Novelty Assessment Report

Paper: TileLang: Bridge Programmability and Performance in Modern Neural Kernels
PDF URL: https://openreview.net/pdf?id=Jb1WkNSfUB
Venue: ICLR 2026 Conference Submission
Year: 2026
Report Generated: 2025-12-27

Abstract

Modern AI algorithms increasingly adopt fused kernels for performance, but implementing them remains complex due to the lack of fine-grained control in existing compilers like Triton. We introduce TileLang, a controllable programming system for fused neural kernels. TileLang provides explicit tile-level primitives for memory placement, data movement, and parallel scheduling. To guide developers in hardware-aware programming, the TileLang introduces two key techniques: tile inference which models tile programs as fused graphs and automatically deduces tile configuration from partial annotations; and tile recommendation that suggests efficient tile configurations based on hardware profiles and heuristics. TileLang makes it easy to express a wide range of fused attention kernels in under 80 lines of Python code, reducing code size by up to 90% compared to manual implementations. Evaluations show that TileLang achieves up to 5x speedup over Triton on NVIDIA H100 and up to 6 on AMD GPUs, demonstrating its ability to bridge programmability and performance.

Core Task Landscape

This paper addresses: **Programmable Tile-Level System for Fused Neural Kernels**

A total of **26 papers** were analyzed and organized into a taxonomy with **13 categories**.

Taxonomy Overview

The research landscape has been organized into the following main categories:

- **Tile-Level Programming Abstractions and Compilation**
- **Hardware Architecture Design for Tile-Based Acceleration**
- **Optimization Techniques for Tile-Based Systems**

Complete Taxonomy Tree

- Programmable Tile-Level System for Fused Neural Kernels Survey Taxonomy
- Tile-Level Programming Abstractions and Compilation
  - Explicit Tile-Level Programming Languages ★ (2 papers)
  - [0] TileLang: Bridge Programmability and Performance in Modern Neural Kernels (Anon et al., 2026) View paper
  - [3] Triton: an intermediate language and compiler for tiled neural network computations (Philippe Tillet, 2019) View paper
  - Intermediate Compilation Frameworks and Overlays (2 papers)
  - [14] FlexCNN: An end-to-end framework for composing CNN accelerators on FPGA (Suhail Basalama, 2023) View paper
  - [15] DLA: Compiler and FPGA overlay for neural network inference acceleration (Abdelfattah, 2018) View paper
- Hardware Architecture Design for Tile-Based Acceleration
  - Coarse-Grained Reconfigurable Arrays (3 papers)
  - [1] Opal: A 16nm Coarse-Grained Reconfigurable Array SoC for Full Sparse Machine Learning Applications (Po-Han Chen, 2025) View paper
  - [4] Onyx: A 12nm 756 GOPS/W Coarse-Grained Reconfigurable Array for Accelerating Dense and Sparse Applications (Kalhan Koul, 2024) View paper
  - [6] Onyx: A Programmable Accelerator for Sparse Tensor Algebra (Kalhan Koul, 2024) View paper
  - Reconfigurable Processing Elements and Multicore Designs (3 papers)
  - [2] A high energy efficient reconfigurable hybrid neural network processor for deep learning applications (Shouyi Yin, 2017) View paper
  - [9] A Reconfigurable Deep Neural Network on Chip Design with Flexible Convolutional Operations (Kun-Chih Chen, 2022) View paper
  - [10] A reconfigurable neural network processor with tile-grained multicore pipeline for object detection on FPGA (Libo Chang, 2021) View paper
  - Unified Matrix and Neural Processors (2 papers)
  - [7] A unified programmable edge matrix processor for deep neural networks and matrix algebra (Biji George, 2022) View paper
  - [8] Venus: A versatile deep neural network accelerator architecture design for multiple applications (Jiaqi Yang, 2023) View paper
  - Multi-Task and Fault-Tolerant Architectures (1 papers)
  - [11] FPGA Implementation of a Fault-Tolerant Fused and Branched CNN Accelerator With Reconfigurable Capabilities (Rizwan Tariq Syed, 2024) View paper
  - Processing-in-Memory Architectures (2 papers)
  - [12] Modeling analog tile-based accelerators using sst (Benjamin Feinberg, 2022) View paper
  - [17] T-PIM: An Energy-Efficient Processing-in-Memory Accelerator for End-to-End On-Device Training (Jaehoon Heo, 2023) View paper
- Optimization Techniques for Tile-Based Systems
  - Scheduling and Mapping Strategies (2 papers)

