

Ensemble for High Recognition Performance FPGA

Hassab Elgawi Osman

Imaging Science and Engineering Lab, Tokyo Institute of Technology-Japan

osman@isl.titech.ac.jp

Abstract—We describe a flexible and efficient architecture for generic object recognition system based on ensemble classifier in a Field Programmable Gate Array (FPGA) environment. We have shown previously utilizing a bag of covariance matrices as object descriptor improves the object recognition accuracy while speed up the learning process. We extend this technique, and present its hardware architecture, as well as object classifier based on on-line variant of random forest (RF) implemented using Logarithmic Number System (LNS). First, we describe the algorithmic and architecture of our model, comprises several computation modules. Then test and verified the model functionality using numerical simulation. Utilizing examples from GRAZ02 dataset it has been shown that the proposed system gained strong recognition performance over the floating-point and fixed-point precision, even when only 10% training examples are used and is reasonably power efficient.

Index Terms—ensemble learning, random forest (RF), FPGA, LNS, object recognition.

I. INTRODUCTION

Combining multiple classifiers (e.g., decision trees) to build an ensemble is an advanced machine learning technique with substantially classification improvement over a single-based classifiers. *Random forests* (RFs) [1], a representative decision tree-based ensemble has been emerged as a principle machine learning tool combining properties of efficient classifier and feature selection running on general-purpose processor-based (GPP-based) custom-hardware and optimized operating systems. Rather than minimizing training error, RF minimizes the generalization error, while being fast to train, proven not to overfit, and computationally effective ($O(\sqrt{VT} \log T)$, where V is the number of variables and T is the number of observations). These merits make RF a potential tool suited for adaptive classification problems. In addition to its popularity in classification and regression, RF also has been applied to vision problems such as object recognition [2], [3], [4], OCR [5], and for key point recognition [6]. Despite of the appearance success of RF virtually no work has been done to map from its ideal mathematical model to compact and reliable hardware design.

In this paper we present object recognition system implemented on an field programmable gate array (FPGA). We favored FPGA implementation due to its ample flexibility and cost considerations [7], it enables learning algorithm to scale up as process technology allows for ever larger gate counts. As can be seen in Fig.1 the recognition process is composed of automatic representation of objects as covariance matrices follow by a tree-based RF detector that operate in on-line mode. We have shown in [4] utilizing a bag of covariance matrices as object descriptor improves the accuracy of object

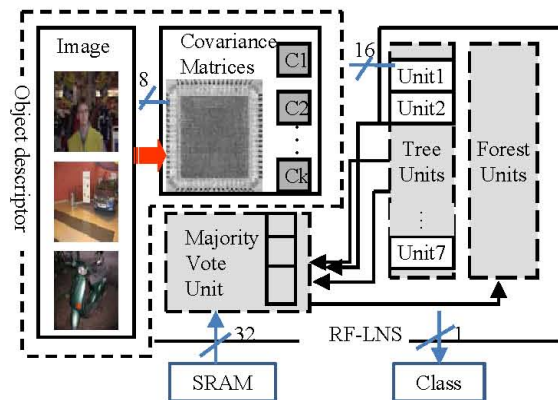


Fig. 1. Object Recognition based on RF-LNS which is optimized to be easily integrated in a System-on-Chip (SoC) platform implementation.

recognition while speed up the learning process, so we are extending this technique, present its hardware architecture. The RF detector is designed using Logarithmic Number Systems (LNS), allows the reduction of the required word-length to 16 bits, and consequently a general-purpose microprocessor of the same word-length is used. The architecture comprises several computation modules, referred to as ‘Covariance Matrices’, ‘Tree Units’, ‘Majority Vote Unit’, and ‘Forest Units’. However, it should be noted that the number of these Units that can be accommodated depends on the hardware platform. The main contribution of our approach (in addition to its impacts on the tradeoff between algorithmic setting accuracy and hardware implementation cost) is three-fold: (1) its direction towards arithmetic complexity reduction using a modified RF based on LNS (RF-LNS). We predict further progress using this approach over Pulsed Neural Network (PNN) approaches, (2) it has been designed in order to be easily integrated in a system-on-chip (SoC), which can perform both automatic feature selection and recognition, and (3) it allows for fair comparison with floating-point (FP) and fixed-point implementations. We test and verified the model functionality using numerical simulation, present results obtained using examples from GRAZ02 dataset [8] and compares with floating-point and fixed-point precision. First, in Section II the paper presents related work and highlight on general constrains in implementing hardware-based recognition systems. Section III shows the object descriptor we used and overview on RF algorithmic settings. In Section IV we present our full architecture and design of our recognition system. We follow with experimental evaluation

and estimation of the required precision in Section V. A brief conclusion appears in Section VI.

