FPGA Implementation of Support Vector Machines for 3D Object Identification

Marta Ruiz-Llata and Mar Yébenes-Calvino

Departamento de Tecnología Electrónica
Universidad Carlos III de Madrid
C/ Butarque 15, 28911, Leganés, Madrid
marta.ruiz-llata@uc3m.es

Abstract. In this paper we present a hardware architecture for a Support Vector Machine intended for vision applications to be implemented in a FPGA device. The architecture computes the contribution of each support vector in parallel without performing multiplications by using a CORDIC algorithm and a hardware-friendly kernel function. Additionally input images are not preprocessed for feature extraction as each image is treated as a point in a high dimensional space.

Keywords: Support Vector Machines, embedded systems, image processing.

1 Introduction

There has been an impressive development of cameras and distributed sensors that allows the positioning of many of them to monitor the environment. Application examples are ambient intelligence, intelligent transportation systems, security, and surveillance. However, to manage such an amount of information from a central computing system is very difficult, so there is a lot of interest in conceiving sensors that solely provide the pertinent information. A key issue is the development of adaptive embedded vision systems, because a great deal of information can be extracted from a single image or a video sequence, and vision tasks are very computer resource consuming.

Neural networks [1] and other learning machines algorithms, as Support Vector Machines (SVM) [2], have been successfully used in vision systems as they can deal with many image classification and pattern recognition tasks. However the implementation of these algorithms on limited resource hardware, such microprocessor or Field Programmable Gate Arrays (FPGAs) is still a very challenging task [3].

In this paper we propose the implementation of an adaptive vision system in FPGA devices. Neural processors for vision [4] and more general applications [5] have been implemented on FPGAs. These devices provide many advantages such solid development tools, easy reprogrammability and fast development time without losing performance with respect full-custom systems design. Additionally a high degree of parallel processing can be achieved compared with other embedded platforms as microcontrollers and DSPs. The algorithm to be implemented in the FPGA device is the feedforward phase of a Support Vector Machine trained to identify different objects from the COIL data set [6] independently of their pose, as described in the software implementation of a SVM for 3D object recognition [2]. SVM have been also implemented on embedded hardware as FPGA [7] and microcontroller [8] with successfully results on classification error for reduced dimension input data. In reference [2], a SVM algorithm is proposed to perform recognition on images regarded as points in a high dimensional space, that is, the system does not require feature extraction so input images are treated as vectors. Our aim is also to process high dimension input vectors for object identification but in a dedicated hardware platform based on FPGA devices.

The structure of the paper is as follows. In the next section we describe the basic of SVMs. In section 3 we present the proposed hardware architecture of the feed forward phase of the SVM classifier. In section 4 we present its implementation for a reduced complexity problem using the Iris data base. Section 5 describes preliminary results using the COIL images data base. We present conclusions and further research in section 6.

2 Theoretical Overview of SVMs

Support Vector Machines, as opposed to Neural Networks, evolve from the theory to the implementation and experiments. Their theoretical foundations are statistical learning theory and structural risk minimization [9]. The non-linear two-class classification task of a training data set of l elements:

$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_l, y_l)\}$ where $\mathbf{x}_i \in R^n$ and $y = \pm 1$

consists on resolving the following quadratic programming problem (CQP) with a series of associated restrictions:

$$\begin{aligned} &maximize \quad -\frac{1}{2}\alpha^T \mathbf{Q} \alpha + \mathbf{r}^T \alpha \\ &subject\ to \quad 0 \le \alpha_i \le C, i = 1, \ldots l \\ &\mathbf{y}^T \alpha = 0 \ . \end{aligned} \tag{1}$$

where $r_i = 1 \ \forall i$, and $\mathbf{Q}$ is a $l \times l$ positive semi-definite symmetric matrix and each element determined by:

$$q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \ . \tag{2}$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function.

The feed-forward estimation or classification of a new, no learned, vector $\mathbf{x}$ is:

$$y(\mathbf{x}) = \mathtt{sgn}\left(\sum_{i=0}^{l} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \ . \tag{3}$$

where parameters α_i and b are given in the learning phase by resolving equation (1). The fact that the summation in equation (3) goes over all training data patterns is not relevant because most α_i parameters result equal to 0. Those training patterns whose corresponding α_i parameter is not equal to 0 are called support vectors (SV). The scarcity of the model results from a sophisticated learning that matches the model capacity to the data complexity ensuring a good performance on the future, previously unseen, data.

Typical Kernels are polynomial functions and Gaussian function. In addition other kernel function have been proposed in the literature; it is highlighted the function named hardware-friendly kernel function [10], which greatly simplifies the SVM feed-forward phase computation in resource constrained hardware while maintains good classification performance respect to the conventional Gaussian kernel [10]. Table 1 shows the examples of kernel functions.

