

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

Shuaow Jia

Beijing University of Posts and Telecommunications

Outline

- Background & Motivation
- The Challenge
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results

Outline

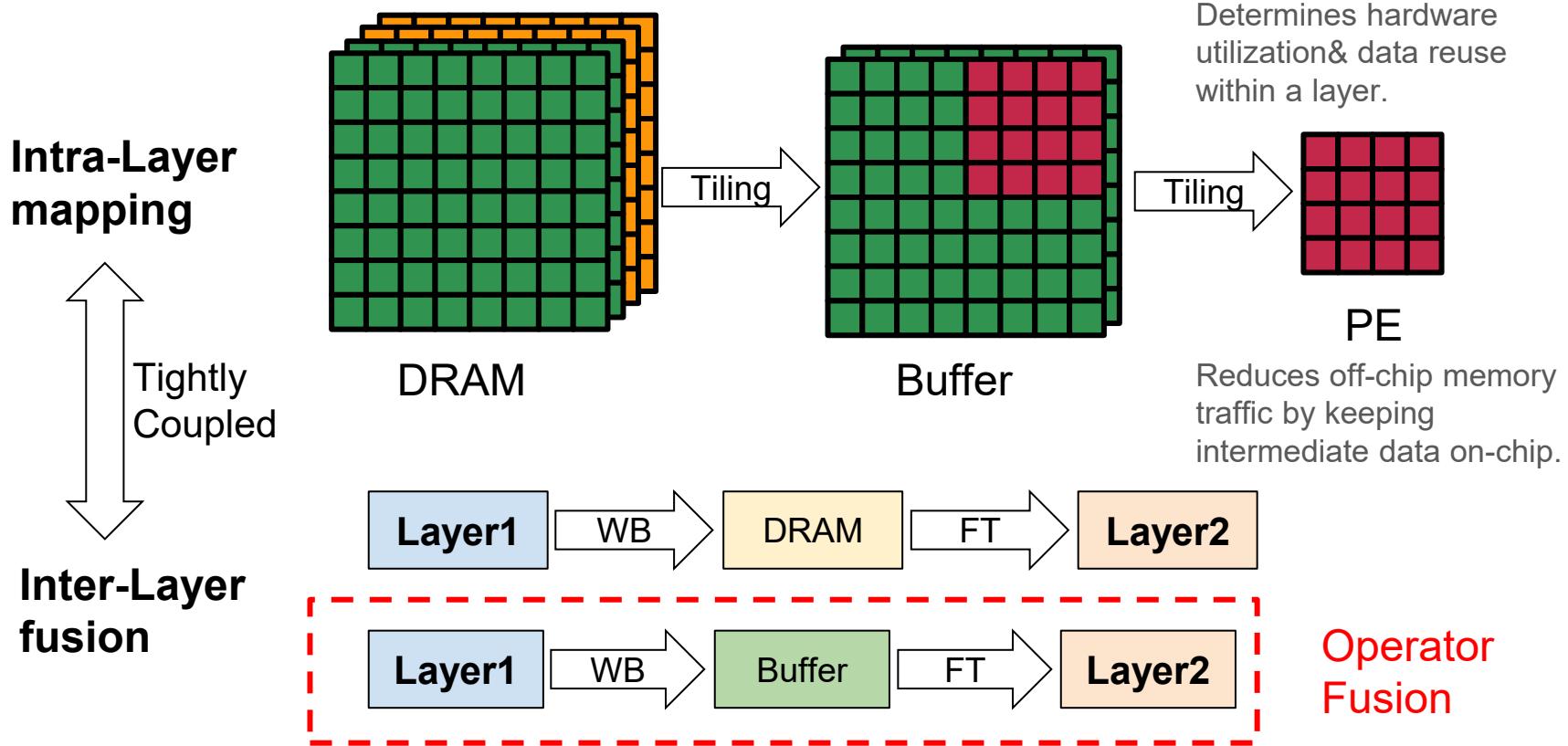
- **Background & Motivation**
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results
- Conclusion

