A Fast Design Space Exploration Framework for the Deep Learning Accelerators: Work-in-Progress

A Fast Design Space Exploration Framework for the Deep Learning Accelerators: Work-in-Progress

Alessio Colucci[1], Alberto Marchisio[1], Beatrice Bussolino[2], Vojtech Mrazek[3],
Maurizio Martina[2], Guido Masera[2], Muhammad Shafique[1,4]
[1]*Technische Universität Wien, Vienna, Austria*
[2]*Politecnico di Torino, Turin, Italy*
[3]*Faculty of Information Technology, IT4Innovations Centre of Excellence, Brno University of Technology, Czech Republic*
[4]*Division of Engineering, New York University Abu Dhabi, UAE*
Email: {alessio.colucci,alberto.marchisio,muhammad.shafique}@tuwien.ac.at,muhammad.shafique@nyu.edu
{beatrice.bussolino,maurizio.martina,guido.masera}@polito.it,mrazek@fit.vutbr.cz

***Abstract*— The Capsule Networks (CapsNets) is an advanced form of Convolutional Neural Network (CNN), capable of learning spatial relations and being invariant to transformations. CapsNets requires complex matrix operations which current accelerators are not optimized for, concerning both *training* and *inference* passes. Current state-of-the-art simulators and design space exploration (DSE) tools for DNN hardware neglect the modeling of training operations, while requiring long exploration times that slow down the complete design flow. These impediments restrict the real-world applications of CapsNets (e.g., autonomous driving and robotics) as well as the further development of DNNs in life-long learning scenarios that require training on low-power embedded devices.**

Towards this, we present *XploreDL*, a novel framework to perform fast yet high-fidelity DSE for both inference and training accelerators, supporting both CNNs and CapsNets operations. *XploreDL* enables a resource-efficient DSE for accelerators, focusing on power, area, and latency, highlighting Pareto-optimal solutions which can be a green-lit to expedite the design flow. *XploreDL* can reach the same fidelity as ARM's SCALE-sim, while providing 600x speedup and having a 50x lower memory-footprint. Preliminary results with a deep CapsNet model on MNIST for training accelerators show promising Pareto-optimal architectures with up to 0.4 TOPS/squared-mm and 800 fJ/op efficiency. With inference accelerators for AlexNet the Pareto-optimal solutions reach up to 1.8 TOPS/squared-mm and 200 fJ/op efficiency.

***Index Terms*—Design Space Exploration, Hardware Accelerator, Capsule Networks, Convolutional Neural Networks, Training.**

I. Introduction

In recent years, Deep Neural Networks (DNNs) have seen a tremendous proliferation in Machine Learning (ML) applications, leading to the massive employment of DNN-based solutions in various fields, like autonomous driving, surveillance, natural language processing, and smart healthcare [1]. Recent studies [2][3][4][5] have highlighted computational power and energy consumption of standard processing devices, like CPUs and GPUs, as the primary bottlenecks in the further advancements of DNN models. Therefore, industry and research trends have moved towards dedicated accelerator-based IPs and ASICs, both for inference [6][7] and training [8]. But ASIC-based designs have long time-to-market, leading to incremental design options. However, DNN architectures are rapidly evolving, as with Capsule Networks [9], so new hardware extensions are required [10], which may not be feasible to deploy considering a short time-to-market. Hence, we propose *XploreDL*, a Framework for Fast Simulation and Design Space Exploration of CNNs and Capsule Networks Training and Inference Accelerators.

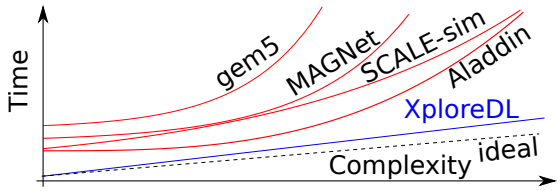

Fig. 1: Computational time to run a complete Design Space Exploration (DSE) simulation for a given neural network vs. Complexity of the design space, considering both the hardware parameters and the neural network configuration. These results are qualitative, gathered from different works on different design spaces [11][12][13][14].

Motivational Study and Research Objective: Several state-of-the-art simulators are limited to compute estimations of the total latency and power of the hardware design, excluding other physical design parameters such as area, sub-circuit power and maximum achievable frequency. However these parameters, even though in their approximated forms, are extremely important at the beginning of the design phase, because they can provide a useful direction to follow for future design choices with detailed analyses. While some simulators tend to have very high accuracy, *at the beginning of the design phase it is much more important to have results in a quick way but with consistency and high fidelity*, to steer the design in the proper direction without waiting for super-long simulations. High fidelity denotes that the comparative analysis between two design choices should hold true/consistent for both the exhaustive and fast search based DSEs, such that the selection of the appropriate design choice is correct and efficient. To compare state-of-the-art simulators with our XploreDL framework, we performed a qualitative study analyzing the computational time of the simulators w.r.t. the complexity of the model/design combination [11][12][13][14]. The results in Fig. 1 show that, while the computational time of the other simulators is exponential, the slope of our XploreDL framework is linear (slightly higher than the ideal case), hence enabling efficient scalability for high-complexity design explorations.

