FADiff: Fusion-Aware Differentiable Optimization for DNN Scheduling on Tensor Accelerators

Shuao Jia
Beijing University of Posts and Telecommunications
Beijing, China
pdxqjiashuao@bupt.edu.cn

Zichao Ling
Beijing University of Posts and Telecommunications
Beijing, China
lingzichao@bupt.edu.cn

Chen Bai
Fudan University
Shanghai, China
cbai@fudan.edu.cn

Bei Yu
The Chinese University of Hong Kong
Hong Kong, China
byu@cse.cuhk.edu.hk

Kang Zhao
Beijing University of Posts and Telecommunications
Beijing, China
zhaokang@bupt.edu.cn

Jianwang Zhai*
Beijing University of Posts and Telecommunications
Beijing, China
zhaijw@bupt.edu.cn

Abstract

Efficiently deploying Deep Neural Networks (DNNs)—particularly Large Language Models (LLMs)—on tensor accelerators is essential for achieving high throughput and energy efficiency at the edge. However, the combinatorial explosion of the joint design space spanning intra-layer mapping and inter-layer fusion renders existing approaches inadequate. In this work, we present FADiff, a gradient-based optimization framework that automatically discovers optimal mapping and fusion strategies to accelerate DNN inference. We first formulate a unified, differentiable analytical cost model that accurately estimates the energy and latency of individual layer mappings as well as diverse fusion configurations. We then combine CP-SAT-based pre-enumeration of feasible discrete candidates with augmented-Lagrangian penalties over continuous relaxations, enabling a constrained gradient-based search that efficiently navigates the vast design space to identify optimal joint strategies. Experimental results show that FADiff consistently outperforms prior methods, delivering superior optimization quality with significantly faster convergence in both energy and latency.

CCS Concepts

• **Hardware → Modeling and parameter extraction**.

Keywords

deep neural network, accelerator architecture, large language model, differentiable analytical model

ACM Reference Format:
Shuao Jia, Zichao Ling, Chen Bai, Bei Yu, Kang Zhao, and Jianwang Zhai. 2027. FADiff: Fusion-Aware Differentiable Optimization for DNN Scheduling on Tensor Accelerators . In *32nd Asia and South Pacific Design Automation Conference (ASPDAC '27), January 25–28, 2027, Tokyo, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3840404.3848216

*Corresponding author.
This work is supported by the National Natural Science Foundation of China (No. 62404021), the Beijing Natural Science Foundation (No. L2603021), the Fundamental Research Funds for the BUPT (No. 2025AI4S20), and the Research Initiation Project for Introduced Talents of BUPT (No. 510124058, 510224062).



This work is licensed under a Creative Commons Attribution 4.0 International License.
ASPDAC '27, Tokyo, Japan
© 2027 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2923-2/2027/01
https://doi.org/10.1145/3840404.3848216

1 Introduction

Efficient deployment of DNNs on resource-constrained hardware, such as edge tensor accelerators, is essential for realizing high performance on-device intelligence. A fundamental challenge in this endeavor is bridging the abstraction gap between high-level models and low-level hardware execution—a task that entails navigating a vast combinatorial design space in which two key decisions interact non-trivially: intra-layer mapping (how a single layer's computation and data are tiled and bound to hardware resources) and inter-layer fusion (how consecutive layers are grouped for joint execution). Together, these decisions govern data reuse, memory traffic, and ultimately energy and latency efficiency. As networks deepen and accelerator architectures grow more complex, this coupled mapping-fusion space becomes prohibitively difficult to explore and optimize.

Existing studies on DNN deployment design space exploration can be broadly categorized into heuristic and learning-based approaches. Heuristic frameworks such as TVM [1], DeFiNES [2], SoMa [3], and Klotski [4, 5] rely on combinatorial or search-based optimization to traverse the vast design space. Learning-based efforts, including Active Learning-guided optimization [6] and Deep Gaussian Transfer Learning [7], aim to improve efficiency through surrogate modeling and experience transfer. Despite their differences, both paradigms fundamentally depend on iterative surrogate evaluation and empirical search, resulting in slow convergence, limited interpretability, and a lack of theoretical grounding.

More recently, gradient-based optimization has emerged as an effective paradigm for accelerating DNN deployment design space exploration. The key insight, pioneered by DOSA [8] and Felix [9], lies in formulating differentiable energy and latency models. This transformation converts the discrete, combinatorial design space into a continuous domain, thereby enabling efficient gradient-based optimization. Instead of heuristic or learning-based algorithms, this paradigm uses analytical gradients in a theoretically sound way, enhancing optimization efficiency and interpretability by explicitly modeling parameter sensitivities.

However, these methods remain confined to layer-wise optimization and do not jointly address inter-layer fusion within the same differentiable framework. Layer fusion—critical for reducing off-chip

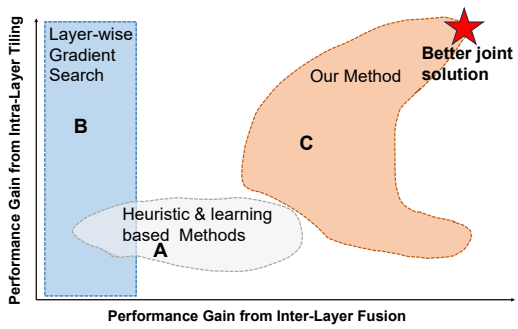


Figure 1: The joint design space of inter-layer fusion and intra-layer mapping. Traditional heuristic and learning-based methods (A) explore only a limited sub-region, while layer-wise gradient approaches (B) remain confined to a single axis. In contrast, our method (C) performs a differentiable search over the full joint space, converging to a superior solution (red ⋆).

memory traffic—is still delegated to separate heuristic or learning-based passes, precluding truly global co-optimization of fusion structure and intra-layer mapping (Figure 1). These shortcomings can be attributed to two fundamental causes.

