Kerneltron: Support Vector 'Machine' in Silicon

Roman Genov and Gert Cauwenberghs

Department of Electrical and Computer Engineering
Johns Hopkins University, Baltimore, MD, 21218, USA
{roman,gert}@jhu.edu
http://bach.ece.jhu.edu

Abstract. Detection of complex objects in streaming video poses two fundamental challenges: training from sparse data with proper generalization across variations in the object class and the environment; and the computational power required of the trained classifier running real-time. The *Kerneltron* supports the generalization performance of a Support Vector Machine (SVM) and offers the bandwidth and efficiency of a massively parallel architecture. The mixed-signal VLSI processor is dedicated to the most intensive of SVM operations: evaluating a kernel over large numbers of vectors in high dimensions. At the core of the *Kerneltron* is an internally analog, fine-grain computational array performing externally digital inner-products between an incoming vector and each of the stored support vectors. The three-transistor unit cell in the array combines single-bit dynamic storage, binary multiplication, and zero-latency analog accumulation. Precise digital outputs are obtained through oversampled quantization of the analog array outputs combined with bit-serial unary encoding of the digital inputs. The 256 input, 128 vector *Kerneltron* measures 3 mm $\times$ 3 mm in 0.5 μm CMOS, delivers 6.5 GMACS throughput at 5.9 mW power, and attains 8-bit output resolution.

1 Introduction

Support vector machines (SVM) [1] offer a principled approach to machine learning combining many of the advantages of artificial intelligence and neural network approaches. Underlying the success of SVMs are mathematical foundations of statistical learning theory [2]. Rather than minimizing training error (*empirical risk*), SVMs minimize *structural risk* which expresses an upper bound on the generalization error, *i.e.,* the probability of erroneous classification on yet-to-be-seen examples. This makes SVMs especially suited for adaptive object detection and identification with sparse training data.

Real-time detection and identification of visual objects in video from examples is generally considered a hard problem for two reasons. One is the large degree of variability in the object class, *i.e.,* orientation and illumination of the object or occlusions and background clutter in the surrounding, which usually necessitates a large number of training examples to generalize properly. The

other is the excessive amount of computation incurred during training, and even in run-time.

Support vector machines have been applied to visual object detection, with demonstrated success in face and pedestrian detection tasks [3–6]. Unlike approaches to object detection that rely heavily on hand-crafted models and motion information, SVM-based systems learn the model of the object of interest from examples and work reliably in absence of motion cues. To reduce the computational burden of real-time implementation to a level that can be accommodated with available hardware, a reduced set of features are selected from the data which also result in a reduced number of support vectors [5]. The reduction in implementation necessarily comes at a loss in classification performance, a loss which is more severe for tasks of greater complexity.

The run-time computational load is dominated by evaluation of a kernel between the incoming vector and each of the support vectors. For a large class of permissible kernels, which include polynomial splines and radial kernels, this computation entails matrix-vector multiplication in large dimensions. For the pedestrian detection task in unconstrained environments [5], highest detection at lowest false alarm is achieved for very large numbers (thousands) of input dimensions and support vectors, incurring millions of matrix multiply-accumulates (MAC) for each classification. The computation recurs at different positions and scales across each video frame.

The *Kerneltron* offers a factor 100-10,000 improvement in computational efficiency (throughput per unit power) over the most advanced digital signal processors available today. It affords this level of efficiency at the expense of specificity: the VLSI architecture is dedicated to massively parallel kernel computation. Speed can be traded for power dissipation. Lower power is attractive in portable applications of kernel-based pattern recognition, such as visual aids for the blind [7].

Section 2 briefly summarizes feature extraction and SVM classification for object detection in streaming video. Section 3 describes the architecture and circuit implementation of the *Kerneltron*. Experimental results, scalability issues and application examples are discussed in Section 4.

2 Object Detection with Support Vector Machines

A support vector machine is trained with a data set of labeled examples. For pattern classification in images, relevant features are typically extracted from the training set examples using redundant spatial filtering techniques, such as overcomplete wavelet decomposition [4]. The classifier is trained on these feature vectors. In run time, images representing frames of streaming video are scanned by moving windows of different dimensions. For every unit shift of a moving window, a wavelet feature vector is computed and presented to the SVM classifier to produce a decision. The general block diagram of such a system is outlined in Fig. 1. A brief functional description of the major components follows next.

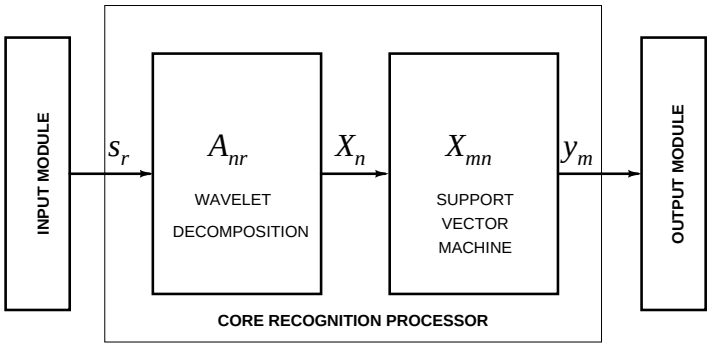


Fig. 1. Functional block diagram of the SVM classifier. The core of the system is a support vector machine processor for general object detection and classification. An overcomplete wavelet decomposition of the incoming sensory data at the input generates redundant input features to the SVM, providing for robust and relatively invariant classification performance.

