Improving Simulation Speed and Accuracy for Many-Core Embedded Platforms with Ensemble Models

E. Paone, N. Vahabi, V. Zaccaria, C. Silvano
Politecnico di Milano, Milano, Italy
paone@elet.polimi.it, nazanin.vahabi@mail.polimi.it,
{zaccaria, silvano}@elet.polimi.it

D. Melpignano, G. Haugou, T. Lepley
STMicroelectronics, Grenoble, France
{diego.melpignano, germain.haugou, thierry.lepley}@st.com

***Abstract*—In this paper, we introduce a novel modeling technique to reduce the time associated with cycle-accurate simulation of parallel applications deployed on many-core embedded platforms. We introduce an ensemble model based on artificial neural networks that exploits (in the training phase) multiple levels of simulation abstraction, from cycle-accurate to cycle-approximate, to predict the cycle-accurate results for unknown application configurations.**

We show that high-level modeling can be used to significantly reduce the number of low-level model evaluations provided that a suitable artificial neural network is used to aggregate the results. We propose a methodology for the design and optimization of such an ensemble model and we assess the proposed approach for an industrial simulation framework based on STMicroelectronics STHORM (P2012) many-core computing fabric.

I. Introduction

Nowadays, finding the best trade-off in terms of the selected figures of merit (such as energy, delay, and area) can be achieved by tuning a range of application parameters through sophisticated design space exploration heuristics. However, the problem is exacerbated by the long simulation time associated with each configuration of the applications running on a virtual platform. Several closed-form models have been proposed to overcome this inefficiency [1]–[4] but we believe that the research area associated with model aggregation has still untapped potential.

In this paper, we introduce a modeling methodology that leverages different simulation models to reduce the overall simulation time. The methodology is inspired by *ensemble learning* [5]. Ensemble methods use multiple models to obtain better predictive performance than it could be obtained from any of the constituent models. Indeed, ensemble models can give better accuracy with respect to a specific model, provided that a suitable aggregation technique is used [6].

While a *pure* ensemble is a technique for combining many *weak learners* (as defined in [5]) in an attempt to produce a *strong* learner, here we are validating a new technique that attempts to produce a *fast predictor* by combining many *slow predictors*.

Our assumption is to have a Low-Level, highly accurate model $\mathcal{M}_{LL}$ that allows, given an application/architecture configuration x, to derive an estimate y of a specific system metric:

$$y = \mathcal{M}_{LL}(x) \tag{1}$$

Design space exploration techniques based on Response Surface Models (RSMs) [7] are an important area of research and development that needs a high level of efficiency in evaluating a huge number of system configurations. The goal of these techniques is to find a set of optimal configurations x that minimize one or more system metrics. Our goal is to improve state-of-the-art modeling methodology to support the next generation of design space exploration tools.

Neural networks have been already used effectively as a surrogate of $\mathcal{M}_{LL}$ [7]. The final model is a closed form expression that can be used to predict with reasonable accuracy the target system metric provided by cycle-accurate simulators. However, before being useful, the model must be trained and the training time can be long depending on the target accuracy.

Our goal is to combine $\mathcal{M}_{LL}$ with a high-level (thus faster), but less accurate, model $\mathcal{M}_{HL}$:

$$\hat{y} = \mathcal{M}_{HL}(x) \tag{2}$$

We expect that, if $\mathcal{M}_{HL}$ is somewhat correlated with $\mathcal{M}_{LL}$ (or with the samples that have been used for training it), then it can be used to improve the prediction accuracy attainable with $\mathcal{M}_{LL}$ (given the same amount of simulation time needed for the training phase). Dually, it can be used to decrease the simulation time associated to a certain degree of accuracy.

The main contribution of this paper is to combine both $\mathcal{M}_{HL}$ and $\mathcal{M}_{LL}$ by applying techniques borrowed from *machine classification*, *statistical analysis of variance* and *multi-objective analysis*. Our goal is to exceed the speed/accuracy trade-off attainable with state-of-the-art methods such as those presented in [7].

As all statistical modeling techniques, our modeling methodology makes sense only when the aggregated models present certain properties. In particular, we assume that there exists a non negligible correlation between all the input models and the target system metric. This assumption has been experimentally verified for $\mathcal{M}_{HL}$ and $\mathcal{M}_{LL}$ simulation platforms used in this work by means of a correlation test. The reported correlation is indeed very strong and it has been exploited in the ensemble model to speed up the overall exploration phase while maintaining the accuracy.