- [19] Aero: Design space exploration framework for resource-constrained cnn mapping on tile-based accelerators (Simei Yang, 2022) View paper
- [21] IsoSched: Preemptive Tile Cascaded Scheduling of Multi-DNN via Subgraph Isomorphism (Zhao Boran, 2025) View paper
- Layer Fusion and Memory Optimization (4 papers)
- [16] A tile-based fused-layer CNN accelerator for FPGAs (Fabrizio Indirli, 2020) View paper
- [23] Memory-Efficient Feature Merging for Residual Connections with Layer-Centric Tile Fusion (Hao Zhang, 2025) View paper
- [24] A tile-based fused-layer approach to accelerate DCNNs on low-density FPGAs (Ahmet Erdem, 2019) View paper
- [25] Memory-aware fusing and tiling of neural networks for accelerated edge inference (Jackson Farley, 2022) View paper
- Machine Learning-Driven Optimization (2 papers)
- [13] A Machine Learning Approach to Optimizing CNN Deployment on Tile-Based Systems-on-Chip (Baisi, 2024) View paper
- [22] Autonomous Task-Tiling and Deep Neural Architecture Search for Intermittent Systems (Banerjee, 2024) View paper
- Specialized Convolution Acceleration (2 papers)
- [18] Accelerating deformable convolution networks with dynamic and irregular memory accesses (Cheng Chu, 2023) View paper
- [20] Energy-efficient accelerator design for deformable convolution networks (Xu Dawen, 2021) View paper
- On-Chip Learning and Adaptation (1 papers)
- [26] Hardware-Software Co-Design for On-Chip Learning in AI Systems (M. L. Varshika, 2023) View paper
- Reconfigurable Hardware Acceleration Methods (1 papers)
- [5] Investigation of reconfigurable hardware acceleration for low-power embedded neural networks (Li, 2024) View paper

Narrative

Core task: programmable tile-level system for fused neural kernels. The field centers on designing systems that partition neural computations into tiles—spatial or logical blocks—enabling efficient execution of fused operations on specialized hardware. The taxonomy reveals three main branches: Tile-Level Programming Abstractions and Compilation focuses on languages and compilers that expose tile-level parallelism to programmers, exemplified by explicit programming models like Triton Compiler[3] and TileLang Bridge[0]; Hardware Architecture Design for Tile-Based Acceleration explores reconfigurable and domain-specific architectures such as coarse-grained reconfigurable arrays (CGRAs) like Opal CGRA[1] and Onyx CGRA[4], as well as specialized accelerators like Venus Accelerator[8]; and Optimization Techniques for Tile-Based Systems addresses scheduling, memory management, and fusion strategies, including works on autonomous tiling (Autonomous Task Tiling[22]) and layer-centric fusion (Layer Centric Fusion[23]). These branches are tightly coupled: programming abstractions must map efficiently to hardware substrates, while optimization techniques bridge the gap between high-level code and low-level execution.

A particularly active line of work involves explicit tile-level languages that give developers fine-grained control over data movement and compute scheduling, balancing productivity with performance. TileLang Bridge[0] sits squarely in this space, offering a programming interface that abstracts tile-level operations while remaining close to hardware semantics. It shares conceptual ground with Triton Compiler[3], which similarly targets GPU tile programming but emphasizes ease of use for kernel fusion. In contrast, hardware-centric approaches like Opal CGRA[1] and Reconfigurable Hardware Acceleration[5] prioritize architectural flexibility and energy efficiency, often requiring more specialized compilation flows. The tension between programmer-friendly abstractions and hardware-specific optimizations remains a central theme: some works lean toward domain-specific languages with rich semantics, while others favor lower-level primitives that expose more control. TileLang Bridge[0] appears to navigate this trade-off by providing a structured yet expressive tile abstraction, positioning itself as a bridge between high-level neural frameworks and tile-based execution models.

Related Works in Same Category

The following **1 sibling papers** share the same taxonomy leaf node with the original paper:

1. Triton: an intermediate language and compiler for tiled neural network computations

Authors: Philippe Tillet, H. T. Kung, David Cox | **Year/Venue**: 2019 | **URL**: View paper

Abstract

The validation and deployment of novel research ideas in the field of Deep Learning is often limited by the availability of efficient compute kernels for certain basic primitives. In particular, operations that cannot leverage existing vendor libraries (e.g., cuBLAS, cuDNN) are at risk of facing poor device utilization unless custom implementations are written by experts â usually at the expense of portability. For this reason, the development of new programming abstractions for specifying cus...

