POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

A Framework for Exploring Fused Layers in DNN Dataflows for Spatial Accelerators

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Matteo Salvatore Currò

Advisor: Prof. Cristina Silvano

Co-advisor: Prof. Leandro Fiorin

Academic year: 2024-2025

Introduction

Deep neural networks (DNNs) demand billions of operations per inference.
Deep neural networks (DNNs) dominate modern computing across numerous applications. However, their immense computational cost requires billions of operations for a single inference pass. To meet strict edge energy constraints, spatial architectures (SAs) exploit massive data-level parallelism, though their efficiency relies heavily on optimal mapping, the schedule determining how loop iterations are tiled, ordered, and distributed across the hardware. Even for a single convolutional layer on a moderately complex SA, the resulting set of all valid mappings, the *map space*, exceeds 10^{19} candidates, driving the need for analytical cost models and automated exploration tools. Despite the possibility of selecting the optimal mapping for each individual layer, the classic *layer-by-layer* execution model introduces a critical constraint: after one layer completes, its output tensor is written to off-chip DRAM, only to be read back when the next layer begins. Because DRAM accesses are highly energy-expensive, this intermediate-activation traffic frequently dominates the energy budget and degrades latency. Layer fusion addresses this by computing multiple consecutive layers in a single interleaved schedule. By tiling the execution, intermediate activations are produced and consumed entirely within on-chip buffer. Hardware implementations, such as DepFiN [3], adopt a *depth-first* inter-layer scheduling: the execution remains sequential across layers, but the SA computes an intermediate feature-map tile that is buffered exclusively in on-chip and immediately consumed by the next layer, eliminating redundant external-memory round trips for intermediate. However, the mapping employed by DepFiN is fixed at architecture-definition time and therefore lack the flexibility to efficiently serve diverse workloads.
To overcome this rigidity, an analytical model for fused-layer mapping is needed, yet fusing F layers into a unified loop nest exponentially increases the number of loop dimensions, introduces strict producer–consumer dependencies between layers, and creates new inter-layer data-reuse opportunities. The dimensional proliferation leads to a combinatorial explosion in the design space: for a 10-layer fusion on DepFiN, it expands the map space by over 400 orders of magnitude compared to its single-layer equiva-

lent. Taming this space requires enforcing cascading tile-size dependencies between layers to prune infeasible mappings. Furthermore, since the mapping of a single layer is natively expressed as a loop nest, it is highly desirable to maintain this representational consistency across the extended fused-layer map space. No prior tool represents both single-layer and fused-layer mappings under a unified, coherent loop-nest abstraction while preserving heterogeneous per-layer dataflow freedom.
Therefore, our work formally explores the fused-layer dataflow design space through a unified analytical cost mode. This study formalizes the fused-layer map space, develops an exact constraint-based pruning strategy, constructs a per-layer analytical cost model to evaluate performance. Finally, an extensive design-space exploration across two SAs and four workloads quantifies the conditions under which layer fusion provides a tangible benefit versus a detrimental overheads.

Problem Formulation

Transitioning from a single-layer execution model to multi-layer fusion is not a simple concatenation of operations; it fundamentally alters the mapping problem, introduces qualitatively new points to the problem to address, new mapping decisions and exponentially expands the map space.

Dimension proliferation and Map Space Explosion. A single 2-D convolution has $|D| = 6$ loop dimensions. When F layers are fused into a unified loop nest, each layer contributes its own set of dimensions. The total dimension count grows as:

$$|D_{\text{fused}}| = \underbrace{4F}_{\substack{\text{per-layer:}\\ C_i,R_i,S_i,Z_i}} + \underbrace{2(F-1)}_{\substack{\text{intermediate}\\ \text{spatial: } Y_i,X_i}} + \underbrace{2}_{\substack{\text{output}\\ P,Q}} = 6F.$$

For the 10-layer FSRCNN fusion (10 fused + 1 output) used in validation, this yields $|D_{\text{fused}}| = 66$, an order-of-magnitude increase.
The new map-space cardinality is the product of loop permutations at every memory level and factor allocations across all hierarchy levels:

$$|\mathcal{MS}| = \underbrace{(|D|!)^{L_m}}_{\text{loop permutations}} \cdot \underbrace{\prod_{d\in D}\prod_{p|d}\binom{v_p(d)+L-1}{L-1}}_{\text{factor allocations}},$$

where L_m is the number of memory levels and L the total number of levels (memory + spatial).