II. HARDWARE-BASED MACHINE LEARNING

Perhaps motivated by the high computational complexity of many software-oriented machine vision algorithms, there have been many attempts to create hardware implementations which are able to identify and localize one or more object in a given scene or an image, achieve high recognition performance. There are studies about Pulsed Neural Network (PNN) that employ Pulsed Neuron (PN) or Spiking Neuron, focused on sound signal processing and object localization and processing. The PN models employ leak integrators as there internal potentials, and have the ability to adapt, much better than traditional neural nets. Therefore the models can deal with temporal signals without the windowing process. The Kerneltron [9], [10], developed at John Hopkins is a SVM classification module, with a system precision resolution of no more than 8 bits. In [11] hardware implementation of Decision Trees (DTs) is proposed. However to the best of our knowledge, ours is the first attempt to implement RF in hardware. We predict further progress using this approach.

A. Hardware implementations: problems and constraints

Any kind of implementation of machine learning algorithms for vision problems, be it analog, digital, optical, brings along various constraints:

- *Algorithmic design*: Automatic optimize settings of the parameters
- *Accuracy and efficiency*: Hardware implementations can only offer limited accuracy. Floating-point operations are costly and complex in terms of hardware. Fixed-point implementation of floating-point operations is one of the classical techniques which may speed up the algorithm with marginally lose in precision.
- *Area*: The tradeoff between accuracy required and hardware (chip) area available. Accuracy often comes at the price of an area penalty.

B. logarithmic Number System (LNS)

LNS is an alternative way to represent numbers beside the conventional fixed-point and floating-point (FP) arithmetics. The LNS represents a number by the exponent in a certain base and a sign bit. The multiplication of two numbers is simply the sum of the two numbers' exponent parts, $\log_2(x \cdot y) = \log_2(x) + \log_2(y)$. However, the addition of two LNS numbers, $\log_2|(X, Y)| = X + \log_2|1 + 2^{Y-X}|$ is not a linear operation and requires two fixed-point adder/subtractors, and lookup-tables (LUTs) process (also classed Function Generators (FGs)). The size of LNS adders increases exponentially as the operands' word lengths increase. Thus the LNS arithmetic systems usually have advantages of low precision and constant relative error, which are desirable features for portable devices.

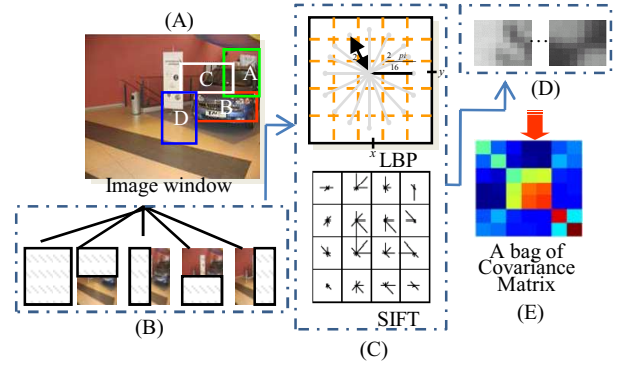


Fig. 2. (A) Rectangles are examples of possible regions for histogram features. Stable appearance in Rectangles A, B and C are good candidates for a car classifier while regions D is not. (C) Top, Points sampled to calculate the LBP around a point (x, y) . Bottom, the use of standard invariant feature (SIFT). (D) Any region can be represented by a covariance matrix. Size of the covariance matrix is proportional to the number of features used.

III. ALGORITHMIC CONSIDERATIONS

The proposed object recognition approach consists of two basic models, a model for object descriptor based on covariance matrices [12], [4] and a classifier based on on-line variant of RF implemented on FPGA using LNS.

A. Covariance Matrices Descriptor

We have used bag of covariance matrices¹ (Fig.2), to represent an object region.