2.1 Overcomplete Wavelet Decomposition

An overcomplete wavelet basis enables the system to handle complex shapes and achieve a precise description of the object class at adequate spatial resolution for detection [4]. The transformation of the sensory data **s** into the feature vector **X** is of the linear form

$$X_n = \sum_{r=1}^{R} A_{nr}\ s_r, \quad n = 1, \cdots, N, \tag{1}$$

where the wavelet coefficients A_{nr} form an overcomplete basis *i.e.,* $N > R$.

In visual object detection overcomplete Haar wavelets have been successfully used on pedestrian and face detection tasks [4, 5]. Haar wavelets are attractive because they are robust and particularly simple to compute, with coefficients A_{nr} that are either -1 or 1.

2.2 Support Vector Classification

Classification of the wavelet transformed features is performed by a support vector machine (SVM) [1]. From a machine learning theoretical perspective [2], the appealing characteristics of SVMs are:

1. The learning technique generalizes well even with relatively few data points in the training set, and bounds on the generalization error can be directly estimated from the training data.
2. The only parameter that needs tuning is a penalty term for misclassification which acts as a regularizer [8] and determines a trade-off between resolution and generalization performance [9].

3. The algorithm finds, under general conditions, a unique separating decision surface that maximizes the margin of the classified training data for best out-of-sample performance.

SVMs express the classification or regression output in terms of a linear combination of examples in the training data, in which only a fraction of the data points, called "support vectors," have non-zero coefficients. The support vectors thus capture all the relevant data contained in the training set. In its basic form, a SVM classifies a pattern vector $\mathbf{X}$ into class $y \in \{-1, +1\}$ based on the support vectors $\mathbf{X}_m$ and corresponding classes y_m as

$$y = \text{sign}(\sum_{m=1}^{M} \alpha_m \; y_m \; K(\mathbf{X}_m, \mathbf{X}) - b), \tag{2}$$

where $K(\cdot, \cdot)$ is a symmetric positive-definite kernel function which can be freely chosen subject to fairly mild constraints [1]. The parameters α_m and b are determined by a linearly constrained quadratic programming (QP) problem [2, 10], which can be efficiently implemented by means of a sequence of smaller scale, subproblem optimizations [3], or an incremental scheme that adjusts the solution one training point at a time [11]. Most of the training data $\mathbf{X}_m$ have zero coefficients α_m; the non-zero coefficients returned by the constrained QP optimization define the support vector set. In what follows we assume that the set of support vectors and coefficients α_m are given, and we concentrate on efficient run-time implementation of the classifier.

Several widely used classifier architectures reduce to special valid forms of kernels $K(\cdot, \cdot)$, like polynomial classifiers, multilayer perceptrons[1], and radial basis functions [13]. The following forms are frequently used:

1. Inner-product based kernels (*e.g.,* polynomial; sigmoidal connectionist):

$$K(\mathbf{X}_m, \mathbf{X}) = f(\mathbf{X}_m \cdot \mathbf{X}) = f(\sum_{n=1}^{N} X_{mn} \; X_n) \tag{3}$$

2. Radial basis functions (L_2 norm distance based):

$$K(\mathbf{X}_m, \mathbf{X}) = f(\|\mathbf{X}_m - \mathbf{X}\|) = f((\sum_{n=1}^{N} |X_{mn} - X_n|^2)^{\frac{1}{2}}) \tag{4}$$

where $f(\cdot)$ is a monotonically non-decreasing scalar function subject to the Mercer condition on $K(\cdot, \cdot)$ [2, 8].

With no loss of generality, we concentrate on kernels of the inner product type (3), and devise an efficient scheme of computing a large number of high-dimensional inner-products in parallel. Computationally, the inner-products comprise the most intensive part in evaluating kernels of both types (3) and (4).

[1] with logistic sigmoidal activation function, for particular values of the threshold parameter only

Indeed, radial basis functions (4) can be expressed in inner-product form:

$$f(\|\mathbf{X}_m - \mathbf{X}\|) = f((-2\mathbf{X}_m \cdot \mathbf{X} + \|\mathbf{X}_m\|^2 + \|\mathbf{X}\|^2)^{\frac{1}{2}}), \tag{5}$$

where the last two terms depend only on either the input vector or the support vector. These common terms are of much lower complexity than the inner-products, and can be easily pre-computed or stored in peripheral registers.

