# A Novel FPGA-based SVM Classifier

Markos Papadonikolakis #1, Christos-Savvas Bouganis #2

# *Electrical and Electronic Engineering Department, Imperial College London*
*Exhibition Road, South Kensington, London, SW7 2AZ, UK*
1 markos.papadonikolakis07@imperial.ac.uk
2 christos-savvas.bouganis@imperial.ac.uk

*Abstract*—**Support Vector Machines (SVMs) are a powerful supervised learning tool, providing state-of-the-art accuracy at a cost of high computational complexity. The SVM classification suffers from linear dependencies on the number of the Support Vectors and the problem's dimensionality. In this work, we propose a scalable FPGA architecture for the acceleration of SVM classification, which exploits the device heterogeneity and the dynamic range diversities among the dataset attributes. Furthermore, this work introduces the first FPGA-oriented cascade SVM classifier scheme, which intensifies the custom-arithmetic properties of the heterogeneous architecture and boosts the classification performance even more. The implementation results demonstrate the efficiency of the heterogeneous architecture, presenting a speed-up factor of 2-3 orders of magnitude, compared to the CPU implementation, while outperforming other proposed FPGA and GPU approaches by more than 7 times.**

## I. Introduction

Support Vector Machines [1] are a powerful machine learning method, providing good generalization performance for a wide range of regression and classification tasks. In SVMs, a training dataset, consisting of pairs of input vectors and desired outputs, is used to model and construct the decision function of the system and, hence, they are considered as an instance of supervised learning. During the training phase, the system identifies the *Support Vectors* (SVs), which are those data points that can best build a separation model for the classes. Those vectors are then used to predict the class of any future data point during the classification phase.

While the SVM training can be considered as an online or offline task, the classification is mostly performed real-time on newly obtained data. Face detection, speech recognition, bioinformatics or geostatistical analysis require online classification and have real-time constraints. However, the SVM classification is a computationally expensive task, linearly dependent on the classification data load, the population of the Support Vectors and the problem's dimensionality. For large-scale problems, the classification task becomes very time consuming and urgent needs for acceleration arise.

In this work, a fully scalable heterogeneous FPGA architecture for the acceleration of the SVM classification is proposed. The objective is the exploitation of the precision requirements of the problem's attributes, in order to use the available resources of a modern FPGA device in the most efficient manner. The maximum parallelization factor for the SVM training processor is achieved by maintaining the device's ratio between DSPs and logic resources. Moreover, this work proposes a cascade classifier, which enhances the characteristics of the heterogeneous classifier and further exploits the relationship between the precision and the resource utilization. The implementation results show that a significant gain on throughput or resource usage can be obtained for large-scale cases, where the available resource budget does not allow for a fully-unrolled implementation, or where extra throughput is desired.

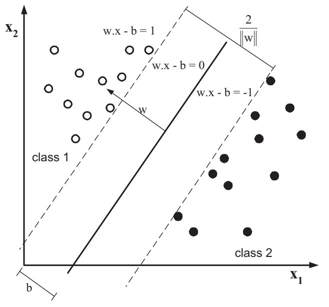

Fig. 1. SVM separating hyperplane.

The rest of this paper is organized as follows: Section II gives the theoretical background on the SVM training and classification. In Section III, the heterogeneous FPGA implementation for the SVM classifier is presented, among with the cascade classifier architecture. Section IV provides the implementation results, while the conclusion is presented in Section V.

## II. Non-linear Support Vector Machines

On a two-class classification problem, the SVMs objective is the construction of a separating hyperplane $\mathbf{w}\cdot\mathbf{x}-b=0$ to attain maximum separation between the classes, as shown in Fig. 1. The classes' hyperplanes are parallel to the separating one, lying on each of its sides. The Euclidean Distance between the two hyperplanes is 2/$\|\mathbf{w}\|$, thus the objective of SVMs is to minimize $\|\mathbf{w}\|$:

$$\min \tfrac{1}{2}\|\mathbf{w}\|^2, \text{ s.t. } y_i(K(\mathbf{w},\mathbf{x}_i)-b)\geq 1,\ 1\leq i\leq N. \quad (1)$$

where $K(\cdot,\cdot)$ denotes a kernel function [2], $\mathbf{x}_i$ is the training data, label $y_i$ denotes the belonging class of datum $\mathbf{x}_i$, $\mathbf{w}$ is the perpendicular vector to the hyperplane direction, $b$ is the offset to the origin and $N$ is the training set size.