Let I be an input color image. Let F be the dimensional feature image extracted from I

$$F(x, y) = \phi(I, x, y) \quad (1)$$

where the function ϕ can be any feature maps (such as intensity, color, etc). For a given region $R \subset F$, let $\{z_k\}_{k=1 \dots n}$ be the d dimensional feature points inside R . We represent the region R with the $d \times d$ covariance matrix C_R of feature points.

$$C_R = \frac{1}{n-1} \sum_{k=1}^n (z_k - \mu)(z_k - \mu)^T \quad (2)$$

where μ is the mean of the region R centered at the point.

B. Labeling the Image

We gradually build our knowledge of the image, from features (Fig.2(C)) to covariance matrix (Fig.2(D)) to a bag of covariance matrices (Fig.2(E)). Our first step is to form a covariance matrix C from image features such that each feature Z in C has an intensity value $\mu(z)$ and an associated variance $\lambda^{-1}(z)$, so λ is the inverse variance, also called the precision. Next, we group covariance matrices as a set of spatially group feature in C that are likely to share common label into a bag of covariance matrices. Then estimate the bag of covariance matrix likelihoods $P(I_i|C, I_i)$ and the likelihood that each bag of covariance matrices is homogeneously labeled. We use this representation to

¹Covariance matrix correspond to small, nearly uniform regions in the image

automatically detect any target in images. We then apply on-line RF learner to select object descriptors and to learn an object classifier.

C. RF for Recognition

A detailed discussion of Breiman's RF [1] learning algorithm is beyond our scope here. We refer the reader to [13], [4] for a good introduction to our on-line settings of RF and the details of solutions to on-line classification and incremental vision. However, in order to simplify the further discussion, we briefly define some fundamental terms:

Decision-tree. For the k -th tree, a random covariance matrix C_k is generated, independent of the past random covariance matrices C_1, \dots, C_{k-1} , and a tree is grown using the training set of positive (contains the object relevant to the class) and negative (does not contain the object) image I , and covariance feature C_k . The decision generated by a random tree corresponds to a covariance feature selected by learning algorithm. Each tree casts a unit vote for a single matrix, resulting in a classifier $h(I, C_k)$.

Forest. Given a set of M decision trees, a forest is computed as ensemble of these tree-generated base classifiers $h(I, C_k)$, $k = 1, \dots, n$, using a majority vote.

Majority vote. If there are M Decision Trees, the majority voting method will give a correct decision if at least $\text{floor}(M/2) + 1$ decision trees gives correct outputs. If each tree has probability p to make a correct decision, then the forest will have the following probability P to make a correction decision.

$$P = \sum_{i=\text{floor}(M/2)+1}^b \binom{M}{i} p^i (1-p)^{M-i} \quad (3)$$

D. On-line RF for Recognition

To obtain an on-line algorithm, the steps above must be on-line where the current base classifier is updated whenever a new sample arrives. In particular our on-line RF involves two steps in inferring the object category (Algorithm 1). First, based on covariance object descriptor we develop a new, conditional permutation scheme for the computation of feature importance measure. Second, the fixed set tree K is initialized, then individual trees in RF are incrementally generated by specifically selected covariance matrix from the bag of covariance matrices. For updating, any on-line learning algorithm may be used, but we employ a standard Karman filtering technique [15].

IV. HARDWARE ARCHITECTURE

A. FPGA Architecture

All FPGAs consist of three major components: 1) Logic blocks (LBs); 2) I/O blocks; and 3) programmable routing, as shown in Fig.3(A). A logic block (LB) is functionally complete logic circuits, partitioned to LB size, mapped and routed, and place in an interconnect framework to perform a desired operation. Field programmability is achieved through

Algorithm 1 On-line Random Forests

- 1: Initially select the number K of trees to be generated.
 - 2: **for** $k = 1, 2, \dots, K$ **do**
 - 3: \hat{T} bootstrap sample from T initialize $e = 0, t = 0, T_k = \phi$
 - 4: **Do until** $T_k = N_k$
 - 5: Vector C_k that represent a bag of covariance is generate
 - 6: Construct Tree $h(I, C_k)$ using any decision tree algorithm
 - 7: Each Tree makes its estimation based on a single matrix from the bag of covariance matrices at I
 - 8: Each Tree casts a vote for most popular covariance matrix at image I
 - 9: The popular covariance matrix at I at is predicted by selecting the matrix with max votes over h_1, h_2, \dots, h_k
 - 10: $h_l = \arg \max_y \sum_{k=1}^K I(h_k(x) = y)$
 - 11: Return a hypothesis h_l
 - 12: **end for**
 - 13: **Get** the next sample set
 - 14: **Output:** Proximity measure, feature importance, a hypothesis h
-