The computation of the inner-products takes the form of matrix-vector multiplication (MVM), $\sum_{n=1}^{N} X_{mn}\ X_n;\ m = 1, \ldots M$, where M is the number of support vectors. For large scale problems as the ones of interest here, the dimensions of the matrix $M \times N$ are excessive for real-time implementation even on a high-end processor. As a point of reference, consider the pedestrian and face detection task in [5], for which the feature vector length N is 1,326 wavelets per instance, and the number of support vectors M is in excess of 4,000. To cover the visual field over the entire scanned image at reasonable resolution (500 image window instances through a variable resolution search method) at video rate (30 frames per second), a computational throughput of 75×10^9 multiply-and-accumulate operations per second, is needed. The computational requirement can be relaxed through simplifying and further optimizing the SVM architecture for real-time operation, but at the expense of classification performance [4, 5].

3 Kerneltron: Massively Parallel VLSI Kernel Machine

The *Kerneltron* offers the computational power required for the unabridged SVM architecture to run in real-time, for optimal out-of-sample classification performance. The architecture is described next.

3.1 Core Recognition VLSI Processor

At the core of the system is a recognition engine, which very efficiently implements kernel-based algorithms, such as support vector machines, for general pattern detection and classification. The implementation focuses on inner-product computation in a parallel architecture.

Both wavelet and SVM computations are most efficiently implemented on the same chip, in a scalable VLSI architecture as illustrated schematically in Fig. 2. The diagram is the floorplan of the Kerneltron, with matrices projected as 2-D arrays of cells, and input and output vector components crossing in perpendicular directions alternating from one stage to the next. This style of scalable architecture also supports the integration of learning functions, through local outer product parameter updates [12], compatible with the recently developed incremental SVM learning rule [11]. The architecture maintains low input/output data rate. Digital inputs are fed into the processor through a properly sized serial/parallel converter shift register. A unit shift of a scanning moving window in an image corresponds to one shift of a new pixel per classification cycle, while a single scalar decision is produced at the output.

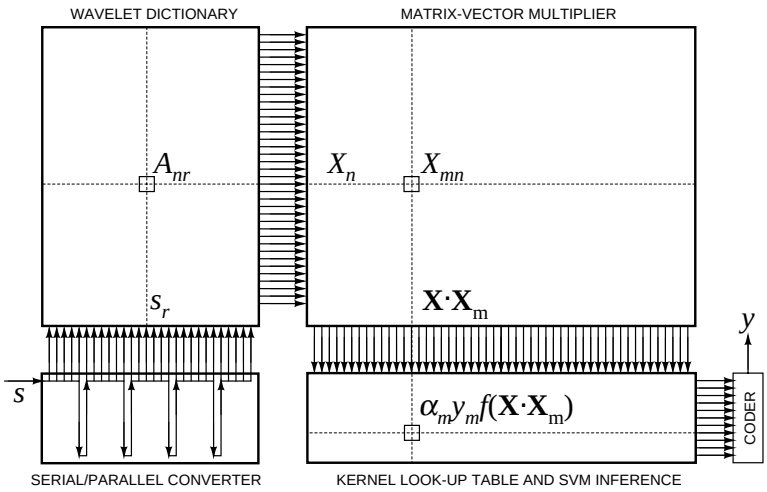


Fig. 2. The architecture of the core recognition processor, combining overcomplete wavelet decomposition with generalized support vector machine classification. Communication with outside modules is through a serial digital input/output interface for maximal flexibility and programmability, while the core internal computations are parallel and analog for optimal efficiency.

The classification decision is obtained in digital domain by thresholding the weighted sum of kernels. The kernels are obtained by mapping the inner-products $\mathbf{X} \cdot \mathbf{X}_m$ through the function $f(\cdot)$ stored in a look-up table.

By virtue of the inner-product form of the kernel, the computation can be much simplified without affecting the result. Since both the wavelet feature extraction and the inner-product computation represent linear transformations, they can be collapsed into a single linear transformation by multiplying the two matrices:

$$W_{mr} = \sum_{n=1}^{N} X_{mn} A_{nr}. \tag{6}$$

Therefore the architecture can be simplified to one that omits the (explicit) wavelet transformation, and instead transforms the support vectors.[2] For simplicity of the argument, we proceed with the inner-product architecture excluding the overcomplete wavelet feature extraction stage, bearing in mind that the approach extends to include wavelet extraction by merging the two matrices.

[2] Referred to the input prior to wavelet transformation, support vectors $\mathbf{s}_m$ need to be transformed *twice*: $W_{mr} = \sum_{n=1}^{N} \sum_{p=1}^{S} A_{np} A_{nr} s_{mp}$.

3.2 Mixed-Signal Computation

Computing inner-products between an input vector $\mathbf{X}$ and template vectors $\mathbf{W}_m$ in parallel is equivalent to the operation of matrix-vector multiplication (MVM)

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n, \tag{7}$$

with N-dimensional input vector X_n, M-dimensional output vector Y_m, and $M \times N$ matrix of coefficients W_{mn}. The matrix elements W_{mn} denote the support vectors X_{mn}, or the wavelet transformed support vectors (6) for convenience of notation.[3]

Internally Analog, Externally Digital Computation The approach combines the computational efficiency of analog array processing with the precision of digital processing and the convenience of a programmable and reconfigurable digital interface.

