–92, Nov. 2013.
- [131] O. Yusuke, I. Makoto, and A. Kunihiro. Hierarchical Multi-Chip Architecture for High Capacity Scalability of Fully Parallel Hamming-Distance Associative Memories. *IEICE Trans. Electron.*, E87-C(11):1847–1855, 2004.
- [132] C. T. Zahn and R. Z. Roskies. Fourier Descriptors for Plane Closed Curves. *IEEE Transactions on Computers*, C-21(3):269–281, 1972.
- [133] F. Zhou and F. De la Torre. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–134. IEEE, 2012.
- [134] S. Zhu. *Associative Memory as a Bayesian Building Block*. PhD thesis, Portland State University, 2008.
- [135] X. Zhu, X. Yang, C. Wu, J. Wu, and X. Yi. Hamming network circuits based on cmos/memristor hybrid design. *IEICE Electronics Express*, 10(12):1–9, 2013.
- [136] F. Zokaee, H. R. Zarandi, and L. Jiang. Aligner: A process-in-memory architecture for short read alignment in rerams. *IEEE Computer Architecture Letters*, 17(2):237–240, 2018.

- [137] A. M. Zyarah and D. Kudithipudi. Neuromemrisitive Architecture of HTM with On-Device Learning and Neurogenesis. *J. Emerging Technologies and Computing Systems*, 15(3):24:1–24:24, May 2019.