Table 1: Map-space comparison.

Quantity	**Single**	**11-layer**
Dimensions ($\|D\|$)	6	66
Memory levels (L_m)	5	5
Total levels (L)	7	7
Permutation term	$\sim 10^{11}$	$\sim 10^{460}$
Factor-allocation term	$\sim 10^{13}$–10^{20}	$\sim 10^{55}$
Total map-space	$\sim 10^{24}$–10^{32}	$> 10^{515}$

The fused map space exceeds 10^{515}, over 400 orders of magnitude larger than the single-layer case.

Cascading inter-layer tile-size dependencies. Fusion introduces strict producer–consumer constraints. Selecting a two dimensional output spatial tile of size $P \times Q$ for the final layer mathematically forces every preceding layer to compute a progressively larger tile to account for overlapping filter halo. Accounting for strides along the spatial dimensions, this constraint propagates backward according to:

Layer intermediate footprint:
$$\big(\text{stride}_h \cdot (P-1) + R\big) \times \big(\text{stride}_w \cdot (Q-1) + S\big)$$

Eliminating more DRAM traffic but progressively tighten the on-chip memory constraints for every fused layer.
This backward propagation dictates that inter-layer tile sizes are *strictly dependent*. A tiling choice made for the outermost layer rigidly constrains the required intermediate tile sizes at every preceding level of the computation.
A 1-D output layer tile of size P forces every layer i to produce a progressively larger tile to account for the cumulative filter halo:

$$\text{Layer } i: \quad \Big(P + \textstyle\sum_{j=i+1}^{F-1}(R_j - 1)\Big) \qquad (1)$$

The Need for Per-Layer Mapping and Flexible Execution Granularity. A naive strategy for transitioning to multi-layer fusion mapping representation, is mathematically compose the layers by completely merging their loop indices into a monolithic mathematical formula. However, this over-simplification eliminates the

explicit representation of the intermediate feature maps, introducing sever architectural modeling limitations:

- **Loss of Per-Layer Mapping Freedom:** DNN workloads frequently compose layers with different topologies, dictating different ideal dataflows in the optimal mapping. Forcing all fused layers to share the exact same loop tiling and ordering destroys the ability to express heterogeneous dataflows.
- **Back-to-Back Execution:** Collapsing intermediate variables forces the operations into a strict back-to-back schedule. Because no loop variable can advance the producer without advancing the consumer, the producer executes a MAC operation to generate the output that the consumer must immediately use as input, eliminating opportunities for temporal buffering.

Therefore, a robust formulation of the fused-layer design space must mathematically decouple the iteration spaces of the fused layers. It must explicitly represent intermediate dimensions to allow for distinct, per-layer mapping choices.

Related Works

Layer-by-layer dataflow exploration. With the advent of SAs, analytical modeling frameworks and automated mappers have been developed to navigate the single-layer map space. Timeloop[2] introduced a nested-loop representation of workloads and a mapper–model paradigm, achieving rapid energy and throughput estimates. ZigZag[4] unifies tiling and loop ordering, enabling pruning heuristics that heavily reduce the search space, at the risk of eliminating optimal solutions. FactorFlow [5] further improves analytical efficiency by modeling spatial fanout levels explicitly, and inverting the traditional mapper paradigm to achieve up to 182× faster runtimes for convolution workloads. Its lightweight evaluation engine makes it the natural baseline for this study, where the exponential map-space growth of fusion demands minimal per-evaluation overhead.