switches (Transistors controlled by memory element or fuses) and each I/O block is programmed to act as an input or output, as required, i.e., N -input LUTs can implement any n -input boolean function. The programmable routing is also configured to make the necessary connections between logic blocks, and from logic blocks to I/O blocks. The processing power of an FPGA is highly dependent on the processing capabilities of its LBs and the total number of LBs available in the array. Generally, FPGAs use logic blocks that contain one or more LUT, typically with at least four-inputs. A four-input LUT can implement any binary function of four logic inputs. Fig.3(B) shows the architecture of a simple LB containing one four-input LUT and one flip-flop for storage. LBS generally contain dedicated carry circuitry in order to simplify implementation of adders and multipliers. In our implementation, we keep the arithmetic operations as a fixed-point addition for which current FPGAs are applicable.

B. Transform into Log-domain

Implementing FP arithmetic operations on FPGAs is very expensive in terms of the number of logic elements required. Therefore it is often necessary to work with fixed-point representations when implementing algorithms. Rather than adapting the FP arithmetic we based on LNS, eliminate the need for multiplications and division, allowing all operations to be carried out using shifts and additions. In LNS, a number x is represented in signed magnitude form, i.e., as a pair (S, e) , where $x = (-1)^s (r)^e$, S being the sign bit (which is either 0 or 1 according to the sign of x) and e being the signed exponent of the radix r . The exponent e is expressed in fixed-point binary mode with say, I bits for the integer part and F bits for the fractional part and one bit for the sign of the exponent, i.e., with a total of $(I + F + 1)$ bits.

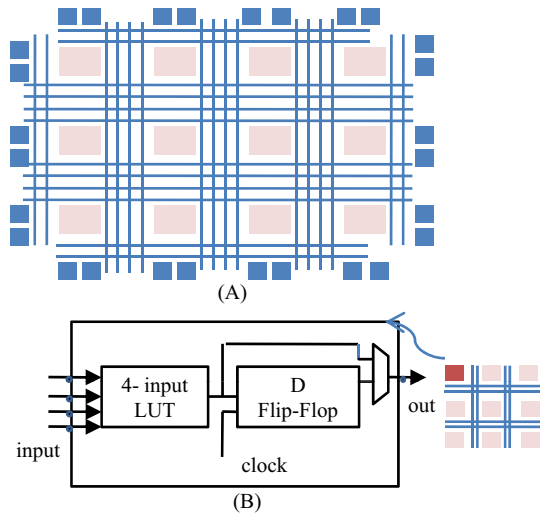


Fig. 3. (A) Granularity and interconnection structure of generic Xilinx FPGA. (B) An architecture of a logic block with one, four-input LUT use for implementation of memory and shift registers.

If the radix is considered to be 2, then the smallest number that can be represented using the scheme is 2^{-N} , where $N = (s^I - 1) + (1 - 2^{-F}) = (2^I - 2^{-F})$. The ratio between two consecutive numbers is equal to $r^{2^{-F}}$, and the corresponding precision e is roughly $(\ln r)2^{-F}$. Typically, if $I = 5$, $F = 30$, and $r = 2$, we can have a precision of 30 bits in radix 2. However, for the purpose of comparison with the precision of FP representation, e will be assumed as $2^{-23} (\approx 10^{-7})$. Numbers closer to zero, are represented with better precision in LNS than FP systems. However, LNS depart from FP in that, the relative error of LNS is constant and LNS can often achieve equivalent signal-to-noise ratio with fewer bits of precision relative to FP architectures.

