Visual Object Recognition Accelerator Based on Approximate In-Memory Processing

Yeseong Kim, Mohsen Imani, Tajana Rosing
University of California San Diego
{yek048, moimani, tajana}@ucsd.edu

A part of this work has been accepted and presented in IEEE ICCAD 2017 conference and a part is underview in ISCA 2018 conference.
Y. Kim, M. Imani, T. Rosing "ORCHARD: Visual Object Recognition Accelerator Based on Approximate In-Memory Processing", IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2017.

ABSTRACT

Machine learning has been widely investigated as an autonomous solution which performs diverse classification tasks, e.g., voice recognition and visual object detection. However, running these tasks in conventional computer architectures requires high energy consumption due to large data movement and memory accesses. In this paper, we propose a novel accelerator design, called CHOIR, which performs the classification tasks based on a co-designed boosting algorithm. Since CHOIR processes most classification tasks inside emerging non-volatile memory blocks, we can significantly mitigate computation overhead incurred by the enormous amount of data. In our evaluation conducted on circuit- and device-level simulations, we show that the CHOIR successfully performs practical classification tasks with high accuracy, e.g., more than 96% for practical image recognition problems. In addition, our design significantly improves the performance and energy efficiency by up to 376**x** and 1896**x**, respectively, compared to the existing processor-based implementation.

1. CHOIR DESIGN

The proposed CHOIR accelerates the classification model training and inference by the interplay of three main algorithms: Genetic Algorithm (GA), Decision Tree (DT), and AdaBoost. Figure 1 shows an architectural overview of the proposed CHOIR design. The two main modules, *trainer* and *predictor*, perform the model learning and the inference task, respectively. All sub block arrays in the two modules are designed using the resistive memory and the multi-stage CAMs discussed in Section 1.1, while the microcontroller executes high-level algorithm controls and invokes their in-memory computing functionalities.

In the model learning step, CHOIR learns classification rules from the given training dataset and generates multiple decision trees as the base learners. There are two components dedicated for the model training, *sample selector* and *DT learner*. The *sample selector* consists of Chromosome MEM blocks, which implement a genetic algorithm (GA), and RepCH-MEM blocks, which store the output of the algorithm. The *DT Learner* consists of Stump Generator blocks which are also a set of resistive memory devices. This component trains a decision tree by recursively generating decision stumps using the output of sample selector, i.e., RepCH-MEM. The generated decision tree is stored into a DT-MEM of the predictor module as a base learner. We generate multiple decision trees for a classification problem based on the AdaBoost algorithm.

In the prediction step, the generated DT-MEMs of a predictor module are used to perform the class inference. Note

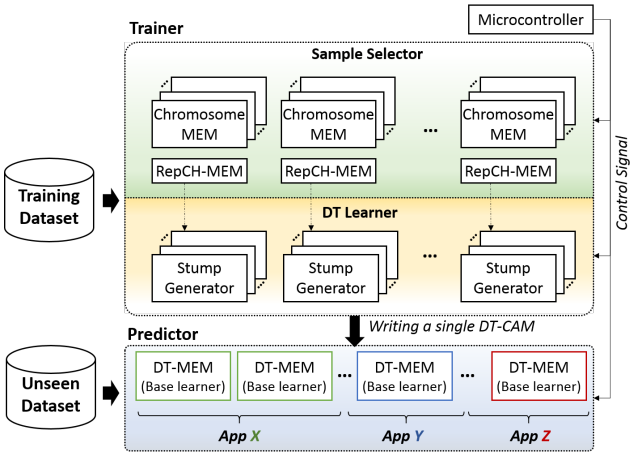

Figure 1: An Architectural Overview of CHOIR

that CHOIR can store different sets of DT-MEMs for multiple applications. In fact, the size of one DT-MEM is reasonably small, e.g., around 8 KBytes in our experimental setup. This allows us to support various applications with small memory footprints. In the next section, we discuss how we adapt the AdaBoost algorithm for the CHOIR design.