Firstly, while general in principle, heuristic and learning-based methods face severe scalability barriers when addressing the combinatorially large, high-dimensional design space of joint mapping and fusion. For instance, Gaussian-process-based Bayesian Optimization (BO) requires inverting an $N \times N$ covariance matrix, incurring $O(N^3)$ cost in the number of evaluated samples N; in high-dimensional design spaces, N must grow rapidly to explore the space, making BO quickly intractable. Secondly, existing gradient-based approaches, though computationally efficient, are inherently limited to intra-layer optimization. Their fundamental constraint stems from the fact that inter-layer fusion decisions are inherently discrete—choosing which layers to merge admits no natural gradient. This discontinuity breaks the assumptions underlying standard backpropagation, forcing these methods to treat layers as independent and optimize each in isolation.

To bridge this gap, we propose FADiff, a gradient-based optimization framework that unifies intra-layer mapping and inter-layer fusion into a single, jointly differentiable optimization problem, as illustrated in Figure 2. At the heart of FADiff lies a differentiable analytical cost model that captures the complex interaction between mapping and fusion strategies in a unified form. This unified formulation enables efficient gradient-based co-optimization of both problems, yielding high-performance and hardware-valid DNN deployment solutions. The main contributions are as follows:

- We develop an analytical cost model that jointly captures intra-layer mapping and inter-layer fusion behaviors, enabling a unified differentiable formulation for energy and latency trade-offs on tensor accelerators.
- We embed validity constraints into the CP-SAT solver and add an augmented-Lagrangian penalty term to the objective, ensuring gradient-based optimization in continuous space with consistently feasible solutions.
- We demonstrate that FADiff achieves an average of 38.8% reduction in Energy-Delay Product (EDP) compared with state-of-the-art optimization approaches.

The remainder of this paper is organized as follows. Section 2 formulates the joint mapping–fusion problem, Section 3 details the proposed FADiff framework, Section 4 presents the experimental results, and Section 5 concludes the paper.

2 Problem Formulation

In this section, we formalize the joint mapping–fusion deployment problem addressed in this work.

We model the DNN as a directed acyclic graph (DAG), i.e., $G = (V, E)$, where each vertex $v \in V$ represents a computational layer and each edge $e \in E$ encodes a tensor dependency, thereby capturing the producer-consumer relationships that govern fusion legality. A *deployment strategy* specifies the complete spatio-temporal execution of the network on the target hardware through intra-layer mapping and inter-layer fusion. Our objective is to identify the strategy that minimizes the EDP subject to on-chip buffer capacity and dataflow constraints.

3 Methodology

Existing deployment frameworks for tensor accelerators face a fundamental trade-off: non-differentiable combinatorial search incurs prohibitive cost, while differentiable alternatives remain confined to layer-wise mapping and neglect fusion entirely. As illustrated in Figure 2, we resolve this tension with a unified differentiable framework that co-optimizes layer mapping and fusion end-to-end.

In this section, we present the continuous representation of strategies in Section 3.1, followed by the differentiable energy and latency model in Section 3.2, the differentiable modeling of optimization constraints in Section 3.3, and the end-to-end optimization framework in Section 3.4.

3.1 Continuous Representation of Strategies

The central obstacle to gradient-based optimization is the discrete nature of mapping and fusion decisions. We overcome this by introducing continuous parameterizations that relax both strategy spaces into a jointly differentiable formulation.

Mapping Representation. Each layer's mapping is fully characterized by its tiling factors across all problem dimensions and memory levels. We adopt a 7-dimensional problem space $\{N, K, C, P, Q, R, S\}$, where N is batch size, K and C are output and input channel counts, P and Q are output spatial dimensions, and R and S are kernel height and width. This formulation subsumes both CONV and GEMM (the latter setting P=Q=R=S=1). Tiling targets a 4-level memory hierarchy: $L3$ (DRAM), $L2$ (scratchpad), $L1$ (accumulator), and $L0$ (PE registers). We denote the temporal tiling factor of dimension d at level m by $T_{t,d,m}$, and the corresponding spatial factor by $T_{s,d,m}$; where unambiguous, we write $T_{d,m}$ to refer to either.