Relationship Analysis

Both papers belong to the Explicit Tile-Level Programming Languages category, providing domain-specific languages for fine-grained control over tile configuration and memory management in neural kernel programming. They overlap in offering tile-based abstractions for GPU programming with explicit control over memory hierarchy, data movement, and parallel execution. However, the original paper (TileLang) emphasizes a unified fused tile-level dataflow graph (FTG) with tile recommendation and inference for automated optimization guidance, while the candidate paper (Triton) focuses on a C-based language with LLVM-based IR and polyhedral-inspired tile-level compiler passes, representing an earlier generation approach with less automated configuration support.

Contributions Analysis

Overall novelty summary. The paper introduces TileLang, a programmable tile-level system providing explicit primitives for memory placement, data movement, and parallel scheduling in fused neural kernels. Within the taxonomy, it resides in the 'Explicit Tile-Level Programming Languages' leaf, which contains only two papers total. This sparse population suggests the research direction—domain-specific languages offering fine-grained tile control—remains relatively underexplored compared to broader hardware architecture branches. The sibling paper (Triton Compiler) shares the goal of GPU tile programming but emphasizes ease of use over explicit control, indicating TileLang occupies a distinct niche prioritizing programmability with hardware awareness.

The taxonomy reveals TileLang sits adjacent to 'Intermediate Compilation Frameworks and Overlays,' which abstract hardware details through compiler infrastructures rather than exposing tile primitives. Neighboring branches include 'Coarse-Grained Reconfigurable Arrays' (3 papers) and 'Reconfigurable Processing Elements' (3 papers), focusing on physical architectures rather than programming abstractions. The 'Optimization Techniques' branch addresses scheduling and fusion strategies but assumes existing programming models. TileLang's explicit tile-level primitives distinguish it from compiler-only frameworks while its programmability separates it from hardware-centric designs, positioning it at the intersection of abstraction and control.

Among 20 candidates examined across three contributions, the 'Unified fused tile-level dataflow graph (FTG) representation' shows one refutable candidate from 10 examined, suggesting some overlap in graph-based modeling approaches. The 'Programmable tile-level abstractions' contribution examined 10 candidates with zero refutations, indicating potential novelty in the explicit primitive design. The 'Tile recommendation and inference framework' was not evaluated against prior work in this limited search. The modest search scope (20 papers, not exhaustive) means substantial related work may exist beyond top-K semantic matches, particularly in compiler optimization or dataflow modeling domains.

Based on the limited 20-candidate search, TileLang appears to contribute novel explicit tile primitives in a sparsely populated research direction, though the FTG representation shows some prior overlap. The analysis covers top semantic matches and immediate taxonomy neighbors but does not exhaustively survey compiler frameworks, GPU programming models, or dataflow graph literature. A broader search might reveal additional related work in tensor compiler design or hardware-software co-design that was not captured in this scope.

This paper presents **3 main contributions**, each analyzed against relevant prior work:

Contribution 1: Programmable tile-level abstractions for neural kernel development

Description: The authors introduce a tile-level programming model that provides explicit primitives for memory placement, data movement, and parallel scheduling. Unlike existing compilers that rely on opaque optimization passes, TileLang gives developers direct control over hardware resources through user-visible intrinsics for buffer allocation, data transfer orchestration, custom memory layouts, and parallelism strategies.

This contribution was assessed against **10 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

---

1. The next 700 accelerated layers: From mathematical expressions of network computation graphs to accelerated gpu kernels, automatically

URL: View paper

Brief Assessment

Accelerated GPU Kernels[33] focuses on automatic compilation from mathematical expressions to GPU kernels, not on providing explicit tile-level programming primitives for memory placement and parallel scheduling as described in the original contribution.

---

2. Training of deep learning pipelines on memory-constrained GPUs via segmented fused-tiled execution

URL: View paper

Brief Assessment

Segmented Fused Tiled[32] focuses on memory management for training large-image deep learning pipelines through tiling and checkpointing, not on providing programmable tile-level abstractions with explicit primitives for memory placement, data movement, and parallel scheduling in neural kernel development.

---

3. Aero: Design space exploration framework for resource-constrained cnn mapping on tile-based accelerators

URL: View paper

Brief Assessment

Aero Design Space[19] focuses on design space exploration for CNN mapping on tile-based accelerators with AIMC arrays, not on providing programmable tile-level abstractions for neural kernel development. The candidate addresses resource allocation and mapping strategies rather than developer-facing programming primitives for memory placement and data movement.