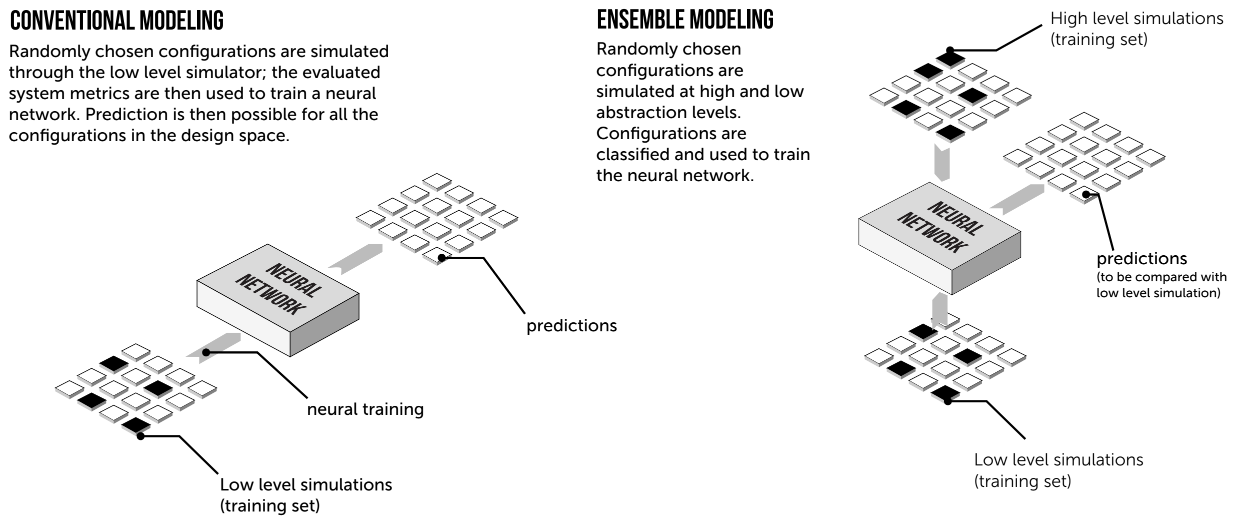
CONVENTIONAL MODELING
Randomly chosen configurations are simulated through the low level simulator; the evaluated system metrics are then used to train a neural network. Prediction is then possible for all the configurations in the design space.
NEURAL NETWORK
predictions
neural training
Low level simulations (training set)
ENSEMBLE MODELING
Randomly chosen configurations are simulated at high and low abstraction levels. Configurations are classified and used to train the neural network.
High level simulations (training set)
NEURAL NETWORK
predictions (to be compared with low level simulation)
Low level simulations (training set)

Fig. 1. Conventional modeling (left) compared to the proposed ensemble modeling (right).

The paper is organized as follows. Section II introduces the proposed ensemble modeling methodology, while Section III describes the experimental setup and the experimental results. Section IV summarizes some relevant previous work and finally Section V reports some concluding remarks.

II. A METHODOLOGY FOR ENSEMBLE MODELING OF APPLICATIONS AND ARCHITECTURES

The original motivation behind this work is the following: is it possible to improve closed-form models, such as the one considered in [7], by using an *ensemble* of abstraction levels? Ensemble modeling is gaining more and more momentum in various fields related to data mining [5]; along this direction, it seemed reasonable to us considering model "stacking" to improve the accuracy of the final model, as defined in [8].

While conventional stacking is meant to improve only the accuracy of the final model by combining the original trained models, our technique tries to combine models that can provide a good trade-off in terms of simulation time (used for the training) and accuracy. *Thus we will extend the conventional notion of ensemble learning to multiple dimensions*, such as efficiency and accuracy level.

It is widely understood that different abstraction models (cycle-accurate, cycle-approximate, functional) can provide different overlapped views of the same simulated platform. There evidently exists some correlation among the various models that can be leveraged to improve the speed of the overall prediction phase. The problem is, however, how to best combine each model to reduce the simulation time (which, for closed-form models, is typically associated with training time).