C. Object Recognition Architecture based on RF-LNS

Fig.4 Shows RF-LNS object classifier proposed in this paper. The classifier consists of three main design block (a) The LG block; (b) The ACC block; and (c) The SIGM block. The Covariance Unit in Fig.1 contains all the features extracted from an image in a form of bag of covariance matrices. The output of covariance descriptor becomes the input of the RF-LNS classifier. However, Function ϕ given by eq(1) consists of float values which require much place for storing in an FPGA memory. In order to reduce the hardware cost, we propose to approximate the function ϕ using LG. This function will transform float elements of the ϕ into binary elements. For base learner we compute 16 covariance matrices in 32 bit memory. Each base learner (decision trees) is treated as a Tree Unit, estimated by a single covariance matrix selected from bag of covariance. Basically the decision trees consist of two types of nodes: *decision nodes*, corresponding to state variables and *least nodes*, which correspond to all possible covariance features that can be taken. In a decision node a decision is taken about one of the input. Each least node stores the state values for the corresponding region in the image, meaning

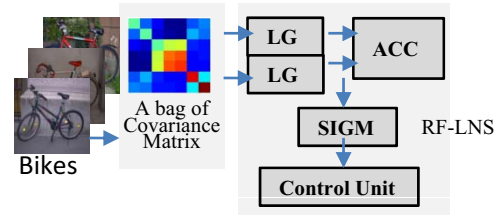


Fig. 4. Object Recognition Architecture

that a leaf node stores a value for each relevant covariance matrix that can be taken. The tree starts out with only one leaf that represents the entire image region. So in a leaf node a decision has to be made whether the node should be split or not. ACC block that does the accumulation operations at each node. Once a tree is constructed it can be used to map an input vector to a least node, which corresponds to a region in the image. Then a decision tree can be converting into an equivalent Tree Unit by extracting one logic function per class from the tree structure. Each tree gives a unit vote for its popular object class. Forest Unit is ensemble of trees grown incrementally to a certain depth. The object is recognized as the one having the majority vote, stored at Majority Vote Unit. The SIGM block that performs the sigmoid evaluation function for majority votes

V. EVALUATION METHODOLOGY

We have shown that objects in an image can be automatically represented by bag of covariance matrices, then on-line RF within LNS is employed to learn to recognize objects. We now demonstrate the usefulness of this frame work in the area of recognition generic objects such as bikes, cars, and persons.

A. Dataset and Tools

The functionality of the proposed system was simulated, and the hardware is synthesized and programmed. We have used data derived from the GRAZ02² dataset [8], a collection of 640×480 24-bit color images. As can be seen in Fig.3, this dataset has three object classes, bikes, cars and persons. Table I reports the number of images and objects in each class, 380 images are available for background class .

TABLE I
NUMBER OF IMAGES AND OBJECTS IN EACH CLASS IN THE GRAZ02 DATASET.

Dataset	Images	Objects
Bikes	365	511
Cars	420	770
Persons	311	785
Total	1096	2066

²available at <http://www.emt.tugraz.at/~pinz/data/>

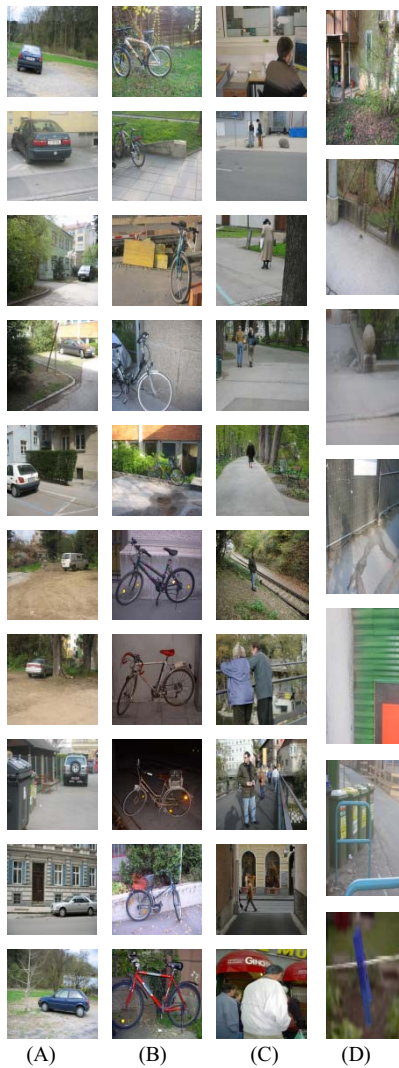


Fig. 5. Examples from GRAZ02 dataset [8] for four different categories: A) cars, B) bikes, C) persons, and D).