---

4. A Machine Learning Approach to Optimizing CNN Deployment on Tile-Based Systems-on-Chip

URL: View paper

Brief Assessment

ML CNN Deployment[13] focuses on optimizing CNN deployment on tile-based SoCs using machine learning to predict execution latency and resource allocation. It does not provide programmable abstractions for memory placement, data movement, or parallel scheduling at the tile level for neural kernel development.

---

5. Exact tile-based segmentation inference for images larger than gpu memory

URL: View paper

Brief Assessment

Tile Segmentation Inference[31] focuses on tile-based image segmentation for out-of-core inference on large images, addressing GPU memory constraints for semantic segmentation. This is fundamentally different from TileLang's tile-level programming abstractions for neural kernel development, which provide explicit control over memory placement, data movement, and parallel scheduling in AI kernel optimization.

---

6. TileLang: A Composable Tiled Programming Model for AI Systems

URL: View paper

Brief Assessment

TileLang Composable[28] focuses on a composable tiled programming model with dataflow-centric tile operators and scheduling primitives, whereas the original paper emphasizes explicit primitives for memory placement, data movement, and parallel scheduling with direct hardware control through user-visible intrinsics.

---

7. Tilelink: Generating efficient compute-communication overlapping kernels using tile-centric primitives

URL: View paper

Brief Assessment

TileLink Overlapping[29] focuses on distributed compute-communication overlapping using tile-centric primitives for inter-device coordination, not on general tile-level programming abstractions for single-device neural kernel optimization with explicit memory placement and scheduling control.

---

8. Register tiling for unstructured sparsity in neural network inference

URL: View paper

Brief Assessment

Register Tiling Sparsity[27] focuses on sparse matrix multiplication optimization through register tiling and unroll-and-sparse-jam transformations for pruned ML models. It does not address tile-level programming abstractions, memory placement primitives, or parallel scheduling controls that developers can directly manipulate—the core of the original paper's contribution.

---

9. UMDAM: A Unified Data Layout and DRAM Address Mapping for Heterogenous NPU-PIM

URL: View paper

Brief Assessment

UMDAM Layout[34] focuses on data layout and DRAM address mapping for heterogeneous NPU-PIM systems in LLM inference, not on tile-level programming abstractions for neural kernel development. The candidate addresses memory organization challenges rather than providing programmable primitives for kernel developers.

---

10. Tile-based architecture exploration for convolutional accelerators in deep neural networks

URL: View paper

Brief Assessment

Tile Architecture Exploration[30] focuses on hardware accelerator design with tile-based processing elements for DCNNs, not on programmable software abstractions for kernel development. The candidate addresses architectural exploration for hardware implementation, while the original contribution concerns software-level programming primitives for memory placement and scheduling.

Contribution 2: Unified fused tile-level dataflow graph (FTG) representation

Description: The system represents tile-level programs as a unified FTG that captures dataflow and tiling structure, where nodes represent tile operators and edges encode data dependencies. This graph-based representation enables systematic analysis and transformation at tile granularity, supporting both tile recommendation and tile inference optimization techniques.

This contribution was assessed against **10 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

---

1. Leda: Leveraging Tiling Dataflow to Accelerate SpMM on HBM-Equipped FPGAs for GNNs

URL: View paper

Brief Assessment

Leda SpMM[38] focuses on sparse matrix multiplication acceleration using tiling formats and dataflow scheduling for GNN workloads on HBM FPGAs, not on unified graph representations for tile-level kernel compilation or optimization frameworks.

---

2. Welder: Scheduling deep learning memory access via tile-graph

URL: View paper

Prior Art Analysis

Welder Scheduling[37] demonstrates that a tile-level dataflow graph representation for capturing tiling structure and data dependencies was already proposed and implemented prior to the original paper. Both systems represent computation as graphs where nodes are tile operators and edges encode data dependencies, enabling systematic analysis at tile granularity. The candidate paper explicitly describes 'tile-graph, an abstraction that facilitates fine-grained data management at tile level' and states that 'each node in the graph processes one data tile of a tensor at a time,' which directly parallels the original paper's FTG concept.

Evidence

Evidence 1 - **Rationale**: Both papers introduce a tile-level graph abstraction as their core contribution. Welder's 'tile-graph' predates TileLang's 'FTG' and serves the same fundamental purpose of representing computation at tile granularity. - **Original**: using a unified fused tile-level dataflow graph (ftg), tilelangstreamlines kernel development by unifying tile recommendation, which guides developers with hardware-aware defaults, and tile inference, which automates completion through constraint propagation. - **Candidate**: in this paper, we introduce welder , a deep learning compiler that optimizes the execution efficiency from a holistic memory access perspective. the core of welder is tile-graph, an abstraction that facilitates fine-grained data management at tile level.