Proposed structure for ensemble modeling. Conventional modeling tries to overcome lengthy simulations by using closed form models (such as neural networks [7], see Figure 1). In this scenario, low level simulations are used to train the model by tuning its structure to minimize a certain measure of error. The error can be computed by considering either configurations not belonging to the training set or belonging to a bigger set (potentially as large as the overall design space).

Closed form models are ideally very *light* given their straightforward mathematical representation. The main drawback, however, is due to the time needed to simulate the training samples. In our approach, we assume that $\mathcal{M}_{LL}$ has been already used to derive a suitable neural surrogate meta-model. As can be seen in Figure 1, the final goal of the model is to make educated *predictions* about the remaining set of configurations, by resorting to a selected random subset of training simulations.

In this paper, we will show that a conventional meta-model can be extended to an *ensemble* model that, taking into account the information coming from a higher level of abstraction $\mathcal{M}_{HL}$, provides either better accuracy or better performance (see again Figure 1). Although the meta-model currently used in conventional modeling could be seen as another "higher-than-high" level of abstraction, in this paper we consider it only as an intermediate step towards ensemble modeling.

How to combine models to improve low level simulation accuracy and/or speed. The methodology is based on the following phases:

- Identify the **structure of the ensemble model**. In the case of the neural network, this consists of identifying the number of hidden layers and the number of neurons, by analyzing the associated design space. Since this problem can be of exponential complexity, suitable heuristics should be applied.
- Identify the correct **percentage of training samples** to be used for both $\mathcal{M}_{HL}$ and $\mathcal{M}_{LL}$ and the overall training set size. This should be done by considering the relative length of simulation of each configuration, which might depend on architecture, software resource usage and/or application-level parameters.
- **Characterize statistically the distribution of the prediction errors** and assess the statistical properties of the model.

As outlined above, we are interested in improving state-of-the-art conventional modeling, thus, in this paper, we will extend traditional approaches based on feed-forward neural networks [7].

In our methodology for ensemble modeling, we are considering the following parameters:

- The **number of hidden layers** and the **number of neurons per layer** in the NN setup, which change the internal geometry of the NN.
- The composition of the training sets in terms of $\mathcal{M}_{LL}$ and $\mathcal{M}_{HL}$ configurations. This is considered in proportion to the total number of configurations of the application design space.

We formally define a *design point* (either produced by $\mathcal{M}_{LL}$ or $\mathcal{M}_{HL}$) by a tuple:

$$c = \langle \boldsymbol{\alpha}, \boldsymbol{\pi} \rangle \tag{3}$$

where:

- $\boldsymbol{\alpha}$ is the array of values for application parameters (also referred to as input configuration);
- $\boldsymbol{\pi}$ is the array of metric values associated to the input configuration $\boldsymbol{\alpha}$.

The possible values for $\boldsymbol{\alpha}$ depend on the design space for the specific application to be explored.

In order to distinguish between $\mathcal{M}_{LL}$ or $\mathcal{M}_{HL}$ points, we apply a coloring technique. Beside the application input parameters, we add a flag parameter (either 0 or 1) to the input configuration of each design point.

$$c = \langle \boldsymbol{\alpha}', \boldsymbol{\pi} \rangle \tag{4}$$

$$\boldsymbol{\alpha}' = [\alpha_0, \ldots, \alpha_{n-1}, \phi] \tag{5}$$

where ϕ is the flag to classify the two types of training configurations:

$$c = \langle [\alpha_0, \ldots, \alpha_{n-1}, 1], \boldsymbol{\pi}_{LL} \rangle \tag{6}$$

$$c = \langle [\alpha_0, \ldots, \alpha_{n-1}, 0], \boldsymbol{\pi}_{HL} \rangle \tag{7}$$

We expect the NN model to be able to learn the commonalities and differences from the information gathered from the two simulation types; in practice, we expect it to learn the correlation – if any – between type 0 and type 1 metrics.

III. Application of the Proposed Ensemble Model and Experimental Results

In this section we present a methodology to assess the proposed ensemble model. An industrial multi-cluster embedded platform was chosen as reference platform because the complexity of multi-cluster architectures can be modeled at different abstraction levels. At the same time, the customization space for applications targeted to this type of device is too large to think about profiling all possible configurations on a cycle-accurate simulator.