Deploying Large Models on Edge Accelerators

The Workload Demand (e.g. GPT-3)	Hardware (e.g., Gemmini)
Model Scale: Large Language Models	Target Device: Edge Tensor Accelerators
Workload Type: <ul style="list-style-type: none">• GPT-3 (6.7B Parameters)• Transformer Layers (MHA + FFN)	Compute Resources: <ul style="list-style-type: none">• 16x16 to 32x32 Systolic Array• Limited PE Parallelism
Memory Footprint: <ul style="list-style-type: none">• Gigabytes of Weights & Activations• High Bandwidth Requirement	On-Chip Memory (SRAM): <ul style="list-style-type: none">• L1 Buffer: 8 KB - 64 KB• L2 Buffer: 8 KB - 512 KB
Goal: High Throughput & Low Latency	Bottleneck: DRAM Bandwidth & Capacity

Deploying GB-scale models on KB-scale buffers creates a massive efficiency gap. We must optimize data movement

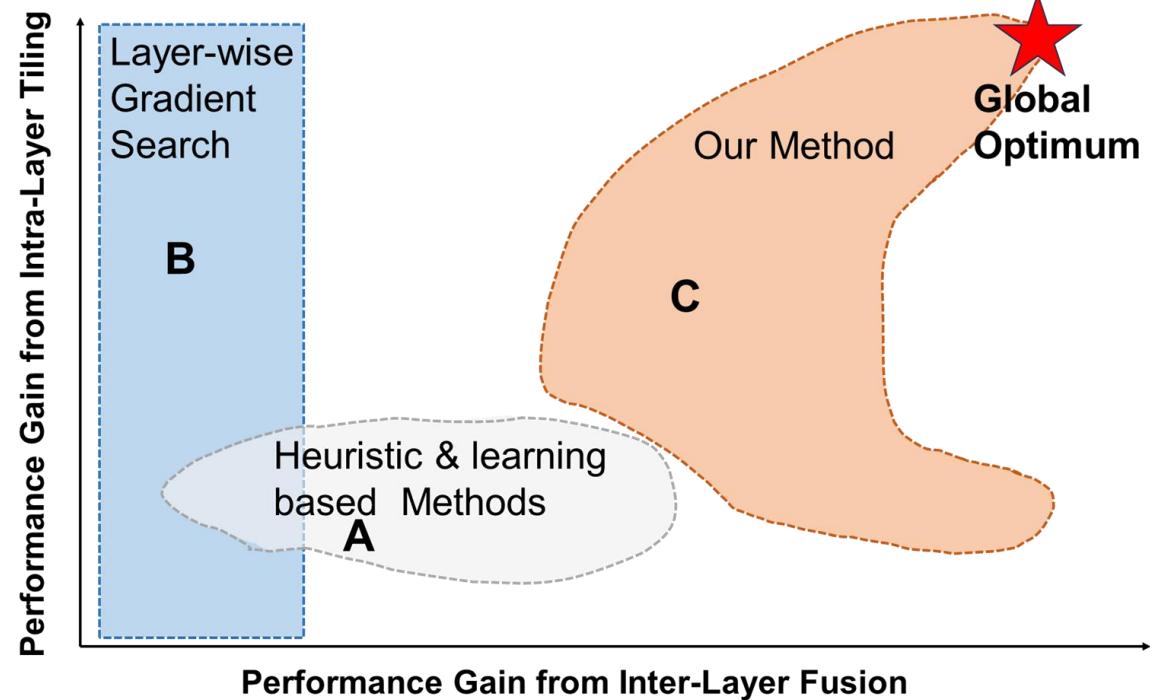
The Coupled Design Space: Mapping & Fusion



