In-memory Machine Learning using Hybrid Decision Trees and Memristor Crossbars

Akash Chavan
Dept. of Computer Science and Engineering
Oakland University
Rochester, MI, USA
akashchavan@oakland.edu

Pranav Sinha
Dept. of Computer Science and Engineering
Oakland University
Rochester, MI, USA
pranavsinha@oakland.edu

Sunny Raj
Dept. of Computer Science and Engineering
Oakland University
Rochester, MI, USA
raj@oakland.edu

2023 IEEE International Symposium on Smart Electronic Systems (iSES) | 979-8-3503-8324-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISES58672.2023.00058

***Abstract*—We propose a method to design in-memory hybrid decision tree (HDT) circuits using memristor crossbars. Decision Trees (DTs) are a well known machine learning algorithm that carries multiple benefits when compared to deep neural networks. They are easily interpretable, fast and they require less data to train. These benefits make them a popular choice in wide-ranging applications that include edge devices and particle physics. We propose a HDT coupled with a multilayer perceptron (MLP) for creating a flexible nonlinear decision boundary which leads to better accuracy. Using this approach, we obtain a test accuracy of 0.90 for the MNIST dataset, which outperforms the state-of-the-art (SOTA). We map this decision tree onto crossbars which are purely memristor based. They utilize zero transistor and one memristor per cell and employ sneak-paths for flow-based in-memory computations. Due to the absence of transistors, our designs are radiation degradation resistant, serving their application in radiation-rich environments, and require less switching energy, making them energy efficient.**

***Index Terms*—In-memory computation, memristors, flow-based computing, decision trees, multivariate decision trees, multilayer perceptron.**

I. Introduction

Flow-based memristor crossbar designs have demonstrated state-of-the-art energy efficiency in various standard benchmark circuits, including the RevLib benchmark [1], [2]. Yet, devising circuits using flow-based computing for general functions has proven challenging. Existing flow-based computing methods employ Binary Decision Diagrams (BDDs) and similar data structures as an intermediate step, potentially leading to exponential complexity, particularly in cases involving common functions like multiplication [3], [4]. As a result, creating flow-based circuits for intricate machine learning algorithms via BDDs presents significant challenges. In this paper, we depart from BDD-related data structures and design machine learning circuits using decision trees [5].

DTs, with their lower complexity and reduced energy requirements during inference, offer practicality for energy-limited settings like edge computing and mobile devices [6], [7]. DTs excel in scenarios with limited training data, making them the preferred choice for prediction tasks where data availability is scarce [8]. Adversarial attacks pose a significant threat to neural networks, causing unexpected outputs with minor input changes, rendering them unsuitable for safety-critical systems [9]. DTs, in contrast, do not encounter these problems and can be applied in scenarios demanding safety assurances, resilience, confidentiality, and protection [10], [11]. Furthermore, neural network's output is not easy to interpret, while the output generated by DTs can be explained easily [12]. DTs are also employed when prioritizing speed and energy efficiency among machine learning algorithms for processing input data [7].

Concisely, our paper introduces an algorithm for generating decision tree circuit designs that leverage the advantages of in-memory flow-based computing with memristor crossbars including improved throughput, and resistance to radiation and drift. Our work encompasses several key contributions:

- We introduce a HDT algorithm that enhances accuracy over univariate DTs, leveraging multiple features to produce an optimal decision boundary.
- We evaluated our algorithm on the popular MNIST dataset [13] and obtain a test accuracy of 0.90 which outperforms the SOTA [14]. While this design occupies slightly more space and uses more energy, a pure memristor-based circuit still carries many advantages including application in radiation-rich environments and less switching energy for its operation.