We will show that an ensemble model based on artificial neural networks (NNs) provides a significant reduction of the simulation time, while still guaranteeing results close to the low level simulation model.

Target architecture and platform simulator. The target platform for the applications used in this work is STMicroelectronics STHORM, a low-power many-core computing fabric also known as Platform 2012 [9]. STHORM consists of a Globally Asynchronous Locally Synchronous (GALS) fabric of clusters (4 in our configuration), connected through an asynchronous global Network-on-Chip (GANOC). The STHORM cluster is composed of a multi-core computing engine, called ENCore, and a cluster controller. The ENCore cluster hosts 16 processing elements (PEs); the base processing element of the ENCore engine is a STxP70-V4 processor, a dual-issue customizable 32-bit RISC core of STMicroelectronics with a 32-bit floating-point unit.

STHORM SDK (version 2012.2) includes a simulation platform that can be configured in terms of simulation speed and of accuracy with regard to the hardware architecture. We selected the *posix-posix* simulation model for $\mathcal{M}_{HL}$ and the *posix-xp70* one for $\mathcal{M}_{LL}$:

- In *posix-posix*, the host processor and each STHORM PE are modeled as POSIX threads and the STHORM memory hierarchy is also modeled. Application and OpenCL runtime code run natively on the workstation for both host and STHORM side.
- In *posix-xp70* the host code (application and runtime) is executed natively on the workstation in a POSIX thread. The STHORM device is modeled as an architecture accurate platform based on xp70 Instruction Set Simulator (ISS), on which the OpenCL runtime and the application kernel execution are simulated.

Benchmark applications. Our approach being application-specific, we selected three applications for image processing implemented with OpenCL 1.1 APIs [10] and targeted to STHORM: **FAST** corner detection, Scale-Invariant Feature Transform (**SIFT**) and MultiView (**MV**) Stereo-Matching. These applications were profiled on the STHORM simulation platform to generate the databases of configurations corresponding to $\mathcal{M}_{HL}$ and $\mathcal{M}_{LL}$.

FAST, SIFT and MV are all characterized by a high degree of parallelism, however they differ in terms of complexity and customizability they provide. The customization is enabled by a set of application parameters, that can be of two types:

- *application specific* parameters, which affect both the Quality-of-Service (QoS) provided by the application and the computational load;
- *platform resource* parameters, which impact on the OpenCL runtime (kernel scheduling and memory access patterns), without affecting the quality of application results.

The possibility to change the *platform resource* parameters allows for improved application portability and optimization for the specific target device.

The main characteristics of the three applications are summarized in Table I. FAST and SIFT have the same design complexity in terms of number of parameters. However, while in SIFT the two parameters are both application specific, FAST also exposes a parameter that changes the parallelization degree on the fabric clusters. MV has a larger design space (six input parameters) and 4 out of 6 parameters affect the OpenCL runtime. For all test applications, we consider the metric *cl-cycles*, which represents the number of CPU cycles for execution of the OpenCL kernels. This metric depends on both application specific and platform parameters.

TABLE I. DESIGN SPACE EXPLORATION FOR TARGET APPLICATIONS.

	FAST	SIFT	MV
Size of design space	64	20	486
Time (h) for complete LL simulation	6.75	12.58	603.15
Time (min) for complete HL simulation	0.55	0.73	44.48
Platform resource parameters	1	0	4
Application-specific parameters	1	2	2

Design space of the ensemble model and preliminary correlation analysis. Table II shows the selected configuration parameters of the neural network model. The LL% and HL% values represent the percentages of design space that are explored using $\mathcal{M}_{LL}$ and $\mathcal{M}_{HL}$, respectively, to estimate the clock cycles (*cl-cycles*). The set of mixed configurations (in terms of LL% accurate and HL% approximate samples) is used for training and validation of the NN model. Then, by querying the NN model, it is possible to predict the clock cycles (*cl-cycles*) for those configurations that were not simulated on $\mathcal{M}_{LL}$ during the first phase. The prediction error for a specific NN configuration is calculated as the relative Root Mean Square (RMS) error with respect to the solutions obtained by using only $\mathcal{M}_{LL}$.

