Education Abstract: Design Space Exploration for Deep Learning at the Edge

Andy D. Pimentel
University of Amsterdam, The Netherlands
a.d.pimentel@uva.nl

***Abstract*—The AI revolution, fueled by effective Deep Learning approaches, has seen a recent shift towards processing the AI workloads closer to the user, at the Edge. This paper addresses the instrumental role of system-level design space exploration (DSE) methods for achieving efficient inference of deep-learning models on resource-constrained devices at the Edge.**

I. Introduction

Deep Learning methods have become an immensely important driver for AI-based applications in a variety of application fields like computer vision, natural language processing, automotive, and many more. For the inference of trained deep-learning models, we are currently witnessing a shift from Cloud-based execution of the AI workloads to executing them closer to the user, at the Edge. Such so-called Edge AI typically provides lower latencies as well as increased privacy when performing model inference. However, deploying and inferring large Deep Neural Networks (DNNs) on edge devices is challenging because these devices usually have limited power/compute/memory resources.

System-level design space exploration (DSE) [1], [2] has traditionally been an important approach for optimizing the mapping of application workloads onto resource-constrained embedded computer systems. With such DSE, a large variety of different design alternatives can be explored, such as the number and type of processor cores to deploy for the computations, the spatial and temporal binding of application tasks to processor cores, and so on.

In this paper, we will addresses the instrumental role of system-level DSE for achieving efficient inference of DNN models on resource-constrained (embedded) devices at the Edge. Although there is a wealth of techniques for ’fitting’ large DNNs to resource-constrained devices, including model pruning, quantization and distillation [6], we will focus the discussion on two specific directions in the domain of Edge AI: hardware-aware Neural Architecture Search (NAS) and distributed inference of large DNNs on multiple edge devices.

II. Design Space Exploration (DSE)

System-level DSE explores different system implementation solutions while simultaneously considering multiple optimization objectives, such as performance, power/energy consumption, and memory footprint. The search for optimal design solutions typically entails two distinct elements: i) The evaluation of a single design solution in terms of the considered optimization objectives. For this, measurements on real systems or estimates using either analytical or simulation models are used [1], [2]. ii) The search strategy for covering and navigating through the vast design space, spanned by all free variables such as the number and type of processor cores to deploy, the binding of application tasks to cores, etc. Here, metaheuristics, such as simulated annealing, ant colony optimization and genetic algorithms, are popular approaches. This is because these algorithms search the design space for optimal solutions using only a finite number of design point evaluations, and thus being able to handle large design spaces. However, they cannot guarantee to find the global optimum. In the scope of system-level DSE, especially genetic algorithms have proven to be highly effective [3]–[5].

III. Hardware-aware Neural Architecture Search

Neural Architecture Search (NAS) is an approach with which a proper architecture of a DNN (i.e., yielding optimal accuracy) can be automatically determined by means of searching the DNN design space [6]. Traditional NAS strategies are often unaware of the target device the DNN should be executed on, which may result in multiple, expensive (re-)design iterations when a DNN solution does not fit on a resource-constrained device. To address this issue, we have developed a hardware-aware NAS approach, based on a genetic algorithm as underlying search algorithm, that considers multi-objective optimization. This allows our NAS to simultaneously optimize the DNN for accuracy as well as for objectives that are specific for the target hardware device such as memory usage, inference latency, and energy consumption. In contrast to traditional NAS techniques, wherein the training of a candidate DNN to assess its accuracy is considered an isolated task, our work uses a novel technique called piecemeal-training [7], [8]. In piecemeal-training, a neural network architecture and the related weights are jointly and simultaneously learned by combining concepts of the traditional training process and evolutionary architecture search in a single algorithm.

Our NAS results in multiple Pareto-optimal DNN solutions that provide different trade-offs regarding objectives such as accuracy, memory usage, and energy consumption. As the application needs for systems at the Edge can be significantly affected by changes in the application environment, like a change of the battery level in the edge device, these different Pareto-optimal DNNs can also be exploited at run time. In that case, the system can dynamically switch between different

DNN implementations based on the needs and changes of the application environment [9].

IV. Distributed Inference of DNNs

Another approach for inferring large DNNs at the Edge is to leverage all available resources across multiple edge devices to execute the DNNs by properly partitioning them and running each DNN partition on a separate edge device. As described below, system-level DSE plays an important role here as well.

A. AutoDiCE

The AutoDiCE tool provides a design and programming framework that takes a trained DNN model as input and subsequently allows for efficiently exploring and automatically implementing a range of different DNN partitions on multiple edge devices to facilitate distributed DNN inference [10]. AutoDiCE automates the splitting of a DNN model into a set of sub-models as well as the code generation needed for the distributed and collaborative execution of these sub-models on multiple, possibly heterogeneous, edge devices, while supporting the exploitation of parallelism *among* and *within* the edge devices. Regarding the latter, different processing cores in an edge device (like CPU cores, GPU and NPU) can be utilized to execute DNN layers in parallel in a pipelined fashion, or in a layer-switched manner [12], [13] to exploit the affinity of specific DNN layers to certain processing core types.