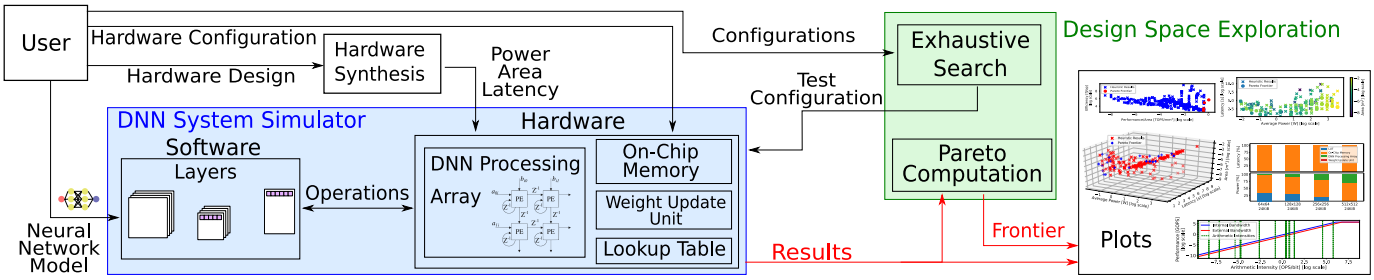

Fig. 2: Workflow and block diagram of the XploreDL tool, including the HW and SW models of the simulator and the DSE.

In a nutshell, with our XploreDL framework we provide the following key contribution: *we perform a systematic Design Space Exploration (DSE), to analyze different architectural solutions, and generate Pareto-frontiers for specialized training and inference accelerators for CapsNets and CNNs, given efficiency and performance-per-area as optimization objectives.*

II. An Overview of XploreDL

XploreDL (see Fig. 2) is a framework to simulate the whole flow of data during the training & inference processes of a generic neural network, focused not only on the traditional CNNs but also on novel models such as CapsNets. It is composed of different sub-components, along with the workflow:

1) **DNN System Simulator**, covering both the software- and the hardware-level simulations of a DNN accelerator. It contains a model for each of the system sub-components:
   a) *DNN Model*, supporting several types of layers (convolutional, matrix, pooling, capsule), which are scalable with respect to their dimension and parameters such as stride, padding, etc. In this stage, the data transfers and the required hardware resources for each layer are computed.
   b) *On-Chip Memory*, which takes into account the read and write accesses from the off-chip DRAM and to the DNN processing array, considering contiguous and non-contiguous word requests;
   c) *DNN Processing Array*, composed of processing elements to provide all the low-level hardware parameters, including the loading of the weights and its complete execution;
   d) *Weight Update Unit*, a small array of adders to update the weights after backpropagation, working in a similar way as the DNN processing array;
   e) *Look-Up-Table (LUT)*, to model the activation functions, receiving the dimension of the elements to be processed.
2) **Exhaustive-Search Algorithm for the DSE**, to test the different input configurations and simulate the output;
3) **Pareto-Optimal Extractor**, to compute the Pareto-optimal solutions from all the simulated ones;
4) **Plotter**, to visualize all the simulated data.

III. Preliminary Results

In the following section we analyze preliminary results on training and inference accelerators for both CapsNets and CNNs.

A. Training Accelerator for DeepCaps

We focus our preliminary analysis on a DSE for the training accelerator on the DeepCaps network [15] on the MNIST dataset [16], a deeper version of the original CapsNet network, whose best solutions achieved 800 fJ/op for energy efficiency and 0.4 TOPS/mm^2 for performance-per-area. In Fig. 3, we show some design points in a 2D plot with energy-efficiency in fJ/op and performance per square millimeter as indicators. This plot provides deep insights into the design space, since it compares relative objectives. We can see a quasi-linear behaviour ① towards lower performance and worse efficiency. This can be related to higher area with the same DNN processing array utilization, leading to increased energy consumption as well as reduced performance-per-area.

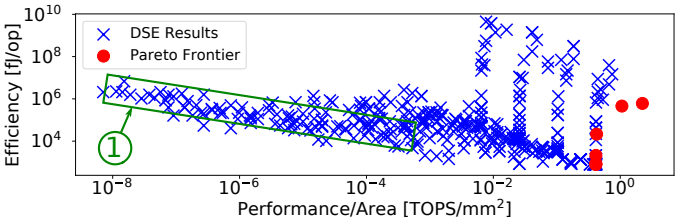

Fig. 3: 2D plot showing a selected subset of explored solutions for the DeepCaps training accelerator with efficiency and performance-per-area as goals.

