Explainable Learning-Based Intrusion Detection Supported by Memristors

Jingdi Chen
George Washington University
jingdic@gwu.edu

Lei Zhang
George Washington University
zjiaz_821117@gwu.edu

Joseph Riem
George Washington University
joseph.riem@gwu.edu

Gina Adam
George Washington University
ginaadam@gwu.edu

Nathaniel D. Bastian
United States Military Academy
nathaniel.bastian@westpoint.edu

Tian Lan
George Washington University
tlan@gwu.edu

***Abstract*—Deep learning based methods have demonstrated great success in network intrusion detection. However, the use of Deep Neural Networks (DNNs) makes it difficult to support real-time, packet-level detections in communication networks that handle high-speed traffic with low latency and energy. To this end, this paper proposes a novel approach to efficiently realize a DNN-based classifier by converting it into a pruned, explainable decision tree and evaluating its hardware implementation using an emerging architecture based on memristor devices, in order to support network intrusion detections on the fly. Preliminary experiments on real-world datasets show that the proposed method achieves nearly four orders of magnitude speed up while retaining the desired accuracy.**

***Index Terms*—Reinforcement Learning, Memristors, Hardware-SoftwareCo-design**

I. Introduction

Deep learning has been successfully applied to many network security problems such as packet inspection and network intrusion detection [1]. However, the superior performance of DNNs comes at the cost of using millions or even billions of parameters to achieve universal function approximation, making DNN-based classifiers not suitable for real-time, packet-level detection. Therefore, an explainable, real-time learning-based solution specifically tailored for networks with high speed traffic and low latency is needed.

For network intrusion detection, we propose a new approach that converts a learned DNN-based policy into a decision tree policy – which is inherently explainable – then prunes the decision tree and implements it using a flexible memristor architecture. The conversion leverages a teacher-student training process developed in [2], where the DNN policy acts as the teacher and generates input-output samples to construct the student decision tree. We further adopt cost complexity pruning (CCP) [3] to prune the branches down to a tractable size for real-time network operations and explore its analog hardware implementation.

To implement an adjustable decision tree for network intrusion detection, adjustable components with high efficiency need to be utilized. Memristor devices are great candidates for this endeavor since fully analog circuits with high efficiency can be implemented. We show that decision trees could benefit from efficient hardware implementations as well [4], particularly in resource-constrained environments, e.g. Internet of Things. This work models the proposed decision tree after pruning by providing adjustable decision boundaries implemented using analog memristor-based circuits.

II. Method Details

We formulate network intrusion detection as a one-step Reinforcement Learning (RL), with packet- or flow-level features as states $\mathcal{S}$, detection outcome as actions $\mathcal{A}$, and detection cross-entropy loss $-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$ as rewards R. We use a stochastic policy $\pi(a|s)$ to determine the probability of selecting action $a \in \mathcal{A}$ in each state $s \in \mathcal{S}$ and optimize the expected discounted long-term reward.

To extract the decision tree from a trained DNN policy for intrusion detection, we adopt a teacher-student training methodology. First, we follow the trajectories generated by the teacher DNN and collect state-action pairs according to distribution over states $s \sim d^{\pi}(s)$ for training. Second, we resample the trajectory of state-action pair $\mathcal{D}$ using the sampling probability function $p(s,a) \propto \left(V^{\pi^*}(s) - \min_{a' \in A} Q^{\pi^*}(s,a')\right) \cdot \mathbf{1}\{(s,a) \in \mathcal{D}\}$, where $V(s)$ and $Q(s,a)$ are the value function and state-action value function of RL. Since there is no Q-function available here, we use the maximum entropy formulation (i.e., the inverse of a Boltzmann policy) to obtain Q values, i.e.,$Q(s,a) = \log \pi^*(s,a)$. $\mathbf{1}\{x\}$ is an indicator function equal to 1 only if x is true. We then retrain the decision tree using CART algorithm [5] on the resampled dataset. To enable efficient implementation using memristors, we adopt cost complexity pruning (CCP) to reduce the number of branches according to the requirements of network operators.

Memristors (also known as resistive switches or ReRAM) are novel, two-terminal memory devices that are ultra-scalable to a few nanometers and can have their internal state programmed in an analog fashion and retained in an energy-efficient manner, similar to an artificial synapse [6], [7]. The memristor-based circuit for one decision tree leaf consists of circuitry for memristor read and programming, operational amplifiers for signal amplification and compensation and a

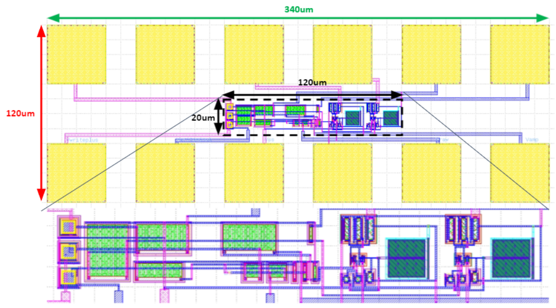

Fig. 1. Design of ReRAM circuit for single node.