II. Related Work

Several hardware designs for decision trees have been introduced in the literature, achieving impressive throughput up to 10^8 inferences per second and energy efficiency in the nJs per inference range [15], [16]. However, these designs mostly incorporate traditional CMOS circuitry, including various cell configurations such as 1T1R and 6T2R, as well as conventional CMOS peripheral components for computation. Sinha and Raj [14] generate purely memristor-based decision tree hardware designs but their approach is less accurate due to the utilization of univariate decision trees. These existing approaches either use CMOS circuitry or lack a methodology to design hardware for multivariate decision trees, which are a more generalized form of univariate trees, capable of achieving enhanced accuracy on specific datasets. In our paper, we present an algorithm for generating hardware designs that cater to both univariate and multivariate decision trees. We

2832-3602/23/$31.00 ©2023 IEEE
DOI 10.1109/iSES58672.2023.00058

follow [14] for the standardized experimentation protocol, mapping of the HDTs onto the crossbar designs, and also for crossbar design verification.

A. Decision Tree

The common DTs, where every node evaluates a singular attribute's value, can be characterized as axis-parallel due to the fact that the assessments at each node equate to axis-parallel hyperplanes within the input space. Subsequently, DTs that involve nodes evaluating a linear combination of attributes have been put forward by various researchers under different names: Linear Machine Decision Tree [17], Multivariate Decision Trees [18], Oblique Decision Trees [19], Randomly Partitioned Multivariate Decision Tree, PCA partitioned Multivariate Decision Tree [20] etc. The tests associated with each node are equivalent to hyperplanes in general position, and they partition the input space into polyhedra [21]. Multivariate decision trees are specifically engineered to encapsulate intricate interactions among multiple features, which offers the potential to enhance predictive performance beyond that of simple univariate decision trees [18].

The concept of neural trees emerged as an integration of neural networks and DTs. Within the literature, neural trees can be categorized into two main groups. The first group leverages DTs as a foundational structure for the neural network. The fundamental concept involves crafting a decision tree and subsequently transforming it into a neural network model. This includes Entropy nets [22], Fast training algorithms for multilayer neural nets [23], Continuous ID3 algorithm [24] etc. The second group uses neural networks as building blocks in DTs. The nonlinear multivariate decision tree with MLPs at the internal nodes was proposed by Guo and Gelfand [25]. Behnke and Karayiannis used competitive learning to form a competitive decision tree architecture named CNET [26]. A hybrid form that contains neural networks at the leaves of the tree and univariate nodes in the non-leaf nodes of the tree was introduced by Utgoff [27]. We employ the MLP model as the non-linear, multivariate decision node. Because MLP is a universal approximator and can approximate any function given a sufficiently big non-linear basis function, a decision tree need not have any node more complex than such a node [28].

B. Flow-based computing

In flow-based computing paradigm, the rows and columns of a crossbar act as nodes, and the memristors provide connections between the nodes. Memristors are assigned labels in the crossbar, which decides whether the memristor is configured with the same value as the input or the negation ($\neg$) of the input. The input, which is dependent on the memristor labels, is loaded on the crossbars during run time. Memristors with a value of 1 are considered to be of low resistance and act as a closed circuit, whereas a value of 0 would be considered to be of high resistance and act as an open circuit. A current is then applied to the bottommost row and the configuration of the memristors decides the route the current takes in the crossbar. Configuration energy required for the memristors are in the Femto Joules scale, making it very energy efficient [29].

An illustrative example is shown in Figure 1, where we use a HDT generated by the proposed algorithm for the MONK's problem dataset available in the UCI machine learning repository [30]. The dataset consists of 6 features and two classes. All the input features have been scaled to be between 0 and 3, for a bit length of 2 per feature. The inequalities at the HDT nodes are used to create a truth table by assigning 0 to values less than or equal to the threshold and 1 otherwise. The boolean formula from truth table is converted into a reduced-order binary decision diagram (ROBDD), which can then be rearranged into a bipartite graph [31], also known as the Binary Classification Graph (BCG). The BCG is then further mapped onto a crossbar as per the algorithms discussed in [14].