The SVM training builds a model that is able to distinguish the belonging class of any future data based on the Support

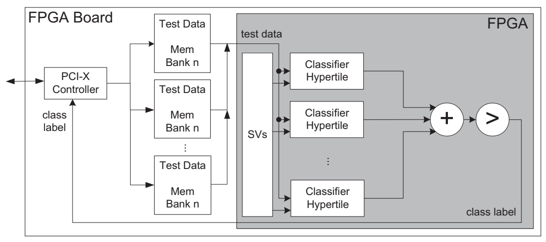

Fig. 2. The FPGA architecture of the SVM classifier.

Vectors obtained by the training dataset. On the classification phase, any new datum $\mathbf{x}$ is classified according to the output of the decision function:

$$f(\mathbf{x}) = \mathrm{sgn}(\sum_{i=1}^{N_{\mathrm{SV}}} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b), \quad (2)$$

where $N_{\mathrm{SV}}$ is the total number of Support Vectors identified in the training phase. It is easily derived that the computational time of the non-linear SVM classification task is linearly dependent on the size of the classification dataset, on the Support Vector population $N_{\mathrm{SV}}$ and on the problem's dimensionality. The decision function $f$ for non-linear SVMs requires massive matrix-vector operations for large datasets. However, matrix-vector computations offer significant parallelization potential, which can be exploited by the parallel hardware resources of an FPGA device.

## III. Hardware Mapping of the SVM Classification

### A. FPGA Architecture of the SVM Classifier

The rationale behind the design of the SVM classifier is the exploitation of the parallel computational power offered by the FPGA heterogeneous resources and the high memory bandwidth of the FPGA internal memories in the most efficient way, in order to speed up the decision function (2). The computation of $f(\mathbf{x})$ involves matrix-vector operations, which are highly parallelizable. Therefore, the problem can be segmented into smaller ones and parallel units can be instantiated for the processing of each subproblem.

The proposed FPGA architecture for the SVM classifier is shown in Fig. 2. The Support Vectors are loaded into the internal FPGA memories. The classification dataset is loaded on the RAMs of the FPGA board, which serve as FIFO units between the host and the FPGA. The data points are streamed into the FPGA and fed to each classifier hypertile, which is the processing unit of the architecture. The hypertile outputs the kernel evaluations $K(\cdot,\cdot)$ , which are then added in parallel. When the sum of (2) is completed, the class of each datum is streamed out by the system. In cases where the SVs cannot fit in the FPGA RAM blocks, the classification is performed into multiple steps. The details of this case though are not addressed in this work.

A previous work presented the design flow for a heterogeneous processing unit for the SVM training phase [3]. Here, the core idea is exploited for the design of a classifier processing unit, which can be instantiated in the proposed

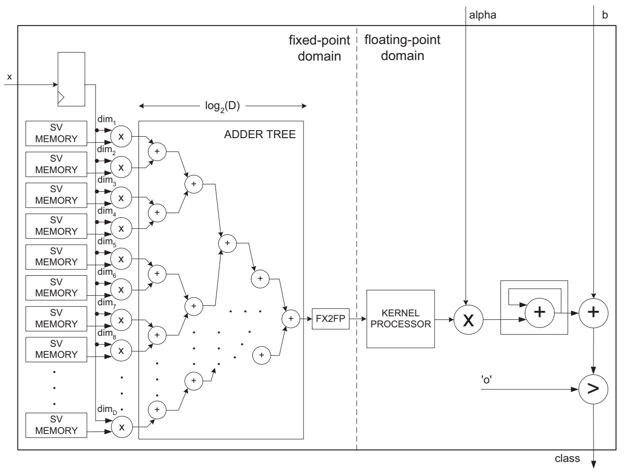

Fig. 3. Hypertile of the heterogeneous SVM classifier.

FPGA classifier, as well as in the proposed cascade scheme. Regarding the need to exploit the different dynamic ranges of a heterogeneous dataset's features, the hardware implementation of an inner product should be efficiently designed. In cases where the dataset contains continuous along with categorical or binary attributes, implementing a hypertile based on a MAC unit is a waste of resources. Instead, dedicating a custom precision multiplier for each attribute significantly reduces the resource usage in cases of high precision diversities among the dataset dimensions. A scheme with parallel multipliers feeding a pipelined adder tree allows for the exploitation of the dynamic range diversities and integrates an adaptive circuit, which uses precisely the resources needed for the data representation.

