We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

10

Random Forest-LNS Architecture and Vision

Hassab Elgawi Osman
Imaging Science and Engineering Laboratory, Tokyo Institute of Technology
Japan

1. Introduction

Combining multiple classifiers (e.g., decision trees) to build an ensemble is an advanced machine learning technique with substantially improvement over 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 model running on general-purpose processor (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, ($O(\sqrt{V}TlogT)$, 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. RF has been applied to vision problems such as object recognition (2–7). It has also been used for OCR (8) and for key point recognition (9). 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 chapter we present object recognition system implemented on a field programmable gate array (FPGA), enables learning algorithm to scale up. Fig.1 shows the general architecture of the proposed recognition system, composed of two main steps, each comprises several computational models. In the first step, objects are automatically represented as covariance matrices followed by a tree-based RF detector that operates on-line. We have shown in (4) utilizing a bag of covariance matrices as object descriptor improves the accuracy of object recognition while speed up the learning process, so we are extending this technique, present its hardware architecture. The on-line RF detector is designed using Logarithmic Number System (LNS) (10), RF-LNS, allows the reduction of the required word-length to 16 bits, and consequently a general-purpose microprocessor of the same word-length can be used. For the compact architecture we made RF-LNS comprises few computation modules, referred to as 'Tree Units', 'Majority Vote Unit', and 'Forest Units'. 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), (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 (FX) implementations. We test and verified the model functionality using numerical simulation, present results obtained using examples from GRAZ02 dataset (11). First, in Section 2 we present related works and highlight on general constrains in implementing hardware-based recognition systems. Section 3 shows the object descriptor we used and overview on RF al-

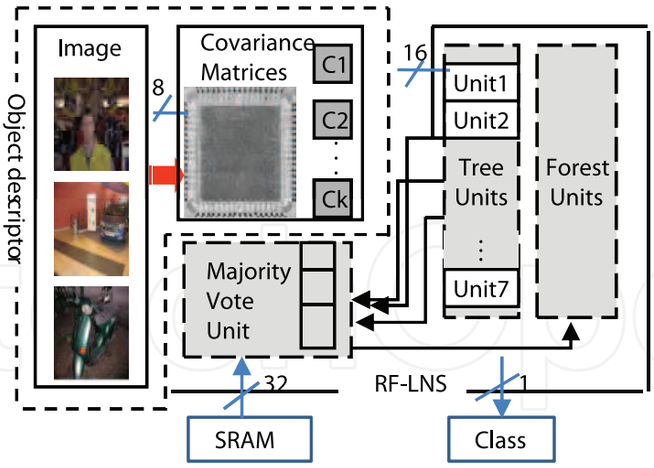


Fig. 1. Object Recognition based on RF-LNS which is optimized to be easily integrated in a System-on-Chip (SoC) platform implementation.

gorithmic settings. In Section 4 we present full architecture and design of our recognition system. We follow with experimental evaluation and estimation of the required precision in Section 5. A brief conclusion appears in Section 6.

2. Hardware-based Machine Learning

Perhaps motivated by the high computational complexity of many software-oriented machine vision algorithms, there have been several attempts to create faster execution hardware implementations which are able to identify and localize objects 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 object localization and processing. The PN models and have the ability to adapt, much better than traditional neural nets. The Kerneltron (12; 13) is a SVM classification module, with a system precision resolution of no more than 8 bits. A fully digital architecture for SVM classification employing the linear and RBF kernels is proposed. The minimal word size they are able to use is 20 bits. In (14) 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.

2.1 Hardware implementations: problems and constraints

Any kind of hardware implementations of machine vision algorithms be it analog, digital, or optical, brings along various constraints:

- *Algorithematic design*: Automatic optimize settings of the parameters.
- *Accuracy and efficiency*: Hardware implementations can only offer limited accuracy. FP arithmetics are costly in terms of the number of logic elements required while FX implementation may speed up the algorithm but is leading to a definitively power consumption 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.

- *GPP vs. FPGA*: A general purpose processor's (GPP) hardware contains all the basic blocks needed to build any logic of mathematical function imaginable but the limitations are in the parallelism available in the program, i.e. performance, and power consumption. FPGA provides flexibility to cope with the current evolving applications but at the cost of large performance, area, power and reconfiguration time penalties.

2.2 logarithmic Number System (LNS)

LNS is an alternative way to represent real numbers/values beside the conventional FP representation. The idea is to convert values into logarithms once and keep them in this representation throughout the entire computation. 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, $\log_2(x \cdot y) = \log_2(x) + \log_2(y)$, divisions and square roots are implemented by fixed-point subtraction and bit shift respectively. 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 (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.