B. Experimental Settings

Our RF-LNS is trained with varying amounts (10%, 50% and 90% respectively) of randomly selected training data. All images not selected for the training split were put into the test split. For the 10% training data experiments, 10% of images were selected randomly with the remainder used for testing. This was repeated 20 times. For the 50% training data experiments, stratified 5×2 fold cross validation was used. Each cross validation selected 50% of the dataset for training and tested the classifiers on the remaining 50%; the test and training sets were then exchanged and the classifiers retrained and retested. This process was repeated 5 times. Finally, for the 90% training data situation, stratified 1×10 fold cross validation was performed, with the dataset divided into ten randomly selected, equally sized subsets, with each subset being used in turn for testing after the classifiers were trained on the remaining nine subsets.

VI. PERFORMANCES

GRAZ02 images contain only one object category per image so the recognition task can be seen as a binary classification problem: bikes vs. background (i.e., non-bikes), people vs. background, and car vs. background. Generalization performances in these object recognition experiments were estimated by statistic measure; the Area Under the ROC Curve (AUC) to measure the classifiers performance. AUC measures of classifier performance that is independent of the threshold, meaning it summarizes how true positive and false positive rates change as the threshold gradually increases from 0.0 to 1.0, i.e., it does not summarize accuracy. An ideal perfect classifier has an AUC of 1.0 and a random classifier has an AUC of 0.5.

A. Finite Precision Analysis

The primary task here is to analyze the precision requirements for performing on-line RF classification in LNS hardware. The RF-LNS precision was varied to ascertain optimal LNS precisions and compare them against the cost of using FP architectures. Tables II, III, and IV give the mean AUC values across all runs to 2 decimal places for RF-LNS and training data amount combinations, for the bikes, cars and people datasets respectively. The performance of RF-LNS is reported with weight quantized with 4, 8, and 16 bits, and for different decision tree depths, from depth = 3 to depth = 7. For example a figure of 85% means that 85% of object images were correctly classified but 15% of the background images were incorrectly classified (i.e. thought to be foreground). For RF-LNS to maintain acceptable performance, 16 bits of precision are sufficient for all GRAZ02 categories, even when only 10% training examples are used. Such low precision required by RF-LNS makes it competitive with FP arithmetic for our generic object recognition application.

B. Efficiency and Hardware area

In order to evaluate the efficiency of RF-LNS classifier in terms of hardware area, 10- and 20-bit fixed-point (FX) implementations were synthesized by Xilinx ISE XST for comparison, and the resulting numbers of slices are shown in Table V. It is note worthy that on most datasets; the RF-LNS takes roughly the same number of slices as the inadequate 10-bit FX version. When compared against the more realistic 20-bit FX version, the RF-LNS classifiers are about one-half the size of the FX classifiers. Our design also achieved a high speed clock rate processing. For the 1-bit RF-LNS, the power dissipation is small, and the area usage on FPGA is less than 2 percents.

VII. CONCLUSIONS AND FUTURE WORKS

Efficient hardware implementations of machine-learning techniques yield a variety of advantages over software solutions: increased processing speed, and reliability as well as reduced cost and complexity. In this paper RF technique is modified so that classification is performed by LNS arithmetic. The model is applied for generic Object recognition task, it

TABLE II

MEAN AUC PERFORMANCE OF RF-LNS ON THE BIKES VS. BACKGROUND DATASET, BY AMOUNT OF TRAINING DATA. PERFORMANCE OF RF-LNS IS REPORTED FOR DIFFERENT DEPTHS (DTH).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.79	0.79	0.77	0.81	0.81	0.81	0.81	0.80	0.83	0.83	0.83	0.83	0.81	0.84	0.83
50%	0.86	0.86	0.82	0.81	0.83	0.88	0.89	0.85	0.88	0.86	0.90	0.90	0.86	0.89	0.89
90%	0.80	0.81	0.81	0.83	0.88	0.87	0.87	0.87	0.88	0.90	0.90	0.91	0.90	0.90	0.90

TABLE III

MEAN AUC PERFORMANCE OF RF-LNS ON THE CARS VS. BACKGROUND DATASET, BY AMOUNT OF TRAINING DATA. PERFORMANCE OF RF-LNS IS REPORTED FOR DIFFERENT DEPTHS (DTH).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.66	0.70	0.70	0.75	0.71	0.68	0.73	0.73	0.76	0.73	0.71	0.75	0.75	0.77	0.75
50%	0.77	0.78	0.77	0.77	0.79	0.79	0.80	0.79	0.81	0.81	0.81	0.80	0.81	0.82	0.83
90%	0.77	0.75	0.75	0.73	0.79	0.81	0.81	0.78	0.78	0.82	0.83	0.83	0.81	0.80	0.85