Fig. 2 shows a correlation analysis between the parameters presented in Table II and the simulation time and RMS error associated with the proposed ensemble model. The area as well as the color intensity of the circles in the plot represent the absolute value of correlation, while the color (blue or red) indicates whether the correlation is, respectively, positive or negative (indicated also by the '+' or '–' sign in each cell). The negative correlation between "Error" (RMS), on the one hand, and both "HL%" and "LL%", on the other hand, confirms that it is possible to improve prediction accuracy by increasing the size of the training set. This is verified both for $\mathcal{M}_{HL}$ and $\mathcal{M}_{LL}$ training samples, which accounts for the adoption of an ensemble model. Indeed, while reducing the RMS prediction error, the utilization of $\mathcal{M}_{HL}$ simulations has a very low impact on the simulation time, which is confirmed by the low correlation between "HL%" and "Sim. Time". Although derived from the MV application, this plot reveals some general results observed also for the other applications (not shown here because of space limitations).

Accuracy analysis of the ensemble model. In Fig. 3, the different configurations for the NN model are plotted with reference to the simulation time (relative to the total time needed to simulate the entire application design space on the low level simulator) and the associated RMS prediction error. The area of the circles is proportional to the number of neurons in the NN: we can observe that medium-small networks best fit our model and allow for low prediction error.

This result is confirmed by the histogram in Fig. 4, which shows the number of NN configurations for different levels of RMS prediction error. The configurations are assigned to one of the three histograms according to the number of layers

TABLE II. PARAMETERS FOR NN MODEL TUNING.

Test	# Layers	# Neurons	HL%	LL%
FAST	1-5	1-5	0,50,100	2,4,6,8,10
SIFT	1-5	1-5	0,50,100	5,10,15,20,25,30
MV	1-5	1-5	0,50,100	2,4,6,8,10

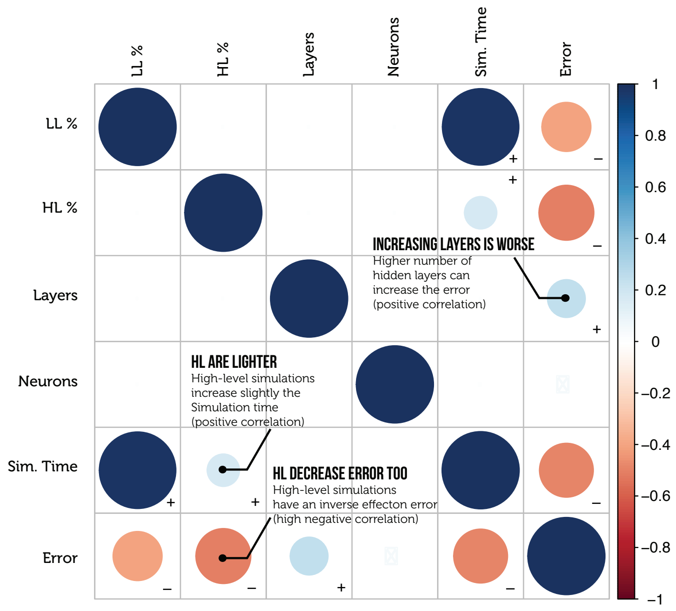

Fig. 2. Correlation analysis applied to the neural network model for the MultiView Stereo-Matching application.

in the NN. In this plot, the highest density distribution for low levels of RMS error corresponds to NN configurations with only one hidden layer. In general, the best configuration we have consistently seen in our experiments is a NN with 1 hidden layer of 5 neurons.

In Fig. 3, the color intensity expresses the percentage of $\mathcal{M}_{HL}$ simulations used for training the NN. Since the cycle-approximate simulator is much faster than the cycle-accurate, the time contribution from adding $\mathcal{M}_{HL}$ simulations is negligible: thus, the overall simulation time is almost proportional to the number of simulations on $\mathcal{M}_{LL}$.

Analysis of variance of the results. Fig. 5 shows the distribution of NN configurations for MV, in relation to the percentage of high-level samples used for training the ensemble model. The configurations with the same number of high-level training samples (either 0%, 50% or 100%) are grouped together in the same vertical window. The vertical gray bar indicates the average RMS error in each group: we observe that by passing from a purely cycle-accurate model (first window from left) to an ensemble model with 50% high-level training samples, the average RMS error on the prediction of metric *cl-cycles* decreases from 0.83 down to 0.51. At the same time, as seen in Fig. 3, the delay due to additional high-level simulations is negligible compared to the time required for cycle-accurate simulations.

