Incremental Learning for Real-time Partitioning for FPGA Applications

Belhedi Wiem, Kammoun Ahmed and Hireche Chabha
Department of Research, Altran Technologies, Rennes, France

Keywords: Hardware/Software Partitioning, Incremental Learning, Classification, Incremental Kernel SVM (InKSVM), Online Learning.

Abstract: The co-design approach consists in defining all the sub-tasks of an application to be integrated and distributed on software or hardware targets. The introduction of conventional cognitive reasoning can solve several problems such as real-time hardware/software classification for FPGA-based applications. However, this requires the availability of large databases, which may conflict with real-time applications.
The proposed method is based on the Incremental Kernel SVM (InKSVM) model. InKSVM learns incrementally, as new data becomes available over time, in order to efficiently process large, dynamic data and reduce computation time. As a result, it relaxes the assumption of complete data availability and provides fully autonomous performance.
Hence, in this paper, an incremental learning algorithm for hardware/software partitioning is presented. Starting from a real database collected from our FPGA experiments, the proposed approach uses InKSVM to perform the task classification in hardware and software. The proposal has been evaluated in terms of classification efficiency. The performance of the proposed approach was also compared to reference works in the literature.
The results of the evaluation consist in empirical evidence of the superiority of the InKSVM over state-of-the-art progressive learning approaches in terms of model accuracy and complexity.

1 INTRODUCTION

Hardware/software partitioning consists of dividing the application's computations between those which will be performed by conventional software (that are sequential instructions) and those that run parallel circuits which will be performed by specific hardware. This is referred to as co-design, the design is twofold, a software design and a hardware design. The co-design approach is then to define all the sub-tasks of an application to integrate and to distribute them on software or hardware targets (Kammoun et al., 2018).

The automatic partitioning of a system specification is a complex issue (considered as a NP-hard problem) due to the high number of parameters to account for. In addition, it requires adapted computing powers. The problem gets more complicated when working on embedded systems that will be subject to real-time constraints, surface consumption, etc.

Faced with the complexity of the software / hardware partitioning problem, several approaches have adopted manual methods to assign each task to the corresponding entity on architecture.

Others have been made in regard of hardware/software partitioning (Shui-sheng et al., 2006), (Wang et al., 2016), (Zhang Tao and Zhichun, 2017), (Ouyang et al., 2017), (Wijesundera et al., 2018), (Yousuf and Gordon-Ross, 2016).

All these approaches are unique in nature and each offers advantages of its own. However, real-time applications require unsupervised learning in order for a fully autonomous performance (Skliarova and Sklyarov, 2019). Hence, in this paper, we proposed an unsupervised learning algorithm for hardawre/software partitioning. Starting for a real database that was collected from our experimentations on FPGA, the proposed work makes the use of Incremental Kernel-SVM in order to perform task-classification into hardware and software. The proposal was evaluated in terms of its classification efficiency and its performance was also compared to benchmark approaches.

The partitioning category uses an automatic method; in this case an optimization algorithm, which takes into account all the parameters of the problem, will be adopted.

In this work, our goal is to develop an algorithm that will naturally group the data into two groups:

Wiem, B., Ahmed, K. and Chabha, H.
Incremental Learning for Real-time Partitioning for FPGA Applications.
DOI: 10.5220/0010202705980603
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 598-603
ISBN: 978-989-758-484-8

hardware tasks and software tasks. Fore this, we use in incremental Kernel SVM (InKSVM) to perform the task classification in hardware and software.

The overall organization of the paper is as follows. After the introduction, we present the various Incremental learning methods algorithms that were used for real-time classification applications in section 2. In section 3, the proposed learning strategy of unsupervised hardawre/software partitioning is presented. In Section 4, experimental results of the proposed approach are presented and compared to those given by benchmark approaches. In Section 5, we summarize results from different perspectives and we conclude the paper.

2 RELATED WORK: INCREMENTAL LEARNING METHODS FOR REAL-TIME CLASSIFICATION

Artificial intelligence has drawn great attention in recent years and it can be found in many practical applications, such as (Belhedi and Hannachi, 2020).