The architecture of the heterogeneous classifier hypertile is presented in Fig. 3. The data path is split in fixed- and floating-point domains. The internal FPGA memories store an SV subset and feed the parallel multipliers, each of which is dedicated to a single data point dimension. During the preprocessing stage, the features have been sorted according to their precision requirements, in order to minimize the adder tree resource usage. The adder tree produces the inner product - or the norm, in the Gaussian case - for the floating-point kernel processor. The fixed-point inner products are interpreted into IEEE754 single precision floating-point format before fed to the kernel. The kernel processor implements the targeted kernel function, while its output is accumulated to produce the final result of the hypertile. The design flow of the hypertile targets to maintain the ratio between the DSP and the logic resource usage at the original ratio of the targeted device, hence minimizing the resource cost and leading to the maximum parallelization factor.

### B. FPGA Architecture of the Cascade Classifier

The computational complexity of the SVM classification has driven the research to the proposal of multiple-SVM systems, such as the cascade classifier. This architecture describes a system of many SVM classifiers in a cascade fashion, where each classifier feeds its output to the next one. The main idea is to bias each cascade level in a way that one of the binary decisions is very confident, while the other is uncertain and propagates the data point to the next cascade level. Burges' cascade [4] uses a different reduced set of vectors for each classifier, while Romdhani [5] proposes a "greedy" cascade where every classifier uses the kernel evaluations of the previous ones. Although these approaches present interesting results and accelerate the overall classification time of SVMs, they are not particularly useful for a hardware architecture of a cascade classifier. The reason is that each SVM classifier added in the cascade requires a full hardware implementation and, in that way, the area cost of a cascade classifier increases proportionally to the number of its units.

This work aims at adding a new dimension to the cascade approach, which is the exploitation of the bit-precision utilized by each classifier in the cascade. The proposed heterogeneous classifier presents an area cost which is directly dependent on the problem's characteristics, the bit-precision of each attribute and the dimensionality. This important quality allows for the design of a cascade classifier, which implements each of the cascade units with a different numerical precision. The first classifier in the cascade is implemented in lower precision, hence with smaller area cost and with larger throughput potential than the full-precision classifier. Certainly, the lower precision is expected to have some impact on the classification accuracy of the module. Thus, the low-precision classifier uses a more relaxed decision function $f_{lp}$, where two classification thresholds $\mathrm{C_{cn}}$ and $\mathrm{C_{cp}}$ are introduced, instead of the sign function of (2). In that way, the decision function of the cascade classifier can be formulated as:

$$F(\mathbf{x}) = \begin{cases} -1 & \text{, if } f_{lp}(\mathbf{x}) \le \mathrm{C_{cn}}, \\ f_{hp}(\mathbf{x}) & \text{, if } \mathrm{C_{cn}} \le f_{lp}(\mathbf{x}) \le \mathrm{C_{cp}}, \\ 1 & \text{, else} \end{cases} \tag{3}$$

, where

$$f_{lp}(\mathbf{x}) = \sum_{i=1}^{N_{\mathrm{LP}}} y_i \alpha_i K_{lp}(\mathbf{x}_i, \mathbf{x}) + b_{lp}, \tag{4}$$

$$f_{hp}(\mathbf{x}) = \sum_{i=1}^{N_{\mathrm{HP}}} y_i \alpha_i K_{hp}(\mathbf{x}_i, \mathbf{x}) + b_{hp}. \tag{5}$$

By choosing a targeted precision for the $LP$ classifier, the desired false rate for the $LP$ classifier indicates the appropriate thresholds $\mathrm{C_{cn}}$ and $\mathrm{C_{cp}}$.

The decision function $f_{lp}$ of the low precision classifier $LP$ uses the set of the support vectors $N_{\mathrm{LP}}$, obtained by training the SVM in low precision. On the other hand, the high precision classifier $HP$ uses a set of $N_{\mathrm{HP}}$ vectors. The $HP$ classifier is not trained over all the training set; instead, the $LP$ classifier operates as a zone pass filter in order to pass a reduced training set to the $HP$ classifier. This decreases the training time of the $HP$ classifier and the size $N_{\mathrm{HP}}$.