The digital representation is embedded in the analog array architecture, with matrix elements stored locally in bit-parallel form

$$W_{mn} = \sum_{i=0}^{I-1} 2^{-i-1} w_{mn}^{(i)} \tag{8}$$

and inputs presented in bit-serial fashion

$$X_n = \sum_{j=0}^{J-1} \gamma_j x_n^{(j)}, \tag{9}$$

where the coefficients γ_j are assumed in radix two, depending on the form of input encoding used. The MVM task (7) then decomposes into

$$Y_m = \sum_{n=0}^{N-1} W_{mn} X_n = \sum_{i=0}^{I-1} 2^{-i-1} Y_m^{(i)} \tag{10}$$

with MVM partials

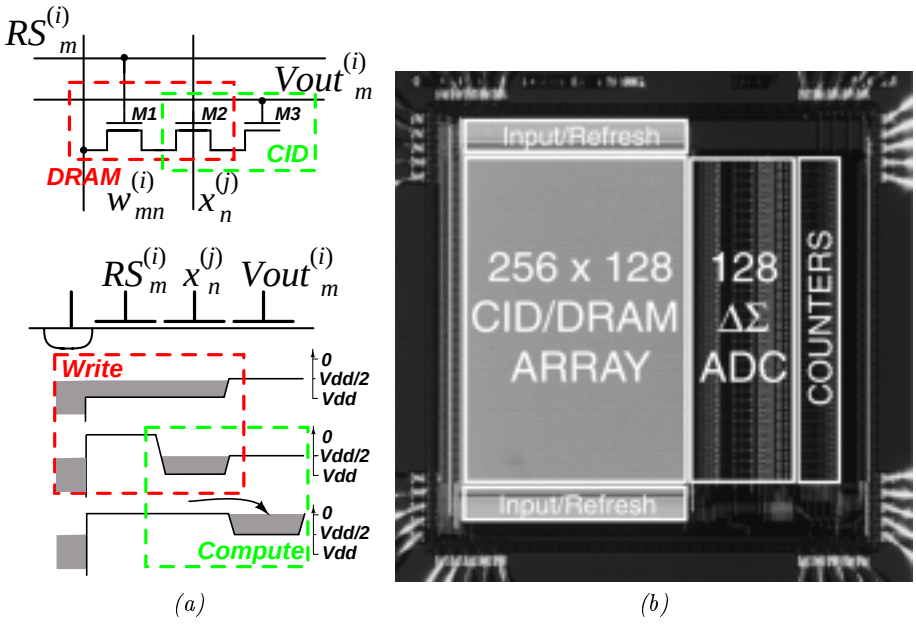
$$Y_m{}^{(i)} = \sum_{j=0}^{J-1} \gamma_j Y_m^{(i,j)}, \tag{11}$$

and

$$Y_m{}^{(i,j)} = \sum_{n=0}^{N-1} w_{mn}^{(i)} x_n^{(j)} \ . \tag{12}$$

The binary-binary partial products (12) are conveniently computed and accumulated, with zero latency, using an analog MVM array [14]-[17]. For this purpose we developed a 1-bit multiply-and-accumulate CID/DRAM cell.

[3] In the wavelet transformed case, $\mathbf{s}$ should be substituted for $\mathbf{X}$ in what follows.



(a) (b)

Fig. 3. *(a)* CID computational cell with integrated DRAM storage. Circuit diagram, and charge transfer diagram for active write and compute operations. *(b)* Micrograph of the *Kerneltron* prototype, containing containing an array of 256 × 128 CID/DRAM cells, and a row-parallel bank of 128 algorithmic $\Delta\Sigma$ ADCs. Die size is 3 mm × 3 mm in 0.5 μm CMOS technology.

CID/DRAM Cell and Array The unit cell in the analog array combines a CID (charge injection device [18]) computational element [16, 17] with a DRAM storage element. The cell stores one bit of a matrix element ${w_{mn}}^{(i)}$, performs a one-quadrant binary-unary (or binary-binary) multiplication of ${w_{mn}}^{(i)}$ and ${x_n}^{(j)}$ in (12), and accumulates the result across cells with common m and i indices. The circuit diagram and operation of the cell are given in Fig. 3 (a). It performs non-destructive computation since the transferred charge is sensed capacitively at the output. An array of cells thus performs (unsigned) binary-unary multiplication (12) of matrix ${w_{mn}}^{(i)}$ and vector ${x_n}^{(j)}$ yielding ${Y_m}^{(i,j)}$, for values of i in parallel across the array, and values of j in sequence over time. A 256 × 128 array prototype using CID/DRAM cells is shown in Fig. 3 (b).