The box-plot diagrams in Fig. 6 show distribution and median of the RMS prediction error on metric *cl-cycles* for FAST and SIFT. Each diagram contains three distributions, corresponding to three different percentages of $\mathcal{M}_{HL}$ samples in the training set. For SIFT the improvement of prediction accuracy is sound, while for FAST we only notice a reduction of RMS error when increasing the number of approximate training samples from 0 to 50%.

We applied the non-parametric Kruskal-Wallis analysis of variance (ANOVA) to assess whether the introduction of $\mathcal{M}_{HL}$ training samples is statistically significant for improvement of

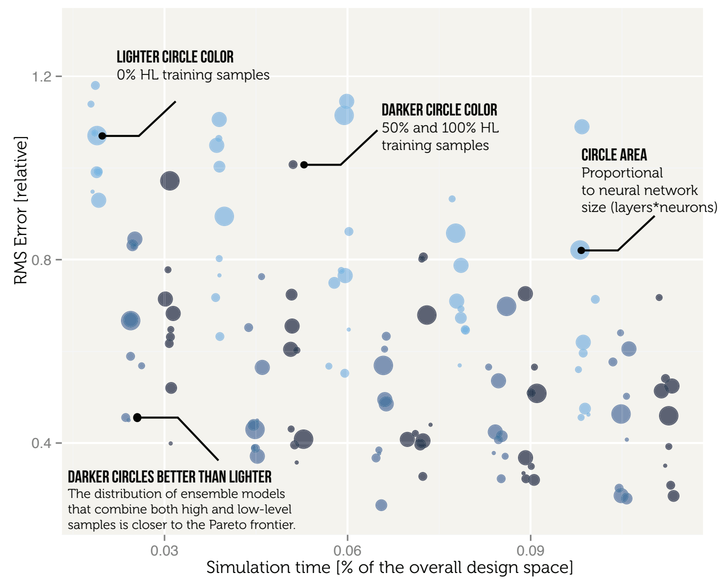

Fig. 3. RMS prediction error on *cl-cycles* vs. simulation time for the MultiView Stereo-Matching application.

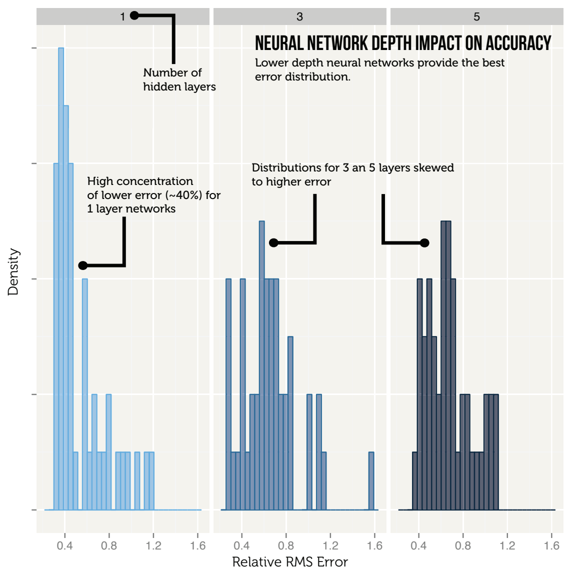

Fig. 4. Distribution of neural network model configurations according to the number of hidden layers for the MultiView Stereo-Matching application: density distribution vs. RMS prediction error.

TABLE III. KRUSKAL-WALLIS ANALYSIS FOR SIGNIFICANCE OF LL% TRAINING SAMPLES ON RMS ERROR.

Test	chi-squared	degree of freedom	p-value
FAST	0.0028	2	0.998
MV	47.686	2	4.4e-11
SIFT	10.308	2	0.006

the model prediction accuracy. From Table III we deduce that the distributions observed for MV and SIFT can be considered significant because of the low p-value[1], while this hypothesis is not verified for FAST (p-value ≈ 1).

Final considerations. The reported experimental results show that the proposed ensemble model enables to achieve up to 30% improvement given the same simulation time required for a pure cycle-accurate model. Alternatively, the same level of accuracy could be achieved by replacing part of the cycle-accurate training set with application configurations profiled on a fast simulation platform, with a one order of magnitude speed-up of the design exploration.

