2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines

A Heterogeneous FPGA Architecture for Support Vector Machine Training

Markos Papadonikolakis, Christos-Savvas Bouganis
Department of Electrical and Electronic Engineering
Imperial College London
London, UK
Email: markos.papadonikolakis07@imperial.ac.uk, christos-savvas.bouganis@imperial.ac.uk

***Abstract*—Support Vector Machines is a powerful supervised learning tool. Its training phase, however, is a time-consuming task and heavily dependent on the training dataset size and dimensionality. In this work, we propose a scalable FPGA architecture for the acceleration of SVM training, which exploits the heterogeneous nature of the device and the diversities of the precision requirements among the dataset attributes. The maximum parallelization potential is obtained by maintaining the usage of DSPs and logic resources at the initial ratio of the FPGA device. The results demonstrate the efficiency of the heterogeneous architecture in both homogeneous and heterogeneous datasets. The proposed architecture outperforms other proposed designs by more than 6 times, in terms of raw computational speed.**

I. Introduction

Support Vector Machines (SVMs) [1] are a popular supervised learning method for classification and regression problems. In supervised classification, the machine is first trained using a training data set. For a two-class classification problem, the objective of SVMs is to construct 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 $2/||\mathbf{w}||_2$, thus the objective of SVMs is to maximize the distance between the classes' hyperplanes or, in other words, to minimize $\|\mathbf{w}\|_2$:

$$\min \tfrac{1}{2}\|\mathbf{w}\|_2^2, \text{ s.t. } y_i(K(\mathbf{w}, \mathbf{x}_i) - b) \geq 1, \ 1 \leq i \leq N, \quad (1)$$

where $K(\cdot, \cdot)$ denotes a kernel function, $\mathbf{x}_i$ is the training data, label y_i denotes the belonging class of datum $\mathbf{x}_i$ and takes the values -1,1, $\mathbf{w}$ is the perpendicular vector to the hyperplane direction, b is the offset to the origin and N is the training set size. The training task focuses on the identification of those training samples that lie closest to the hyperplane and determine its direction; these samples are called *Support Vectors* (SVs).

The SVM training time is heavily dependent on the training dataset size and the problem's dimensionality. Especially for online learning applications with real-time constraints, this dependence turns SVM training into the bottleneck of the overall performance. Therefore, much research is focused on the SVM training phase, proposing algorithms and methods to speed-up this time consuming task.

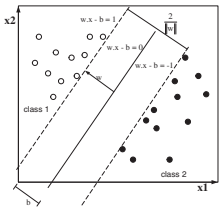

Figure 1. SVM separating hyperplane.

Solving the SVM training problem by using QP techniques is a demanding and computationally expensive task, especially for large high-dimensional datasets. Therefore, many algorithms have been proposed to decompose this large QP problem into smaller ones. Algorithms like Sequential Minimum Optimization (SMO) [2] and SVM$^{\text{LIGHT}}$ [3] solve the SVM problem analytically and they apply to linear and non-linear SVM. The execution time of these algorithms scale superlinearly with the size N of the training set. Some other works, [4], approach the problem from a geometric point of view. The most important characteristic of all these software approaches is that they lead to analytical solutions of the training problem and that, for each iteration, the kernels between a data point and all the other dataset points need to be computed. Details on the targeted algorithm can be found in [4] and [5].

Depending on the problem's nature, training datasets can be categorized as homogeneous or heterogeneous. Homogeneous datasets are often met in imaging applications, where the dynamic range requirements among the dataset features are equal. However, many real world datasets present high diversities among the dynamic ranges of their features. The attributes of these heterogeneous datasets can be continuous, indexing, categorical or boolean. The efficiency and performance of the SVM training system can be maximized by customizing the use of the available computing resources. This is a strong motivation for targeting SVM training solvers on computing devices which can exploit the potential of custom precision arithmetic, like FPGAs.