To improve linearity and to reduce sensitivity to clock feedthrough, we use differential encoding of input and stored bits in the CID/DRAM architecture using twice the number of columns and unit cells as shown in Fig. 4 (a). This amounts to exclusive-OR (XOR), rather than AND, multiplication on the analog array, using signed, rather than unsigned, binary values for inputs and weights, ${x_n}^{(j)} = \pm 1$ and ${w_{mn}}^{(i)} = \pm 1$.

In principle, the MVM partials (12) can be quantized by a bank of flash analog-to-digital converters (ADCs), and the results accumulated in the digital

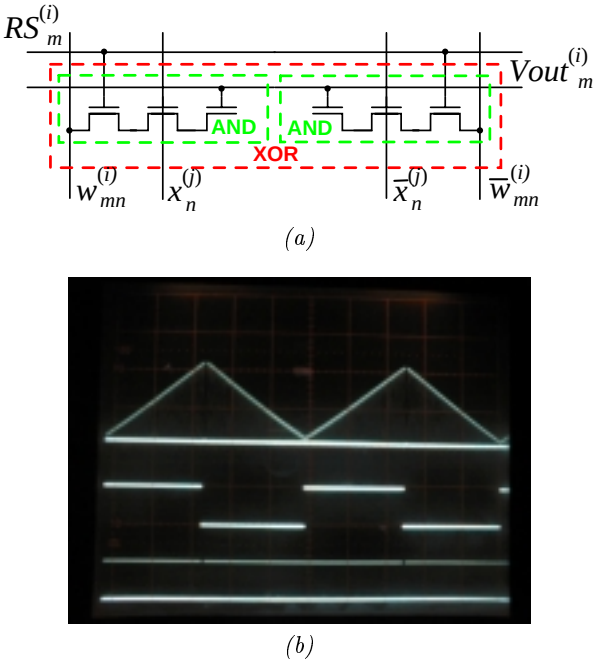


Fig. 4. *(a)* Two charge-mode AND cells configured as an exclusive-OR (XOR) multiply-and-accumulate gate. *(b)* Measured linearity of the computational array configured for signed multiplication on each cell (XOR configuration). Waveforms shown are, *top to bottom*: the analog voltage output, $Vout_m^{(i)}$, on the sense line; input data (in common for both input, ${x_n}^{(j)}$, and weight, ${w_{mn}}^{(i)}$, shift register); and input shift register clock.

domain according to (11) and (10) to yield a digital output resolution exceeding the analog precision of the array and the quantizers [19]. Alternatively, an oversampling ADC accumulates the sum (11) in the analog domain, with inputs encoded in unary format ($\gamma_i = 1$). This avoids the need for high-resolution flash ADCs, which are replaced with single-bit quantizers in the delta-sigma loop.

Oversampling Mixed-Signal Array Processing The precision of computation is limited by the resolution of the analog-to-digital converters (ADC) digitizing the analog array outputs. The conventional delta-sigma ($\Delta\Sigma$) ADC design paradigm allows to reduce requirements on precision of analog circuits to attain high resolution of conversion, at the expense of bandwidth. In the presented architecture a high conversion rate is maintained by combining delta-sigma analog-to-digital conversion with oversampled encoding of the digital inputs, where the delta-sigma modulator integrates the partial multiply-and-accumulate outputs (12) from the analog array according to (11).

Fig. 5 depicts one row of matrix elements W_{mn} in the $\Delta\Sigma$ oversampling architecture, encoded in $I = 4$ bit-parallel rows of CID/DRAM cells. One bit of a unary-coded input vector is presented each clock cycle, taking J clock cycles to

