Supervised Hardware/Software Partitioning Algorithms for FPGA-based Applications

Belhedi Wiem and Hannachi Marwa
Department of Research, Altran Technologies, France

Keywords: Hardware/Software Partitioning, Linear Regression, Logistic Regression, Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Deep Neural Network (DNN).

Abstract: Real time systems require the cooperation of the reconfigurable hardware and the software in order to boost the application performance in terms of both energy and time. However, the integration of these systems presents a hardware/software co-design challenges in terms of both time minimization and autonomy; hence, the importance of hardware/software partitioning algorithms.
Here, we present a selection of artificial intelligence based-approaches that we apply in order to solve the hardware/software classification task in real-time systems. For this, the used database consists of a collection of real experiments that were conducted in Altran Technologies. The tested classification algorithms include Linear Regression model optimized with gradient descent, logistic regression, Support vector machine (SVM), Linear Discriminant Analysis (LDA), and deep neural network (DNN).
Results show the applicability of these methods and the high accuracy of the task type decision.

1 INTRODUCTION

The rapid evolution of industrial applications poses a unique challenge requiring high computing performance, flexibility, energy efficiency, autonomy, and real-time processing in order to meet the requirements. For this, numerous solutions were proposed to solve the tradeoff between high computational complexity and time/energy consumption for real-time applications.

In fact, processors are very efficient in data management and control, but they are less efficient in computation and data processing because of their sequential architecture. On the other hand, hardware solutions such as Field-Programmable Gate Array (FPGAs), Application-Specific Integrated Circuit (ASICs) have a parallel architecture that allows them to achieve high computing power. However, their performance in the management and data control is outperformed by the processors. Hence, the cooperation of these hardware and software solutions in heterogeneous systems would boost the application performance in terms of both energy and time (Mohamed et al., 2018)

The heterogeneous nature of this type of systems makes it possible to use resources adapted to each type of request. When heavy processing is required, such as video processing, dedicated hardware blocks may be used while the processor is used for lighter tasks. Unlike dedicated hardware blocks, the processor has the advantage that it can be programmed easily, thus making the system scalable. We therefore see a rivalry between performance and scalability. Indeed, they allow effective treatment of certain requests while retaining the programmable aspect of the processor. However, they remain dedicated to one type of treatment. Heterogeneous systems have been successfully employed for different tasks including real-time locating systems (Alawieh et al., 2015), video processing (Hoozemans et al., 2019), and for matrix multiplication, watermarking, filtering, nearest neighborhood, and AES decryption (Rethinagiri et al., 2015).

Despite the effectiveness of these systems, they still require the optimization of several parameters, such as hardware/software tasks partitioning in order to minimize time and energy consumption.

In this paper, we propose to apply several machine learning algorithms in order to optimize Hardware/Software partitioning. The selected algorithms are the linear regression, logistic regression, support vector machine (SVM), linear discriminant analysis (LDA), and the deep neural network (DNN).

The remainder of this paper is organized as fol-

Wiem, B. and Marwa, H.
Supervised Hardware/Software Partitioning Algorithms for FPGA-based Applications.
DOI: 10.5220/0009149708600864
In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020) - Volume 2*, pages 860-864
ISBN: 978-989-758-395-7; ISSN: 2184-433X

lows. Section 2 introduces the used database, and overviews and discusses the classification algorithms. Section 3 presents our preliminary results. Section 4 outlines our conclusions and describes our current and future work.

2 HARDWARE/SOFTWARE CLASSIFICATION

2.1 Database

The database is of a collection of experiments on FPGA tasks that were conducted in Altran Technologies. As illustrated in Figure 1, it consists of several tasks with their respective Execution time (ET), Energy, Allocations together with their respective types (Hardware or Software).

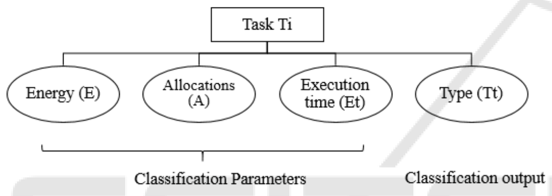

Figure 1: Database Architecture.

In the Task type (Tt), the value ”1” denotes that the task is mapped as hardware, and ”0” denotes that the task is software.