1.1 Memory-Based Inference of CHOIR

Figure 2a illustrates an example of a decision tree. This decision tree has multi-level decisions using two *decision stumps*, i.e., the two decision nodes of the tree. For each decision stump, different features of a F-dimension data point, $\mathbf{v}$, are considered, i.e., v_a and v_b for $0 \leq a < F$ and $0 \leq b < F$. The leaf node includes the probabilities of each class, $\mathbf{p}$. In this example, K is 2.

In the CHOIR design, a DT-MEM implements a decision tree based on the concept of *auto-associative memory* which repeatedly activates a row using internal memory data. Figure 2b shows the structure of the DT-MEM that does the prediction of the example decision tree inside the memory. The DT-MEM has four memory components, feature, value, node type, and data. The value memory component exploits a CAM structure, which supports an in-memory search functionality, and others are designed as normal memory blocks. Each component except the feature one has $2D_{max}$ memory rows for each edge, while the feature part has D_{max} rows, a design parameter that determines the maximum number of decision stumps. There is another design parameter, K_{max}, which is the maximum number of classes supported by CHOIR.

Each row of the feature part corresponds to one decision stump, and its connected two rows of other parts include information about the two children nodes of the decision stump. For the decision tree shown in Figure 2a, to represent the decision stump of the root node, i.e., $v_a > \alpha$, the 0-th row of the feature part stores the feature index (32-bit integer), i.e., a, and the connected two rows of the value part are set by α and $\alpha + \epsilon$ (32-bit floating point values),

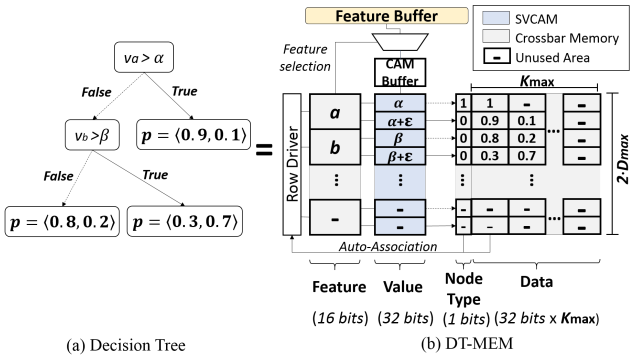

Figure 2: A decision tree example and equivalent DT-MEM structure

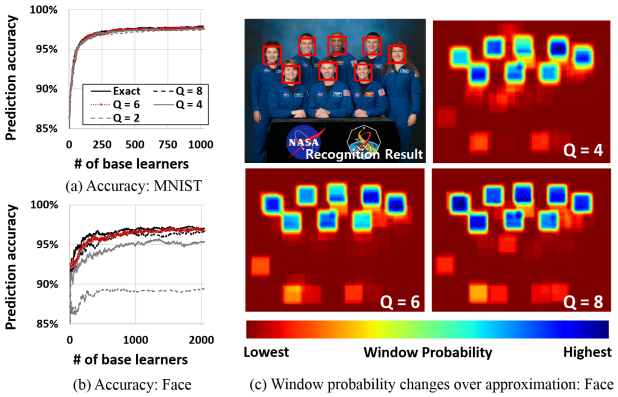

Figure 3: Accuracy changes for different HOG approximation levels

respectively. ϵ is a 32-bit number whose all elements are 0's except the last bit of 1. The 1-th row of the value part and its connected row pair have the information about the decision stump of $v_b > \beta$ in the same way.

The data part stores either i) a row index (32-bit integer) if the child is another decision stump or ii) probability values **p** (32-bit floating point array) if the child is a leaf node. The child type, i.e., decision stump or leaf, is indicated by the 1-bit flag stored in the node type part. For example, since the left child of the root node is another decision stump, the node type flag is set to 1, and the first of 32 bits of the *data* part store the row index of the child decision stump, i.e., 1. In contrast, for the right child, which is a leaf node with the probability values, the node type flag is set by 0 and the data part stores **p**.