III. Approach

The objective of this paper is to design energy-efficient flow-based circuits for machine learning by employing multivariate decision trees, aiming for enhanced accuracy compared to basic univariate decision trees. We achieve this by developing multivariate decision trees, an extension of the univariate model, expected to improve accuracy on similar datasets. We implement a non-linear multivariate decision tree with multilayer perceptrons at the internal nodes as proposed by Guo and Gelfand [25]. In the context of a standard multivariate decision tree, all available features are collectively utilized at the decision node to arrive at a decision, as a result, these types of trees need more memristors in the synthesized crossbar designs, making them inefficient in terms of space and energy utilization. This utilization of all features can be unnecessary and can result in suboptimal circuit designs in terms of efficiency. To address this challenge, we implement a two-fold strategy where, at the tree level, we limit the number of features to a predefined value K and, at the node level, we allow the node the flexibility to dynamically select a number of features ranging from 2 to K, opting for the value that achieves an optimal data partition at the node. This results in a decision tree where some nodes use less number of features while some use more resulting in optimum crossbar size. On top of this, to strike a balance between energy efficiency and accuracy, we follow an approach similar to Yildiz and Alpaydin [28] where we compare univariate and non-linear multivariate splits based on information gain and take the split with higher information gain and continue tree generation. This ensures that the final decision tree is more accurate compared to simple univariate decision trees while being energy and space-efficient. We experimentally verify that this hybrid approach produces decision trees that are more accurate compared to univariate decision trees.

The pseudo-code for our multilayer perceptron training is presented in Algorithm 1. We first explain how the multilayer perceptron is trained efficiently at each node. This is discussed in Section III-A. Then, in Section III-B, we use MLP in the decision tree training framework.

Algorithm 1 MLP training at decision node

Require: Sample matrix X, label matrix y and, number of features K
Ensure: Multivariate partitions m_l, m_r
Let $C = \{c_1, ..., c_M\}$ be the set of M classes at the decision node.
Partition C into C_L and C_R
Initialize b_f, b_m
for $f = 2$ to K **do**
    Feature selection
    Train MLP using selected features
    $b_f \leftarrow$ best f features
    $b_m \leftarrow$ best trained MLP
end for
$m_l \leftarrow \{x \,|\, x \in X$ and $f(x[b_f]) \leq 0\}$, $m_r \leftarrow \{x \,|\, x \in X$ and $f(x[b_f]) > 0\}$ {$f(.)$ denotes input-output mapping of the model b_m}
return m_l, m_r

Algorithm 2 HDT training

Require: Dataset $\mathbf{X}, \mathbf{y}$; hyperparameters K
Ensure: Decision tree T
Assign X, y to root node of tree T
$stack.push(RootNode)$
while $stack$ not empty **do**
    $N \leftarrow stack.pop()$
    Find univariate split S_u for N
    Find multivariate split S_m using *Algorithm 2*
    if InformationGain(S_u) > InformationGain(S_m) **then**
        $N.split \leftarrow S_u$ {Select univariate split}
    else
        $N.split \leftarrow S_m$ {Select multivariate split}
    end if
    Create child nodes C_1, C_2 partitioned by $N.split$
    if C_1 does not satisfy stopping criteria **then**
        $stack.push(C_1)$
    end if
    if C_2 does not satisfy stopping criteria **then**
        $stack.push(C_2)$
    end if
end while
return Decision tree T

A. MLP Training

The algorithm takes a sample matrix X, a label matrix y, and the maximum number of features K as input. It aims to generate multivariate partitions m_l and m_r. The following subsections describe the steps to obtain these multivariate partitions in more detail.