2.2 Linear Regression Model Optimized with Gradient Descent

In this paper, we admit that the task type (Tt) can be modeled as a linear function of the execution time (ET), the energy (E), and the allocation (A) as:

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + b \tag{1}$$

where x_1 is the execution time (ET), x_2 is the energy (E), x_3 is the allocations (A), y is the task type (Tt), α_1, α_2 , α_3 are the mixing coefficients, and b is the intercept. While training the regression model, it calculates the cost function which measures the Root Mean Squared error between the predicted task type $\bar{y}$ and true one. The model targets, therefore, to minimize the cost function defined as:

$$RMSE = \frac{1}{N}\sum_{i=1}^{N}(\bar{y}_i - y_i)^2 \tag{2}$$

In order to minimize the cost function, the best coefficient must be found. For this, initial values of have to be set randomly and then the model iteratively update these values in order to minimize the cost function until it reaches the minimum. The best values will be then employed to predict the task type in the most accurate manner it can. To do so, the gradient descent algorithm is employed. This latter consists of a commonly used optimization function that adjusts weights according to the cost error (Ruder, 2016) as:

$$\begin{cases} \frac{d}{d\alpha_1} = \frac{2}{M}\sum_{i=1}^{M} -x_{1i}(y_i(\alpha_1 x_{1i} + \alpha_2 x_{2i} + \alpha_3 x_{3i} + b)) \\ \frac{d}{d\alpha_2} = \frac{2}{M}\sum_{i=1}^{M} -x_{2i}(y_i - (\alpha_1 x_{1i} + \alpha_2 x_{2i} + \alpha_3 x_{3i} + b)) \\ \frac{d}{d\alpha_3} = \frac{2}{M}\sum_{i=1}^{M} -x_{3i}(y_i - (\alpha_1 x_{1i} + \alpha_2 x_{2i} + \alpha_3 x_{3i} + b)) \\ \frac{d}{db} = \frac{2}{M}\sum_{i=1}^{M} -(y_i - (\alpha_1 x_{1i} + \alpha_2 x_{2i} + \alpha_3 x_{3i} + b)) \end{cases} \tag{3}$$

2.3 Logistic Regression

Logistic regression is the statistical technique used to predict the relationship between inputs, that can be either continuous or categorical, and predicts the probability that an observation falls into one of two categories (Maxwell, 2015). The logistic regression is an extension of linear regression for classification problems. In fact, it models the probabilities for classification problems with two possible outcomes.
The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression, since the outcome in logistic regression is a probability between 0 and 1. Hence, the weighted sum is transformed by a logistic function to a probability.

Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$log(\frac{P_i}{1-P_i}) = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \tag{4}$$

where

$$P_i = P_r(y = 1 | x = x_i) \tag{5}$$

2.4 Linear Discriminant Analysis (LDA)

The Linear Discriminant Analysis (LDA) aims to find linear combinations of features that characterize or separate two or more classes (Antuvan and Masia, 2019). In fact, it projects a dataset onto a lower-dimensional sub-space with good class separability in order to reduce computational costs and to avoid overfitting.

Hence the first step of the LDA algorithm is to calculate the separability between different classes. This latter is the distance between the mean of different classes defined as:

$$S_b = \sum_{i=1}^{g} N_i(\bar{x}_i - \bar{x}_i)(\bar{x}_i - \bar{x}_i)^T \tag{6}$$

where x is a sample (i.e., row) and g is the total number of samples within a given class.

After that, the distance between the mean and sample of each class, also called the within class variance, is calculated as:

$$S_w = \sum_{i=1}^{g}(N_i - 1)S_i = \sum_{i=1}^{g}\sum_{j=1}^{N_i}(\bar{x}_{i,j} - \bar{x}_i)(\bar{x}_{i,j} - \bar{x}_i)^T \quad (7)$$