However, in real world problems, not all the data is always always available at the very beginning. For instance, this is the case for autonomous systems(e.g.,autonomous driving and robotics) which need a continuous adjustment, as new data is available. Moreover, other systems need human feedback. In such situations, classic or batch models retrain from scratch, which requires high computational complexity and training time.

Hence incremental-learning-based algorithms were proposed in the literature in order to solve real-time challenges.

More precisely, for the sake of solving classification problems for non-linearly separable data, many incremental classifiers have been proposed in the literature. The rest of this section reviews and discusses a selection of Incremental classifiers that are namely: Online Random Forest (ORF)(Lakshminarayanan et al., 2014), incremental Learning Vector Quantization (ILVQ)(Shui-sheng et al., 2006), Learn++ (LPP)(Polikar et al., 2001), Stochastic Gradient Descent (SGD)(Bottou, 2010), and Incremental Extreme Learning Machine (IELM). A performance comparaison of these classifiers with respect to the proposed approach is presented in section 4

Online Random Forest (ORF) (Lakshminarayanan et al., 2014) is an incremental version of the Extreme Random Forest. In fact, it goes a step further in order to refine the prediction.

A predefined number of trees grows continuously by adding splits whenever enough samples are gathered within one leaf. Tree ensembles are very popular, due to their high accuracy, simplicity and parallelization capability.

In fact, instead of using a predetermined set of data at the start, the ORF injects new data during the process. It works by creating new trees whenever there are enough sample based on the result of existing trees and adds those to the forest.

Incremental Learning Vector Quantization (ILVQ) (Shui-sheng et al., 2006) extends the Generalized Learning Vector Quantization (GLVQ) to a dynamically growing model by continuous insertion of new prototypes. The (GLVQ)(Liang et al., 2006) is an improvement of the basic method in which reference vectors are updated based on the steepest descent method in order to minimize the cost function. The cost function is determined so that the obtained learning rule satisfies the convergence condition.

Learn++ (LPP) (Polikar et al., 2001) utilizes an ensemble of classifiers by generating multiple hypotheses using training data sampled according to carefully tailored distributions. The outputs of the resulting classifiers are combined using a weighted majority voting procedure. In essence, both Learn++ and AdaBoost which is it inspired by generating an ensemble of weak classifiers, each trained using a different distribution of training samples. The outputs of these classifiers are then combined using Littlestone's majority-voting scheme to obtain the final classification rule.

Stochastic Gradient Descent (SGD) (Bottou, 2010) As the data size and means of stocking it had gone up over the last decade, the SGD is an attempt to help processing these faster and thus reduces the computing time, which is the limiting factor in the current statistical machine learning methods. A more precise analysis uncovers qualitatively different tradeoffs for the case of small-scale and large-scale learning problems. The large-scale case involves the computational complexity of the underlying optimization algorithm in non-trivial ways. Unlikely optimization algorithms such as stochastic gradient descent show amazing performance for large-scale problems. In particular, second order stochastic gradient and averaged stochastic gradient are asymptotically efficient after a single pass on the training set.

Incremental Extreme Learning Machine (IELM) (Liang et al., 2006) is a variant of the ELM algorithm, which are feedforward neural networks for classification and feature learning with a single layer or multiple layers of hidden nodes, where the parameters of hidden nodes (not just the weights connecting inputs to hidden nodes) need not be tuned. In OS-ELM, the parameters of hidden nodes (the input weights and biases of additive nodes or the centers and impact factors of RBF nodes) are randomly selected and the output weights are analytically determined based on the sequentially arriving data. One of the main strength of IELM is its versatility, as it can both handle data arriving one by one or chunk-by-chunk with varying chunk size.