Since the number of different DNN mapping possibilities on multiple edge devices is vast, the AutoDiCE framework features a DSE methodology to automatically search for proper distributed CNN inference implementations [11]. The DSE utilizes a genetic algorithm with a so-called Split-Point Encoding approach that is tailored to efficiently searching for good DNN partitionings and mappings. Moreover, we accelerate the searching process by using a multi-stage hierarchical DSE approach. At every stage, we perform DSE at two hierarchical levels. In the first level, we use analytical models to approximate the throughput, memory and energy consumption of DNN partitionings/mappings. The solutions found in the first level together with Pareto-optimal solutions from a previous DSE stage are utilized as the parents for the second level DSE. In this second level, we evaluate each design point using real measurements taken from AutoDiCE-generated DNN inference implementations to determine the Pareto solutions for a next DSE stage. The output of the last DSE stage provides the final Pareto-optimal solutions.

B. Robust Partitioning of DNNs

Efforts to distribute the DNN computations and coefficients over multiple edge devices collaboratively, such as discussed above, generally assume that all participating devices are always available and thus do not consider the presence of device failures. However, failures due to, e.g., connectivity issues or depleted batteries of edge devices can seriously inhibit the proper execution of distributed DNN inference, leading to a significant accuracy drop of the DNN model or even a complete system failure. We have therefore introduced a novel partitioning method, called RobustDiCE, for robust distribution and inference of DNN models over multiple edge devices [14]. This method can tolerate intermittent and permanent device failures in a distributed system at the Edge. To this end, it features both system robustness, i.e., DNN inference can continue execution even if one or more edge devices fail to function properly, and model robustness, i.e. preserving the inference accuracy of the DNN model as much as possible when some of the intermediate DNN inference results are lost due to failed devices. Model robustness is achieved by evaluating the relative importance of each neuron in the DNN model and then partitioning these different neurons of each DNN layer into different groups (to be mapped to the various edge devices) as 'evenly' as possible. This method combines partial neuron replication and importance-aware neuron clustering and, as such, provides a tunable trade-off between robustness (i.e., retaining model accuracy after failures) and resource utilization. For a user-provided level of robustness, RobustDiCE deploys DSE to find implementations that optimize for DNN inference accuracy, per-device energy and memory consumption, and overall system throughput.

The recently introduced EASTER methodology extends RobustDiCE to learn robust distribution strategies for Transformer models against device failures [15], also considering the trade-off between robustness and resource utilization.

References

[1] A. D. Pimentel, "Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration", IEEE Design & Test, Vol. 34 (Nr. 1), Jan. 2017.

[2] A. D. Pimentel, "Methodologies for Design Space Exploration", Handbook of Computer Architecture, Springer, 2022.

[3] C. Erbas, S. Cerav-Erbas and A. D. Pimentel, "Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design", IEEE Trans. on Evolutionary Computation, pp. 358-374, Vol. 10 (No. 3), June 2006.

[4] Z. J. Jia, T. Bautista, A. Nunez, M. Thompson, and A. D. Pimentel, "A System-level Infrastructure for Multi-dimensional MP-SoC Design Space Co-exploration", ACM TECS, Vol. 13 (Nr. 1s), Nov. 2013.

[5] W. Quan and A.D. Pimentel, "Towards Exploring Vast MPSoC Mapping Design Spaces using a Bias-Elitist Evolutionary Approach", DSD 2014.

[6] G. Menghani, "Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better", ACM Comput. Surv. 55(12), Dec. 2023.

[7] D. Sapra and A. D. Pimentel, "Designing Convolutional Neural Networks with Constrained Evolutionary Piecemeal Training", Applied Intelligence, Springer, July, 2021.

[8] D. Sapra and A. D. Pimentel, "Constrained Evolutionary Piecemeal Training to Design Efficient Neural Networks", IEA/AIE 2000.

[9] S. Minakova, D. Sapra, T. Stefanov, and A. D. Pimentel, "Scenario Based Run-time Switching for Adaptive CNN-based Applications at the Edge", ACM TECS, Vol. 21 (Nr. 2), March 2022.

[10] X. Guo, A. D. Pimentel, and T. Stefanov, "Automated Exploration and Implementation of Distributed CNN Inference at the Edge", IEEE Internet of Things Journal, Vol. 10(7), 2023.

[11] X. Guo, A. D. Pimentel, and T. Stefanov, "Hierarchical Design Space Exploration for Distributed CNN Inference at the Edge", ITEM 2022.

[12] E. Aghapour, D. Sapra, A. D. Pimentel, and A. Pathania, "ARM-CO-UP: ARM COoperative Utilization of Processors", ACM TODAES, 2024.

[13] E. Aghapour, Y. Shen, D. Sapra, A. D. Pimentel and A. Pathania, "PiQi: Partially Quantized DNN Inference on HMPSoCs", ISLPED 2024.

[14] X. Guo, A.D. Pimentel, and T. Stefanov, "RobustDiCE: Robust and Distributed CNN Inference at the Edge", ASP-DAC 2024.

[15] X. Guo, Q. Jiang, Y. Shen, A. D. Pimentel, T. Stefanov, "EASTER: Learning to Split Transformers at the Edge Robustly", IEEE TCAD, in press, 2024.