The procedure flow for the cascade classifier is shown in Fig. 4. The $LP$ classifier is first trained over the entire training dataset. Then, it classifies the training set according to (3),

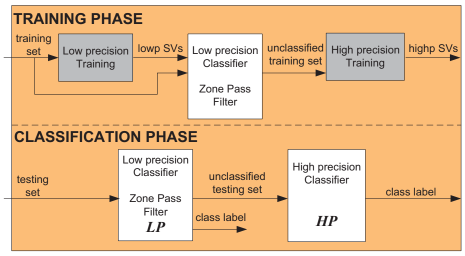

Fig. 4. Training and Classification flow of the cascade classifier.

but with training thresholds $\mathrm{C_{tn}}$ and $\mathrm{C_{tp}}$. These thresholds control the size of the training set for the $HP$ classifier. In the classification phase, the new data points are fed to the $LP$ classifier, which uses (3) and makes a more relaxed classification decision. Since the high-precision SV set is generally a subset of the $LP$ one, it is expected that $N_{\mathrm{HP}} \le N_{\mathrm{LP}}$ and a common pool memory for SVs can be used.

The objective is to attain a larger front-end throughput for the $LP$ classifier compared to the initial throughput of the heterogeneous classifier. That means that the $LP$ classifier requires more heterogeneous hypertiles in parallel than the initial heterogeneous approach, in order to increase its parallelization factor. At the same time, the $HP$ classifier, which is not fully-unrolled, must have a throughput sufficient enough in order not to cause any back pressure to the front-end classifier. The thresholds $\mathrm{C_{cn}}$ and $\mathrm{C_{cp}}$ determine the throughput ratio $a$ between the $HP$ and $LP$ classifiers, which controls the unrolling factor of the $HP$ circuit. The desired ratio $a$ specifies some discrete design points and the optimum design for the cascade classifier can be found through an iterative algorithm.

## IV. Implementation Results

### A. *Performance of the Heterogeneous SVM Classifier*

The targeted device for the proposed architecture was the Altera's Stratix III EP3SE260. The results can be easily expanded to other targeted devices by changing the resource constraints of the design flow. The architecture is captured in VHDL and the floating-point modules are generated by the Altera tools and the Altera floating-point compiler [6]. The targeted operating frequency of all produced designs ranges between 180-250MHz.

The proposed architecture is evaluated for a set of synthetic datasets that have the same number of total bits per datum $\mathbf{x}$ but different precision distribution among their features. The histograms of these synthetic datasets are shown in Fig. 5. The proposed FPGA classifier was compared to a C implementation on a PC with a 3GHz Intel CORE 2 DUO and 2GB of RAM. Fig. 6 shows that the proposed FPGA classifier presents a speed-up of two orders of magnitude. It can be derived that, even when the total bit width of all the attributes per data point is equal, the dynamic range distribution among the features can offer significant reduction on the resource usage and, hence, important performance gain.

The performance of the heterogeneous classifier was also evaluated on datasets which present a uniform distribution of

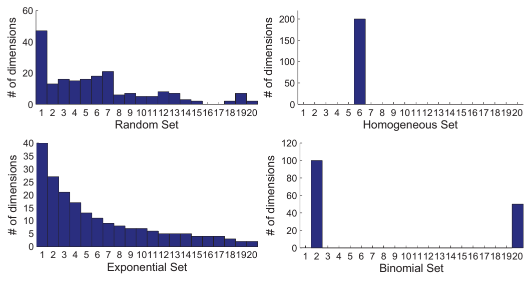

Fig. 5. Histograms of synthetic datasets.

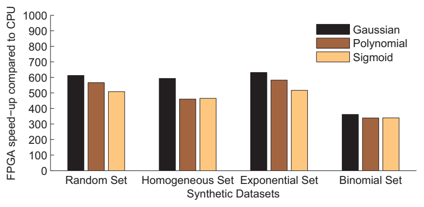

Fig. 6. Achieved speed-up for synthetic datasets.

bit-precision across their attributes. The computational complexity of (2) and the FPGA execution time depend linearly on the classification problem size. Fig. 7 shows the achieved speed-up of the FPGA compared to the CPU as a function of the SV size for a range of classification problem sizes. The spikes on the speed-up factor are caused by the SV reloads, when the SV set cannot fit entirely on the FPGA memories. However, it is easily observed that, for larger datasets, the SV reloads affects the total execution time much less and the FPGA speed-up smoothens.