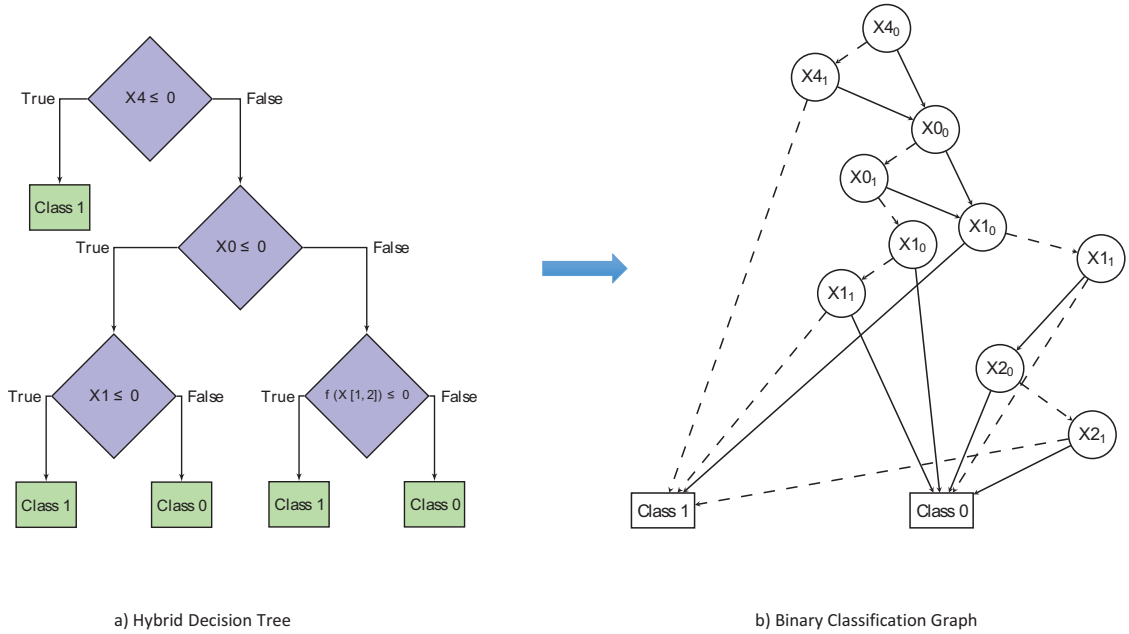
1) Class Partitioning: Guo and Gelfand [25] introduced a nonlinear multivariate decision tree with multilayer perceptrons at the internal nodes. To address the binary nature of the tree nodes, they devised a heuristic for grouping $M > 2$ classes into two. This process entails a nested optimization approach. In the inner optimization, gradient descent is employed to determine weights minimizing mean-squared error, akin to standard neural network training, enabling the identification of a suitable split for the two distinct class groups. The outer optimization leverages the exchange heuristic to perform a local search with backtracking, achieving the best split of M classes into two groups with a time complexity of $O(M^2)$. Loh and Shih [32] proposed an unsupervised 2-means clustering algorithm to initially group classes into two superclasses. Our approach aligns with the strategies of [25], [32] but adopts a simpler and deterministic method for class grouping. We construct two distinct subsets of approximately equal size: C_L, comprising samples from the most frequent class and those closely related, and C_R, encompassing samples from the remaining classes.

2) Feature Selection: The key concept is that specific dimensions within the instance space, as it reaches a particular node, may exhibit little variation and can be deemed redundant. Avoiding these features may increase the generalization ability and reduce node complexity [33]. We iterate through the range of features from $f = 2$ to K and select the best features that can distinguish between the two sets effectively, which can then be efficiently implemented using memristor crossbars. To do this, we use the usual machine-learning strategies for feature selection. We perform the chi-squared test [34] similar to [35] to evaluate the relevance and significance of each feature with respect to the target variable and select the f best features based on their scores.

3) Training: Selected features f are used to train a MLP to separate classes C_L and C_R. We store the best model obtained along with the best features while iterating over the range of features. Using the best model we partition the data into left and right partitions similar to [25].

B. HDT Training

The steps to train the HDT are presented in algorithm 2. The algorithm takes as input a training sample matrix $\mathbf{X}$, containing the feature vectors, the corresponding class label matrix $\mathbf{y}$, and K, the max number of features to use for multivariate decisions. The algorithm outputs a decision tree T, where each decision node is either univariate or non-linear multivariate and the leaf node with the assigned class label. The algorithm maintains a list of nodes that are being worked on and needs further processing using the $stack$ data structure. It is initialized with the root node that contains the complete dataset. In each iteration of the while loop, the best univariate split obtainable using a single feature is calculated; this step utilizes the same Information Gain criteria used by the popular ID3 algorithm. Then, using the MLP, a non-linear multivariate split is obtained, and depending on the information gain of both splits either of the two is selected for inclusion in the decision tree. Finally, the data splits produced by the selected decision are inserted into the $stack$ for further processing if they do not meet the stopping criteria. If the selected splits meet the stopping criteria, they are assigned a label and not