Although gradient-based optimization requires continuous tiling factors $\{T_{t,d,m}\}$ and $\{T_{s,d,m}\}$, the design space is inherently discrete. We therefore optimize continuous relaxations at all memory levels except the outermost, whose factors are derived from the total dimension size to guarantee that the product along each dimension does not exceed it. An augmented-Lagrangian penalty ensures $T_{d,m} \geq 1$ (Section 3.3). To reconcile continuous optimization with discrete feasibility, we pre-enumerate all valid mapping candidates per layer using the CP-SAT solver and periodically project each relaxed factor

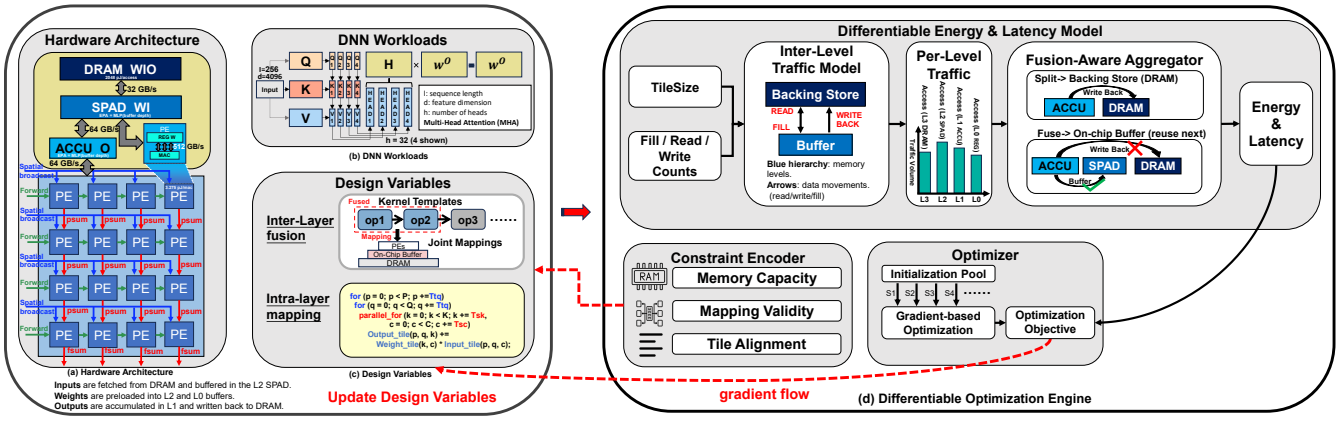


Figure 2: Overview of FADiff. Left (regions a–c): hardware architecture, DNN workloads, and design variables. Right (region d): differentiable optimization engine.

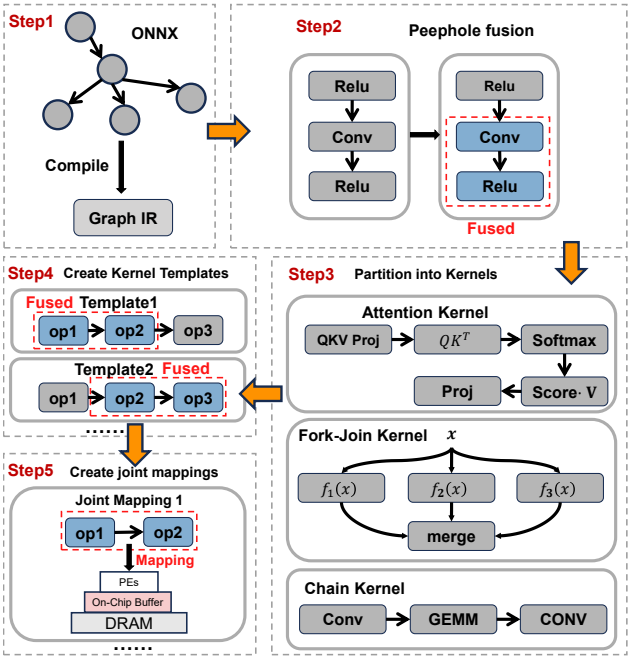


Figure 3: Overview of the five-step pipeline for generating the layer fusion design space.

onto its nearest valid candidate, maintaining consistency with the discrete space throughout training.

Kernel-Template-Based Layer Fusion Representation. Existing gradient-based methods optimize intra-layer mapping effectively but do not model cross-layer fusion. We introduce a differentiable, template-based fusion representation that captures the topology of modern DNNs while remaining compatible with layer-wise mapping optimization.

As shown in Fig. 3, we compile the model's ONNX graph into a graph-level IR, apply peephole fusion of lightweight operators (e.g., ReLU, Clip) into preceding compute-bound layers, and partition the result into optimizable subgraphs called *kernels*. Representative kernel topologies include *attention* (Q/K projection → score MatMul → Softmax → V MatMul), *residual* (two branches merging at Add), *fork-join* (a shared ancestor splitting into paths that reconverge at Concat or Mul), and *chain* (a linear sequence of Conv/GEMM/MatMul).

For each kernel $k \in \mathcal{K}$, we construct a finite, size-tunable template library $\mathcal{T}_k$. Each template $\tau \in \mathcal{T}k$ encodes (1) fused operator groups $\mathcal{G}k, \tau$, whose depth determines the intermediate buffer footprint; (2) tile-alignment constraints between producers and consumers (Section 3.3); and (3) intermediate-tensor residency constraints that retain data on-chip (Section 3.3). All templates are generated by a CP-SAT solver, guaranteeing correctness and hardware-constraint satisfaction by construction.

Within each fused group $G \in \mathcal{G}_{k,\tau}$, the constituent nodes are jointly mapped: their combined tiling simultaneously determines buffered tile sizes, producer-consumer alignment, and on-chip residency—all of which must hold for the fusion to be valid. We use the CP-SAT solver to pre-enumerate per-node mapping candidates under these hard constraints; the subsequent optimization over this candidate pool is described in Section 3.4.