Limitations of SOTA

1. Heuristic/Learning-based (e.g., TVM, BO, GA): Limited exploration due to scalability barriers ($O(N^3)$ cost)

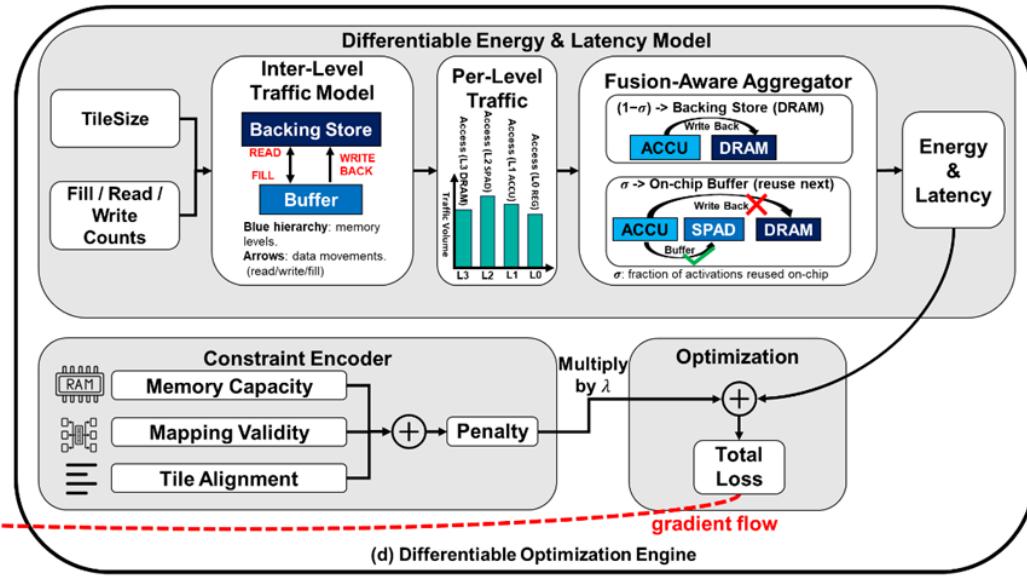
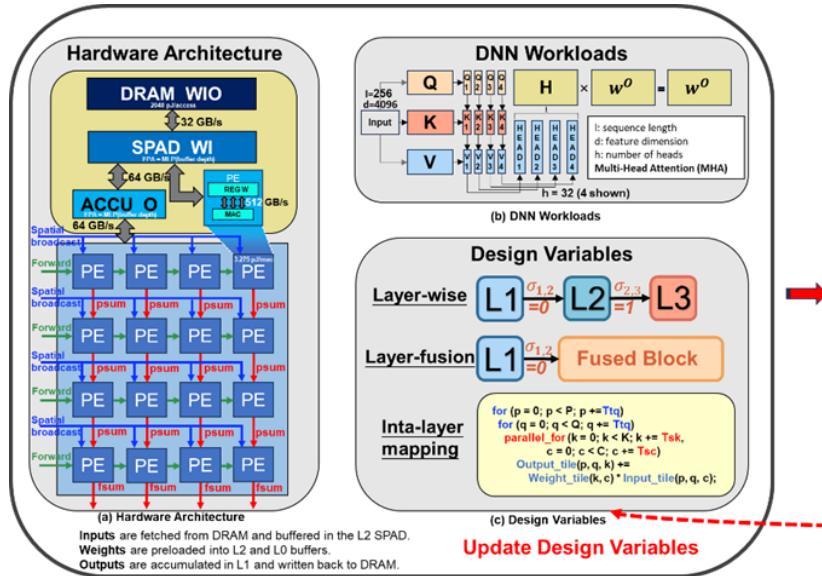
2. Existing Differentiable Methods (e.g., DOSA): Confined to single-layer optimization; cannot handle discrete fusion.



Outline

- Background & Motivation
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results

FADiff: A Unified Differentiable Optimization Framework



Outline

- Background & Motivation
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results

Intra-Layer Mapping (Integer Tiling)

Problem: Tiling factors $T_{d,m}$ must be discrete integers (divisors of loop size).

Solution: Gumbel-Softmax Relaxation.

- Assign a logit to each candidate divisor based on proximity.
- Sample a differentiable probability vector p_j
- Calculate the expected value: $\hat{d} = \sum p_j \cdot d_j$

Result: The tiling factor becomes a weighted sum, allowing gradients to flow to the logits.