a) Hybrid Decision Tree

b) Binary Classification Graph

Fig. 1. A HDT is trained using the MONK's problem dataset. The inequalities at the decision nodes in the HDT are transformed into a truth table. This truth table is then employed to construct an equivalent ROBDD, effectively substituting the decisions. This transformation results in the conversion of the HDT into a BCG, which subsequently serves as the basis for deriving the crossbar design.

TABLE I
ACCURACY, CROSSBAR SIZE, AND ENERGY UTILIZATION COMPARISON BETWEEN UNIVARIATE AND HYBRID DECISION TREES. HDTS ARE MORE ACCURATE PROVIDED ITS CAPABILITY TO CREATE A NONLINEAR BOUNDARY USING MULTIPLE FEATURES.

	Accuracy			Size		Energy (pJ)	
Dataset	DT [14]	HDT	Delta	DT [14]	HDT	DT [14]	HDT
Iris	0.9	**0.93**	0.03	20×18	10×9	0.58	0.24
Wine	0.92	**1.00**	0.08	41×39	48×48	1.34	1.62
Banknote	**1.00**	**1.00**	0	51×48	36×34	1.6	1.10
Car-evaluation	0.86	**0.90**	0.04	10×14	34×34	0.18	1.00
Ionosphere	0.96	**0.99**	0.03	51×45	58×51	1.52	1.88
Balance-scale	0.79	**0.90**	0.11	283×260	92×93	8.84	3.05
Indian-Diabetes	0.77	**0.81**	0.04	435×408	132×128	14.92	4.76
Tic-tac-toe	0.96	**0.98**	0.02	32×32	254×242	1.2	8.61
Monk1	0.8	**0.92**	0.12	28×25	27×24	0.84	0.76
Statlog-shuttle	**1.00**	**1.00**	0	178×167	384×385	5.8	15.02
MNIST	0.86	**0.90**	0.04	3391×3237	14068×13935	118.12	506.16

processed further. We utilize the majority class proportion-based stopping criteria proposed in [20].

IV. RESULTS

We train our HDT algorithm on multiple datasets from the UCI machine learning repository and the MNIST dataset [13], [30]. The crossbar design is synthesized from the obtained tree and is compared with the ones generated through the univariate decision tree proposed in [14] on the basis of accuracy, size and energy utilization. We performed all the tests on an AMD Ryzen 9 7950X 5.7 GHz CPU with 128 GB of memory.

Figure 2 shows the crossbar design generated from the BCG. Each memristor has a label associated with it. The label decides whether the memristor is configured with the same value as the input or the negation ($\neg$) of the input. It also tells the feature index and the bit value. Once the crossbar design is ready, depending on the input, the values for the labels are loaded with either 0 or 1. During execution a current is applied to the bottommost row and the output is received in one of the top M rows, M being the number of classes, which is 2 in this case. The memristor values decide the route the current takes in the crossbar, and since the current ends up in the Class 0 line, we classify the input to be of Class 0.

We also tune hyperparameters to obtain the best accuracy. We primarily make use of three hyperparameters: maximum features to utilize K, δ to calculate the stopping criteria, and maximum tree depth. We perform a grid search with a range of [2, number of features] for K, [0.85, 0.995] for δ, and [1, 10] for the maximum tree depth. To train the decision trees, each feature of the input is scaled to be between 0 and 2^L-1, where