3 PROPOSED APPROACH

Let x_i be the training vectors and $y_i = \pm 1$ are their corresponding labels. The goal of the SVM-based classification is to find the optimal separating function that reduces to a linear combination of kernels on the training data as follows:

$$f(x) = \sum_{j=1}^{N} \alpha_j K(x_j, x) + b \tag{1}$$

The coefficients α_j are obtained by minimizing the following quadratic objective function subject to the lagrange multiplier (b) and with the symmetric positive definite matrix (Q) constrains:

$$min_{0 \leq \alpha_j \geq C} : W = \sum_{i,j} \alpha_i Q_{ij} \alpha_j - \sum_i \alpha_i + b \sum_i y_i \alpha_i \tag{2}$$

Hence, as $Q = y_i y_j K(x_i, x_j)$ is positive definite, and K are positive-definite, then the Karush-Kuhn-Tucker (KKT) condition on the loss function W are sufficient for optimality and are written as:

$$\begin{cases} g_i = \frac{dW}{d\alpha_i} = \sum_i Q_{ij}\alpha_j + y_i f(x_i) - 1 \begin{cases} \geq 0, if \alpha_i = 0 \\ = 0, \\ if 0 < \alpha_i < C \\ \leq 0, otherwise \end{cases} \\ \frac{dW}{db} = \sum_i y_j \alpha_j = 0 \end{cases} \tag{3}$$

Hence, the KKT condition divides the dataset into three sets as:

- The first set, S, consists of support vectors that are strictly located on the margin ($y_i f(x_i) = 1$).
- The second set consists of error support vectors that exceed the margin.
- The third set consists of non-support vectors.

Before a new data is added, the KKT condition is satisfied for all the training samples. The key idea is to maintain equilibrium on all data points by updating the Lagrange multiplier α_i in order to satisfy the KKT condition that can be also expressed as:

$$\begin{cases} \Delta g_i = Q_{ic}\Delta\alpha_c + \sum_{j \in S} Q_{ij}\Delta_j + y_i \Delta b, \\ \forall i \in \{1, ..., l\} \cup \{c\} \\ \Delta \frac{dW}{db} = y_c \Delta\alpha_c + \sum_{j \in S} \Delta\alpha_j = 0 \end{cases} \tag{4}$$

where α_c is the coefficient being incremented of the new data point x_c outside the initial database. Since $g_i = 0$ for the margin vectors inside S, the equation 4 can be rewritten in matrix form as:

$$\begin{pmatrix} \Delta g_c \\ \Delta g_s \\ \Delta g_0 \\ 0 \end{pmatrix} = \begin{pmatrix} y_c & Q_{c,s} \\ y_s & Q_{s,s} \\ y_0 & Q_{0,s} \\ 0 & y_s^T \end{pmatrix} \begin{pmatrix} \Delta b \\ \Delta\alpha_s \end{pmatrix} + \Delta\alpha_c \begin{pmatrix} Q_{c,c}^T \\ Q_{c,s}^T \\ Q_{c,0}^T \\ y_c \end{pmatrix} \tag{5}$$

Hence, in equilibrium:

$$\Delta b = \beta \Delta\alpha_c \tag{6}$$

and

$$\Delta\alpha_j = \beta_j \Delta\alpha_j, \forall j \in D \tag{7}$$

where the sensitivity coefficients are give by

$$\begin{pmatrix} \beta \\ \beta_{s1} \\ \vdots \\ \beta_{sl_s} \end{pmatrix} = -A \begin{pmatrix} y_c \\ Q_{s1c} \\ \vdots \\ Q_{sl_sc} \end{pmatrix} \tag{8}$$

Where $A = Q^{-1}$ and $\beta_j = 0$ for all j outside S. Hence, according to the equation 4, the margin change according to:

$$\Delta g_i = \gamma_i \Delta\alpha_c, \forall i \in \cup\{c\} \tag{9}$$

where the margin sensitivity γ_i is expressed as:

$$\gamma_i = Q_{ic} + \sum_{j \in S} Q_{ij}\beta_j + y_i\beta, \forall i \notin S \tag{10}$$

$\gamma_i = 0$ for all i in S.
Hence, IKSVM efficiently updates the previously trained model.

4 EVALUATION

4.1 Comparison with Incremental Learning Methods

In order to provide empirical evidence of the superiority of the proposed model, several experiments are conducted. First we have compared it with state-of-the-art incremental learning approaches in terms of accuracy and model complexity. This experiment shows the advantages of incremental learning over batch learning. For this experiment, tests were conducted using several artificial databases whose descriptions are reported in Table 1. According to Figure 1 and Figure 2, the proposed approach provides better accuracy compared to batch in terms of accuracy values, since incremental models yield a cleaner solution.