Fused-layer dataflow exploration. DeFiNES [4] extends ZigZag to cross-layer schedules, formalizing the depth-first design space along tile size, overlap storing mode, and fuse depth. However, DeFiNES restricts cross-layer tiling to spatial dimensions, prohibiting channel tiling. By supporting channel tiling, this study explores a richer design space.
LoopTree [2] proposes an extensive design space and a highly accurate model for fused-layer dataflows. Yet it inherits Timeloop's exhaustive mapping strategies, which struggle to scale in a fused-layer paradigm. Furthermore, LoopTree adopts a tree-based representation that departs from the nested-loop abstraction, whereas the framework we proposed maintains consistency with loop-nest formalism.
At the graph level, Optimus [1] determines *which* layers to fuse, it is strictly complementary to our work.

Proposed Solution

Our study develops an analytical framework for fused-layer mapping that can represent, prune, and evaluate the multi-layer map space. The framework builds upon the FactorFlow [5] factor-based abstraction and introduces three modeling concepts and a generalized per-layer cost model.

1. Aliasing: naming the intermediate dimensions. When layers are fused, each intermediate activation serves a dual role: output of layer i and input of layer $i+1$. Without explicit naming, the fused layers collapse into a rigid back-to-back scalar schedule with no per-layer mapping freedom. *Aliasing* resolves this by introducing concrete loop variables (Y_i, X_i, Z_i) for each intermediate tensor, registering them in the coupling so that the producer indexes the tensor through native dimensions while the consumer indexes it through affine dim-sums involving its own variables. The intermediate dimensions can be tiled, reordered, and parallelized independently.

2. Decoupling: per-layer mapping independence. Because aliasing assigns syntactically distinct variable names to each layer, the per-layer dimension sets become disjoint: $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$. This *decoupling* guarantees that iterating over any dimension of one layer leaves all other layers stationary. As a consequence, the unified dataflow can encode a *different stationarity* for each layer within a single

loop nest.

3. Producer–consumer feasibility pruning. While aliasing and decoupling provide mapping freedom, they introduce a risk: the loop nest no longer guarantees every intermediate datum is produced before consumption. To tame the resulting map-space explosion, our work utilizes a pruning rule to systematically eliminate infeasible mappings before cost evaluation. The validator checks constraints not at isolated memory levels, but at boundaries defined by consecutive producer dimension instances (zones), which can span multiple memory levels. It mathematically verifies that the cumulative producer tile size (X_{cum}) is large enough to cover the consumer's spatial and channel access patterns (Q_{zone} and S_{zone}) Cascading tile-size inequalities are rigorously enforced across width, height, and channel axis.

$$\begin{aligned} Q_{\text{zone}} + S_{\text{zone}} - 1 &\le X_{\text{cum}}, && \text{(Width)} \\ P_{\text{zone}} + R_{\text{zone}} - 1 &\le Y_{\text{cum}}, && \text{(Height)} \\ C_{i+1,\,\text{zone}} &\le Z_{i,\,\text{cum}}. && \text{(Channel)} \end{aligned}$$

Every candidate mapping move triggers the validator; moves that violate any constraint are rolled back immediately. The pruning is *exact*: it removes only physically infeasible mappings, preserving every valid candidate.
For instance, consider a width-axis configuration with full extents $X = 18$, $Q = 16$, $S = 3$ on a three-level hierarchy (DRAM, Global Buffer, Register). Here subscripts denote the *hierarchy level*, not the layer index. A candidate mapping tiles these dimensions as shown in Figure 1 (from outermost to innermost): $Q_1 = 4$, $X_1 = 3$, $S_1 = 1$, $Q_2 = 4$, $X_2 = 2$, $S_2 = 1$, $Q_3 = 1$, $S_3 = 3$, $X_3 = 3$. The three X loops partition the nest into three *zones*;

Table 2: All three zones pass: the mapping tried is feasible on the width axis.