Table 1. Kernel Functions

Kernel Funcions	
$\left[(\mathbf{x}^T\mathbf{x}_i)+1\right]^d$	Polinomial of degree d
$e^{\left(-\left\|\mathbf{x}_i-\mathbf{x}\right\|^2/\sigma^2\right)}$	Gaussian
$2^{-\gamma\left\|\mathbf{x}_i-\mathbf{x}\right\|_1}$	Hardware-friendly

3 Proposed System Architecture

The goal is to map the classification algorithm of a Support Vector Machine represented by equation (3) into an FPGA. The basic architecture is represented in Fig. 1.

We compute the sign of the addition of $(n_{SV}+1)$ terms, being n_{SV} the number of support vectors, obtained from the training algorithm (equation (1)). Support vectors are those whose α_i parameter is not 0, so the summation in equation

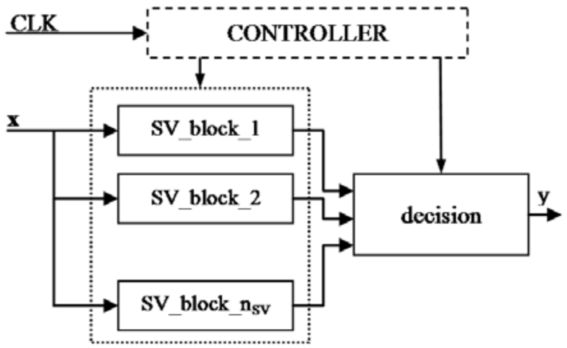

Fig. 1. Architecture for the calculation of the feed forward phase on an SVM classifier

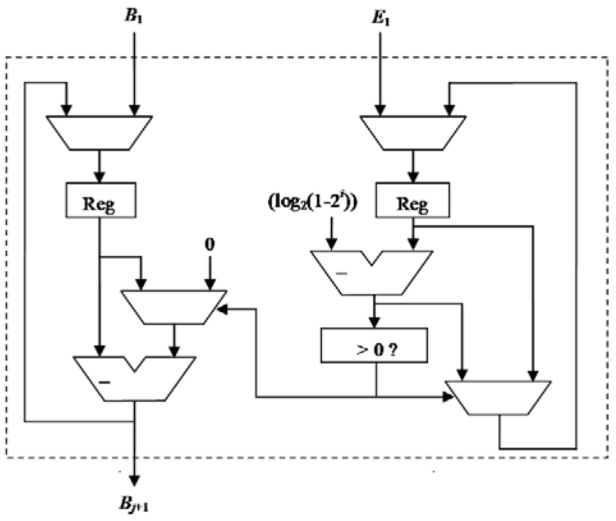

Fig. 2. Architecture for performing a single step of the hardware friendly kernel computation based on a CORDIC algorithm

(3) apply to the support vectors plus parameter b. The calculation of the terms for each support vector are performed in parallel by the blocks named SV_block. The additional block (Controller) is a state machine that controls the signal flow and the operation stages to compute the kernel function as described below.

The function of each SV_block is the computation of the hardware-friendly kernel represented in Table 1 as described in reference [10]. It is possible to compute the kernel function without resorting to any multiplication or function evaluation procedure by using a COrdinate Rotation DIgital Computer (CORDIC) algorithm. CORDIC algorithms are a class of hardware-efficient algorithms that provide iterative solutions based on shifts and adds for the calculation of trigonometric and transcendental functions [11].

As described in [10] the first step is to rescale the SVM learning parameters to guarantee their correct coding in the hardware. Their new values are between 0 and $1-2^{-k}$, being k the number of bits assigned to represent the parameters. The scaling procedure does not affect the classification function (equation (3)) as the sign remains unchanged. The evaluation of the kernel function is performed by the iterative algorithm described by equations (4) and (5):

$$B_{j+1} = B_j(1 + d_j 2^{-j}) \ . \tag{4}$$

$$E_{j+1} = E_j - \mathtt{log}_2(1 + d_j 2^{-j}) \ . \tag{5}$$

where $d_j = \{-1, 0\}$ is chosen such that $\left|E_{j+1}\right| \leq \left|E_j\right|$. Starting values are $B_1 = \alpha_i$ and for support vector $\mathbf{x}_i$ and input vector $\mathbf{x}$. After k iterations (being k the number of bits assigned to represent the parameters) we obtain the solution:

$$B_{k+1} = B_1 2^{E_1} = \alpha_i 2^{-\gamma \|\mathbf{x}_i - \mathbf{x}\|_1} = \alpha_i K(\mathbf{x}_i, \mathbf{x}) \,. \tag{6}$$

The architecture for computing equations (4) and (5) of the CORDIC algorithm is represented in Fig.2. It is based on the architectures proposed in references [10] and [11].