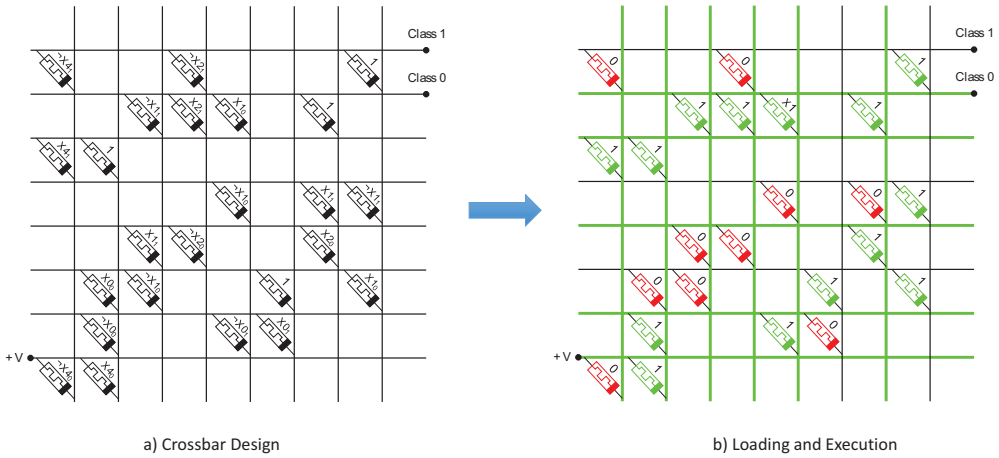


Fig. 2. (a) Crossbar design for classifying the MONK's problem dataset with labeled memristors, obtained from the BCG from Figure 1 and (b) loading and execution of the crossbar for input data in class 0. Current is initially applied to the lowest row and all the wires with a current in them have been shown in green. A green class 0 wire indicates the classification output as 0.

TABLE II
COMPARISON OF OUR METHOD TO THE SOTA. WE REPORT TREE-DEPTH, ENERGY, AREA, AND ACCURACY NUMBERS FOR THE MNIST DATASET.

Method	Process (nm)	Tree Depth	Energy (nJ)	Area (mm^2)	Accuracy
Intel X5560 [36]	45	6	2.04×10^7	-	-
Nvidia Tesla M2050 [36]	40	6	1.10×10^7	-	-
Xilinx Virtex-6 [36]	40	6	3.51×10^5	-	-
ASIC [37]	65	-	1.87×10^5	6.50	-
ASIC [38]	65	-	4.60×10^5	2.30	-
ASIC IMC [39]	65	6	19.4	0.56	-
ACAM [40]	65	10	1.28	1.80	-
DT (worst-case) [14]	70	10	2.05×10^{-2}	5.23×10^{-3}	-
DT (MNIST) [14]	70	10	1.74×10^{-2}	9.40×10^{-4}	0.82
HDT (MNIST)	70	10	0.5	0.96	**0.90**

L is the bit length of the feature. The decision tree is trained on 80% of the dataset while keeping the rest for testing. The memristors used are the ones synthesized by Goux et al. which have energy utilization and cell width of 10 fJ and 70 nm respectively [41]. The number of programmable memristors in the crossbar multiplied by 10 fJ yields the energy utilization of the complete crossbar design. To compare univariate and hybrid decision trees, we report the best accuracy among feature bit lengths from 1 to 8 and their corresponding energy utilization, and the crossbar size of the design.

Table I shows the comparison between the univariate and hybrid decision trees based on their highest accuracy among the tested 1 to 8-bit length, energy consumption and space utilization. As expected, the test accuracy of HDT outperforms the univariate DT. However, this increase in accuracy comes with a substantial increase in the crossbar size which leads to a higher energy consumption.

The comparison of our designs to the SOTA is presented in Table II. In our setup, we follow a strategy of [14] where we train our model for the popular MNIST dataset with 80% of the data for training and the rest for testing. The maximum tree depth is set to be 10. The univariate test accuracy is 0.82 with a crossbar area of $9.40 \times 10^{-4} mm^2$ consuming 1.74×10^{-2}nJ of energy [14]. However, when we incorporate multiple features using the hybrid decision tree, we obtain a test accuracy of 0.90 with a crossbar area of $0.96 mm^2$ consuming 0.5nJ of energy. The increase in size can be attributed to the increase in the number of features that were considered while training the hybrid decision tree.