Zone	Q_z	S_z	X_{cum}	Check
1 (X_3 – X_2)	1	3	3	$1+3-1 = 3 \le 3$
2 (X_2 – X_1)	4	1	6	$4+1-1 = 4 \le 6$
3 (global)	16	3	18	$16+3-1 = 18 \le 18$

4. Per-layer cost model. The single-layer cost model is generalized on a per-layer basis.

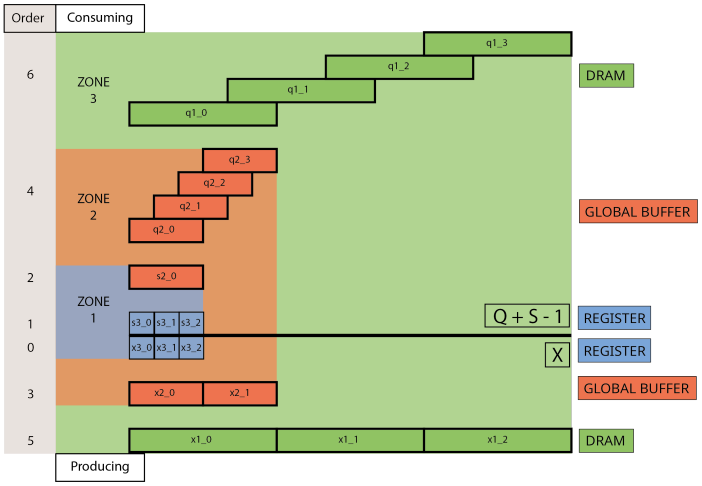

Figure 1: Zone-based pruning validation.

Reads and writes are attributed to each layer through per-layer dimension filtering. The energy is decomposed as:

$$\text{WMOPs}_i = \sum_{l \in \text{levels}} e_l \cdot \left(R_l^{(i)} + W_l^{(i)}\right),$$

where e_l is the per-access energy at level l. The latency model detects bandwidth-limited stalls per layer per level:

$$\Lambda_l^{(i)} = \max\left(\Lambda_{l,\text{rd}}^{(i)},\, \Lambda_{l,\text{wr}}^{(i)}\right), \qquad \Lambda = \max_l \sum_{i=0}^{F-1} \Lambda_l^{(i)}.$$

Validation

The framework is validated against DepFiN [3], a 12 nm depth-first chip with a 16×128 PE array, 1056 KB Feature Memory, and 524 KB Weight Memory. The validation reproduces the execution of an 11-layer CNN workload

1. **Ideal latency**: the model predicts 37.94 M cycles, matching the compute-bound reference (Total MACs/PEs) exactly.
2. **DRAM bypass**: zero intermediate DRAM traffic, reproducing DepFiN's depth-first fusion guarantee.
3. **On-chip stalls and feature shifter**: the model explicitly captures the hardware's feature shift register, demonstrating how intra-PE temporal reuse reduces the required Feature Memory (FMEM) bandwidth. The framework reveals that without this mechanism, memory-bound stalls would cause a 10% latency overhead. With the shifter active, steady-state execution achieves zero stalls.

4. **Weight Memory**: zero stall cycles, confirming sufficient bandwidth.

These results establish that our work faithfully captures the data-movement behavior of a real fused-layer chip.

Experiments

Our work is experimentally evaluated through on a systematic design-space exploration covering two SAs (DepFiN-like and Eyeriss-like), four CNN workloads (FSRCNN, MC-CNN –activation-dominant; VGG16, ResNet18 – weight-dominant), three execution modes (full fusion, partial fusion, single-layer), and two architecture-sizing scenarios (Scenario A sized for full fusion, Scenario B sized for partial fusion).

DRAM traffic elimination. Full fusion reduces total DRAM traffic by a 94.8% (FSRCNN), 85.3% (MC-CNN), 59.9% (VGG16), and 25.1% (ResNet18), confirming that fusion eliminates the dominant off-chip transfers for activation-heavy workloads.