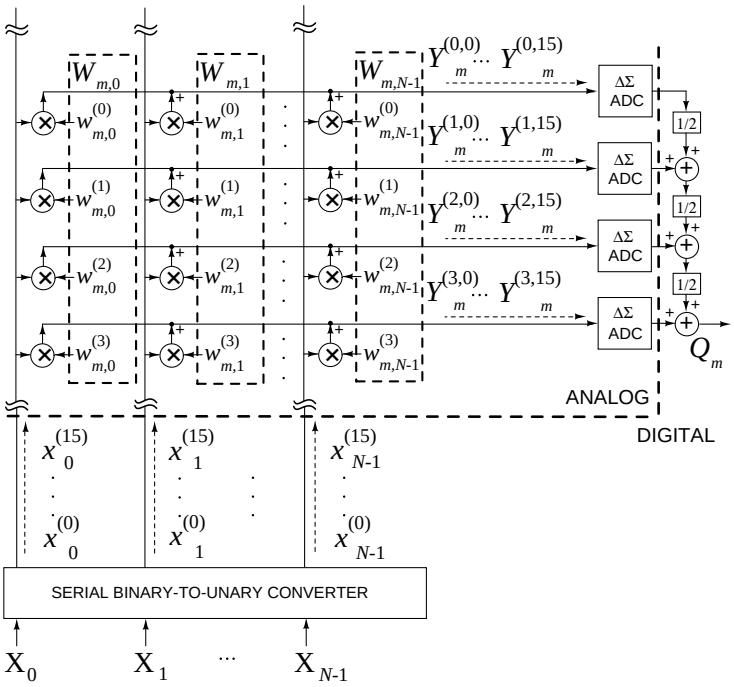


Fig. 5. Block diagram of one row of the matrix with binary encoded elements $w^{(i)}{}_{mn}$, for a single m and $I = 4$. Data flow of bit-serial unary encoded inputs $x^{(j)}{}_n$ and corresponding partial product outputs $Y^{(i,j)}{}_m$, with $J = 16$ bits. The full product for a single row $Y^{(i)}{}_m$ is accumulated and quantized by a delta-sigma ADC. The final product is constructed in the digital domain according to (10).

complete a full computational cycle (7). The data flow is illustrated for a digital input series $x_n{}^{(j)}$ of $J = 16$ unary bits.

Over J clock cycles, the oversampling ADC integrates the partial products (12), producing a decimated output

$$Q_m^{(i)} \approx \sum_{j=0}^{J-1} \gamma_j Y_m^{(i,j)} \ , \tag{13}$$

where $\gamma_j = 1$ for unary coding of inputs. Decimation for a first-order delta-sigma modulator is achieved using a binary counter. Higher precision are obtained in the same number of cycles J by using a residue resampling extended counting scheme [21].

Additional gains in precision can be obtained by exploiting binomial statistics of binary terms in the analog summation (12) [20]. In the present scheme, this would entail stochastic encoding of the digital inputs prior to unary oversampled encoding.

Table 1. Measured Performance

Technology	0.5 μm CMOS
Area	3mm × 3mm
Power	5.9 mW
Supply Voltage	5 V
Dimensions	256 inputs × 128 templates
Throughput	6.5 GMACS
Output Resolution	8-bit

4 Experimental Results and Discussion

4.1 Measured Performance

A prototype *Kerneltron* was integrated on a 3×3 mm^2 die and fabricated in 0.5 μm CMOS technology. The chip contains an array of 256×128 CID/DRAM cells, and a row-parallel bank of 128 algorithmic $\Delta\Sigma$ ADCs. Fig. 3 (b) depicts the micrograph and system floorplan of the chip.

The processor interfaces externally in digital format. Two separate shift registers load the templates (support vectors) along odd and even columns of the DRAM array. Integrated refresh circuitry periodically updates the charge stored in the array to compensate for leakage. Vertical bit lines extend across the array, with two rows of sense amplifiers at the top and bottom of the array. The refresh alternates between even and odd columns, with separate select lines. Stored charge corresponding to matrix element values can also be read and shifted out from the chip for test purposes. All of the supporting digital clocks and control signals are generated on-chip.

Fig. 4 (b) shows the measured linearity of the computational array, configured differentially for signed (XOR) multiplication. The case shown is where all complementary weight storage elements are actively set, and an alternating sequence of bits in blocks N is shifted through the input register.[4] For every shift in the input register, a computation is performed and the result is observed on the output sense line. The array dissipates 3.3 mW for a 10 μs cycle time. The bank of $\Delta\Sigma$ ADCs dissipates 2.6 mW yielding a combined conversion rate of 12.8 Msamples/s. Table 1 summarizes the measured performance.

4.2 System-Level Performance

Fig. 6 compares template matching performed by a floating point processor and by the *Kerneltron*, illustrating the effect of quantization and limited precision in the analog array architecture. An 'eye' template was selected as a 16×16 fragment from the *Lena* image, yielding a 256-dimensional vector. Fig. 6 (c) depicts the two-dimensional cross-correlation (inner-products over a sliding window) of the 8-bit image with the 8-bit template computed with full precision. The same

[4] ${w_{mn}}^{(i)} = 1$; ${x_n}^{(j)} = -1$ for $n = 1, \ldots N$; and ${x_n}^{(j)} = -1$ for $n = N+1, \ldots 2N$.