B. Inference Accelerator for AlexNet

The following preliminary results can be compared with state-of-the-art simulators, as they deal with a very common CNN model (AlexNet [17]) in a widely explored context (inference). Running a similar DSE with SCALE-sim [18] lead to 600x slower execution and 50x higher memory footprint, validating the less resource-intensive XploreDL implementation. Moreover, our best solutions provide up to 200 fJ/op for energy efficiency and 1.8 TOPS/mm^2 for performance per area. SCALE-sim does not provide hardware-level characteristics, so we resorted to MAGNet [14] as comparison. Its solutions reach 52 fJ/op and 2.6 TOPS/mm^2. The divergence is related to the different synthesis libraries (45nm CMOS by XploreDL vs 16nm FinFET by MAGNet), but also to different dataflow optimizations between XploreDL and MAGNet, the latter suited for detailed analysis in the late stages of the design flow.

In Fig. 4, we present the energy-efficiency in fJ/op and performance per square millimeter. We can see a vertical pattern ② of solutions with similar performance-per-area but different efficiencies, as they have more on-chip memory

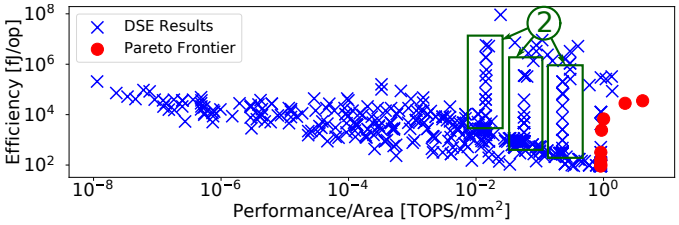

Fig. 4: 2D plot showing a selected subset of explored solutions for the AlexNet inference accelerator using efficiency and performance-per-area as objectives. Performance-per-area is quite high on average at the cost of worse energy efficiency.

with smaller DNN processing array, thus obtaining similar performance but lower efficiency.

IV. Observations and Design Guidelines

The above-discussed preliminary results showed that our XploreDL framework, while giving fast and approximated results, provides useful takeaways for directing the next design choices towards a better efficiency. Hence, we can derive these **key observations**:

- Due to the huge demand of application-driven design solutions for DNNs and a wide design space, a fast and fidelitous multi-objective DSE is key for converging towards high-efficient architectures in the future design steps.
- Most of the Pareto-optimal solutions are low-power and low-area, while there are a few outliers with much lower latency at the expense of power and area. These solutions could become the starting points for further analysis when designing latency-critical systems, which are generally less tight on the power and area constraints.
- Further optimizations of the on-chip memory and the accelerator can potentially have a significant impact, as their power consumption varies greatly across different configurations.

Acknowledgments

This work has been partially supported by the Doctoral College Resilient Embedded Systems which is run jointly by TU Wien's Faculty of Informatics and FH-Technikum Wien.

References

[1] Y. LeCun et al. "Deep learning". In: *Nature Cell Biology* (2015).
[2] V. Sze et al. "Efficient Processing of Deep Neural Networks: A Tutorial and Survey". In: *Proceedings of the IEEE* (2017).
[3] A. Marchisio et al. "Deep Learning for Edge Computing: Current Trends, Cross-Layer Optimizations, and Open Research Challenges". In: *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2019, pp. 553–559.
[4] M. Shafique et al. "Robust Machine Learning Systems: Challenges,Current Trends, Perspectives, and the Road Ahead". In: *IEEE Design Test* 37.2 (2020), pp. 30–57.
[5] M. Capra et al. "An Updated Survey of Efficient Hardware Architectures for Accelerating Deep Convolutional Neural Networks". In: *Future Internet* 12.7 (July 2020), p. 113. (Visited on 07/16/2020).
[6] N. P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit". In: *ISCA* (2017).
[7] M. A. Hanif et al. "MPNA: A Massively-Parallel Neural Array Accelerator with Dataflow Optimization for Convolutional Neural Networks". In: *CoRR* abs/1810.12910 (2018). arXiv: 1810.12910. URL: http://arxiv.org/abs/1810.12910.
[8] E. Qin et al. "SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training". In: *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2020, pp. 58–70.
[9] S. Sabour et al. "Dynamic Routing Between Capsules". In: *NIPS* (2017).
[10] A. Marchisio et al. "CapsAcc: An Efficient Hardware Accelerator for CapsuleNets with Data Reuse". In: *DATE*. 2019, pp. 964–967.
[11] A. Nocua Cifuentes et al. "ElasticSimMATE: A fast and accurate gem5 trace-driven simulator for multicore systems". In: *ReCoSoC*. 2017.
[12] A. Akram. "A Comparison of x86 Computer Architecture Simulators". In: *CASRL* (2016).
[13] Y. S. Shao et al. "Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures". In: *ISCA* (2014).
[14] R. Venkatesan et al. "MAGNet : A Modular Accelerator Generator for Neural Networks". In: *ICCAD*. 2019.
[15] J. Rajasegaran et al. "DeepCaps: Going Deeper with Capsule Networks". In: *CVPR*. 2019.
[16] Y. LeCun et al. *THE MNIST DATABASE of handwritten digits*. 1998.
[17] A. Krizhevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.
[18] A. Samajdar et al. "SCALE-Sim: Systolic CNN Accelerator Simulator". In: *arXiv*. 2018.