Intra-Layer Mapping (Integer Tiling)

An example: Optimizing continuous parameter $T_{cont} = 6.4$ for a discrete loop dimension $N = 64$.

- **Candidate Generation (Factorization)**
 - Valid Set d_j : {1,2,4,8,16,32,64}
- **Proximity Logits (Distance Metric)**
 - Assign score based on distance: $l_j = -\alpha(T_{cont} - d_j)^2$
 - $d = 8: -(6.4 - 8)^2 = -2.56$; $d = 4: -(6.4 - 4)^2 = -5.76$
- **Probabilistic Sampling (Gumbel-Softmax)**
 - Formula: $p_j = \text{Softmax}\left((l_j + g_j)/\tau\right)$
 - Outcome: A probability vector, e.g., $P(8) \approx 0.8, P(4) \approx 0.15$.
- **Straight-Through Estimator (STE)**
 - Forward Pass (Hardware Valid): $d^* = 8$ (Integer).
 - Backward Pass (Differentiable): $\hat{d} = \sum p_j \cdot d_j \approx 7.1$

Inter-Layer Fusion (Fusion Strategy)

Problem: Fusion is inherently binary (Yes/No).

Solution: Continuous Variable $\sigma_i \in [0,1]$.

- $\sigma_i \approx 0$: No fusion (DRAM Write-back).
- $\sigma_i \approx 1$: Full fusion (On-chip Reuse).

Result: Fusion becomes a continuous "degree of reuse," enabling joint optimization with mapping.

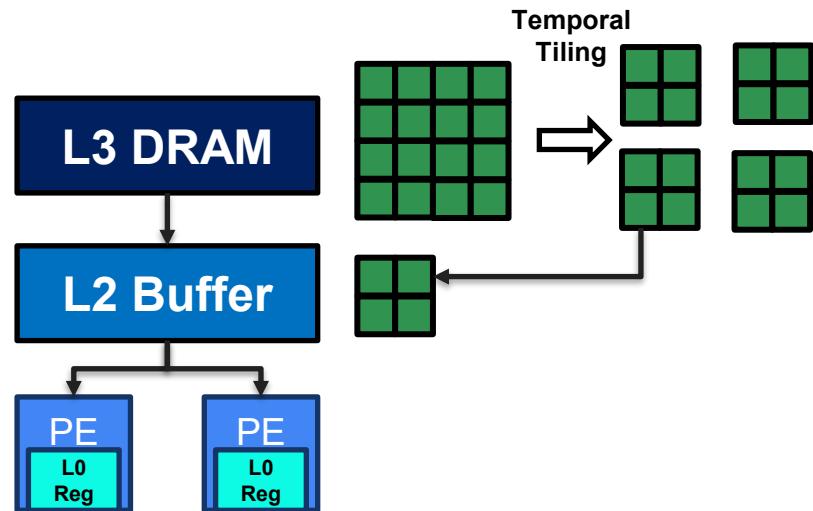
Outline

- Background & Motivation
- Overview of FADiff
- Methodology
 - Unified Representation
 - **Differentiable Cost Model**
 - Constraints & Optimization
- Evaluation and Results

Intra-Layer Data Fill Traffic Model

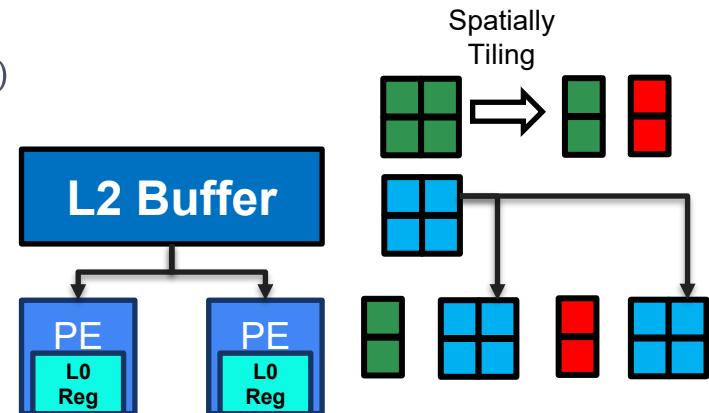
- **Fill Traffic:**

- Traffic from higher level $i + 1$ to lower level i .
- Formula: $\text{Fill}(L_i, T) = \text{TileSize}(i, T) \times \text{FetchCount}(i, T)$