In addition, the proposed method is applied for solving industry applications: hardawre/software partitioning for FPGA-based applications. For this, the database is of a collection of experiments that were conducted in Altran Technologies.

The studied incremental methods are namely: Online Random Forest (ORF), Incremental Learning Vector Quantization (ILVQ), Learn++ (LPP), Incremental Extreme Learning Machine (IELM) and Stochastic Gradient Descent (SGD).

Table 1: Evaluated datasets.

Incremental Method	**Description**			
	Train	Test	Features	Classes
Border	4000	1000	2	3
Overlap	3960	990	2	4
Letter	16000	4000	16	26
DNA	1400	1186	180	3

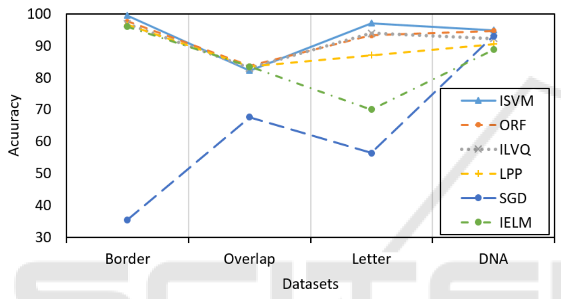

Figure 1: Comparison against incremental learning methods in terms of accuracy.

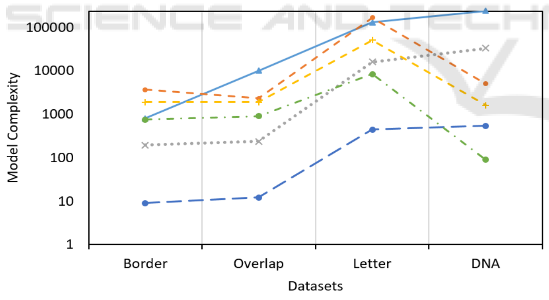

Figure 2: Comparison against incremental learning methods in terms of model complexity.

4.2 Application of Real-time Hardawre/Software Partitioning for FPGA-based Applications (Wiem et al., 2018)

4.2.1 Database

The database is of a collection of experiments that were conducted in Altran Technologies. As described in (Belhedi and Hannachi, 2020), it consists of several tasks with their respective Execution time (ET), Energy, Allocation, and type (Hardware or Software).

The allocation step is one of the most important in the partitioning process. In fact, by definition, the Allocation is to find the best set of components which allows to implement the functionalities of a given system. However, the sheer number of available software and hardware makes the task extremely complex.

4.2.2 Comparison of Partitioning Results with Conventional Approaches

The comparison of partitioning results with respect to conventional Approaches is illustrated in Table 2. Results are reported for the proposed method as well as 1)Lee(Lee et al., 2007), 2)Lin(Lin et al., 2006), 3)GHO(Lee et al., 2009), 4)GA(Zou et al., 2004), as well as the Hardware orient partition (HOP).

The results illustrated in Table 2 show the superiority of the proposed approach in terms of both accuracy and execution time.

5 CONCLUSIONS AND FUTURE WORK

The problem of software / hardware partitioning is approached in many ways depending on the application and architecture models considered. In this paper, this problem was effectively solved based on AI algorithms.

In this paper, IKSVM was used. In fact, InKSVM learns incrementally, as new data becomes available over time, in order to efficiently process large, dynamic data and reduce computation time. As a result, it relaxes the assumption of complete data availability and provides fully autonomous performance as it efficiently updates the previously trained model.

In order to provide empirical evidence of the superiority of the proposed model, several experiments are conducted. First we have compared it with state-of-the-art incremental learning approaches in terms of accuracy and model complexity. This experiment shows the advantages of incremental learning over batch learning. For this experiment, tests were conducted using several artificial databases whose descriptions are reported in Table 1. According to Figures 1 and 2, the proposal provides better accuracy compared to batch in terms of accuracy values, since incremental models yield a cleaner solution.

In addition, the proposed method is applied for solving industry applications: hardawre/software partitioning for FPGA-based applications. For this, the database is of a collection of experiments that were conducted in Altran Technologies.

Table 2: Comparison of partitioning results against conventional Approaches.

