Mind Mappings: Enabling Efficient Algorithm-Accelerator Mapping Space Search
Extended Abstract

Kartik Hegde, Po-An Tsai*, Sitao Huang,
Vikas Chandra+, Angshuman Parashar*, Christopher W. Fletcher
University of Illinois at Urbana-Champaign, *NVIDIA, +Facebook

1. Motivation

The slowing of Moore's Law coupled with a fast-growing demand for efficient compute has ushered in the era of specialized hardware. Although specialized, hardware accelerators still provide some degree of configurability and programmability [9, 14, 7, 7, 10, 5] to cope with the diversity, i.e., different parameterizations of an algorithm or even a range of algorithms, in the targeted domain.

Such flexibility creates a new problem: given a workload, we must find an efficient mapping, i.e., parameterized instance, of that workload onto the specialized architecture. We refer to this problem as *map space search*. A mapping can involve various decisions, such as how to allocate valuable on-chip buffer capacity to each data structure, how to schedule computation temporally and spatially across the system, and so on.

Map space search is a critical problem facing the community today for several reasons. First, prior work has shown that execution efficiency is very sensitive to the choice of mapping [4, 12, 7, 11, 9, 10, 14]. Second, the same studies illustrate how the optimal mapping varies significantly depending on problem size and parameters (e.g., CNN layer shapes), resource availability, performance and power requirements, etc. This suggests that map space search will constitute an increasing *recurring cost*, as accelerators are continuously re-targeted for new problems and parameterizations.

Making matters worse, accelerators lack the consistent hardware-software abstraction that ISAs provide in the general-purpose computing world. This has resulted in map space search tools being tied to specific accelerators; or worse, map space search requiring manual expert-driven analysis (usually by the accelerator designer).

2. Limitations of the State of the Art

Prior work has proposed tools and algorithms (i.e., *Mappers*) to automate map space search but all existing approaches have serious limitations due to the complexity of the search space [7, 11, 3, 1, 15]. First, the search space is often high dimensional, causing a combinatorial explosion in map space size and rendering exhaustive techniques ineffective [11]. Second, the search space is both non-convex (featuring many local minima) and non-smooth (not differentiable). This has forced prior work [3, 1, 15] to perform the search using *black-box optimization techniques* [6], limiting search quality and increasing search cost.

To illustrate these issues, consider black-box search approaches such as Simulated Annealing (SA)[8], Genetic Algorithms (GA)[13], etc. These algorithms are called "black box" as the search algorithm can only interact with the *cost function* through its evaluation interface: given an input mapping, compute the *cost*, where "cost" is some efficiency metric such as performance, energy, EDP, etc., and the "cost function" is evaluated using an architectural simulator or the actual hardware itself. During the search, black-box algorithms use the cost function to evaluate the neighbors of the current mapping to take the next best step that maximally improves the cost.

The above process is conceptually implemented as a two-level loop where each outer loop iteration is a step, i.e., change in mapping that improves cost, and the inner loop is a local search that evaluates the cost of different neighboring mappings to determine the next best step to take. Since black-box optimizers do not have access to information such as gradients and Hessians of the cost function, the inner loop is often stochastic [8] or based on heuristics learnt during the search [2]. Therefore, the number of evaluations in the inner loop exponentially increases with the map space dimensionality, as higher dimensionality implies more neighbors to consider per step.

3. Key Insight

To address the challenges above, this paper proposes *Mind Mappings*, a scalable and automated method to quickly and effectively perform map space search. *The key idea is to approximate the cost function with a smooth, differentiable function called a surrogate, and to use the surrogate to perform a more powerful gradient-based search.* The differentiable surrogate enables us to replace the expensive inner loop (Section 2) to choose the next best step with constant work (i.e., independent of map space dimensionality): at each step in the search space, we compute the gradient of the surrogate at the current mapping and choose the next mapping based on gradient descent. As gradients by definition point at the steepest descent, the new mapping maximally reduces the cost of the current mapping in the neighborhood, leading to a local minima. With sufficient random restarts, Mind Mappings leads to high quality solutions.

4. Main Artifacts

Mind mappings proposes a two phase optimization method as shown in Figure 1.

Phase 1: Smooth function approximation. First, we approximate the otherwise non-smooth complex search space