The FPGA architectures proposed in previous works, [5], [6], map well on homogeneous SVM problems. However, the potential of high variety among the features dynamic ranges is not explored. In this work, a fully scalable

978-0-7695-4056-6/10 $26.00 © 2010 IEEE
DOI 10.1109/FCCM.2010.39

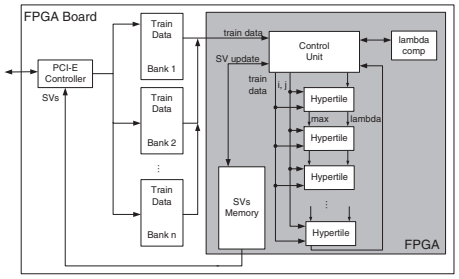

Figure 2. FPGA architecture for Gilbert's algorithm on the SVM training.

heterogeneous FPGA architecture for the acceleration of the SVM training 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. The main contribution of this work is the proposal of an FPGA architecture for the SVM training problem, which fully exploits the parallel processing power of an FPGA device and offers scalability and adaptivity to the classification problem's nature, with respect to the available resource constraints. This is the first work in the field of accelerating SVM training through an FPGA device that makes full use of the custom number presentation supported by the device and aims at full utilization of its heterogeneous components.

II. FPGA Architecture

The design of the FPGA architecture is driven by the need to exploit the parallelization potential of the targeted algorithm, in order to speed-up the SVM training application. The most time consuming part of the algorithm is the evaluation of the kernel functions. These computations are required for the projection of all dataset points N along a line, a task which is easily parallelizable. Thus, the objective is to efficiently map the FPGA resources in order to decompose the original problem into smaller ones and employ parallel processing units to solve the subproblems.

The proposed FPGA Architecture is shown in Fig. 2. The training data are downloaded to the FPGA board through the available PCI-E bus. The training set is stored in the RAM banks of the FPGA board, in case the dataset cannot fit in the internal FPGA memories. Each subproblem is mapped to a hypertile, which performs the projection task. The control unit is responsible for the algorithmic steps and the I/O interfaces. In this way, the algorithm iteration time is solely dependent on the projection task.

A. Hypertile Analysis of a Heterogeneous Architecture

Considering the hardware implementation of kernel functions, the issues regarding their dynamic range requirements

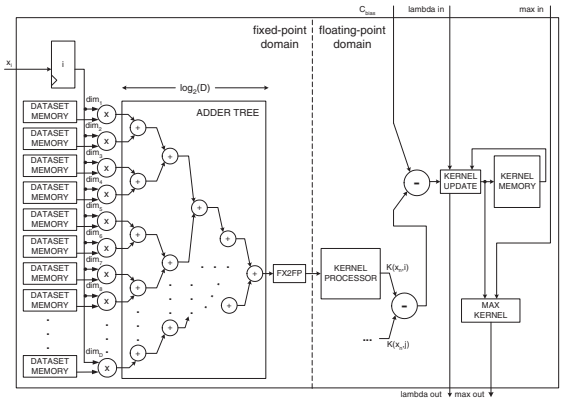

Figure 3. Hypertile of the heterogeneous FPGA architecture.

should be first addressed. For a homogeneous dataset with P bits per dimension and dimensionality D, an inner product representation would require $2 \cdot P + \log_2(D)$; fixed-point arithmetic is adequate for a wide range of input dataset characteristics. However, the high-dimensionality of a kernel and its increased dynamic range require a different approach from the latter case. Floating-point precision is essential for the hardware implementation of kernel functions.

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. A MAC unit is a good choice when the precision requirements among all dimensions are the same. Nevertheless, if the dataset contains continuous along with categorical or binary attributes, implementing MAC units is a waste of resources. Instead, dedicating a custom precision multiplier for each attribute significantly reduces the resource usage in cases with high precision diversities among the dataset dimensions. A scheme with parallel multipliers feeding a pipelined adder tree allows the exploitation of the dynamic range diversities and integrate an adaptive circuit, which uses precisely the resources needed for the data representation.

The architecture of the heterogeneous hypertile is presented in Fig. 3. The data path is split in fixed- and floating-point domains. The internal FPGA memories store the subproblem training data and feed the parallel multipliers, each of which is dedicated for a single data point attribute. The features of each data point are efficiently packed and stored in the same memory lines of the hypertile RAM blocks, since the parallel inner product blocks need access to all data point's attributes in each clock cycle. The features are sorted and added according to their precision requirements in order to minimize the adder tree resource usage. The adder tree produces the inner product for the floating-point kernel processor. The fixed-point inner products are interpreted into floating-point format before fed to the kernel.