Then the lower dimensional space P_{LDA} is constructed so that the between class variance is maximized and the within class variance is minimized. Let P be the lower dimensional space projection, the P_{LDA} is determined as:

$$P_{LDA} = \underset{P}{argmax}\frac{P^T S_b P}{P^T S_w P} \quad (8)$$

2.5 Support Vector Machine (SVM)

For a number N of features, support vector machine (SVM) algorithm aims to find a hyperplane in an N-dimensional space that distinctly classifies the data (Gola et al., 2019). In fact, given a labeled training data, SVM outputs an optimal hyperplane that categorizes new examples (Maxwell, 2015). In fact, it can be summed up as an optimization problem depending only on the multipliers α_i:

$$\begin{cases} \underset{\alpha}{max}L_D(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}, y_i y_{i'}, \langle x_i, x_{i'}\rangle \\ s.t \;\; \alpha_i \geq 0 \;\; \forall i \\ and \;\; \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases} \quad (9)$$

x_i and y_i are respectively the input and output observations, $\langle x_i, x_{i'}\rangle$ is the inner product of the observations i and i', α_i are the ponderation coefficients that define the support points.

2.6 Deep Neural Network (DNN)

As indicated in the DNN architecture illustrated in Figure 2 the DNN model consists of four layers, two of which are hidden layers. In the input layer, three variables are set as independent inputs, including the execution time (TE), the energy (E), and the allocation (A), while the task type label is set as the output (Hw/Sw decision). Hence, the values of the three-neurons of the input layer are passed through two hidden layers that have multiple neurons with activation function in all neurons. The final output layer, which indicates of the task type, has one neuron.
In the proposed architecture, the Relu function is used as the activation function in all layers except in the last layer (output layer) where the Sigmoid function is used in order to map the output to the [0, 1] (Nie et al., 2019).

Firstly, the input data should be vectorized and fed into the network. A series of matrix operations is operated on the input data layer by layer through the two hidden layers. Each input sample is multiplied by weights and added to bias that are updated after every epoch. Finally, the activation functions are applied to the result (Agostinelli et al., 2014).

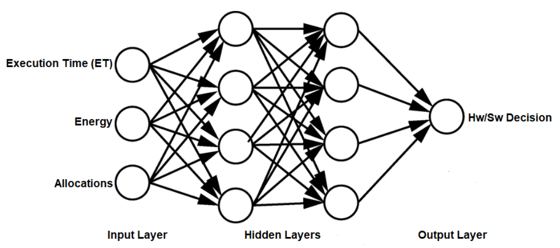

Figure 2: DNN Architecture.

3 RESULTS AND DISCUSSION

In this work, the point is to describe what qualities in a partitioning task contributes to decide whether it is hardware or software. In other words: If the task is, for instance, of a certain execution time, energy and allocations, there probability that a certain task is a hardware. In order to quantify the probability of a task regarding software or hardware, several classification approaches were used. Performance results are reported in terms of training and test accuracy as well as minimum mean-square error (MMSE) defined as (Wiem et al., 2018):

$$MMSE = \frac{1}{N}\sum_{i=1}^{N}(\bar{y}(k) - y(k)) \quad (10)$$

The comparative results are shown in Tables 1 and 2.

The predicted classes were compared with the true ones for the training set of 70% randomly chosen data from the dataset. The remaining data constitute the test set. The training is conducted through 100 epochs, where an epoch corresponds to a learning on all the data. In this work, the number of epochs was chosen to satisfy the trade-off between the estimation accuracy and the training time. To test and evaluate the network, the test set was used (population of 30% of the whole database).

In the classification process, we observed that almost all the learning models grouped tasks under certain characteristics. For example, tasks that possess high execution time and high energy in one group

Table 1: Training/Test accuracies.

Classification Method	Training accuracy	Test accuracy
LR	0.53	0.52
BR	0.87	0.60
LDA	0.93	0.40
SVM	0.87	0.80
DNN	0.86	0.80

Table 2: Classification results in terms of MMSE.

Classification Method	MMSE
LR	0.3404
BR	0.6324
LDA	0.7745
SVM	0.3638
DNN	0.4472

(hardware tasks). The observation on the classification results favors SVM learning algorithm for classification problems since the correctness percentage is high compared to the competitive algorithms. Though, the differences are not much.

The results reported in Tables 1 and 2, therefore, prove the success of these approaches recognize task type through the implicit relationships between input and output variables. From these results, both DNN and SVM are the most successful approaches to predict the task type with the highest accuracy and without complex procedure.