Layer fusion thus reduces to a two-level problem: discrete template selection per kernel, followed by continuous joint-mapping optimization within each fused group.

Template selection is made differentiable via Gumbel-Softmax with straight-through estimation (STE). Learnable logits $l_{k,\tau}$ yield the relaxed categorical distribution:

$$p_{k,\tau} = \frac{\exp\big((l_{k,\tau} + g_{k,\tau})/\lambda\big)}{\sum_{\tau' \in \mathcal{T}_k} \exp\big((l_{k,\tau'} + g_{k,\tau'})/\lambda\big)}, \tag{1}$$

where $g_{k,\tau} \sim \text{Gumbel}(0, 1)$ and λ is the temperature.

For the joint mapping within each fused group, we optimize continuous factors—consistent with the standalone mapping formulation—while enforcing feasibility through augmented-Lagrangian penalties (Section 3.3) and periodic projection onto the pre-enumerated candidate set (Section 3.4).

Unified Hierarchical Representation. FADiff parameterizes the design space with two coupled groups. *Continuous mapping factors* $\{T_{t,d,m}\}$ and $\{T_{s,d,m}\}$ govern tiling and spatial-unrolling decisions for every layer across the memory hierarchy; the same parameterization applies to both standalone and fused layers, with feasibility maintained through augmented-Lagrangian penalties and periodic

rounding. *Discrete fusion logits* $\{l_{k,\tau}\}$ select, for each detected kernel $k \in \mathcal{K}$, a fusion template specifying its constituent layers and pipeline depth. Formally, the complete parameter set is:

$$\Theta = \left\{ T^{(l)}_{t,d,m},\ T^{(l)}_{s,d,m} \right\} \ \cup\ \left\{ l_{k,\tau} \mid k \in \mathcal{K},\ \tau \in \mathcal{T}_k \right\}. \tag{2}$$

Both groups are jointly optimized: mapping factors via gradient descent on continuous relaxations (Section 3.1), template selection via Gumbel-STE over a CP-SAT–pre-enumerated feasible set—enabling FADiff to co-optimize mapping and fusion in a single differentiable pass.

3.2 Differentiable Energy and Latency Model

We develop a unified differentiable cost model that estimates end-to-end energy and latency for a complete deployment strategy, enabling gradient-based exploration of the joint design space.

Data Traffic. Data movement dominates both latency and energy in tensor accelerators, often rivaling computation cost [10]. We formulate total inter-level data traffic as a differentiable function of continuous mapping parameters and model layer fusion's impact on on-chip reuse via the kernel-template representation, ensuring gradient propagation through the full memory hierarchy.

(1) Fill Traffic. For tensor $T \in \{I, W\}$, the fill traffic into level L_i is:

$$\mathrm{Fill}(L_i, T) = \mathrm{TileSize}(i, T) \cdot \mathrm{FetchCount}(i, T), \tag{3}$$

$$\mathrm{TileSize}(i, T) = \prod_{d \in \mathcal{D}(T)} \prod_{k=0}^{i} T_{t,d,k} \cdot T_{s,d,k}, \tag{4}$$

$$\mathrm{FetchCount}(i, T) = \prod_{d \in \mathcal{D}(T)} \prod_{k=i+1}^{M} T_{t,d,k}, \tag{5}$$

where $\mathcal{D}(T)$ denotes the set of problem dimensions associated with T (e.g., $\{R, S, C, K\}$ for weights) and L_M is the outermost level (DRAM).

(2) Read Traffic. Read traffic captures data transferred from a lower (slower) to a higher (faster) memory level upon request. We distinguish two types: *inter-memory reads*, which move entire tiles between levels, and *PE-supplying reads*, which feed data directly to the compute array.

(a) Inter-Memory Reads. For tensor $T \in \{I, W\}$, the read traffic from L_{i+1} is:

$$\mathrm{Read}(L_{i+1}, T) = \mathrm{TileSize}(i, T) \cdot \mathrm{FetchCount}(i, T). \tag{6}$$

No broadcast reuse applies, as no broadcast network exists between memory levels.

(b) PE-Supplying Reads. Reads from the innermost buffer L_1 feed the PE array. The traffic equals total MACs divided by the spatial broadcast factor:

$$\mathrm{Read}(L_1, T) = \frac{\mathrm{Ops}}{\mathrm{Bcast}_T}, \qquad \mathrm{Bcast}_T = \prod_{d \notin \mathcal{D}(T)} T_{s,d,m}, \tag{7}$$

where Ops is the layer's MAC count and Bcast_T captures multicast reuse over dimensions not indexed by T.

(3) Write-Back Traffic. Output tensors and partial sums reside only in the L1 accumulator and L3 DRAM (bypassing L2/L0), yielding two write-back types.

(a) Inter-Memory Write-back. Completed output tiles drain from L1 to DRAM:

$$\mathrm{WriteBack}(L_i, T) = \mathrm{TileSize}(i, T) \cdot \mathrm{WriteCount}(i, T), \tag{8}$$

where $\mathrm{WriteCount}(i, T)$ is determined by the outer temporal loop iterations, analogous to FetchCount in Equation (5).