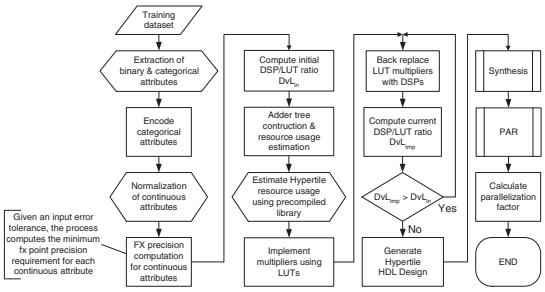

Figure 4. Heterogeneous architecture, FPGA design flow.

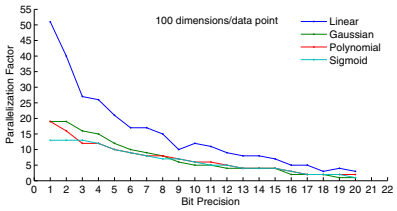

Figure 5. Parallelization factor scaling as a function to bit precision per feature for homogeneous problems.

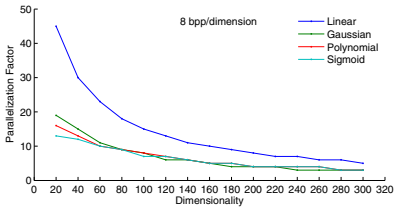

Figure 6. Parallelization factor scaling as a function to dimensionality of homogeneous problems.

B. Hypertile Design Flow

The floating-point domain of the hypertile is designed in IEEE754 single floating-point precision and the hardware design is not dependent on the dataset characteristics. However, the fixed-point domain is customized according to the precision of each feature. The proposed design flow implements the most efficient adder tree, with respect to the problem characteristics. This is achieved by sorting the features in descending precision order and computing the minimum required precision for each node. Furthermore, the design maintains a good balance between instantiated DSPs and LUTs, in order to produce the most area efficient circuit.

The design flow of the heterogeneous hypertile is presented in Fig. 4. After analyzing the input dataset, the tool extracts the categorical and binary attributes and encodes them using the minimum required precision. The dataset is then normalized, a necessary step for the SVMs functionality, so as to prevent a high-offset feature to govern the computations. The dynamic ranges of continuous features are computed with a specified error tolerance, allowing for future data diversities. Given the targeted device, the initial ratio between available DSPs and LUTs is computed. Consequently, the algorithm sorts the features in descending precision order to produce the most cost-efficient adders possible and computes the required precision for each node of the fixed-point circuit. The adder tree is then constructed and the tool estimates the hypertile resource usage, using a precompiled library of the floating-point modules.

Concerning the DSP allocation process to the hypertile, the parallel multipliers are initially instantiated using LUTs and the DSP/LUT ratio is updated. Starting from the larger precision feature, the LUT-based multipliers are back-replaced with DSP ones, until the initial ratio is reached. Thereafter, the hypertile HDL design is automatically generated and, finally, the flow can compute the maximum number of parallel hypertile instantiations that can fit into the targeted FPGA device.

III. Results

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 critical path in all produced designs is located in the floating-point domain. The targeted operating frequency ranges between 200-250MHz and 160-200MHz for the linear and the non-linear SVM case, respectively. Since the projection task governs the execution time of the algorithm, the parallelization factor achieved by the parallel hypertile instances is a reliable and good metric for the SVM training speed-up.

The proposed heterogeneous architecture scales well in homogeneous problems. Fig. 5 illustrates the achieved parallelization factor of the heterogeneous architecture as a function to the bit precision of a 100-feature homogeneous dataset, for the linear case and all the targeted kernels. Both parallelization factors scale in an exponential-like fashion. Fig. 6 presents the scaling of the achieved parallelization factor as a function to a homogeneous problem's dimensionality with fixed 8-bit precision per feature, which is common to image processing applications. The performance of the proposed architecture is similar to the one of our previous homogeneous architecture [5].