The additional elements that compound each SV_block are a memory which store k precomputed values of $\log_2(1 + d_j 2^{-j})$, a memory to store the α_i (B1) parameter and the support vector $\mathbf{x}_i$ obtained from the training step, all represented with k bits. Finally an additional subblock (E1_block) has been included to perform the calculation of E_1 (see Fig.2) from input vector $\mathbf{x}$ and support vector $\mathbf{x}_i$. The architecture of SV_block is represented in Fig.3. Here, E1_block is very important because it is designed in order to guarantee the convergence of the CORDIC algorithm. E_1 is restricted in the range (-1,0] thus its real value is divided into an integer part I and a decimal part F, so that the integral part is included at the end by shifting I times the output of the CORDIC algorithm.

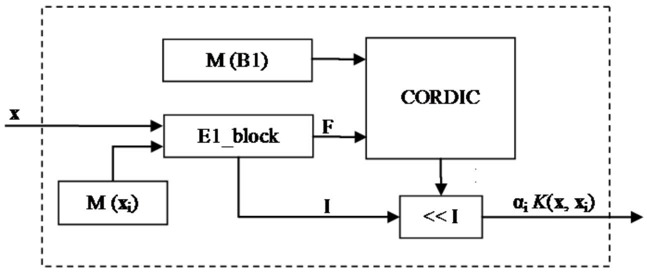

Fig. 3. Architecture for the calculation of the contribution of one support vector (SV_block)

4 Iris Data Set Implementation

We have implemented a two-class classifier for the simplified Iris dataset. The target device is a low cost Altera EP2C20 Cyclone II. This dataset is composed of 120 samples of two components vector that are non-linearly separable. The training step has been performed with Matlab software using the bioinformatics toolbox where the hardware-friendly kernel (Table 1) has been included. Fig.4 shows the training data set, the classification boundary and the resulting support vectors.

The classification hardware has been implemented with Altera Quartus II design software using schematic design and VHDL. The architecture is that represented in Fig.1 just including as many SV_blocks as support vectors are provided in the training. The resolution chosen to represent vector components and SVM parameters is 8 bits, so the classification output is obtained after 10

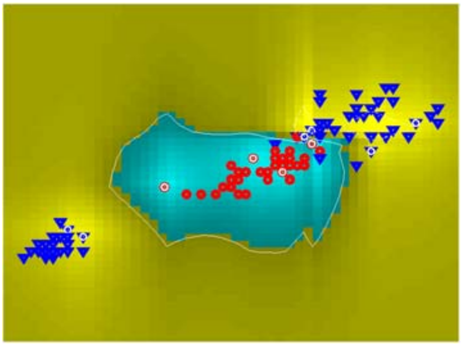

Fig. 4. SVM training and software testing

clock cycles, 8 consumed by the CORDIC algorithm. The maximum clock rate achieved, limited by E1_block, is 33 MHz, so each classification is performed in $0.3\mu s$. The performance of the SVM algorithm depends on the value of parameter γ in the hardware-friendly kernel function, the value of C in equation (1) and the number of elements randomly chosen in the training step. We used $C = 10$ and $\gamma = 2$. Good classifications results are obtained when about 10 support vector are selected in the training. In this case the classification error in the hardware SVM is below 10%, quite higher than the classification error performed by software simulation with the same number of support vectors (5%). This is due to low precision representation of the data in the hardware classifier, but it can be compensated by choosing a classifier with a higher number of support vectors. Increasing the number of support vector does not compromise classification speed as all support vectors are evaluated in parallel. Each SV_block occupies about 180 logic elements. The classifier with 10 support vectors takes 11% of logic cells of the target device and no additional hardware resources.

5 First Image Classification Results

The proposed hardware architecture is to be tested on the COIL database consisting of 7,200 images of 100 objects, there are 72 different angular views of each object. They are color images of 128×128 pixels. Reference [2] proposes a SVM for a multiclass classification of the COIL database using the images directly as inputs to the classifier. First, the images are converted to 8 bits levels gray-scale images and their resolution is reduced to 32×32 pixels, so they are represented as a vector with 1024 components. They obtain excellent recognition rates even with noise corrupted images, shifted images and partial occlusions. We will also follow the same strategy to deal with images as high dimensional input vector. Fig.5 shows four examples of images to be classified.

In this case we also performed the training step with Matlab software using the bioinformatics toolbox where the hardware-friendly kernel has been included.