(b) Accumulation Write-back. Partial sums are spatially reduced within the PE array before writing to L1:

$$\mathrm{WriteBack}(L_1, T) = \frac{\mathrm{Ops}}{\mathrm{Reduce}_T}, \qquad \mathrm{Reduce}_T = \prod_{d \notin \mathcal{D}(T)} T_{s,d,m}. \tag{9}$$

Latency Model. We adopt a roofline-style model assuming full overlap of computation and memory access. Each layer's latency is the maximum of compute and memory-bound terms; total latency sums over all layers:

$$\mathrm{Latency}(l_{i'}) = \max\left(\frac{\mathrm{Ops}}{\mathrm{PEs}},\ \max_i \frac{\mathrm{Access}(L_i)}{BW_i} \right), \tag{10}$$

where PEs is the effective PE count (determined by $T_{s,d,k}$), $\mathrm{Access}(L_i)$ aggregates fill, read, and write-back traffic at level L_i, and BW_i is its effective bandwidth (Figure 2(a)).

Energy Model. The per-layer energy sums computation and data-movement contributions; total model energy aggregates across layers:

$$E_{\mathrm{compute}} = \mathrm{Ops} \cdot \mathrm{EnergyPerOp}, \tag{11}$$

$$E_{\mathrm{datamove}} = \sum_i \mathrm{Access}(L_i) \cdot \mathrm{EPA}_i, \tag{12}$$

where EnergyPerOp and EPA_i are listed in Figure 2(a). The optimization objective is the EDP. Because all model components are constructed differentiably, gradients of the EDP with respect to mapping parameters $T_{t,d,m}$, $T_{s,d,m}$ and fusion logits $\{l_{k,\tau}\}_{\tau \in \mathcal{T}_k}$ are obtained directly via automatic differentiation.

3.3 Differentiable Modeling of Optimization Constraints

We enforce constraints through a two-pronged strategy. *Hard* constraints—mapping validity, memory capacity, and adjacent-tile alignment—are embedded in the CP-SAT solver during pre-enumeration, ensuring every discrete candidate is feasible by construction. *Soft* augmented-Lagrangian penalties provide differentiable guidance between projection steps, discouraging infeasible drift of the continuous trajectory. Because mapping variables are periodically projected onto the nearest valid candidate and the final solution is always evaluated post-projection, feasibility is guaranteed regardless of intermediate relaxation.

Mapping-Validity Constraints. Mapping validity enforces legal temporal and spatial tiling via three conditions. Each factor must satisfy $T_{d,m} \geq 1$ to exclude degenerate mappings. Spatial parallelism is bounded by the array size:

$$\prod_d T_{s,d,m} \leq N_{\mathrm{PE}}, \tag{13}$$

where m is the memory level attached to the PE array. Finally, temporal and spatial factors must exactly tile each dimension:

$$\prod_m T_{t,d,m} \cdot T_{s,d,m} = D_d, \quad \forall\, d. \tag{14}$$

Memory-Capacity Constraints. For each fused group $G \in \mathcal{G}_{k,\tau}$, the weights and activations of all constituent nodes must co-reside

Algorithm 1 Multi-Start End-to-End Optimization

Input: $\Theta, \mathcal{K}, N, R, \Delta$
Output: $\Theta^\star$

$\{\Theta_n^{(0)}\}_{n=1}^{N} \leftarrow \textsc{MaximinInit}(N)$
$\Theta^\star \leftarrow \emptyset, \quad \mathcal{J}^\star \leftarrow +\infty$
for $n = 1$ to N **do**
  $\Theta \leftarrow \Theta_n^{(0)}$
  for $r = 1$ to R **do**
    sample $\{\tau_k\}_{k\in\mathcal{K}}$ from $\{l_{k,\tau}\}$ via STE
    update mapping factors by gradient descent
    if $r \bmod \Delta = 0$ **then**
      project mapping factors to nearest valid candidates
    if converged **then break**
  project mapping factors to valid candidates
  if $\mathcal{J}(\Theta) < \mathcal{J}^\star$ **then**
    $\Theta^\star \leftarrow \Theta, \quad \mathcal{J}^\star \leftarrow \mathcal{J}(\Theta)$
return $\Theta^\star$

in the on-chip buffer. The aggregate storage at level i is:

$$\text{SizeReq}(G, i) = \sum_{v\in G} \big(S_W(v, i) + S_I(v, i)\big) \le C_i, \quad \forall\, G,\ i, \tag{15}$$

where S_W and S_I are the per-node weight and activation footprints derived from Equation (4), and C_i is the physical capacity of level i.

Adjacent-Tile Alignment Constraints. Within each fused group $G \in \mathcal{G}_{k,\tau}$, the output tile of layer v_i must match the input tile of its successor to ensure correct inter-layer data reuse:

$$\mathbf{o}_{v_i} = \mathbf{i}_{v_{i+1}}, \qquad \forall\,(v_i, v_{i+1}) \in G,\ \forall\, G \in \mathcal{G}_{k,\tau}, \tag{16}$$

where $\mathbf{o}_{v_i}=(o_p, o_q, o_k)$ and $\mathbf{i}_{v_{i+1}}=(i_h, i_w, i_c)$ are the output and input tile shapes, respectively.

All three constraint classes are enforced dually: as augmented-Lagrangian penalty terms that provide differentiable guidance during continuous optimization, and as hard constraints in the CP-SAT solver that guarantee the pre-enumerated candidate set is exclusively feasible. Periodic projection of mapping variables onto their nearest valid candidate maintains feasibility throughout.