V. CONCLUSION

We develop high-accuracy decision tree acceleration circuits based on flow-based memristor crossbars. These circuits incorporate sneak paths and 0T1R memristor crossbars to resist issues like resistance drift and radiation-induced degradation. Our approach termed the hybrid decision tree, combines a MLP to create flexible, nonlinear decision boundaries. We also experimentally verify that our HDTs produce flow-based designs that are more accurate compared to univariate DTs. We verify this by testing the proposed algorithm on various datasets including the MNIST dataset where we achieve a test accuracy of 0.90, which outperforms the SOTA.

VI. ACKNOWLEDGMENT

We acknowledge support from NSF Award #2245756 to Sunny Raj.

REFERENCES

[1] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "Revlib: An online resource for reversible functions and reversible circuits," in *38th International Symposium on Multiple Valued Logic (ismvl 2008)*. IEEE, 2008, pp. 220–225.

[2] S. Thijssen, S. K. Jha, and R. Ewetz, "Compact: Flow-based computing on nanoscale crossbars with minimal semiperimeter," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 232–237.

[3] R. E. Bryant, "On the complexity of vlsi implementations and graph representations of boolean functions with application to integer multiplication," *IEEE transactions on Computers*, vol. 40, no. 2, pp. 205–213, 1991.

[4] P. Woelfel, "Bounds on the obdd-size of integer multiplication via universal hashing," *Journal of Computer and System Sciences*, vol. 71, no. 4, pp. 520–534, 2005.

[5] D. H. Moore II, "Classification and regression trees, by leo breiman, jerome h. friedman, richard a. olshen, and charles j. stone. brooks/cole publishing, monterey, 1984,358 pages, $27.95," *Cytometry*, vol. 8, no. 5, pp. 534–535, 1987.

[6] R. S. Somesula, R. Joshi, and S. Katkoori, "On feasibility of decision trees for edge intelligence in highly constrained internet-of-things (iot)," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, 2023, pp. 217–218.

[7] M. Al Moteri, S. B. Khan, and M. Alojail, "Machine learning-driven ubiquitous mobile edge computing as a solution to network challenges in next-generation iot," *Systems*, vol. 11, no. 6, p. 308, 2023.

[8] A. S. Cornell, W. Doorsamy, B. Fuks, G. Harmsen, and L. Mason, "Boosted decision trees in the era of new physics: a smuon analysis case study," *Journal of High Energy Physics*, vol. 2022, no. 4, pp. 1–36, 2022.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[10] D. Puthal, E. Damiani, and S. P. Mohanty, "Secure and scalable collaborative edge computing using decision tree," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2022, pp. 247–252.

[11] A. Akavia, M. Leibovich, Y. S. Resheff, R. Ron, M. Shahar, and M. Vald, "Privacy-preserving decision trees training and prediction," *ACM Transactions on Privacy and Security*, vol. 25, no. 3, pp. 1–30, 2022.

[12] Y. Izza, A. Ignatiev, and J. Marques-Silva, "On tackling explanation redundancy in decision trees," *Journal of Artificial Intelligence Research*, vol. 75, pp. 261–321, 2022.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] P. Sinha and S. Raj, "Designing energy-efficient decision tree memristor crossbar circuits using binary classification graphs," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–9.

[15] G. Pedretti, C. E. Graves, S. Serebryakov, R. Mao, X. Sheng, M. Foltin, C. Li, and J. P. Strachan, "Tree-based machine learning performed in-memory with memristive analog cam," *Nature communications*, vol. 12, no. 1, pp. 1–10, 2021.

[16] M. Rakka, M. E. Fouda, R. Kanj, and F. Kurdahi, "Dt2cam: A decision tree to content addressable memory framework," *arXiv preprint arXiv:2204.06114*, 2022.

[17] P. E. Utgo and C. E. Brodley, "Linear machine decision trees," Citeseer, Tech. Rep., 1991.

[18] C. Brodley and P. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, pp. 45–77, 04 1995.

[19] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *Journal of artificial intelligence research*, vol. 2, pp. 1–32, 1994.