3. Algorithmtic Considerations

The proposed object recognition approach consists of two basic models, a model for object descriptor based on covariance matrices (4; 15) and a classifier based on on-line variant of RF implemented on FPGA using LNS. First we introduce the algorithmtic settings of each model.

3.1 Covariance Matrices Descriptor

We have used bag of covariance matrices (Fig.2), to represent an object region.
Let I be an input color image. Let F be the dimensional feature image extracted from I

$$F(x,y) = \phi(I,x,y) \tag{1}$$

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

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

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

3.2 Image Labeling

We gradually build our knowledge of the image from features to covariance matrix to a bag of covariance matrices. Starting by forming covariance matrix C from image features such that each feature Z in C has intensity $\mu(z)$ and associated variance $\lambda^{-1}(z)$, so λ is the inverse variance (precision). We then group covariance matrices as a set of spatially grouped feature in C that are likely to share common labels into a bag of covariance matrices.
Covariance matrix. Different regions of an object may have different descriptive powers and, hence, a difference impact on learning and recognition (Fig.2A). Following (15), we represent image objects with five covariance matrices $C_{i=1\cdots 5}$ of the feature computed inside R (Fig.2B),

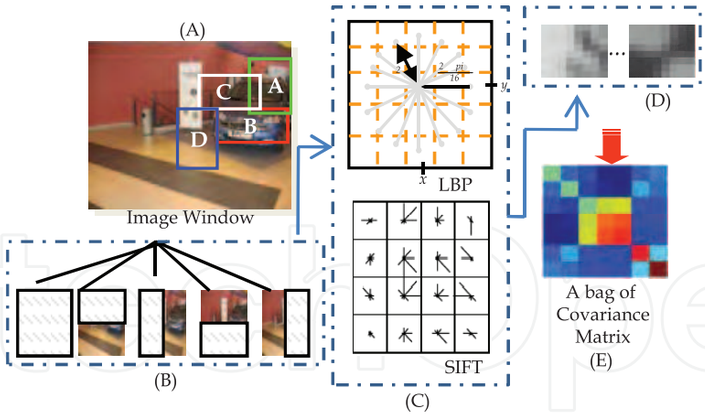


Fig. 2. (A) Rectangles are examples of possible regions for histogram features. Stable appearance in Rectangles A, B and C are good candidates for a car classifier while regions D is not. (C) Top, points sampled to calculate the LBP around a point (x,y). Bottom, the use of standard invariant feature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

noting that features in the covariance matrix may be used in multiple image locations.

Color. Color is described by taken Ohta space histogram values of pixels $(I_1 = R + G + B/3$, $I_2 = R - B$, $I_3 = (2G - R - B)/2)$. This histogram is chosen because it is less sensitive to variations in illumination. Ohta values for each pixel in an image are clustered using k-means, e.g., each pixel in image I is assigned to the nearest cluster center, then histogram frequency is normalized.

Appearance. We have used histograms of Local Binary Patterns (LBPs) for representing each feature's appearance in some appearance space. Fig.2C depicts the points that must be sampled around a particular point (x,y) in order to calculate the LBP. In our implementation, each sample point lies at a distance of 2 pixels from (x,y). Instead of the traditional 3×3 rectangular neighborhood, we sample neighborhood circularly with two different radii (1 and 3). The resulting operators are denoted by $LBP_{8,1}$ and $LBP_{8,1+8,3}$, where subscripts tell the number of samples and the neighborhood radii.

A bag of covariance matrices. A bag of covariance which is a concatenation of Ohta color space histogram, and appearance model based on LBP and Scale Invariant Feature Transform (SIFT) of different features of an image region is presented in Fig.1E. 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 automatically detect any target in images. We then apply on-line RF learner to select object descriptors and to learn an object classifier.

3.3 RF for Recognition

A detailed discussion of Breiman's RF (1) learning algorithm is beyond our scope here, 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 $C_1,\ldots,C_{k-1}$, 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 $h(I, C_k)$, $k = 1, \ldots, n$, using a majority vote.

Majority vote. If there are M Decision Trees, the majority voting method will give a correct decision if at least $floor(M/2) + 1$ 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.

$$P = \sum_{\mathrm{i=floor(M/2)+1}}^{b} \left(\begin{array}{c} M \\ i \end{array} \right) p(1-p) \tag{3}$$

3.4 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.

Algorithm 1 On-line Random Forests

1: Initially select the number K of trees to be generated.
2: **for** $k = 1, 2, \cdots, K$ **do**
3: $\grave{T}$ 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 $h_1, h_2, \ldots, h_k$
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

4. Hardware Architecture

4.1 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 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.