3.4 End-to-End Optimization Framework

We solve the joint problem via multi-start gradient descent (Algorithm 1). N trajectories are initialized with a maximin criterion over the full parameter set Θ (Section 3.1) to improve coverage of the non-convex landscape.

Continuous mapping factors for both standalone and fused-group layers are optimized directly and projected every Δ iterations onto the nearest valid candidate in the pre-enumerated set—avoiding the scalability issues of direct discrete selection over large mapping spaces (Section 4.4). Kernel-template selection is handled via Gumbel-STE: hard discrete choices in the forward pass, soft relaxation in the backward pass. Throughout, augmented Lagrangian penalties (Section 3.3) steer trajectories toward feasible, high-quality regions so that final projections yield strong—not merely legal—solutions. After convergence, all factors are discretized and the best trajectory is returned as $\Theta^\star$.

4 Evaluation

In this section, we present the experimental setup in Section 4.1, followed by the validation of the differentiable cost models in Section 4.2, the comparison of optimization quality in Section 4.3, and the ablation study in Section 4.4.

4.1 Experimental Setup

All experiments run on a dual-socket Intel® Xeon™ Platinum 8375C server with 512 GB DRAM. We target Gemmini [11] under two configurations that span a representative embedded design space: a *large* variant (32 × 32 systolic array, 64 KB L1, 512 KB L2) and a *small* variant (16 × 16, 8 KB L1/L2), enabling us to characterize scaling behavior across distinct compute-to-memory ratios. Memory bandwidth, DRAM energy-per-access (EPA), and MAC energy are adopted from DOSA [8]; on-chip buffer EPA is derived from an MLP surrogate fitted to CACTI [12] data. All energy and area parameters are summarized in Figure 2(a). We report energy-delay product (EDP) as the primary deployment-quality metric.

4.2 Validation of Differentiable Cost Models

We first validate single-layer accuracy against Timeloop [13] / Accelergy [14] across standard, depthwise, pointwise, large-kernel convolutions, and fully connected layers. The model achieves 96% accuracy in predicting memory access counts, perfect rank correlation for latency (Kendall's τ=1.0, Spearman's ρ=1.0), and strong rank correlation for energy (τ=0.78, ρ=0.92).

We then assess multi-layer fused operators against DeFiNES [2], whose analytical model is calibrated to silicon. Because absolute values are sensitive to process and microarchitectural assumptions, we compare only ranking consistency. As shown in Figure 4, Z-score–normalized latency and energy trends closely track the DeFiNES reference for both two- and three-layer fusion, confirming that our differentiable cost model preserves the relative ordering required for reliable gradient-based optimization.

4.3 Comparison of Optimization Quality

Methodological Baselines. To isolate the benefit of gradient-based search in this high-dimensional space, we compare against a Genetic Algorithm (GA) [15] and Bayesian Optimization (BO) [16], both operating on the identical differentiable cost model and candidate space. All experiments use the Large-Gemmini configuration (Section 4.1). FADiff launches N=20 independent trajectories of R=1,000 gradient steps with projection interval Δ=100; the Gumbel temperature anneals from λ=8→1 (phase 1) and 1→0.1 (phase 2). Candidate pools are capped at 10,000 (standalone) and 30,000 (fused group) entries to balance enumeration cost against feasible-space coverage. All methods share the same total evaluation budget unless otherwise noted.

As shown in Figure 5, our gradient-based method achieves substantially lower EDP in far less wall-clock time than GA and BO under an equal budget—underscoring the efficiency of analytical gradients over heuristic or surrogate-based search in high-dimensional mapping spaces. To contextualize these dynamics within end-to-end deployment quality, we additionally run Random Search (RD), GA, and BO on the full model suite; their best EDP is reported alongside DOSA [8] and our method in Table 1.

Table 1: EDP Comparison Across Methods and Gemmini Configurations.

Model	Gemmini large					Gemmini small				
	RD	GA	BO	DOSA [8]	FADiff	RD	GA	BO	DOSA [8]	FADiff
VGG19	3.890e+13	2.100e+13	4.200e+13	4.540e+13	**1.730e+13**	1.510e+14	9.060e+13	2.030e+14	7.380e+13	**4.710e+12**
VGG16	2.560e+13	1.780e+13	2.830e+13	1.610e+13	**1.210e+13**	1.040e+14	4.930e+13	7.860e+13	1.700e+13	**3.240e+12**
MobileNetV1	3.760e+12	1.670e+12	2.010e+12	3.230e+12	**1.600e+12**	7.200e+12	4.810e+12	6.690e+12	1.030e+13	**3.620e+12**
GPT3-6.7B	2.170e+14	7.920e+13	1.110e+14	8.330e+13	**4.810e+13**	1.530e+15	1.170e+15	1.140e+15	1.910e+15	**3.950e+14**
ResNet18	6.570e+11	4.300e+11	1.130e+12	5.030e+11	**2.740e+11**	1.080e+12	5.710e+11	1.500e+12	8.600e+11	**5.640e+11**
Average	5.718e+13	2.402e+13	3.689e+13	2.971e+13	**1.587e+13**	3.587e+14	2.631e+14	2.860e+14	4.024e+14	**8.143e+13**

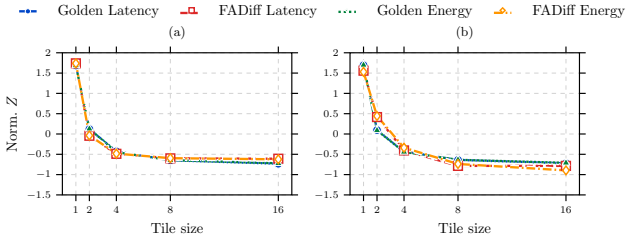


Figure 4: Comparison of Z-score normalized trends under different tile sizes. Left: two-layer fusion. Right: three-layer fusion.