Based on this structure, the DT-MEM performs the in-memory decision task by following iterative steps. (i) It first starts by activating one row. During the initial run, the first row is activated for the decision stump of the root node. Then, a feature v_t is selected based on the index t of the feature part, and placed into the CAM buffer. At the same time, the two rows connected to the value part are enabled. (ii) In the next cycle, the value part performs the CAM-based similarity search for the two enabled rows, e.g., α and $\alpha + \epsilon$; that is, the first row (dotted arrow) is selected if the value is smaller or equal than α. Otherwise, the second row (solid-line arrow) is selected. Thus, for the node type and data part, only one row is activated. (iii) Once the one row is activated, its node type bit determines whether it proceeds the further search. In this example, if $v_a \leq \alpha$, the activated flag is 1, meaning that it needs to test another decision stump. The row index of the next decision stump is stored in the first 32-bits of the *data* component. The row driver decodes the row index and process the decision stump by going back to the step (i). By processing the iterations until the `node type` flag is 0, we can identify probability values as the result of the DT-MEM. Once each DT-MEM identifies the decision probabilities in parallel, `CHOIR` adds the probabilities of all DT trees to create the final prediction. All these computations happen inside the memory without external accesses to the stored data for the decision tree model. This reduces the data movement overhead, thus significantly improving performance of the whole prediction procedure which many base learners involve.

2. EXPERIMENTAL RESULTS

2.1 Experimental Setup

We verify and evaluate the proposed design using HSPICE simulator with System Verilog and Synopsys Design compiler in 45nm TSMC technology. The memristor devices are designed with a large OFF/ON resistance ratio to provide stable and large sense margin [1]. To verify recognition quality of `CHOIR` for practical image recognition problems, we consider four problems: text, face, pedestrian, and vehicle recognition with the dataset found in MNIST, Caltech Webfaces, and UCI library [2, 3].

2.2 Classification Accuracy

We first present how `CHOIR` classifies visual objects in practical images. Figure 3a and 3b show the prediction accuracy changes of MNIST and Face respectively, for different numbers of base learners (L) and quantization levels (Q), where the accuracy is defined by the percentage of images whose object classes are correctly predicted. The results show that by selecting the sufficient number of base learners and quantization levels, we can achieve the same level of recognition quality as compared to the precise feature computation (denoted *Exact* in the figure). For example, for MNIST, there is only 0.4% accuracy loss with the most aggressive quantization level, i.e., $Q = 2$ and $L = 1024$, resulting in 97.5% of accuracy. Face model is more sensitive to the quantization level, but `CHOIR` can recognize images 96.7% of the time which incurs only 0.3% error when $Q = 6$ and $L = 2048$. For Pedestrian and Vehicle, `CHOIR` performs the classification for the data computed by Haar-like features without any approximation. For these two models, the recognition precision is 91.0% and 93.8%, respectively.

2.3 Energy and Performance Improvement

Our result shows that for the two models which use the approximate HOG feature extractor, the quantization level affects the power and performance efficiency of `CHOIR` since it uses less memory area. When using $Q = 6$, which has 0.3% of accuracy loss for MNIST workload, `CHOIR` achieves energy efficiency improvements of 1896**x** relative to the server with 376**x** speedup, and 552**x** as compared to ARM Cortex A53 with 2654**x** speedup. The energy and latency of `CHOIR` for this workload are 29 μJ and 2.0 μs. The comparison results of the non-approximate models, used for Pedestrian and Vehicle workloads, are summarized in Figure **??**. The improvements relative to processor-based implementations on server (embedded device) are still significant, e.g., 1352**x** (132**x**) energy improvement and 37**x** (98**x**) speedup for the Pedestrian workload.

3. REFERENCES

[1] Shahar Kvatinsky, et al. Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2015.

[2] Yann LeCun, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[3] Anelia Angelova, et al. Pruning training sets for learning of object categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* IEEE, 2005.