IV. Previous work

Our work is mainly targeted to improving the simulation speed of configurable models that are useful for design space exploration (DSE). Improving DSE can be addressed in two complementary ways, by either minimizing the number of configurations simulated or the time required to evaluate each configuration.

On one side, exhaustive exploration has been carried out by clustering dependent parameters [11], sensitivity analysis of the design space [12], design space pruning [13]–[16], analytical meta-models [17], statistical simulation [18], optimization algorithms such as Pareto simulated annealing [19], evolutionary algorithms [20].

[1]The p-value indicates the probability that the observed difference in means is due to chance (instead of a systematic effect).

On the other side, the reduction of the time for the evaluation of each configuration has been addressed with linear and spline regression [2], [3] and neural networks [4], [7],

This paper leverages these previous works and, in particular, extends neural-network based methodologies with techniques borrowed from ensemble modeling.

V. Conclusions

Traditionally, high-level models are used to speed up the simulation process at the expense of profiling accuracy. Ensemble neural network models that have been trained with both low-level and high-level simulations (mixed training set) show better accuracy with respect to conventional models (training set consisting of only low-level samples). This result can be achieved because, given a time window, ensemble neural network models enable to better exploit that time in terms of high-level (fast) and low-level (slow) simulations. The accuracy improvement reported by using ensemble neural network models has been up to 30% for the target set of embedded applications running on the STHORM platform.

VI. Acknowledgments

This work has been partially funded by the European Commission under the projects FP7-ICT-248716-2PARMA and ARTEMIS-100230-SMECY.

References

[1] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil, “A predictive performance model for superscalar processors,” in *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 161–170.

[2] ——, “Construction and use of linear regression models for processor performance analysis,” in *Symposium on High Performance Computer Architecture*. Austin, Texas, USA: IEEE Computer Society, 2006, pp. 99–108.

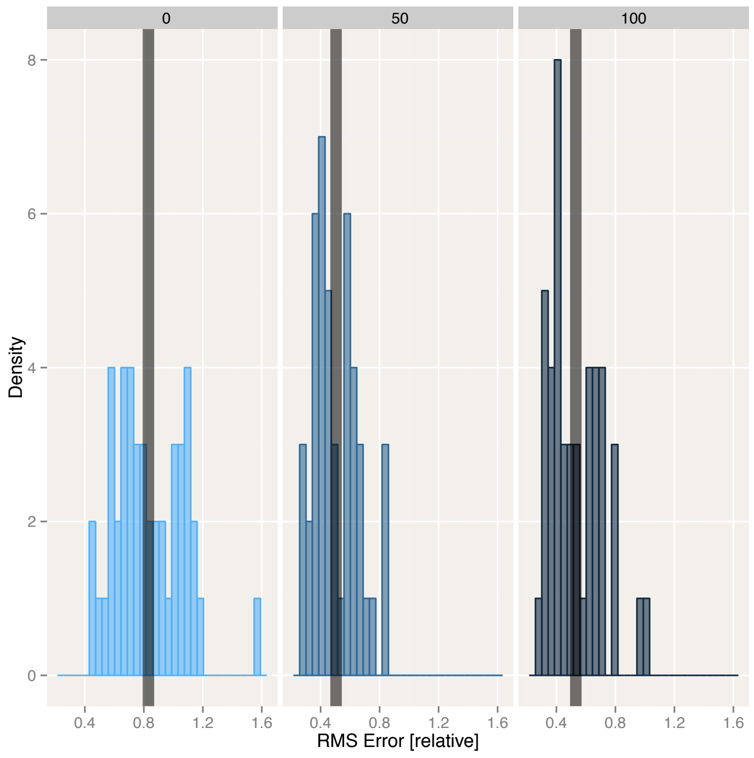

Fig. 5. Distribution of neural network model configurations according to the percentage of high-level training samples for the MultiView Stereo-Matching application: density distribution vs. RMS prediction error.