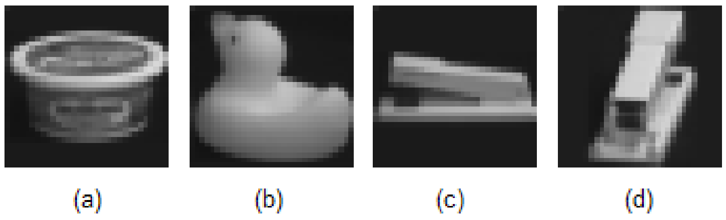

Fig. 5. Examples of modified images of the COIL dataset: (a) Obj72 "cream cheese" rotated 0°, (b) Obj74 "duck toy" rotated 45°, (c) Obj68 "stapler" rotated 0° and (d) Obj68 "stapler" rotated 100°

The classification stage is performed in software that simulates the hardware constrained architecture proposed in section 2. The classification results are compared with those provided by the Matlab toolbox programs.

First goal consists on adapting E1_block and the shift block (Fig.3) to deal with high dimensional input data (1024 vector elements). The first experiment consists on the classification of two object of the COIL database: for example the "duck toy" (obj74) and the "cheese cream" (obj72) (see Fig.5). Twelve images of each class corresponding to twelve different angles of view $(0°, 30°, 60°, \ldots)$ have been used in the training. Learning parameters are $C = 10$ and $\gamma = 2$. After the training all the sample images are selected as support vectors, so the number of support vectors is 12. The other 60 images of each object have been used as test patterns. We achieved 0% classification error both in software SVM and FPGA hardware implementation. We have performed the same experiment with other image pairs from the COIL dataset. We observe that pair to pair comparison of images can be classified without errors when using a number of support vectors between 20 and 40. If we reduce the number of support vectors to 12, the classification error is kept below 5% in the hardware classifier. Worst cases correspond to images whose scale is changed depending on the angle of view as the "stapler" (see Fig.5 c and d). For example, the classification error of the hardware classifier between the "duck toy" and the "stapler" is 3.7%.

6 Conclusions

In this paper we have presented a FPGA implementation of a SVM based on a parallel computation of the support vectors kernel function. The starting point is the method proposed in [10], which provides a hardware-friendly kernel function that can be evaluated using an iterative CORDIC algorithm. Based on this method we have implemented the classification stage of a SVM on a low cost, constrained resources, FPGA device (Altera EP2C20 Cyclone II). The system shows to be very efficient in terms of hardware demanding resources and in terms of classification speed (each classification is performed in 10 clock cycles) compared with related implementations on other hardware platform as a microcontroller [8]. The system performance has been tested with the Iris database and a more sophisticated high dimensional image classification problem. Appealing results have been obtained for low resolution (32×32) image classification without needing a preprocessing stage for feature extraction. Each image is treated as a point in a high dimensional space.

Next step will be to expand the architecture to a multiclass classification problem, being able to identify each object from all the others in the COIL dataset. For multiclass classification (q classes) we need to evaluate the procedure. As a first approach we will follow the same strategy as in reference [2] where it is necessary the training of $q(q-1)/2$ between two classes and the evaluation of $q-1$ classifiers in the hardware. We have estimated that in the target FPGA device up to 8 classifiers (with 12 support vectors each) can be evaluated in parallel. In this case of the image classification problem an external memory is needed to store the support vectors.

References

1. Malamas, E.N., Petrakis, E.G.M., Zervakis, M., Petit, L., Legat, J.: A survey on industrial vision systems, applications and tools. Image and Vision Computing 21, 171–188 (2003)
2. Pontil, M., Verri, A.: Support Vector Machines for 3D Object Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(6), 637–646 (1998)
3. Bouyssounouse, B., Sifakis, J. (eds.): Embedded Systems Design. LNCS, vol. 3436. Springer, Heidelberg (2005)
4. Yang, F., Paindavoine, M.: Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification. IEEE Transactions on Neural Networks 14(5), 1162–1175 (2003)
5. Lee, P., Costa, E., McBader, S., Clementel, L., Sartori, A.: LogTOTEM: A Logarithmic Neural Processor and its Implementation on an FPGA Fabric. In: IJCNN 2007, Orlando, FL (2007)
6. `http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php`
7. Anguita, D., Boni, A., Ridella, S.: A Digital Architecture for Support Vector Machines: Theory, Algorithm and FPGA Implementation. IEEE Transactions on Neural Networks 14(5), 993–1009 (2003)
8. Boni, A., Pianegiani, F., Petri, D.: Low-Power and Low-Cost Implementations of SVM for Smart Sensors. IEEE Transactions on Instrumentation and Measurements 56(1), 39–44 (2007)
9. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
10. Anguita, D., Pischiutta, S., Ridella, S., Sterpi, D.: Feed-Forward Support Vector Machine Without Multipliers. IEEE Transactions on Neural Networks 17(5), 1328–1331 (2006)
11. Andraka, R.: A survey of CORDIC algorithms for FPGA based computers. In: Proc. ACM/SIGDA 6th Int. Symp.Field Programmable Gate Array, Monterey, CA, pp. 191–200 (1998)