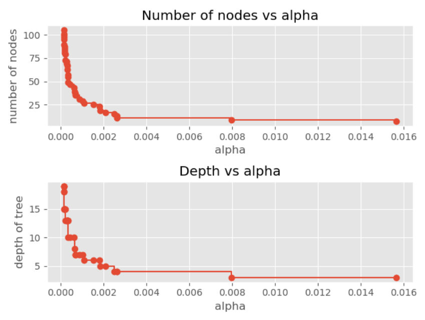

Fig. 2. Greater values of ccp alpha increase the number of nodes pruned.

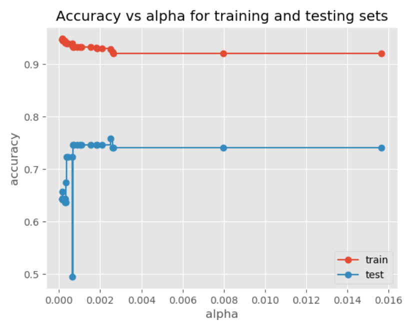

Fig. 3. Setting ccp alpha to 0.002 maximizes the testing accuracy.

block for comparison with the input data. The design is inspired by [4], but was further optimized to reduce the number of resistors and be suitable for an integrated circuit tape-out. The goal was to reduce the area and the delay as needed for the proposed application. The design was simulated in Cadence and taped-out in SkyWater 130nm, which is one of the few technologies for memristor tape-outs available to academic researchers currently.

III. Experiments

We evaluate the Decision Tree Policy Extraction algorithm on the UNSW-NB15 dataset to perform a binary classification to identify malicious traffic from begin. We train a classifier using Multi-layer perceptron (MLP) with one hidden layer with 100 neurons for 8000 steps, extract decision tree policy, and prune it using CCP pruning technique which is parameterized by the cost complexity parameter, ccp alpha.

Without pruning, the extracted decision tree has more than 10,000 nodes. We show in Figure 2 that the number of nodes and tree depth decreases as ccp alpha increases. Then in Figure 3, we see that when ccp alpha is set close to zero, the tree overfits, leading to only 50% testing accuracy. In this example, setting ccp alpha to 0.002 maximizes the testing accuracy and got a decision tree with 16 nodes which is possible to implement in memristor.

We also compare the classification accuracy using F-1 scores and inference time of the DNN classifier, Decision Tree(DT) without pruning, Pruned 16-node Decision Tree for memristor-based implementation in Table I. It shows pruning and implementing decision tree that is extracted from DNN policy on memristor provides human-readable interpretations while preserving nearly no degradation in performance and further allow the computing system to have higher density and to be more energy efficient due to characteristics of our memristor-based circuit.

The area of the circuit design for a single node with pads is 120 μm by 340 μm and without the bare pads is 120 μm by 20 μm, with an estimated power consumption of 150μW. The delay in the node from the programmed boundary to the output is 1.5ms. For the proposed 16-node decision tree, the estimated total area consumption in the memristor-supported 130nm technology is $\approx 0.05mm^2$, the maximum power consumption is $\approx 1mW$ per decision and the maximum delay considered for the longest decision path is $\approx 9ms$. It is important to note that these metrics would be improved further if a state-of-the-art transistor node, e.g. 12nm would be used for integration with memristor technology instead of the current 130nm node explored.

TABLE I
Pruning and implementing the DT that is extracted from DNN policy in memristor is faster than other two methods without losing much accuracy compared to DNN policy.

Method	F-1 Score	Inference Time(s)
DNN	0.82	71.5001
DT (10K+ nodes)	0.67	1.1964
DT (16 node, memristor)	0.75	0.009

IV. Conclusions

Our proposed method could interpret the DNN policy into a pruned decision tree efficiently with lower energy and fast prediction speed for real-time network intrusion. These results suggest future research on the DNN policy explaination using decision trees and implementation in energy-efficient programmable memristors to perform real-time detection.

References

[1] G. Alnwaimi, S. Vahid, and K. Moessner, "Dynamic heterogeneous learning games for opportunistic access in lte-based macro/femtocell deployments," *IEEE Transactions on Wireless Communications*, 2015.
[2] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," 2018. [Online]. Available: https://arxiv.org/abs/1805.08328
[3] R. A. O. Leo Breiman, Jerome H. Friedman and C. J. Stone., "Classification and regression trees," 1987.
[4] P. Grothe and J. Haase, "Memristors as adjustable boundaries for an analog implementation of decision trees," 2019.
[5] F. Zhu, M. Tang, L. Xie, and H. Zhu, "A classification algorithm of cart decision tree based on mapreduce attribute weights," 2018.
[6] D. B. S. Yang, J. Joshua and D. R. Stewart., "Memristive devices for computing." *Nature nanotechnology 8.1*, pp. 13–24, 2013.
[7] F. Kiani, J. Yin, Z. Wang, J. J. Yang, and Q. Xia, "A fully hardware-based memristive multilayer neural network," *Science Advances*, vol. 7, no. 48, p. eabj4801, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abj4801