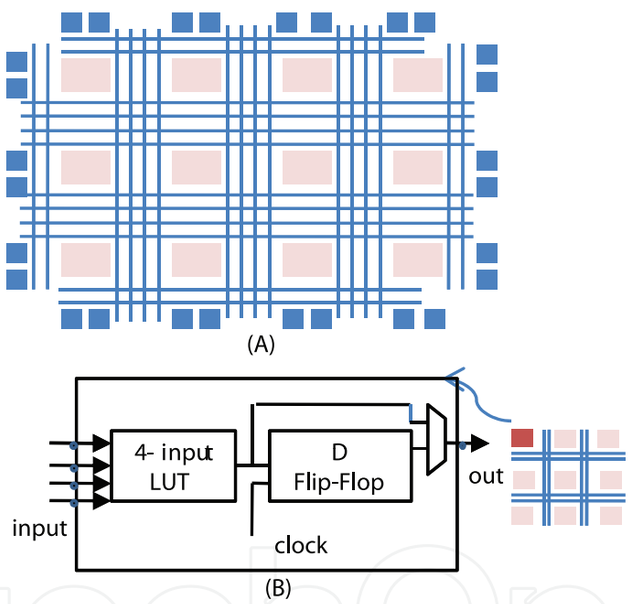


Fig. 3. (A) Granularity and interconnection structure of generic Xilinx FPGA. (B) An architecture of a logic block with one, four-input LUT use for implementation of memory and shift registers.

4.2 Transform into Log-domain

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 (usually in radix 2). The exponent e is expressed in fixed-point binary mode with say, G 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 $(G + F + 1)$ bits. If the radix is considered to be 2, then the smallest number that can be represented using the scheme is 2^{-N}, where $N = (s^G - 1) + (1 - 2^{-F}) = (2^G - 2^{-F})$. 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 $G = 5$, $F = 30$, 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.

4.3 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 blocks (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 'Tree Units' we compute 16 covariance matrices in 32 bit memory. 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 that a least node stores a value for each relevant covariance matrix that can be taken. The tree starts out with only one least node that represents the entire image region then, 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 Units' gives a unit vote for its popular object class. 'Forest Unit' is an 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.

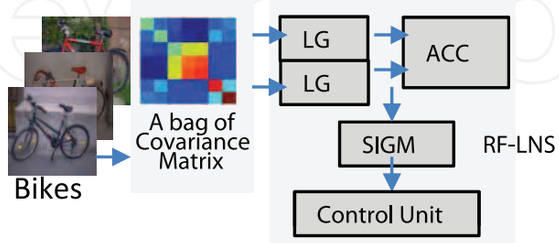


Fig. 4. RF-LNS object classifier Architecture.

5. Evaluation

The functionality of the proposed system was simulated, and the hardware is programmed. We now demonstrate the usefulness of this frame work in the area of recognition generic objects such as bikes, cars, and persons.

5.1 Dataset

We have used data derived from the GRAZ02[1] dataset (11), a collection of 640 $\times$ 480 24-bit color images. As can be seen in Fig.5, this dataset has three object classes, bikes, cars, persons, and in addition to the background class (270 images). This database contains variability with respect to scale and clutter. Objects of interest are often occluded, and they are not dominant in the image. According to (16) the average ratio of object size to image size counted in number of pixels is 0.22 for bikes, 0.17 for people, and 0.9 for cars. Thus this dataset is more complex dataset to learn detectors from, but of more interest because it better reflects the real world complexity. Table 1 reports the number of images and objects in each class, 380 images are available for background class .

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

Table 1. Number of images and objects in each class in the GRAZ02 dataset.

5.2 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 $\times$ 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 $\times$ 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.

6. 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

[1] available at `http://www.emt.tugraz.at/~pinz/data/`

Fig. 5. Examples from GRAZ02 dataset (11) for four different categories: A) cars and ground truth, B) bikes and ground truth, C) persons and ground truth, and D) background.

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.

6.1 Finite Precision Analysis

The primary task here is to analyze the precision requirements for performing recognition. The RF-LNS precision was varied to ascertain optimal LNS precisions and compare them against the cost of using FP architectures. Tables 2, 3, and 4 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.

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	h=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=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 2. 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 (D).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=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 3. 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 (D).

	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)					RF-LNS (16-bit Precision)				
	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=7	D=3	D=4	D=5	D=6	D=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 4. 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 (D).