[20] F. Wang, Q. Wang, F. Nie, W. Yu, and R. Wang, "Efficient tree classifiers for large scale datasets," *Neurocomputing*, vol. 284, pp. 70–79, 2018.

[21] K. P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu, "Enlarging the margins in perceptron decision trees," *Machine Learning*, vol. 41, pp. 295–313, 2000.

[22] I. K. Sethi, "Entropy nets: from decision trees to neural networks," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1605–1613, 1990.

[23] R. P. Brent, "Fast training algorithms for multilayer neural nets," *IEEE Transactions on Neural Networks*, vol. 2, no. 3, pp. 346–354, 1991.

[24] K. J. Cios and N. Liu, "A machine learning method for generation of a neural network architecture: A continuous id3 algorithm," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 280–291, 1992.

[25] H. Guo and S. B. Gelfand, "Classification trees with neural network feature extraction," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1992, pp. 183–184.

[26] S. Behnke and N. B. Karayiannis, "Competitive neural trees for pattern classification," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1352–1369, 1998.

[27] P. E. Utgoff, "Perceptron trees: A case study in hybrid concept representations," *Connection Science*, vol. 1, no. 4, pp. 377–391, 1989.

[28] C. Yildiz and E. Alpaydin, "Omnivariate decision trees," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1539–1546, 2001.

[29] S. Goswami, S. Goswami, and T. Venkatesan, "An organic approach to low energy memory and brain inspired electronics," *Applied Physics Reviews*, vol. 7, no. 2, p. 021303, 2020.

[30] D. Dua and C. Graff, "Uci machine learning repository. university of california, school of information and computer science, irvine, ca (2019)," 2019.

[31] D. Chakraborty and S. K. Jha, "Automated synthesis of compact crossbars for sneak-path based in-memory computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 770–775.

[32] W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica sinica*, pp. 815–840, 1997.

[33] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, pp. 261–283, 2013.

[34] P. E. Greenwood and M. S. Nikulin, *A guide to chi-squared testing*. John Wiley & Sons, 1996, vol. 280.

[35] H. Kim and W.-Y. Loh, "Classification trees with unbiased multiway splits," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 589–604, 2001. [Online]. Available: https://doi.org/10.1198/016214501753168271

[36] B. Van Essen, C. Macaraeg, M. Gokhale, and R. Prenger, "Accelerating a random forest classifier: Multi-core, gp-gpu, or fpga?" in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2012, pp. 232–239.

[37] T.-W. Chen, Y.-C. Su, K.-Y. Huang, Y.-M. Tsai, S.-Y. Chien, and L.-G. Chen, "Visual vocabulary processor based on binary tree architecture for real-time object recognition in full-hd resolution," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 20, no. 12, pp. 2329–2332, 2011.

[38] K. J. Lee, G. Kim, J. Park, and H.-J. Yoo, "A vocabulary forest object matching processor with 2.07 m-vector/s throughput and 13.3 nj/vector per-vector energy for full-hd 60 fps video object recognition," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1059–1069, 2015.

[39] M. Kang, S. K. Gonugondla, S. Lim, and N. R. Shanbhag, "A 19.4-nj/decision, 364-k decisions/s, in-memory random forest multi-class inference accelerator," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2126–2135, 2018.

[40] G. Pedretti, C. E. Graves, S. Serebryakov, R. Mao, X. Sheng, M. Foltin, C. Li, and J. P. Strachan, "Tree-based machine learning performed in-memory with memristive analog cam," *Nature communications*, vol. 12, no. 1, pp. 1–10, 2021.

[41] L. Goux, A. Fantini, G. Kar, Y.-Y. Chen, N. Jossart, R. Degraeve, S. Clima, B. Govoreanu, G. Lorenzo, G. Pourtois *et al.*, "Ultralow sub-500na operating current high-performance TiN\Al2O3\HfO2\Hf\TiN bipolar RRAM achieved through understanding-based stack-engineering," in *2012 Symposium on VLSI Technology (VLSIT)*. IEEE, 2012, pp. 159–160.