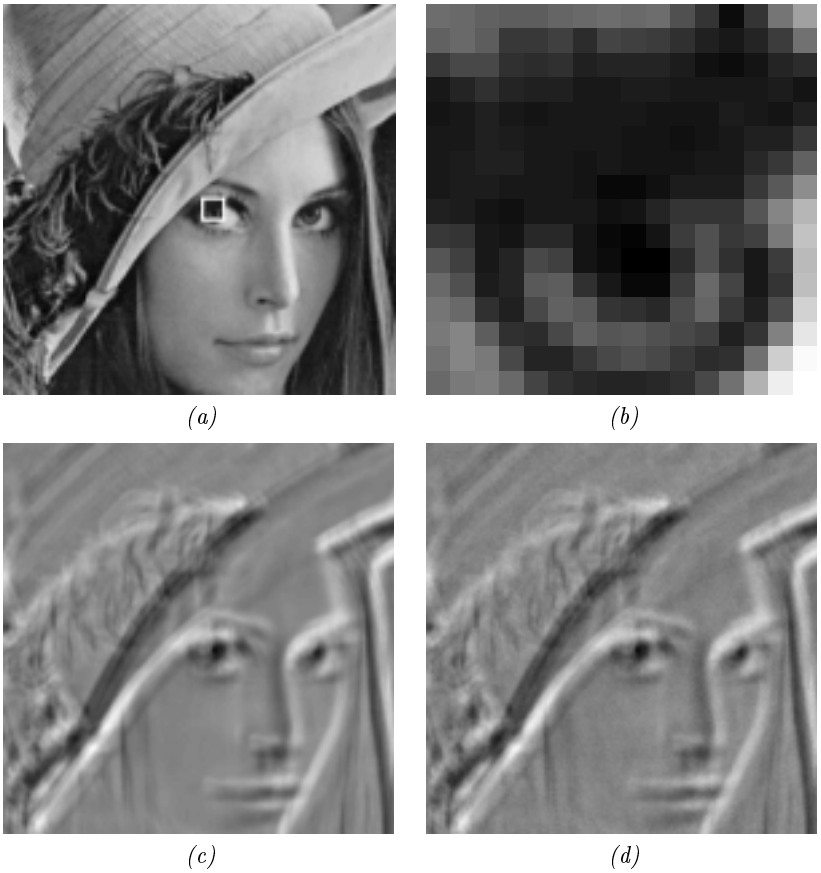


Fig. 6. Cross-correlation of fragments of *Lena (a)* and the eye template *(b)* computed by a 32-bit floating point processor with 8-bit encoded inputs *(c)* and by *Kerneltron* with 8-bit quantization and 4-bit encoded inputs *(d)*.

computation performed by the *Kerneltron*, with 4-bit quantization of the image and template and 8-bit quantization of the output, is given in Fig. 6 (d). Differences are relatively small, and both methods return peak inner-product values (top matches) at both eye locations in the image.[5] The template matching operation is representative of a support vector machine that combines nonlinearly transformed inner-products to identify patterns of interest.

[5] The template acts as a spatial filter on the image, leaking through spectral components of the image at the output. The *Lena* image was mean-subtracted.

4.3 Large-Scale Computation

The design is fully scalable, and can be expanded to any number of input features and support vectors internally as limited by current fabrication technology, and externally by tiling chips in parallel.

The dense CID/DRAM multiply-accumulate cell ($18\lambda \times 45\lambda$ where λ is the technology scaling parameter) supports the integration of millions of cells on a single chip in deep submicron technology, for thousands of support vectors in thousand dimensional input space as the line-width of the fabrication technology continues to shrink. In 0.18 μm CMOS technology (with $\lambda = 0.1\mu m$), 64 computational arrays with 256×128 cells each can be tiled on a 8mm $\times$ 8mm silicon area, with two million cells integrated on a single chip.

Distribution of memory and processing elements in a fine-grain multiply-and-accumulate architecture, with local bit-parallel storage of the W_{mn} coefficients, avoids the memory bandwidth problem that plagues the performance of CPUs and DSPs. Because of fine-grain parallelism, both *throughput* and *power dissipation* scale linearly with the number of integrated elements, so every cell contributes one kernel unit operation and one fixed unit of dissipated energy per computational cycle. Let us assume a conservative cycle time of 10 μs. With two million cells, this gives a computational throughput of 200 GOPS, which is adequate for the task described in Section 2.2. The (dynamic) power dissipation is estimated[6] to be less than 50 mW which is significantly lower than that of a CPU or DSP processor even though computational throughput is many orders of magnitude higher.

4.4 Applications

The *Kerneltron* benefits real-time applications of object detection and recognition, particularly in artificial vision and human-computer interfaces. Applications extend from SVMs to any pattern recognition architecture that relies on computing a kernel distance between an input and a large set of templates in large dimensions.

Besides throughput, power dissipation is a main concern in portable and mobile applications. Power efficiency can be traded for speed, and a reduced implementation of dimensions similar to the version of the pedestrian classifier running on a Pentium PC (27 input features) [4, 5] could be integrated on a chip running at 100 μW of power, easily supported with a hearing aid type battery for a lifetime of several weeks.

One low-power application that could benefit a large group of users is a navigational aid for visually impaired people. OpenEyes, a system developed for this purpose [7] currently runs a classifier in software on a Pentium PC. The software solution offers great flexibility to the user and developer, but limits the mobility of the user. The *Kerneltron* offers the prospect of a low-weight, low-profile alternative.

[6] The parameters of the estimate are: $\lambda = 0.1\mu m$; 3 V power supply; 10 μs cycle time.

5 Conclusions

A massively parallel mixed-signal VLSI processor for kernel-based pattern recognition in very high dimensions has been presented. Besides support vector machines, the processor is capable of implementing other architectures that make intensive use of kernels or template matching. An internally analog, externally digital architecture offers the best of both worlds: the density and energetic efficiency of a charge-mode analog VLSI array, and the convenience and versatility of a digital interface.