The proposed heterogeneous classifier is compared to previous FPGA and GPU works ([7], [8], [9]) on the MNIST dataset, which is a very popular benchmark in SVMs. MNIST is a handwritten digits recognition problem of 60K homogeneous data points with $D$=784, using samples of 28$\times$28 images. [7] reports a raw computational speed of the SVM classification core of 40 GMACs (Giga Multiply-ACcumulate operations per second) on a Xilinx Virtex-5 LX330T. The proposed heterogeneous classifier architecture achieves 282.24 GMACs, which is 7.056$\times$ higher than in [7]. In [7], the dot-products are implemented only with DSP units and the available logic resources and the bit-precision requirements are not exploited. The works on GPUs ([8], [9]) formulate the MNIST problem as an even-vs-odd digit classification. On a 2.5K classification set, [8] reports an execution time of 1.98s, while [9] needs 8.4s to classify 10K vectors. The proposed heterogeneous architecture achieves 7.8$\times$ and 8.55$\times$ speed-ups compared to [8] and [9], respectively.

Moreover, the proposed cascade classifier architecture is compared to the proposed heterogeneous classifier on the MNIST dataset. Using a Gaussian kernel, the full-precision training identifies 9,438 SVs and the classification accuracy is 98.96%. The baseline heterogeneous architecture presents a parallelization factor of 2 for the targeted device and the overall speed-up is 480.14$\times$ compared to the CPU classification. The resource budget of the targeted device allows for 1 $HP$ and 4 parallel $LP$ heterogeneous classifiers for the cascade scheme. The $HP$ classifier of the cascade is half-rolled and requires the 50% of the resource utilization of the fully-unrolled heterogeneous classifier. The classification speed-up of the cascade classifier was doubled for the same resource utilization as the baseline heterogeneous classifier.

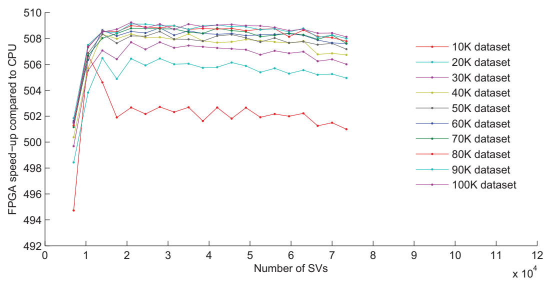

Fig. 7. Achieved speed-up on uniform datasets function to the size of the support vectors set.

## V. Conclusion

This work presents a fully scalable heterogeneous FPGA architecture for the acceleration of the SVM classification. By exploiting the dynamic range diversities among the training problem features, the word-length optimizations in the fixed-point domain allow for efficient usage of the available resources. The proposed architecture achieved to speed-up the CPU classification execution time by 2-3 orders of magnitude, while outperforming over 7$\times$ other works on FPGAs and GPUs. Furthermore, this work introduces a novel cascade classifier scheme, which exploits the characteristics of the heterogeneous architecture and results in a hardware-friendly approach with even higher performance capabilities.

## References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

[2] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London*, 1909.

[3] M. Papadonikolakis and C.-S. Bouganis, "A heterogeneous fpga architecture for support vector machine training," in *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, 2010, pp. 211–214.

[4] C. J. Burges, "Simplified support vector decision rules," in *In International Conference on Machine Learning*, 1996, pp. 71–77.

[5] S. R. M. Ratsch and T. Vetter, "Efficient face detection by a cascaded support vector machine expansion," in *A Royal Society of London Proceedings Series*, 2004, pp. 460:3283–3297.

[6] M. Langhammer, "Floating point datapath synthesis for fpgas," in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, Sept. 2008, pp. 355–360.

[7] S. Cadambi, I. Durdanovic, V. Jakkula, M. Sankaradass, E. Cosatto, S. Chakradhar, and H. Graf, "A massively parallel fpga-based coprocessor for support vector machines," in *FCCM '09*, 2009, pp. 115–122.

[8] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 104–111.

[9] A. Carpenter, "cusvm: A cuda implementation of support vector classification and regression," 2009. [Online]. Available: http://patternsonascreen.net/cuSVM.html