4 CONCLUSION AND FUTURE WORK

In this paper, we investigated the feasibility of supervised learning algorithms for hardware/software partitioning. Experiments were conducted using a database that were collected from real tasks within Altran technologies.

In fact, In general, this paper talks about classification of functions in terms of their implementation through hardware or software to raise the efficiency of the system in terms of time and energy consumption for FPGA-based Applications. Basically, it introduces hardware/software partitioning algorithms in order to solve the classification task in real-time systems. Such a system consists of 6 parts. First , the database where we contain several tasks with 3 parameters, Energy, Allocations, Execution time and all together with their types. Second, Linear Regression model as the task type is represented linearly with the three database parameters each multiplied by coefficient. This model aims to minimize the cost by finding the best proper coefficient value for each parameter. Third, Logistic Regression, where it differs from the linear regression model as the calculation of the weight based on the probability that falls into one of two categories. Mainly, it is the statistical technique that used to predict the relationship between inputs. Fourth, Linear Discriminant Analysis (LDA), where it aims to characterize or separate different classes that share linear features in order to optimize the computations. Fifth, Support Vector Machine (SVM), as it used to classify the data by finding a hyperplane in an N-dimensional space where N is the number of features. Finally, the last part of this paper is Deep Neural Network (DNN). DNN consists of 4 layers where 2 are hidden. The inputs of the trained DNN are the three database parameters and the output is the item type. Each layer have number of neurons with activation function. After applying these algorithms on real experimentations on FPGA and CPU, we found that they are efficient especially DNN and SVN algorithms where they were more accurate in classifying task type.

Hence, we prove the applicability of these methods and the high accuracy of the task type decision.

The tested models were trained offline based on the collected data from real experimentations on FPGA and CPU. The evaluation results show the effectiveness of AI in solving the partitioning problem. For real-time applications, it is important for the tested methods to have a good generalization ability so that they can still work effectively when the conditions of online deployment could not be satisfied in the training stage (Mohammad and Hannachi, 2019) , (Mohamed, 2019). Hence, motivated by the promising results, future work may address the unsupervised learning for a hardware/software partitioning in order to achieve a fully autonomous real-time system.

REFERENCES

Agostinelli, F., Hoffman, M., Sadowski, P., and Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.

Alawieh, M., Kasparek, M., Franke, N., and Hupfer, J. (2015). A high performance fpga-gpu-cpu platform for a real-time locating system. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 1576–1580. IEEE.

Antuvan, C. W. and Masia, L. (2019). An lda-based approach for real-time simultaneous classification of movements using surface electromyography. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):552–561.

Gola, J., Webel, J., Britz, D., Guitar, A., Staudt, T., Winter, M., and Mücklich, F. (2019). Objective microstructure classification by support vector machine (svm) using a combination of morphological parameters and textural features for low carbon steels. *Computational Materials Science*, 160:186–196.

Hoozemans, J., van Straten, J., Viitanen, T., Tervo, A., Kadlec, J., and Al-Ars, Z. (2019). Almarvi execution platform: Heterogeneous video processing soc platform on fpga. *Journal of signal processing systems*, 91(1):61–73.

Maxwell, R. (2015). *The Routledge companion to labor and media*. Routledge.

Mohamed, K. (2019). Model order reduction method for large-scale rc interconnect and implementation of adaptive digital pi controller. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(10):2447–2458.

Mohamed, K., Mehdi, A., and Abdelkader, M. (2018). Lyapunov-global-lanczos algorithm for model order reduction & adaptive pi controller of large scale electrical systems. *Scientia Iranica*, 25(3):1616–1628.

Mohammad, N. and Hannachi, M. (2019). Optimized placement approach on reconfigurable fpga. In *International Journal of Modeling and Optimization 9*.

Nie, X., Cao, J., and Fei, S. (2019). Multistability and instability of competitive neural networks with non-monotonic piecewise linear activation functions. *Nonlinear Analysis: Real World Applications*, 45:799–821.

Rethinagiri, S. K., Palomar, O., Moreno, J. A., Unsal, O., and Cristal, A. (2015). Trigeneous platforms for energy efficient computing of hpc applications. In *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*, pages 264–274. IEEE.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

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