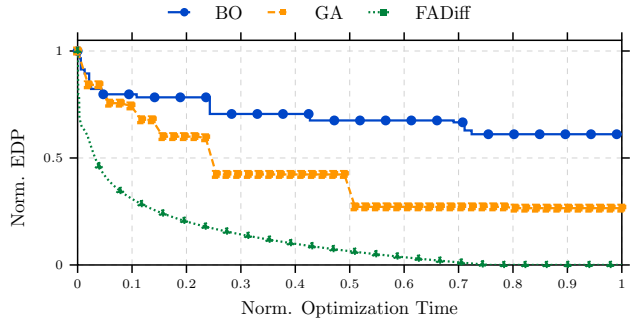


Figure 5: EDP vs. optimization time for GA, BO, and our gradient-based method; with the same optimization budgets, our method converges faster to lower EDP.

Differentiable Framework Baselines. We compare FADiff against DOSA [8], which shares our gradient-based paradigm but optimizes each layer independently without modeling fusion. Both methods are evaluated on the two Gemmini configurations (Section 4.1) under equal budget. The workload suite spans representative CNNs [17–19] and a GPT-3-6.7B decoder-only Transformer [20], for which we optimize MHA [21] and FFN blocks (tensor dimensions in Figure 2(b); FFN hidden size 16,384).

As shown in Table 1, FADiff achieves the lowest EDP across all ten workload–hardware combinations, reducing EDP by **38.8%** on average relative to the best competing baseline. Gains are especially pronounced on Small-Gemmini, indicating that joint fusion-and-mapping optimization becomes increasingly critical under tight resource budgets. The underlying design space is astronomically large—rendering RD intractable, BO prohibitively expensive, and

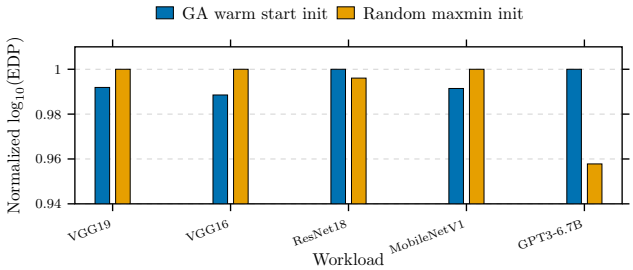


Figure 6: Effect of initialization strategy on optimization quality. GA warm-start and random (maximin) initialization are compared in terms of final EDP.

GA prone to premature convergence. By unifying fusion structure and per-layer mapping in a single differentiable formulation, FADiff enables gradient-driven search to efficiently navigate this otherwise intractable space.

4.4 Ablation Study

Effect of Initialization Strategy. We compare two strategies under an equal evaluation budget: (1) random initialization from 100 maximin-sampled points, and (2) GA warm-start from a single elite individual after 10 generations (population 10). Figure 6 reports EDP on the Large Gemmini across all five workloads. GA warm-start consistently outperforms on structurally regular CNNs (MobileNetV1, VGG16, VGG19), where brief evolutionary pre-exploration steers optimization toward favorable basins. Conversely, random initialization prevails on GPT-3, whose large, strongly coupled design space is sensitive to the premature bias introduced by GA seeding. ResNet18 exhibits intermediate behavior, with neither strategy dominating.

Effect of Design Space Size. Continuous variables are periodically projected onto pre-enumerated discrete candidate sets whose cardinality directly shapes the optimization landscape. Figure 7 sweeps candidate-set sizes for standalone layers and fused groups on Large Gemmini across three workloads. Larger spaces generally lower EDP; the gain is most pronounced for GPT-3 6.7B—especially in the fused-group dimension—confirming that strongly coupled workloads demand a rich fusion-aware search space. ResNet18 saturates at moderate cardinality; MobileNetV1 saturates early, its compact space well-covered even at small budgets. Returns diminish once added candidates cease to introduce genuinely distinct solutions.

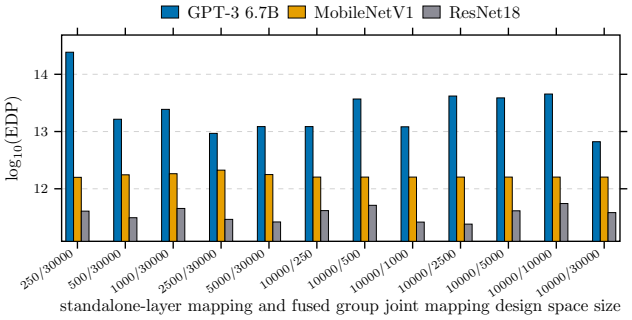


Figure 7: Effect of design space size on optimization quality. The x-axis varies the standalone and fused-group joint design space sizes, and the y-axis reports $\log_{10}$(EDP).