TABLE IV

MEAN AUC PERFORMANCE OF RF-LNS ON THE PERSONS VS. BACKGROUND DATASET, BY AMOUNT OF TRAINING DATA. PERFORMANCE OF RF-LNS IS REPORTED FOR DIFFERENT DEPTHS (DTH).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.83	0.73	0.77	0.77	0.79	0.77	0.74	0.80	0.79	0.81	0.80	0.78	0.81	0.81	0.82
50%	0.79	0.80	0.79	0.78	0.83	0.81	0.83	0.83	0.80	0.84	0.85	0.86	0.85	0.82	0.85
90%	0.80	0.80	0.81	0.78	0.83	0.81	0.82	0.82	0.80	0.86	0.88	0.86	0.83	0.83	0.87

TABLE V

SLICES USED FOR DIFFERENT TREE UNITS FOR EACH DATASET DURING SYNTHESIS.

Dataset	Tree Units	16-bit LNS	10-bit FX	20-bit FX
Bikes	3	315	219	576
	4	498	407	713
	5	611	622	878
	6	823	835	1103
	7	1010	974	1345
Cars	3	277	283	603
	4	397	476	783
	5	536	694	866
	6	784	943	1002
	7	989	1287	1311
Persons	3	336	318	409
	4	534	535	657
	5	765	689	845
	6	878	926	1127
	7	1123	1158	1287

REFERENCES

- [1] L. Breiman, "Random Forests," *Machine Learning*, 45(1):5-32, 2001.
- [2] F. Moomsmann, B. Triggs, and F. Jurie. "Fast discriminative visual codebooks using randomized clustering forests," *NIPS* 2006
- [3] J. Winn and A. Criminisi. "Object class recognition at a glance," *CVPR*, 2006.
- [4] H. Elgawi Osman, "A binary Classification and Online Vision," *In Proc.Int'l. Joint Conference on Neural Networks, IJCNN*, 2009.
- [5] Y. Amit and D. Geman. "Shape quantization and recognition with randomized trees," *Neural Computation* 9(7):1545-1588, 1997
- [6] V. Lepetit, P. Laguerre, and P. Fua. "Randomized trees for real-time keypoint recognition," *In Proc. CVPR*, 2005.
- [7] Ngo H. T. and Asari K. V., "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Trans. of Circuits and Systems for Video Technology*, 15(3):436-444, 2005.
- [8] A. Oplet, M. Fussenegger, A. Pinz and P. Auer. "Generic object recognition with boosting," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(3) pp. 416-431, 2006.
- [9] R. Genov and G. Cauwenberghs. "Kerneltron: Support Vector Machine in Silicon," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1426-1434, 2003.
- [10] R. Genov, S. Chakrabarty and G. Cauwenberghs. "Silicon Support Vector Machine with On-Line Learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 3, pp. 385-404, 2003.
- [11] M. Muselli and D. Liberati. "Binary Rule Generation via Hamming Clustering," *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1258-1268, 2002.
- [12] O. Tuzel, F. Porikli, and P. Meer. "Region covariance: A fast descriptor for detection and classification," *In Proc. ECCV*, 2006.
- [13] H. Elgawi Osman, "Online Random Forests based on CorrFS and CorrBE," *In Proc.IEEE workshop on online classification, CVPR*, pp.1-7, 2008.
- [14] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, "Classification and regression trees," *Wadsworth Inc.*, Belmont, California, 1984.
- [15] K. P. Karman and A. von Brandt, "Moving object recognition using an adaptive background memory in Time-varying Image Processing and Moving Object Recognition," Capellini, Ed., vol. II. Amsterdam, The Netherlands: Elsevier, pp. 297307, 199

shows that at low precision the RF-LNS hardware has significant area savings, shorter word length compared to the fixed-point alternative and statistically has the same performance as software simulations. Our hardware-friendly algorithm that has been described here is general towards the implementation of FPGA. With these characteristics, RF-LNS may be a good way for designing a real-time low power object recognition systems. Further exploring precision requirements for hardware RF-LNS, noise analysis to determine the robustness of the hardware classifier and expanding LNS hardware architectures to other machine learning algorithms are reserved for future works.