6.2 Efficiency and Hardware area

The efficiency of RF-LNS classifier is evaluated in terms of the number of slices. This is simply equivalent to hardware area required to achieve acceptable performance. Table 5 shows number of slice used by RF-LNS classifier as compared with 10- and 20-bit fixed-point (FX) implementations. The number of slices is reported for different Tree Unit for each dataset. RF-LNS takes almost the same number of slices as 10-bit FX but less than one-half of 20-bit FX implementation. This is interesting because 10-bit FX implementation has been widely recognized for not acceptable performance, particularly for recognition problem. Our design

also achieved 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.

7. 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 shows that at low precision the RF-LNS hardware has significant area savings compared to the fixed-point alternative. With these characteristics, RF-LNS may be a good way for designing a real-time low power object recognition systems. Our future goals include 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.

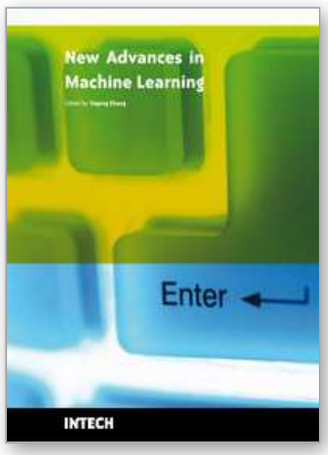
Dataset	Tree Units	16-bit LNS	10-bit FX	20-bit FX
Bikes	3	315	219	576
	4	498	407	713
	5	611	622	878
	6	823	835	1103
	7	1010	974	1345
Cars	3	277	283	603
	4	397	476	783
	5	536	694	866
	6	784	943	1002
	7	989	1287	1311
Persons	3	336	318	409
	4	534	535	657
	5	765	689	845
	6	878	926	1127
	7	1123	1158	1287

Table 5. Slices used for different tree units for each dataset.

8. 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," In *Proc. NIPS* 2006.
[3] J. Winn and A. Criminisi. "Object class recognition at a glance," In *Proc. CVPR*, 2006.
[4] H. Elgawi Osman, "A binary Classification and Online Vision," In *Proc. IJCNN*, 2009. pp.1142-1148
[5] A.Bosch, A.Zisserman, X.Munoz, "Image Classification Using Random Forests and Ferns," *ICCV*, pp.1-8, 2007.
[6] J. Shotton, M. Johnson, R. Cipolla, "Semantic Texton Forests for Image Categorization and Segmentation," In *Proc. CVPR*, pp.1-8 2008.
[7] F. Schroff, A. Criminisi, and A. Zisserman, "Object Class Segmentation using Random Forests," In *Proc. BMVC* 2008.

[8] Y. Amit and D. Geman. "Shape quantization and recognition with randomized trees," *Neural Computation* 9(7):15451588, 1997.
[9] V. Lepetit, P. Lagger, and P. Fua. "Randomized trees for real-time keypoint recognition," In *Proc. CVPR*, 2005.
[10] H. Elgawi Osman, "Hardware-based solutions utilizing Random Forests for Object Recognition," In *Proc. ICONIP*, Part II, LNCS 5507, pp. 760-767, 2008.
[11] A. Oplet, M. Fussenegger, A. Pinz and P. Auer. "Generic object recognition with boosting," *TPAMI* 28(3):416-431, 2006.
[12] R. Genov and G. Cauwenberghs. "Kerneltron: Support Vector Machine in Silicon," *IEEE Transactions on Neural Networks*, 14(5):1426-1434, 2003.
[13] R. Genov, S. Chakrabartty and G. Cauwenberghs. "Silicon Support Vector Machine with On-Line Learning," *IJPRAI*, 17(3):385-404, 2003.
[14] M. Muselli and D. Liberati. "Binary Rule Generation via Hamming Clustering," *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1258-1268, 2002.
[15] O. Tuzel, F. Porikli, and P. Meer. "Region covariance: A fast descriptor for detection and classification," In *Proc. ECCV*, pp.589-600, 2006.
[16] A. Opelt. and Pinz A. "Object Localization with boosting and weak supervision for generic object recognition," *In Kalvianen H. et al. (Eds.) SCIA 2005*, LNCS 3450, pp.862-871, 2005



New Advances in Machine Learning

Edited by Yagang Zhang

ISBN 978-953-307-034-6
Hard cover, 366 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The purpose of this book is to provide an up-to-date and systematical introduction to the principles and algorithms of machine learning. The definition of learning is broad enough to include most tasks that we commonly call “learning” tasks, as we use the word in daily life. It is also broad enough to encompass computers that improve from experience in quite straightforward ways. The book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides a good introduction to many approaches of machine learning, and it is also the source of useful bibliographical information.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen