# Efficient Training and Hardware Co-Design of Machine Learning Models

Mohammad Amir Mansoori and Mario R. Casu

Politecnico di Torino, Turin, Italy
{mohammadamir.mansoori,mario.casu}@polito.it

**Abstract.** To implement a Machine Learning (ML) model in hardware (Hw), usually a first Design Space Exploration (DSE) optimizes the model hyper-parameters in search of the best ML performance, while a second DSE finds the configuration with the best Hw performance. Multiple iterations of these steps might be needed as the optimal ML model may not necessarily be implementable. To reduce the design-time and provide the designer with a single exploration environment, we propose a general framework based on Bayesian Optimization (BO) and High-Level Synthesis (HLS), which performs at once both DSEs generating efficient Pareto curves in the space of ML and Hw performance.

**Keywords:** Machine Learning (ML), Hardware Acceleration, High Level Synthesis (HLS), FPGAs, Bayesian Optimization

## 1 Introduction

Machine Learning (ML) techniques can be very effective in various edge applications (medical diagnosis, computer vision, robotics) but very often require hardware (Hw) accelerators for power- and cost-efficient inference. The usual design method consists of a Design Space Exploration (DSE) to fine-tune the hyper-parameters of an ML model, followed by another DSE that aims to optimize the Hw design for a given target. In this work we consider small-size FPGAs as Hw target and a high-level design approach using High-Level Synthesis (HLS).

This approach of separate DSE is shown in Fig. 1(a) and is useful for those applications where the ML accuracy has to be maximized and powerful Hw accelerators can be selected to meet the desired performance. However, being bounded to one specific small-size Hw architecture makes the design more challenging, calling for the joint optimization strategy shown in Fig. 1(b). The joint method avoids lengthy iterations that occur when the selected ML model is incompatible with the Hw constraints (e.g., it exceeds the available resources or cannot meet the timing requirements) and can obtain a better trade-off between ML performance and Hw performance.

For this reason, we propose a general framework for the joint optimization of training hyper-parameters and HLS-based hardware configurations based on Multi-Objective Bayesian Optimization (BO) with constraints. This methodology allows for the optimization of multiple hardware parameters in HLS including clock frequency, data precision, and different levels of parallelism (unroll

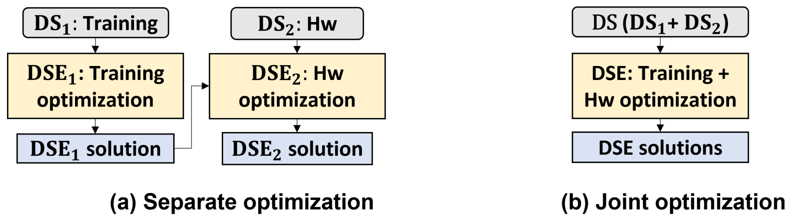

**Fig. 1.** Optimization of training hyper-parameters and hardware configurations: (a) traditional separate DSE, (b) more efficient joint DSE. $DS_1$ and $DS_2$ stand for Design Space of training and hardware design, respectively.

factor or array partitioning), together with the ML training hyper-parameters. BO optimizes black-box objective functions that are expensive to evaluate. It uses Gaussian Processes (GP) as the prior model to quantify the uncertainty of these functions. An acquisition function is then constructed based on the expected improvement of the prior model to decide where to sample the objective functions next [9]. In this work we use Predictive Entropy Search for Multi-Objective Optimization with Constraints (PESMOC) [7] for the acquisition function. In each BO iteration a new sample is suggested by the acquisition function and the objectives are evaluated for the new sample in order to update the Gaussian models and the acquisition functions in the next iteration. The main contributions of this work are as follows:

– An efficient method for the joint optimization of training hyper-parameters and HLS-based hardware configurations in the context of Machine Learning.
– Exploitation of the capabilities of BO for Multi-Objective optimization subject to positive constraints in the design of ML models in hardware.
– Inclusion of the clock frequency as an optimization target, which is currently not optimized in HLS tools for Xilinx devices (Vivado HLS).
– Evaluation of the proposed methodology for the efficient design of a Multi-Layer Perceptron (MLP) type of Neural Network in a Zynq FPGA.

## 2 Related work

To find the optimum hyper-parameters of an ML model during training, more powerful approaches than simple Grid Search and Random Search are in use nowadays, such as Auto-Sklearn, HyperOpt, Auto-Keras, and Keras-Tuner. In [1], a review of commonly-used methods in automated ML has been presented. The algorithms used in these methods can be divided into Reinforcement Learning (RL), BO, and Evolutionary Algorithms (EAs). Although RL and BO share similar characteristics and have been proven effective in several works, one of the difficulties of RL is that the *policies* and *reward function* need customization for each DSE problem, while the BO's Gaussian Process is a natural fit for these optimization problems. EAs are computationally expensive, which led several authors to propose other methods to speed up the computations.

Regarding the DSE aimed at optimizing the hardware configurations, several works have focused on High Level Synthesis (HLS) as the hardware design tool and proposed different methodologies for the optimum selection of HLS *pragmas* ([2], [3]). In [4] Multi-Objective Bayesian Optimization is used to tune the HLS configurations. Although the generated Pareto Fronts are close to the actual optimal points, it could not consider *constraints* in addition to multiple *objectives* in the optimization process which increases the exploration time.

Recently, there has been a growing interest in the joint optimization of hardware and training parameters, specially in the context of Deep Neural Networks (DNNs) for which Hardware-aware Neural Architecture Search (HW-NAS) has been introduced. In the context of HW-NAS, numerous works have been presented for the co-optimization of network architecture and hardware-related parameters that are thoroughly described in a recent review paper [5]. One of the challenges in the recent HW-NAS methodologies is designing efficient algorithms to solve the expensive Multi-Objective Optimization (MOO) problem. According to [5], most of the previous works use single-objective optimization targeting network accuracy *constrained* to hardware-aware characteristics. Other approaches combine multiple objectives into one single function. Evolutionary algorithms are mostly used for MOO in HW-NAS which are computationally expensive. In [6] a two-step BO is used to reduce the complexity of MOO. Our work has features in common with previous HW-NAS works, but is not restricted to DNNs, uses BO as a more efficient exploration method for multi-objective functions, and uses HLS in the optimization loop to obtain an accurate estimation of the Hw performance for a given configuration.

Finding both optimum parameters for ML model training and hardware can be seen as a Multi-Objective Optimization problem with *Constraints* (MOOC). Recently, Garrido-Merchán et al. extended their Spearmint software for applying BO to MOOC and proved its efficiency in designing a DNN architecture in hardware [7]. The hardware area is selected as a constraint and is estimated by Aladdin, a pre-RTL performance estimator for ASIC accelerator design [8]. Unlike our work, FPGA is not considered as a Hw target and HLS is not used.

## 3 Proposed methodology

Fig. 2 illustrates our methodology. The BO takes as input the range of parameters for both training and Hw exploration and aims to optimize at once three objectives: (1) Training error on the ML dataset using floating-point precision in Python; (2) Prediction (i.e., inference) error after Hw implementation, which takes into account fixed-point quantization and is obtained with the HLS C-simulation tool from the Vivado suite; (3) Hw inference performance expressed as clock period times the number of clock cycles as obtained from the HLS tool[1]. The constraints are the FPGA resources (BRAMs, DSPs, FFs, LUTs).

[1] We leave power optimization for future work.

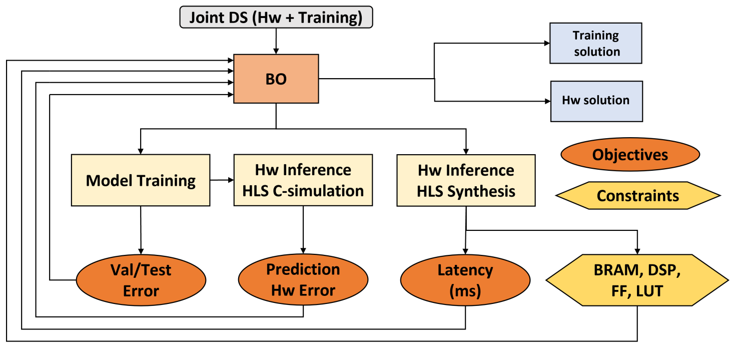

**Fig. 2.** Proposed methodology for the joint optimization of training parameters and HLS-based hardware configurations (BO = Bayesian Optimization).

The BO considers a Gaussian Process for each of the objectives and constraints. The training hyper-parameters and the Hw configuration parameters (HLS *pragmas*, data precision, clock frequency) are updated in each BO iteration based on the maximization of PESMOC acquisition function, which suggests a new sample in the design space to be evaluated by the objective functions. The non-dominated points of the design space are obtained at the end of the iterative process.

## 4 Results

For the evaluation, we used the MNIST dataset to train a Multi-Layer Perceptron (MLP) to be implemented in a Zynq7000 FPGA. We used hls4ml [10] to convert the MLP model to a synthesizable C++ code. The model hyper-parameters and the ranges of the HLS and Hw knobs are in Tab.1.

**Table 1.** Ranges of parameters for the joint training/Hw optimization method.

| **Inputs** | Clk (ns) | Hidden Layers | Neurons | Precision #total | Precision #Integer | Reuse factor | Array Partition | Learning rate | Regul-arization rate |
|---|---|---|---|---|---|---|---|---|---|
| **Ranges** | 4 - 7 | 1 - 3 | 32 - 256 step = 32 | 12 - 16 | 4 - 6 | 1 - 4 | $2^x$ $x = [1 - 8]$ | $1 \times 10^{(-x)}$ $x = [2 - 7]$ | $1 \times 10^{(-x)}$ $x = [2 - 7]$ |

Fig. 3 compares the evolution of the training error using a joint (Fig. 3(a)) and a separate (Fig. 3(b)) optimization approach. Fig. 3(a) shows that both floating-point error during training and fixed-point error in hardware converge as the BO iterations progress. Since the separate method returns only the best training result, Fig. 3(b) shows only the fixed-point inference error and shows an immediately low error in all BO iterations (less than 5%). This is because the separate hardware design starts with a neural network already optimized in terms of training error, which only needs to be tailored to the hardware target.

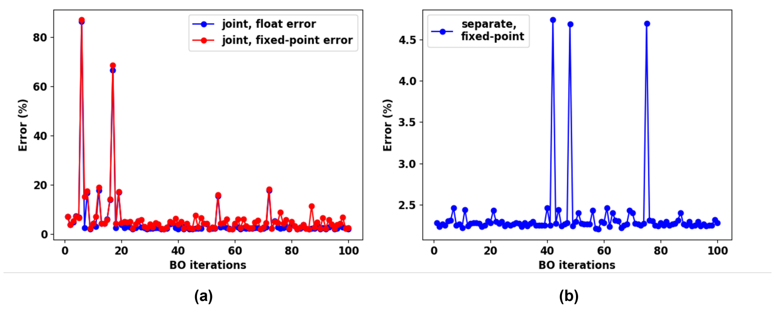

**Fig. 3.** Percentage of training error (float error) and hardware error (fixed-point error) in each BO iteration, (a) proposed joint optimization, (b) separate optimization.

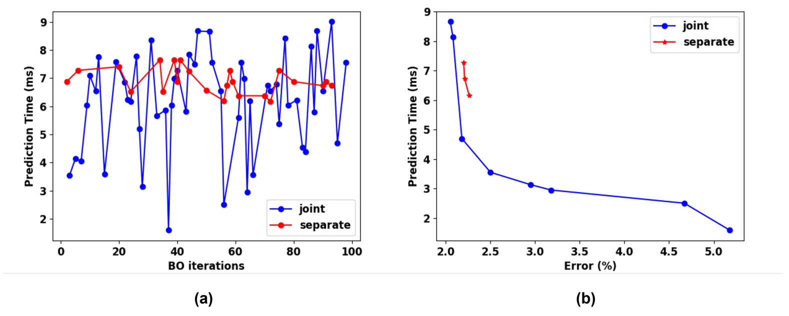

**Fig. 4.** Comparison of (a) prediction time and (b) Pareto fronts.

This last optimization of the separate method, however, is constrained by the initial training. As a result, the BO cannot reach the same latency performance of the joint optimization. This is visible in Fig. 4(a), with the relatively high prediction time for the separate optimization method (red points).

The efficiency of the proposed methodology is apparent in Fig. 4(b), which compares the Pareto curves obtained by separate (red) and joint (blue) optimization methods. In the red curve, the training optimization is done by Keras-Tuner. Note that in this case Keras-Tuner suggests an initial MLP with three layers and a number of neurons that could not fit in the FPGA due to excessive BRAM usage, leading to a failure in the subsequent hardware BO. This required a second iteration to limit the neurons range from $[32 - 256]$ to $[32 - 128]$ in the training DSE, which returned a feasible three-layer MLP with 128 neurons in each layer, low error but relatively high prediction time. The subsequent Hw DSE returned only three (red) Pareto points. On the contrary, the joint method returns many more valid Pareto points (blue) because of its ample maneuverability in the combined trainig and hardware design spaces. Most importantly, the blue points dominate the red ones, as clearly shown in Fig. 4(b).

## 5 Conclusions and future work

We proposed a new strategy, based on Multi-Objective Bayesian Optimization subject to positive constraints, for the efficient design of Machine Learning models in FPGA-based hardware accelerators, which can simultaneously optimize the training hyper-parameters and HLS-based hardware configurations. It optimizes prediction error and latency, subject to FPGA resource constraints. The results show the efficiency of the Pareto sets obtained by the proposed method with near 2× reduction of prediction time without an increase in the prediction error compared to the traditional separate optimization design. In the future, we will evaluate other ML models with this methodology, such as Support Vector Machine (SVM) and Random Forest (RF). Other techniques like Random Search and evolutionary algorithms can be compared with our approach in terms of computational time and efficiency of the Pareto fronts.

**Acknowledgments.** This work was supported by the EMERALD project funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764479.

## References

1. Chen, Y., Song, Q., and Hu, X.: Techniques for automated machine learning. ACM SIGKDD Explorations Newsletter 22.2 (2021) 35-50.
2. Sohrabizadeh, A., et al.: AutoDSE: Enabling Software Programmers Design Efficient FPGA Accelerators. arXiv preprint arXiv:2009.14381 (2020).
3. Zhao, J., et al.: Performance modeling and directives optimization for high-level synthesis on FPGA. IEEE TCAD, 39.7 (2019) 1428-1441.
4. Mehrabi, A., et al.: Bayesian optimization for efficient accelerator synthesis. ACM TACO 18.1 (2020) 1-25.
5. Benmeziane, H., et al.: A Comprehensive Survey on Hardware-Aware Neural Architecture Search. arXiv preprint arXiv:2101.09336 (2021).
6. Parsa, M., et al.: Pabo: Pseudo agent-based multi-objective bayesian hyperparameter optimization for efficient neural accelerator design. 2019 IEEE/ACM ICCAD (2019).
7. Garrido-Merchán, E.C., Hernández-Lobato, D.: Predictive entropy search for multi-objective bayesian optimization with constraints. Neurocomputing 361 (2019) 50-68.
8. Shao, Y.S, et al.: Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures. 2014 ACM/IEEE 41st ISCA (2014).
9. Frazier, P.I: A tutorial on Bayesian optimization. arXiv preprint arXiv:1807.02811 (2018).
10. Fahim, F., et al.: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices. ArXiv preprint arxiv:2103.05579 (2021).