Evidence 2 - **Rationale**: Both systems define computation at tile granularity and use this representation for optimization. Welder's operator-tile concept directly corresponds to TileLang's tile operators in the FTG. - **Original**: under programmable tile abstractions, tilelangprograms can be represented as a unified fused tile-level dataflow graph (ftg). by operating on this ftg, tilelangenables fine-grained reasoning and optimization of ai kernels - **Candidate**: welder defines dnn computation in a fined-grained task granularity named operator-tile. a dnn operator, such as convolution, can be implemented as multiple homogeneous operator-tiles, which are executed either in a streaming or parallel manner to compute all the data tiles in the output tensors

---

3. Optimizing openvx graphs for data movement

URL: View paper

Brief Assessment

OpenVX Optimization[36] focuses on optimizing existing OpenVX dataflow graphs through node fusion and tile size selection for embedded processors, not on introducing a unified tile-level representation with explicit tile operators and tiling structure as first-class IR constructs for kernel compilation.

---

4. Sparsepipe: Sparse Inter-operator Dataflow Architecture with Cross-Iteration Reuse

URL: View paper

Brief Assessment

SparsePipe Dataflow[35] focuses on sparse tensor algebra with inter-operator dataflow for graph processing and scientific computing, not tile-level kernel compilation or tiling structure representation for neural network operators.

---

5. Kitsune: Enabling Dataflow Execution on GPUs with Spatial Pipelines

URL: View paper

Brief Assessment

Kitsune Spatial Pipelines[41] focuses on dataflow execution and spatial pipelines on GPUs, not on tile-level dataflow graph representations for kernel compilation. The candidate's context mentions compiler optimization and software queues but does not describe a unified graph representation for tile operators and tiling structure.

---

6. FlatAttention: Dataflow and Fabric Collectives Co-Optimization for Efficient Multi-Head Attention on Tile-Based Many-PE Accelerators

URL: View paper

Brief Assessment

FlatAttention Collectives[43] focuses on dataflow optimization for multi-head attention on tile-based accelerators using network-on-chip collectives, not on a unified graph representation for general tile-level programs with systematic transformation capabilities.

---

7. Rethinking Tiling and Dataflow for SpMM Acceleration: A Graph Transformation Framework

URL: View paper

Brief Assessment

Graph Transformation SpMM[39] focuses on SpMM (Sparse Matrix-Matrix Multiplication) acceleration through graph transformations for tiling and dataflow optimization. The candidate paper's full text is not available for comparison, preventing assessment of whether it addresses tile-level dataflow graphs in the same context as TileLang's unified FTG representation for neural kernel compilation.

---

8. Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures

URL: View paper

Brief Assessment

Stateful Dataflow Multigraphs[44] operates at a different abstraction level, focusing on data-centric intermediate representation for general program transformations across diverse hardware, rather than specifically representing tile-level operators and tiling structure for kernel compilation as in the original paper's FTG.

---

9. A Unified Synthesis Framework for Dataflow Accelerators Through Multi-level Software and Hardware Intermediate Representations

URL: View paper

Brief Assessment

Unified Synthesis Framework[42] focuses on multi-level intermediate representations for dataflow accelerators and hardware synthesis, not on tile-level programming abstractions for neural kernel compilation. The contexts are fundamentally different.

---

10. PIMapping: A tile-level dataflow optimization framework for PIM-architecture

URL: View paper

Brief Assessment

PIMapping Dataflow[40] focuses on mapping dataflow graphs to PIM architectures for memory-bound operations, while the original paper's FTG represents tile-level programs for GPU kernel optimization with explicit tile operators and data dependencies.

Contribution 3: Tile recommendation and inference framework

Description: The authors develop a two-stage optimization workflow: tile recommendation analyzes the FTG to provide hardware-aware defaults for tile shapes, memory placement, and warp partitions, while tile inference propagates constraints through the graph to automatically complete remaining configurations including memory layouts, software pipelining, and tensorization. This design blends flexible user control with automated optimization.

This contribution was assessed against **0 related papers** from the literature. Papers with potential prior art are analyzed in detail with textual evidence; others receive brief assessments.

Appendix: Text Similarity Detection

No high-similarity text segments were detected across any compared papers.