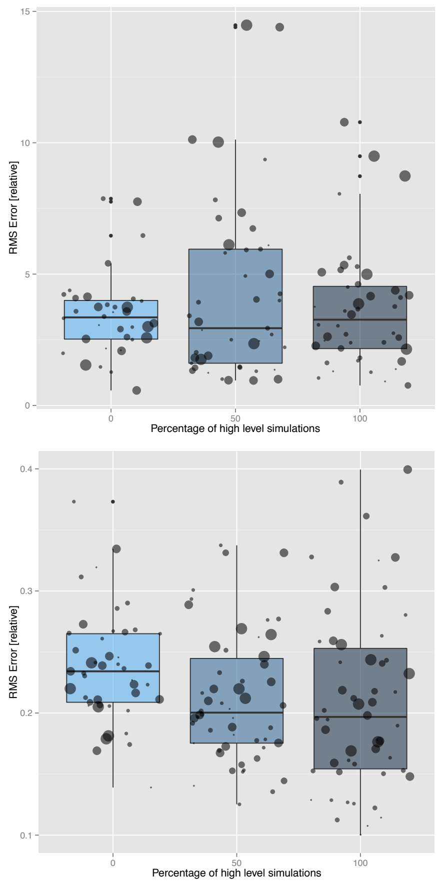

Fig. 6. RMS prediction error vs. HL% training samples for FAST (above) and SIFT (below).

[3] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, vol. 40, no. 5, pp. 185–194, 2006.

[4] E. Ipek, S. A. McKee, R. Caruana, B. R. de Supinski, and M. Schulz, "Efficiently exploring architectural design spaces via predictive modeling," *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, vol. 40, no. 5, pp. 195–206, 2006.

[5] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[6] J. Bennett, S. Lanning, and N. Netflix, "The netflix prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[7] G. Palermo, C. Silvano, and V. Zaccaria, "ReSPIR: A Response Surface-based Pareto Iterative Refinement for application-specific design space exploration," *IEEE Transactions on Computer Aided Design of Integrated Circuits*, vol. 28, no. 12, pp. 1816–1829, Dec. 2009.

[8] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[9] L. Benini, E. Flamand, D. Fuin, and D. Melpignano, "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 983–987.

[10] Khronos Group, "The opencl specification, version 1.1," June 2011, http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf.

[11] T. Givargis, F. Vahid, and J. Henkel, "System-level exploration for pareto-optimal configurations in parameterized systems-on-a-chip," in *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, 2001, pp. 25 –30.

[12] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria, "A sensitivity-based design space exploration methodology for embedded systems," *Design Automation for Embedded Systems*, vol. 7, pp. 7–33, 2002, 10.1023/A:1019791213967. [Online]. Available: http://dx.doi.org/10.1023/A:1019791213967

[13] E. Neema, J. Sztipanovits, and G. Karsai, "Title: Design-space construction and exploration in platform-based design," Tech. Institute for Software Integrated Systems Vanderbilt University Nashville Tennessee, Tech. Rep., 2002.

[14] S. G. Abraham, B. R. Rau, and R. Schreiber, "Fast design space exploration through validity and quality filtering of subsystem designs," Packard, Compiler and Architecture Research, HP Laboratories Palo Alto, Tech. Rep., 2000.

[15] R. Szymanek, F. Catthoor, and K. Kuchcinski, "Time-energy design space exploration for multi-layer memory architectures," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 1, feb. 2004, pp. 318 – 323 Vol.1.

[16] G. Hekstra, G. La Hei, P. Bingley, and F. Sijstermans, "Trimedia cpu64 design space exploration," in *Computer Design, 1999. (ICCD '99) International Conference on*, 1999, pp. 599 –606.

[17] A. Ghosh and T. Givargis, "Cache optimization for embedded processor cores: an analytical approach," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, nov. 2003, pp. 342 – 347.

[18] S. Eyerman, L. Eeckhout, and K. De Bosschere, "Efficient design space exploration of high performance embedded out-of-order processors," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1, march 2006, pp. 1 –6.

[19] G. Palermo, C. Silvano, and V. Zaccaria, "Multi-objective design space exploration of embedded systems," *J. Embedded Comput.*, vol. 1, no. 3, pp. 305–316, 2005.

[20] G. Ascia, V. Catania, A. G. D. Nuovo, M. Palesi, and D. Patti, "Efficient design space exploration for application specific systems-on-a-chip," *Journal of Systems Architecture*, vol. 53, no. 10, pp. 733–750, 2007.