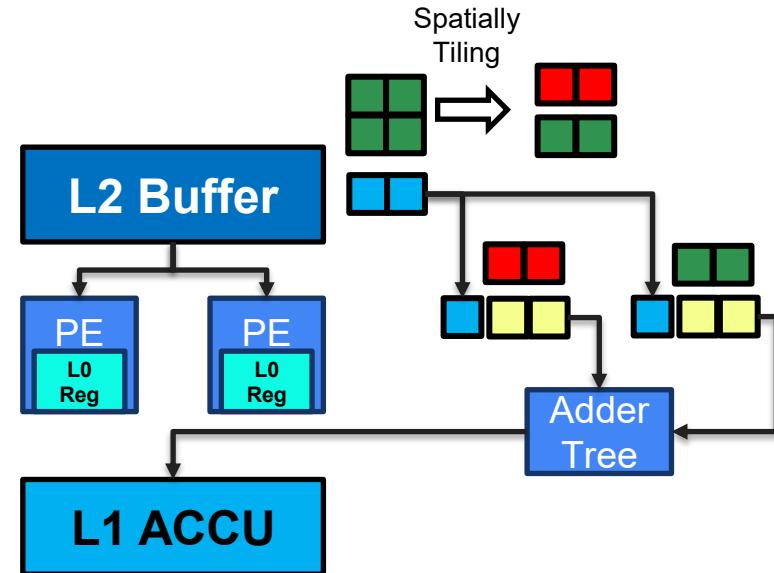
Intra-Layer Data Read Traffic Model

- **Inter-Memory Reads (Data Transfer):**
 - Transferring tiles from lower-level memory (e.g., DRAM) to higher-level (e.g., Scratchpad).
 - Formula: $Read(L_{i+1}, T) = \text{TileSize}(i, T) \cdot \text{FetchCount}(i, T)$
- **PE-Supplying Reads (Computation Feed)**
 - Data fed directly into the PE array for computation.
 - Formula: $Read(L_i, T) = \frac{\text{ops}}{Bcast_T}$
 - The Spatial Broadcast Factor ($Bcast_T$):
 - Quantifies spatial data reuse (e.g., broadcasting weights across rows).
 - Definition: $Bcast_T = \prod_{d \notin \text{dims}(T)} T_{s,d,m}$



Intra-Layer Data Write back Traffic Model

- **Accumulation Write-back (PE to L1):**
 - Moving partial sums from PEs to the Accumulator.
 - Formula: $WriteBack(L_1, T) = \frac{Ops}{Reducer}$
 - The Spatial Reduction Factor (*Reducer*):
 - Definition: $Reducer = \prod_{d \notin dims(T)} T_{s,d,m}$
- **Inter-Memory Write-back (L1 to DRAM)**
 - Off-loading completed output tiles to main memory.
 - Constraint: Outputs typically bypass the L2 Scratchpad.
 - Formula: $WriteBack(L_i, T) = TileSize(i, T) \cdot WriteCount(i, T)$



Fusion-Aware Inter-Layer Traffic

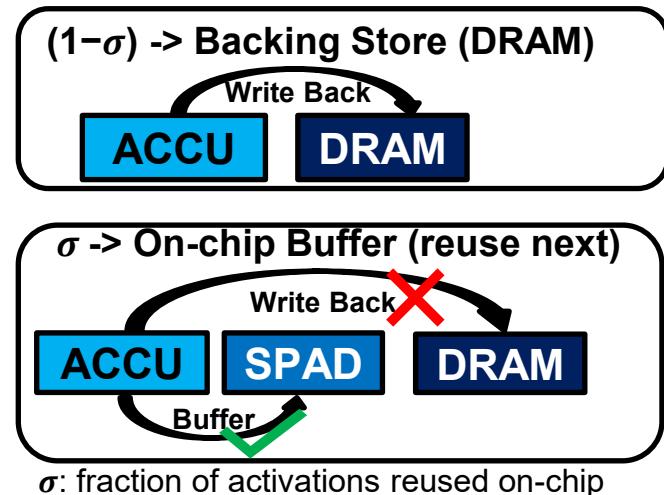
The Fusion Variable $\sigma_i \in [0, 1]$ modulates the traffic boundary.