References

- [0] TileLang: Bridge Programmability and Performance in Modern Neural Kernels View paper
- [1] Opal: A 16nm Coarse-Grained Reconfigurable Array SoC for Full Sparse Machine Learning Applications View paper
- [2] A high energy efficient reconfigurable hybrid neural network processor for deep learning applications View paper
- [3] Triton: an intermediate language and compiler for tiled neural network computations View paper
- [4] Onyx: A 12nm 756 GOPS/W Coarse-Grained Reconfigurable Array for Accelerating Dense and Sparse Applications View paper
- [5] Investigation of reconfigurable hardware acceleration for low-power embedded neural networks View paper
- [6] Onyx: A Programmable Accelerator for Sparse Tensor Algebra View paper
- [7] A unified programmable edge matrix processor for deep neural networks and matrix algebra View paper
- [8] Venus: A versatile deep neural network accelerator architecture design for multiple applications View paper
- [9] A Reconfigurable Deep Neural Network on Chip Design with Flexible Convolutional Operations View paper
- [10] A reconfigurable neural network processor with tile-grained multicore pipeline for object detection on FPGA View paper
- [11] FPGA Implementation of a Fault-Tolerant Fused and Branched CNN Accelerator With Reconfigurable Capabilities View paper
- [12] Modeling analog tile-based accelerators using sst View paper
- [13] A Machine Learning Approach to Optimizing CNN Deployment on Tile-Based Systems-on-Chip View paper
- [14] FlexCNN: An end-to-end framework for composing CNN accelerators on FPGA View paper
- [15] DLA: Compiler and FPGA overlay for neural network inference acceleration View paper
- [16] A tile-based fused-layer CNN accelerator for FPGAs View paper
- [17] T-PIM: An Energy-Efficient Processing-in-Memory Accelerator for End-to-End On-Device Training View paper
- [18] Accelerating deformable convolution networks with dynamic and irregular memory accesses View paper
- [19] Aero: Design space exploration framework for resource-constrained cnn mapping on tile-based accelerators View paper
- [20] Energy-efficient accelerator design for deformable convolution networks View paper
- [21] IsoSched: Preemptive Tile Cascaded Scheduling of Multi-DNN via Subgraph Isomorphism View paper
- [22] Autonomous Task-Tiling and Deep Neural Architecture Search for Intermittent Systems View paper
- [23] Memory-Efficient Feature Merging for Residual Connections with Layer-Centric Tile Fusion View paper
- [24] A tile-based fused-layer approach to accelerate DCNNs on low-density FPGAs View paper
- [25] Memory-aware fusing and tiling of neural networks for accelerated edge inference View paper
- [26] Hardware-Software Co-Design for On-Chip Learning in AI Systems View paper
- [27] Register tiling for unstructured sparsity in neural network inference View paper
- [28] TileLang: A Composable Tiled Programming Model for AI Systems View paper
- [29] Tilelink: Generating efficient compute-communication overlapping kernels using tile-centric primitives View paper
- [30] Tile-based architecture exploration for convolutional accelerators in deep neural networks View paper
- [31] Exact tile-based segmentation inference for images larger than gpu memory View paper
- [32] Training of deep learning pipelines on memory-constrained GPUs via segmented fused-tiled execution View paper
- [33] The next 700 accelerated layers: From mathematical expressions of network computation graphs to accelerated gpu kernels, automatically View paper
- [34] UMDAM: A Unified Data Layout and DRAM Address Mapping for Heterogenous NPU-PIM View paper
- [35] Sparsepipe: Sparse Inter-operator Dataflow Architecture with Cross-Iteration Reuse View paper

- [36] Optimizing openvx graphs for data movement View paper
- [37] Welder: Scheduling deep learning memory access via tile-graph View paper
- [38] Leda: Leveraging Tiling Dataflow to Accelerate SpMM on HBM-Equipped FPGAs for GNNs View paper
- [39] Rethinking Tiling and Dataflow for SpMM Acceleration: A Graph Transformation Framework View paper
- [40] PIMapping: A tile-level dataflow optimization framework for PIM-architecture View paper
- [41] Kitsune: Enabling Dataflow Execution on GPUs with Spatial Pipelines View paper
- [42] A Unified Synthesis Framework for Dataflow Accelerators Through Multi-level Software and Hardware Intermediate Representations View paper
- [43] FlatAttention: Dataflow and Fabric Collectives Co-Optimization for Efficient Multi-Head Attention on Tile-Based Many-PE Accelerators View paper
- [44] Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures View paper