An oversampling configuration relaxes precision requirements in the quantization while maintaining 8-bit effective output resolution, adequate for most vision tasks. Higher resolution, if desired, can be obtained through stochastic encoding of the digital inputs [20].

A 256×128 cell prototype was fabricated in 0.5 μm CMOS. The combination of analog array processing, oversampled input encoding, and delta-sigma analog-to-digital conversion yields a computational throughput of over 1GMACS per milliwatt of power. The architecture is easily scalable and capable of delivering 200GOPS at 50mW of power in a 0.18 μm technology— a level of throughput and efficiency by far sufficient for real-time SVM detection of complex objects on a portable platform.

Acknowledgments: This research was supported by ONR N00014-99-1-0612, ONR/DARPA N00014-00-C-0315 and WatchVision Corporation. The chip was fabricated through the MOSIS service.

References

1. Boser, B., Guyon, I. and Vapnik, V., "A training algorithm for optimal margin classifier," in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp 144-52, 1992.
2. Vapnik, V. *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.
3. Osuna, E., Freund, R., and Girosi, F., "Training support vector machines: An application to face detection," in *Computer Vision and Pattern Recognition*, pp 130-136, 1997.
4. Oren, M., Papageorgiou, C., Sinha, P., Osuna, E. and Poggio, T. "Pedestrian detection using wavelet templates," in *Computer Vision and Pattern Recognition*, pp 193-199, 1997.
5. Papageorgiou, C.P, Oren, M. and Poggio, T., "A General Framework for Object Detection," in *Proceedings of International Conference on Computer Vision*, 1998.
6. H. Sahbi, D. Geman and N. Boujemaa, "Face Detection Using Coarse-to-Fine Support Vector Classifiers," *IEEE Int. Conf. Image Processing (ICIP'2002),* Rochester NY, 2002.
7. S. Kang, and S.-W. Lee,"Handheld Computer Vision System for the Visually Impaired," Proc. of 3rd International Workshop on Human-Friendly Welfare Robotic Systems, Daejeon, Korea, pp. 43-48, 2002.
8. Girosi, F., Jones, M. and Poggio, T. "Regularization Theory and Neural Networks Architectures," *Neural Computation*, vol. **7**, pp 219-269, 1995.

9. Pontil, M. and Verri, A., "Properties of Support Vector Machines," *Neural Computation*, vol. **10**, pp 977-996, 1998.
10. Burges, C., "A Tutorial on Support Vector Machines for Pattern Recognition," in U. Fayyad, editor, *Proceedings of Data Mining and Knowledge Discovery*, pp 1-43, 1998.
11. Cauwenberghs, G. and Poggio, T., "Incremental and Decremental Support Vector Machine Learning," in Adv. Neural Information Processing Systems, Proc. of 2000 IEEE NIPS Conf., Cambridge MA: MIT Press, 2001.
12. Cauwenberghs, G. and Bayoumi, M., *Learning on Silicon, Analog VLSI Adaptive Systems*, Norwell MA: Kluwer Academic, 1999.
13. Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T. and Vapnik, V. "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Functions Classifiers," *IEEE Transactions on Signal Processing* (to appear 1997).
14. A. Kramer, "Array-based analog computation," *IEEE Micro,* vol. **16** (5), pp. 40-49, 1996.
15. A. Chiang, "A programmable CCD signal processor," *IEEE Journal of Solid-State Circuits,* vol. **25** (6), pp. 1510-1517, 1990.
16. C. Neugebauer and A. Yariv, "A Parallel Analog CCD/CMOS Neural Network IC," Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN'91), Seattle, WA, vol. **1**, pp 447-451, 1991.
17. V. Pedroni, A. Agranat, C. Neugebauer, A. Yariv, "Pattern matching and parallel processing with CCD technology," Proc. IEEE Int. Joint Conference on Neural Networks (IJCNN'92), vol. **3**, pp 620-623, 1992.
18. M. Howes, D. Morgan, Eds., *Charge-Coupled Devices and Systems,* John Wiley & Sons, 1979.
19. R. Genov, G. Cauwenberghs "Charge-Mode Parallel Architecture for Matrix-Vector Multiplication," *IEEE T. Circuits and Systems II,* vol. **48** (10), 2001.
20. R. Genov, G. Cauwenberghs, "Stochastic Mixed-Signal VLSI Architecture for High-Dimensional Kernel Machines," to appear in *Advances in Neural Information Processing Systems,* Cambridge, MA: MIT Press, vol. 14, 2002.
21. G. Mulliken, F. Adil, G. Cauwenberghs, R. Genov, "Delta-Sigma Algorithmic Analog-to-Digital Conversion," IEEE Int. Symp. on Circuits and Systems (ISCAS'02), Scottsdale, AZ, May 26-29, 2002. ISCAS'2002.