- **Output Traffic (Layer i)**

- Instead of writing everything to DRAM, we split the flow:
- To DRAM (Write-back): $WriteBack(L_3) = (1 - \sigma_i) \cdot WriteBack_{baseline}$
- To Next Layer (On-chip Copy): $Copy(L_1 \rightarrow L_2) = \sigma_i \cdot WriteBack_{baseline}$

- **Input Traffic (Layer $i + 1$)**

- The next layer reads less from DRAM because it gets data from the previous layer:
- From DRAM (Fill): $Fill(L_2) = (1 - \sigma_i) \cdot Fill_{baseline}$



Energy & Latency Aggregation

- **Latency Model (Roofline-based)**
 - Assumes overlap between compute and memory.
 - Formula: $Latency = \sum_i \max \left(\frac{Ops}{PES}, \max_m \frac{Access(L_m)}{BW_m} \right)$
- **Energy Model**
 - Sum of dynamic energy components.
 - Formula: $Energy = E_{compute} + \sum_m Access(L_m) \cdot EPA_m$

Outline

- Background & Motivation
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results

Constrained Optimization

- **The Total Loss Function:**
 - Formula: $\text{Loss} = \frac{\text{Performance}}{\underbrace{\text{EDP}}_{\text{Constraints}}} + \lambda \cdot \left(\underbrace{P_{map} + P_{mem} + P_{align}}_{\text{Constraints}} \right)$
- **Constraint I: Mapping Validity (P_{map})**
 - **Tiling Validity:**
 - Tiling factors must be ≥ 1 .
 - Formula: $P_{valid} = \sum \left(\max(0, 1 - T_{d,m}) \right)^2$
 - **Spatial Resource Limit:**
 - Parallelism cannot exceed physical PE array size (N_{PE})
 - Formula: $P_{spatial} = \left(\max(0, \prod T_{s,d,m} - N_{PE} \right) \right)^2$

Fusion Constraints (Memory & Alignment)

- **Constraint II: Memory Capacity (P_{mem}):**
 - For any fused group G , total resident data must fit in the buffer capacity C_i .
 - **Formula:**
 - $P_{mem} = (\max(0, \text{SizeReq}(G) - C_i))^2$
 - $\text{SizeReq}(G, i) = \sum_{v \in G} (S_W(v, i) + S_I(v, i))$

Fusion Constraints (Memory & Alignment)

- **Constraint III: Adjacent-Tile Alignment (P_{align})**
 - Ensures the tiling factors constitute a legal schedule.
 - Formula: $P_{align}(G) = \sum_{(v_i, v_{i+1}) \in G} \left\| o_{v_i} - i_{v_{i+1}} \right\|_2^2$

Outline

- Background & Motivation
- Overview of FADiff
- Methodology
 - Unified Representation
 - Differentiable Cost Model
 - Constraints & Optimization
- Evaluation and Results

Experimental Setup & Validation

Hardware Platform	Accelerator	Gemmini Accelerator
Configurations	Config A (Large)	<ul style="list-style-type: none"> • Array: 32×32 • Memory: 512KB L2 • Target: High-end Edge
	Config B (Small)	<ul style="list-style-type: none"> • Array: 16×16 • Memory: 8KB L2 • Target: TinyML / IoT
Workloads	CNNs	ResNet18, VGG16/19, MobileNetV1
	LLM	GPT-3 6.7B
Baselines	Heuristic	<ul style="list-style-type: none"> • Genetic Algorithm (GA) • Bayesian Optimization (BO)
	Gradient-based	• DOSA (Layer-wise SOTA)