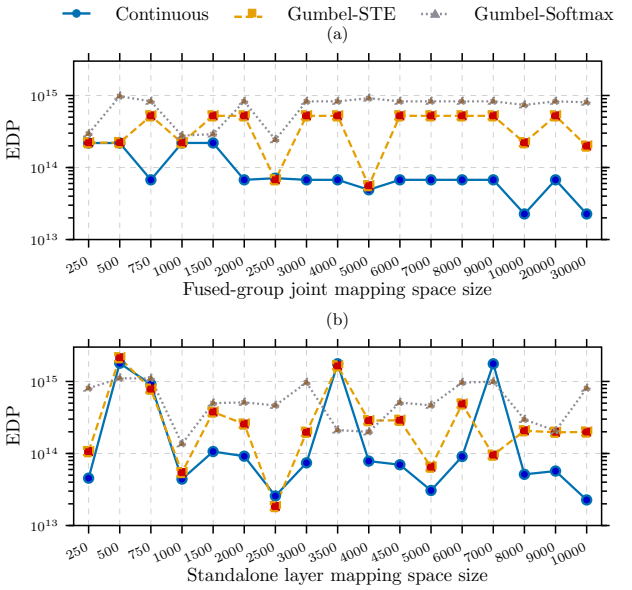


Figure 8: Discrete-optimization strategy comparison. (a) Fused-group size swept; standalone fixed at 10,000. (b) Standalone size swept; fused-group fixed at 30,000.

Continuous Relaxation vs. Gumbel-Softmax with STE. We compare continuous relaxation with periodic projection, Gumbel-Softmax, and Gumbel-STE via two controlled sweeps on GPT-3 6.7B (Large Gemmini) that independently vary standalone and fused-group space sizes (Figure 8). Continuous relaxation dominates increasingly as either space grows: optimizing in a smooth landscape and projecting onto the nearest feasible point preserves local gradient structure at any cardinality. Gumbel-Softmax degrades most severely because its weighted-average forward pass diverges from valid solutions as the categorical space expands. Gumbel-STE mitigates this through hard forward selection but still suffers from gradient noise over large logit spaces. The ranking is consistent: continuous relaxation > Gumbel-STE > Gumbel-Softmax in scalability.

5 Conclusion

We present FADiff, a differentiable framework that jointly optimizes intra-layer mapping and inter-layer fusion. FADiff introduces a unified, hierarchical parameterization in which continuous mapping factors and discrete fusion-template logits are co-optimized through augmented-Lagrangian penalties and Gumbel-STE selection over CP-SAT-pre-enumerated candidates. This formulation preserves gradient structure while guaranteeing hardware-valid solutions at every projection step. By reconciling continuous optimization with discrete constraints, FADiff efficiently navigates an otherwise intractable joint design space, achieving 38.8% average EDP reduction over existing methods.

References

[1] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: An automated end-to-end optimizing compiler for deep learning," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation (OSDI'18)*, 2018, pp. 579–594.

[2] L. Mei, K. Goetschalckx, A. Symons, and M. Verhelst, "Defines: Enabling fast exploration of the depth-first scheduling space for dnn accelerators through analytical modeling," in *Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 570–583.

[3] J. Cai, X. Wang, M. Gao, S. Peng, Z. Zhu, Y. Wei, Z. Wu, and K. Ma, "Soma: Identifying, exploring, and understanding the dram communication scheduling space for dnn accelerators," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 533–548.

[4] C. Bai, X. Wei, Y. Zhuo, Y. Cai, B. Yu, and Y. Xie, "Klotski: Dnn model orchestration framework for dataflow architecture accelerators," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

[5] C. Bai, X. Wei, Y. Zhuo, Y. Cai, H. Zheng, B. Yu, and Y. Xie, "Klotski v2: Improved dnn model orchestration framework for dataflow architecture accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 3, pp. 1045–1057, 2025.

[6] Q. Sun, C. Bai, H. Geng, and B. Yu, "Deep neural network hardware deployment optimization via advanced active learning," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1510–1515.

[7] Q. Sun, C. Bai, T. Chen, H. Geng, X. Zhang, Y. Bai, and B. Yu, "Fast and efficient dnn deployment via deep gaussian transfer learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5380–5390.

[8] C. Hong, Q. Huang, G. Dinh, M. Subedar, and Y. S. Shao, "Dosa: Differentiable model-based one-loop search for dnn accelerators," in *2023 56th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023, pp. 209–224.

[9] Y. Zhao, H. Sharif, V. Adve, and S. Misailovic, "Felix: Optimizing tensor programs with gradient descent," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024, p. 367–381.

[10] Q. Huang, P.-A. Tsai, J. S. Emer, and A. Parashar, "Mind the gap: Attainable data movement and operational intensity bounds for tensor algorithms," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 150–166.

[11] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 769–774.

[12] A. Shafaei, Y. Wang, X. Lin, and M. Pedram, "Fincacti: Architectural analysis and modeling of caches with deeply-scaled finfet devices," in *2014 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2014, pp. 290–295.

[13] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to dnn accelerator evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 304–315.

[14] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[16] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, 2012, pp. 2951–2959.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.
[20] B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, vol. 1, no. 3, p. 3, 2020.
[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.