Methods	**Partitioning results**																						Exec time (us)
	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}	T_{21}	T_{22}	
Proposed	1	1	-1	-1	1	1	1	1	1	-1	1	1	1	1	1	-1	1	-1	1	1	1	-1	20021.60
Lee(Lee et al., 2007)	1	-1	-1	1	1	1	1	1	-1	1	1	1	1	1	1	1	1	-1	1	1	1	1	20022.26
Lin(Lin et al., 2006)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	20151.58
GHO(Lee et al., 2009)	1	-1	1	-1	1	1	1	1	1	1	1	1	-1	1	1	1	1	1	1	1	1	1	20021.66
GA(Zou et al., 2004)	-1	-1	1	-1	-1	1	-1	1	-1	1	1	1	-1	1	1	-1	1	1	1	-1	1	-1	20111.26
HOP	-1	1	-1	-1	1	1	1	1	1	-1	1	-1	1	1	1	-1	1	-1	1	1	1	-1	20066.64

As future work, InKSVM will be implemented on FPGA for a fully autonomous real-time HW/SW partitioning.

REFERENCES

Belhedi, W. and Hannachi, M. (2020). Supervised hardware software partitioning algorithms for fpga based applications. *the 12th International Conference on Agents* *and Artificial Intelligence (ICAART 2020)*, 2:860–864.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Kammoun, A., Hamidouche, W., Belghith, F., Nezan, J.-F., and Masmoudi, N. (2018). Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard. *IEEE Transactions on Consumer Electronics*, 64(4):424–432.

Lakshminarayanan, B., Roy, D. M., and Teh, Y. W. (2014). Mondrian forests: Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148.

Lee, T.-Y., Fan, Y.-H., Cheng, Y.-M., and Tsai, C.-C. (2009). Hardware-software partitioning for embedded multiprocessor fpga systems. *International Journal of Innovative Computing, Information and Control*, 5(10):3071–3083.

Lee, T.-Y., Fan, Y.-H., Cheng, Y.-M., Tsai, C.-C., and Hsiao, R.-S. (2007). Enhancement of hardware-software partition for embedded multiprocessor fpga systems. In *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, volume 1, pages 19–22. IEEE.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423.

Lin, T.-Y., Hung, Y.-T., and Chang, R.-G. (2006). Efficient hardware/software partitioning approach for embedded multiprocessor systems. In *2006 International Symposium on VLSI Design, Automation and Test*, pages 1–4. IEEE.

Ouyang, A., Peng, X., Liu, J., and Sallam, A. (2017). Hardware/software partitioning for heterogenous mpsoc considering communication overhead. *International Journal of Parallel Programming*, 45(4):899–922.

Polikar, R., Upda, L., Upda, S. S., and Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508.

Shui-sheng, Z., Wei-wei, W., and Li-hua, Z. (2006). A new technique for generalized learning vector quantization algorithm. *Image and Vision Computing*, 24(7):649–655.

Skliarova, I. and Sklyarov, V. (2019). Hardware/software co-design. In *FPGA-BASED Hardware Accelerators*, pages 213–241. Springer.

Wang, R., Hung, W. N., Yang, G., and Song, X. (2016). Uncertainty model for configurable hardware/software and resource partitioning. *IEEE Transactions on Computers*, 65(10):3217–3223.

Wiem, B., Mowlaee, P., Aicha, B., et al. (2018). Unsupervised single channel speech separation based on optimized subspace separation. *Speech Communication*, 96:93–101.

Wijesundera, D., Prakash, A., Perera, T., Herath, K., and Srikanthan, T. (2018). Wibheda: framework for data dependency-aware multi-constrained hardware-software partitioning in fpga-based socs for iot devices. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 213–213. IEEE.

Yousuf, S. and Gordon-Ross, A. (2016). An automated hardware/software co-design flow for partially reconfigurable fpgas. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 30–35. IEEE.

Zhang Tao, Zhao Xin, A. X. Q. H. and Zhichun, L. (2017). Using blind optimization algorithm for hardware/software partitioning. *IEEE Access*, 5:1353–1362.

Zou, Y., Zhuang, Z., and Chen, H. (2004). Hw-sw partitioning based on genetic algorithm. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, volume 1, pages 628–633. IEEE.