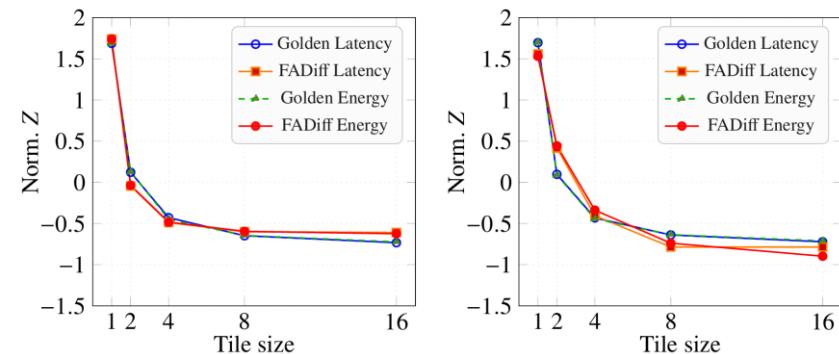


Figure 3: Comparison of Z-score normalized trend. Left: two-layer fusion. Right: three-layer fusion.

Main Results

Table 1: EDP Comparison Across Models and Gemmini Configurations.

Model	Large-Gemmini				Small-Gemmini			
	MICRO'23 [8]	BO [15]	GA [16]	FADiff	MICRO'23 [8]	BO [15]	GA [16]	FADiff
GPT-3 6.7B	1.59×10^{13}	3.67×10^{14}	2.42×10^{14}	1.15×10^{13}	6.14×10^{13}	3.69×10^{15}	2.05×10^{15}	4.65×10^{13}
VGG19	1.16×10^{13}	3.31×10^{14}	2.29×10^{14}	9.62×10^{12}	1.99×10^{13}	8.37×10^{14}	6.41×10^{14}	1.66×10^{13}
VGG16	6.82×10^{12}	1.82×10^{14}	8.65×10^{13}	6.16×10^{12}	9.70×10^{12}	6.44×10^{14}	4.84×10^{14}	8.33×10^{12}
MobileNetV1	2.29×10^{11}	1.60×10^{13}	9.11×10^{12}	1.67×10^{11}	1.44×10^{12}	2.60×10^{13}	2.53×10^{13}	1.37×10^{12}
ResNet18	2.21×10^{10}	4.03×10^{12}	2.98×10^{12}	2.07×10^{10}	2.23×10^{10}	8.13×10^{12}	9.26×10^{12}	2.13×10^{10}
Average	6.91×10^{12}	1.80×10^{14}	1.14×10^{14}	5.49×10^{12}	1.85×10^{13}	1.04×10^{15}	6.42×10^{14}	1.46×10^{13}

Main Results

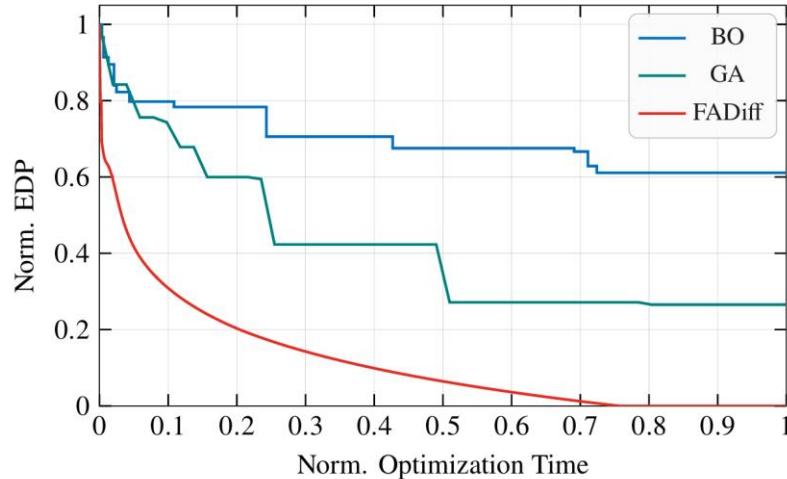


Figure 4: EDP vs. optimization time for GA, BO, and our gradient-based method; with the same time budgets, our method converges faster to lower EDP.