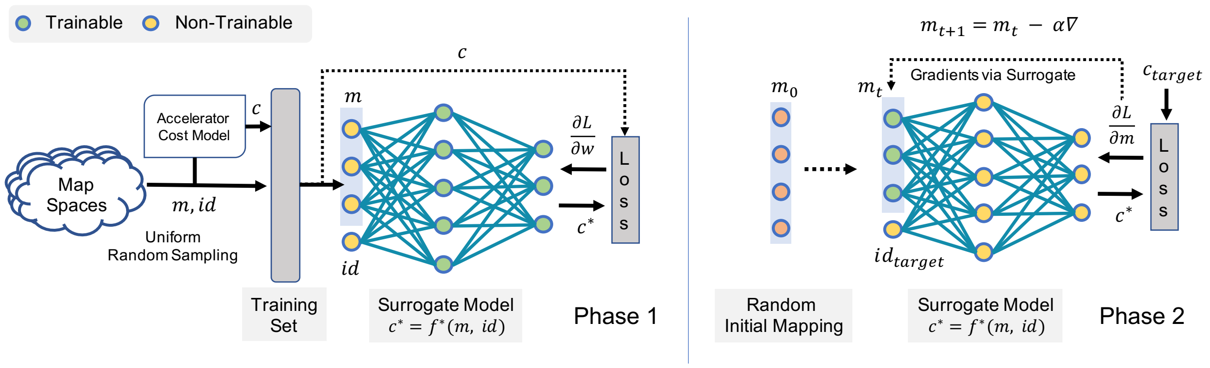

Figure 1: Proposed search procedure. Phase 1: Training the surrogate model $c^* = f^*(m, id)$ based on (mapping, map space id, cost) tuples (m, id, c). DNN weights w are trained with back-prop. Phase 2: Given a map space id id_{target} (problem instance) and a target cost c_{target} (a lower-bound), use the trained surrogate model to iteratively guide a random initial mapping m_0 towards an optimal mapping m_{opt}. In each iteration, m_t is updated using back-propogation with a gradient ∇ of the estimated loss with a learning rate α. The trained model weights w are held constant in this phase.

with a smooth, differentiable surrogate. The surrogate predicts the cost of execution, given a mapping. This surrogate not only enables us to generate gradients, but also saves us from querying the potentially expensive cost function (e.g., run the workload fully) at every step. Importantly, *the surrogate need only be constructed once (offline)* for a given pair of algorithm and accelerator, amortizing the cost of its construction.

Phase 2: Searching over the smooth function. Second, we make a key observation that the smooth surrogate can be used to *directly* generate a guided move in the mapping space, relative to a reference mapping, *that will maximally reduce cost*. As in Figure 1, we start by choosing a random initial mapping. This mapping is then updated at each step via gradients generated by computing the difference between the predicted cost and a target cost, where the target cost may be some theoretical lower-bound cost, e.g., "0."

Not only is this process efficient (as gradients point in the steepest descent direction), it also does not require expert knowledge in the target domain. That is, both predicting the cost given a mapping and finding an optimal mapping given a lower-bound cost are formulated as *learning* problems. i.e., without requiring manual domain-expert analysis.

We refer the reader to Section 4 of the main paper for detailed description of the Mind Mappings approach.

5. Key Results and Contributions

Evaluation. We prototyped Mind Mappings (MM) for tensor accelerators (CNNs and MTTKRP). The proposed search method achieves 1.75×, 2.11×, 1.09× (iso-iteration) and 2.2×, 2.3×, 1.6×(iso-time) better energy-delay-product over techniques from state-of-the-art mappers, simulated annealing, genetic algorithms, and reinforcement learning, respectively, on average.

5.1. Key Take Aways

1. **High Quality Solutions:** Mind Mappings finds mappings whose cost is less than or equal to those found by other popular methods.
2. **Faster Time to Solution:** Mind Mappings uses a much faster surrogate model instead of the expensive cost function, hence enabling higher quality solution in lesser time compared to other approaches.
3. **Optimality:** The mappings found by by Mind Mappings are within 5.3× of the theoretical lower bound (possibly unachievable), pointing to proximity to the achievable global optimum.
4. **Generality:** Mind Mappings generalizes to different algorithms, architectures, and problem shapes (e.g., different CNN Layer shapes), without requiring any expert-intervention, as demonstrated by the evaluations on CNN-Layer and MTTKRP algorithms.

6. Why ASPLOS

Mind Mappings proposes an efficient approach to map space search for programmable accelerators. Hence, Mind Mappings is of potential interest to both accelerator architects and compiler designers; making ASPLOS a suitable venue.

7. Citation for Most Influential Paper Award

Mind Mappings dramatically improved the performance of crucial algorithm-accelerator map-space search problem by re-formulating the problem to enable powerful gradient-based search techniques, thus influencing future works. Proposed solution enabled a target domain-independent approach that generalized to different algorithms and architectures without needing any expert intervention, leading to its wide adoption.

References

[1] Byung Hoon Ahn, Prannoy Pilligundla, and Hadi Esmaeilzadeh. Reinforcement learning and adaptive sampling for optimized dnn compilation. *arXiv preprint arXiv:1905.12799*, 2019.

[2] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[3] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594, 2018.

[4] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ISCA'16.

[5] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[6] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

[7] Kartik Hegde, Rohit Agrawal, Yulun Yao, and Christopher W Fletcher. Morph: Flexible acceleration for 3d cnn-based video understanding. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 933–946. IEEE, 2018.

[8] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[9] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects. *SIGPLAN Not.*, 53(2):461–475, March 2018.

[10] Wenyan Lu, Guihai Yan, Jiajun Li, Shijun Gong, Yinhe Han, and Xiaowei Li. Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks. In *HPCA'17*.

[11] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. Timeloop: A systematic approach to dnn accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 304–315. IEEE, 2019.

[12] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. ISCA'17.

[13] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[14] Fengbin Tu, Shouyi Yin, Peng Ouyang, Shibin Tang, Leibo Liu, and Shaojun Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *VLSI'17*.

[15] Size Zheng, Yun Liang, Shuo Wang, Renze Chen, and Kaiwen Sheng. Flextensor: An automatic schedule exploration and optimization framework for tensor computation on heterogeneous system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 859–873, 2020.