Latency. Fusion delivers 42–48% latency savings for activation-dominant workloads on the DepFiN-like architecture and 16–46% on Eyeriss. For weight-dominant workloads, the outcome is architecture-dependent: Eyeriss still reduces latency (7.8–21.2%) because its distributed per-PE registers avoid a single shared-memory contention point, while DepFiN can be *slower* under full fusion because the very large shared weight memory becomes a bandwidth bottleneck. Partial (pairwise) fusion captures 87–103% of the full-fusion latency benefit, indicating that nearly all the inter-layer saving is realized by fusing just two consecutive layers.

Energy: workload-dependent and architecture-dependent. The energy outcome depends critically on the trade-off between DRAM cost saved and on-chip memory overhead incurred, and is consequently sensitive to the DRAM per-access energy. At the tried DRAM per-access energy of 160 pJ / byte, full fusion is *energy-beneficial* on DepFiN for activation-dominant workloads: FSRCNN single-layer energy is 4.38× the full-fusion value, and MC-CNN is 1.73×, because the eliminated DRAM traffic is massive and the feature-memory overhead modest. While for weight-dominant workloads on DepFiN, the picture reverses: VGG16 single-layer energy is only 7.4% of the full-fusion value, and ResNet18 is 17.3%, because 10–14 MB weight memories inflate per-access energy far beyond the DRAM savings. On Eyeriss, the replicated per-PE register architecture amplifies the on-chip overhead: fusion is roughly energy-neutral for FSRCNN and energy-adverse for all other workloads.

Energy–Delay Product. Full fusion wins on EDP in three cases, where the activation-dominant nature of the workload ensures that both energy and latency favor fusion, amplifying the EDP advantage. For the remaining five combinations, single-layer execution is the more efficient choice.

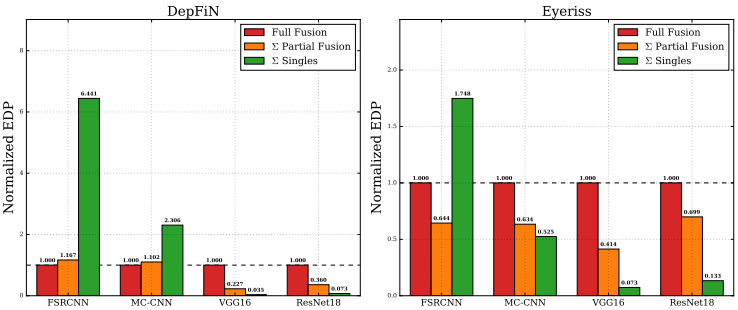

Figure 2: Scenario A: Normalized EDP (Full Fusion = 1.0). Bars below 1.0 indicate lower energy than full fusion. Values annotated above each bar.

Partial fusion as a practical compromise. Scenario B reduced the weight memory by up to 3.1× on DepFiN and the per-PE register by up to 2.1× on Eyeriss, while preserving 96–100% of the full-fusion latency saving for activation-dominant workloads. For activation-dominant workloads on DepFiN, partial fusion won on energy, latency, and EDP simultaneously, making it the strictly dominant strategy.

Super-linear scaling of the energy penalty. Shrinking the Eyeriss PE array to save area demands proportionally larger per-PE registers. This degrades the fusion-to-single energy ratio super-linearly, worsening by 1.5–1.9× for every 2× register increase.