Fig. 7 presents the histograms of some popular classification datasets [7]. Forest Covertype has 581K data points of 54 attributes, Adult's size is 32K with dimensionality D =14, Web is a 300-binary attribute dataset of 49K points and Internet Usage Data is a dataset with 72 categorical and binary attributes, of total 10K instances. MNIST is another homogeneous dataset, like Web. It is a handwritten digits recognition problem with D=784, using 10K training samples of 28×28 images. Its histogram, which is not included in Fig. 7, is similar to the Web one, since both are homogeneous datasets. Fig. 8 shows the parallelization

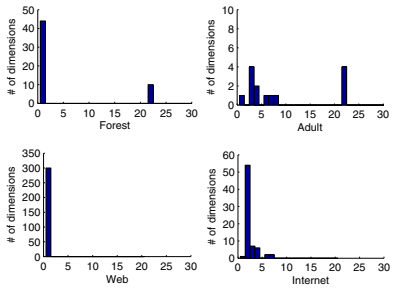

Figure 7. Histograms of real-world datasets.

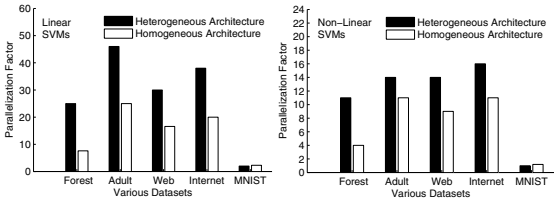

Figure 8. Achieved parallelization factor for real-world datasets.

factors for the targeted problems. In most cases, the heterogeneous architecture outperforms the homogeneous one by a 2-3 factor. Even for the Web problem, which is a homogeneous one, the homogeneous architecture [5] is more area-expensive than the proposed heterogeneous scheme.

The homogeneous architecture presents better parallelization factor for the MNIST problem, which yields up the only disadvantage of the heterogeneous architecture. In the case where a homogeneous hypertile instance cannot fit entirely into the device, it can still be instantiated with less MAC tiles attached to the floating-point domain input; its pipeline will not be fully utilized, but the overall performance is still increased. On the other hand, the parallel multiplier and adder tree scheme of the heterogeneous architecture does not allow for sub-instantiating a hypertile. On the MNIST problem, a previous work [6], which implements the SMO decomposition method, reports a raw computational speed of the SVM processor core of 128 parallel MACs (Multiply-ACcumulate operations) on a Xilinx Virtex-5 LX330T. The targeted Altera device has slightly less number of registers but 4$\times$ more DSPs than the FPGA in [6]. For the non-linear SVMs, the proposed architecture presents 784 MACs, which is 6.125$\times$ higher than in [6]. In [6], the dot-product computations are implemented only with DSP units and the available LUT resources are not used. It should also be mentioned that the SVM processor of [6] only computes the kernels' dot-products and feeds them to the host for the completion of the kernel evaluations, while the proposed hypertile has an embedded fully pipelined floating-point kernel processor. In addition, the achieved operating frequency of the proposed architecture is slightly higher than the one reported in [6].

The implementation results highlight several remarks concerning the performance behavior and scalability of the heterogeneous architecture. In training problems of heterogeneous datasets, the proposed FPGA architecture presents more area-efficient designs than the homogeneous architecture, while maintaining the high throughput achieved by the previous approach. The benefit of the heterogeneous architecture depends on the heterogeneity of the dataset; the larger the dynamic range diversities among the dataset features are, the more area-efficient the hypertile becomes. By maintaining the initial resource ratio, which is constrained by the given FPGA device, the heterogeneous architecture maximizes the achieved parallelization factor. Furthermore, in homogeneous case studies, the proposed design accomplishes to keep the resource usage as low as the homogeneous architecture.

IV. Conclusion

This work presents a fully scalable heterogeneous FPGA architecture for the acceleration of the SVM training problem. The design concept can be used for the mapping of any decomposition methods, whose algorithms are governed by matrix-vector kernel evaluations. By exploiting the dynamic range diversities among the training problem features, a fully customized processing unit is designed. The hypertile design takes full advantage of the custom-precision arithmetic that FPGAs can offer. The efficiency of the proposed design increases with the precision diversities of the dataset attributes.

References

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

[2] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[3] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proceedings of ICML-99*, pp. 200–209. [Online]. Available: citeseer.ist.psu.edu/joachims99transductive.html

[4] S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. Murthy, "A fast iterative nearest point algorithm for support vector machine classifier design," *IEEE Transactions on Neural Networks*, vol. 11, pp. 124–136, 1999.

[5] M. Papadonikolakis and C.-S. Bouganis, "A scalable fpga architecture for non-linear svm training," in *ICFPT 2008*, pp. 337–340.

[6] 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.

[7] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html