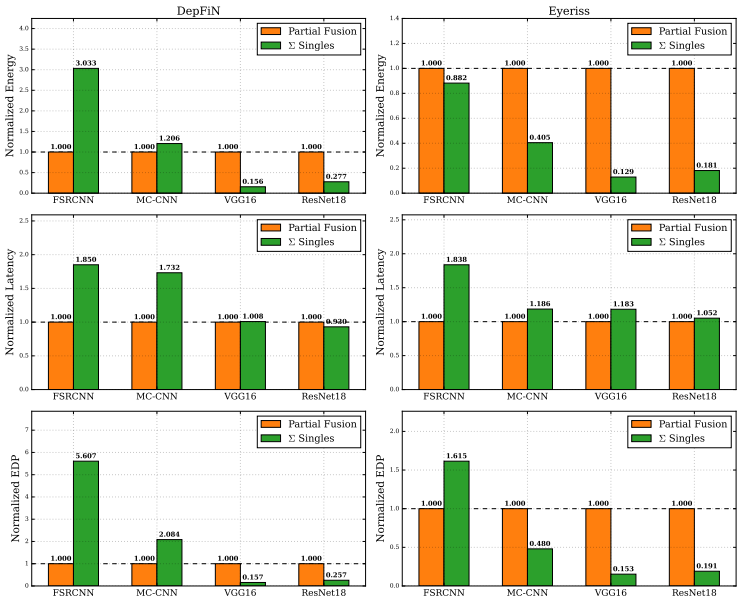

Figure 3: Scenario B: Normalized energy (top), latency (middle), and EDP (bottom), with partial fusion = 1.0. Bars below 1.0 indicate the single-layer sum is more efficient.

Conclusions

The central theoretical contribution of this thesis is demonstrating that the massive fused-layer map space ($> 10^{515}$ configurations for a 10-layer fusion), can be rigorously formalized, navigated, and evaluated within a single, unified loop-nest abstraction. The key enablers are the *aliasing* and *decoupling* mechanisms. Aliasing introduces concrete loop variables for every intermediate activation, having each shared tensor in the unified nest. Decoupling follows directly, the per-layer dimension sets become disjoint, iterating over any dimension of one layer leaves all other stationary. This property guarantees that the unified loop nest can encode a *different dataflow* for each fused layer. No prior fused-layer framework offers this combination of a coherent loop-nest representation with full per-layer mapping independence. Combined with exact producer–consumer feasibility pruning, the framework eliminates only infeasible mappings. We have validated our framework against DepFiN[3]. Our experiments clarify the fundamental trade-offs of layer fusion for the case studies tried. Layer fusion is a *workload-dependent and architecture-dependent optimization* whose energy outcome is sensitive to the DRAM energy cost. At 160 pJ / byte DRAM access energy, fusion is energy-beneficial for activation-dominant workloads on shared-memory architectures, yielding EDP improvements of up to 6.4×. Conversely, weight-dominant models favor single-layer execution due to on-chip memory overhead. Partial fusion emerges as the most practical deployment strategy: it captures nearly all full fusion latency benefit at a fraction of the memory cost.
The primary directions for future work include developing automated heuristic mappers to navigate the pruned fused map space, supporting inter-layer pipeline parallelism, and porting the framework to a compiled language to enable large-scale exploration.

Keywords: deep neural networks, spatial architectures, layer fusion, design space exploration

References

[1] X. Cai, Y. Wang, and L. Zhang. Optimus: towards optimal layer-fusion on deep learning processors. In *Proc. of LCTES'21*, pages 67–79, 2021.

[2] M. Gilbert, Y. N. Wu, J. S. Emer, and V. Sze. Looptree: Exploring the fused-layer dataflow accelerator design space. arXiv preprint, 2024.

[3] K. Goetschalckx, F. Wu, and M. Verhelst. Depfin: A 12-nm depth-first, high-resolution cnn processor for io-efficient inference. *IEEE JSSC*, 58(5):1425–1435, 2023.

[4] L. Mei, K. Goetschalckx, A. Symons, and M. Verhelst. Defines: Enabling fast exploration of the depth-first scheduling space. In *Proc. of HPCA'23*, pages 570–583, 2023.

[5] M. Ronzani and C. Silvano. Factorflow: Mapping gemms on spatial architectures through adaptive programming and greedy optimization. In *Proc. of ASPDAC'25*, pages 706–712, 2025.