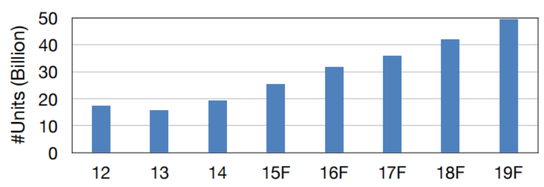
MoteNN: Memory Optimization via Fine-grained Scheduling for Deep Neural Networks on Tiny Devices

Renze Chen[1], Zijian Ding[2], Size Zheng[1], Meng Li[1], Yun Liang[1] [1]PKU [2]UCLA

1. Motivation

1.1. The Era of AIoT on Tiny Devices

- Low-power
- Low-cost
- Rapid Growth
- Wide Application



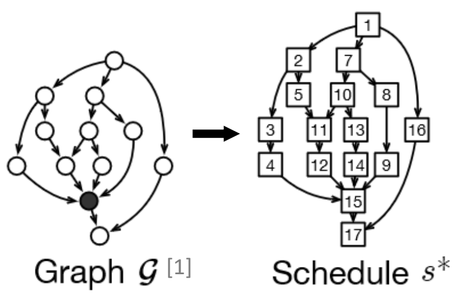
1.2. Challenge: Small Memory Capacity

	Cloud AI	Mobile AI	Tiny AI
Memory (Act)	40GB	16GB	512KB
Memory (Wgt)	40GB	16GB	/
Storage (Wgt)	~TB/PB	512GB	~MB

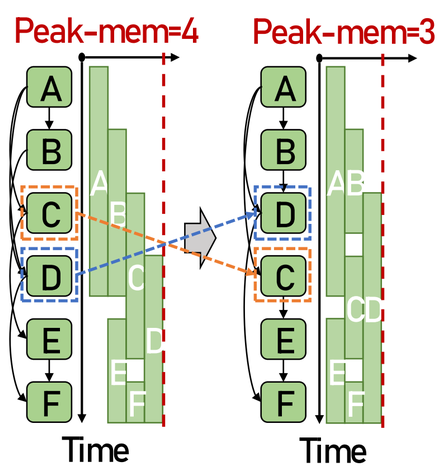
32768x smaller

81920x smaller

1.3. Scheduling for Memory Optimization

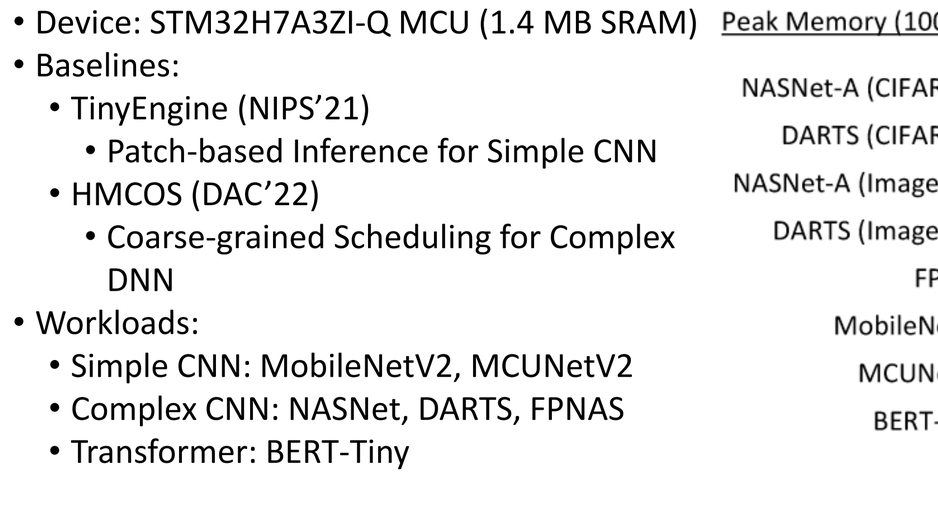


[1] Ordering Chaos: Memory-Aware Scheduling of Irregularly Wired Neural Networks for Edge Devices



- Works like Serenity and HMCOS schedule the operator execution order to manage tensor lifetimes to optimize peak memory usage of activation.

1.4. Fine-grained Scheduling can be Better



(a) Peak Memory: 768 **(b)** Peak Memory: 384

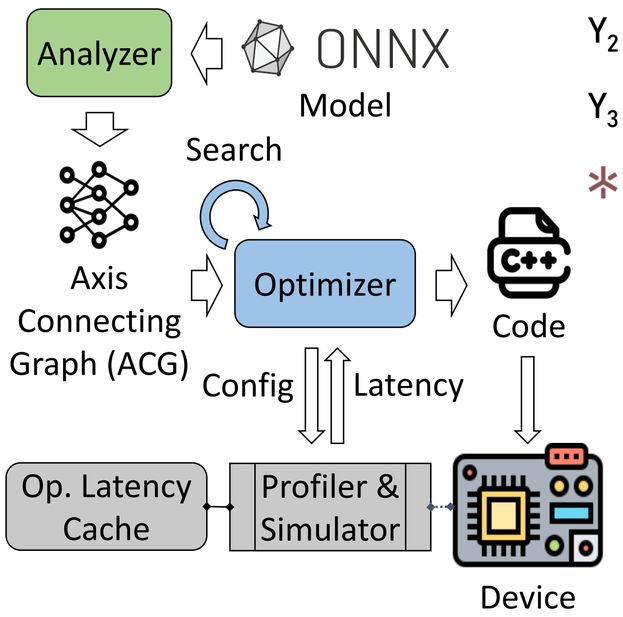
- We find that make the graph finer-grained by partitioning some operators into smaller ones and schedule the fine-grained graph can further reduce the memory usage.

2. Techniques

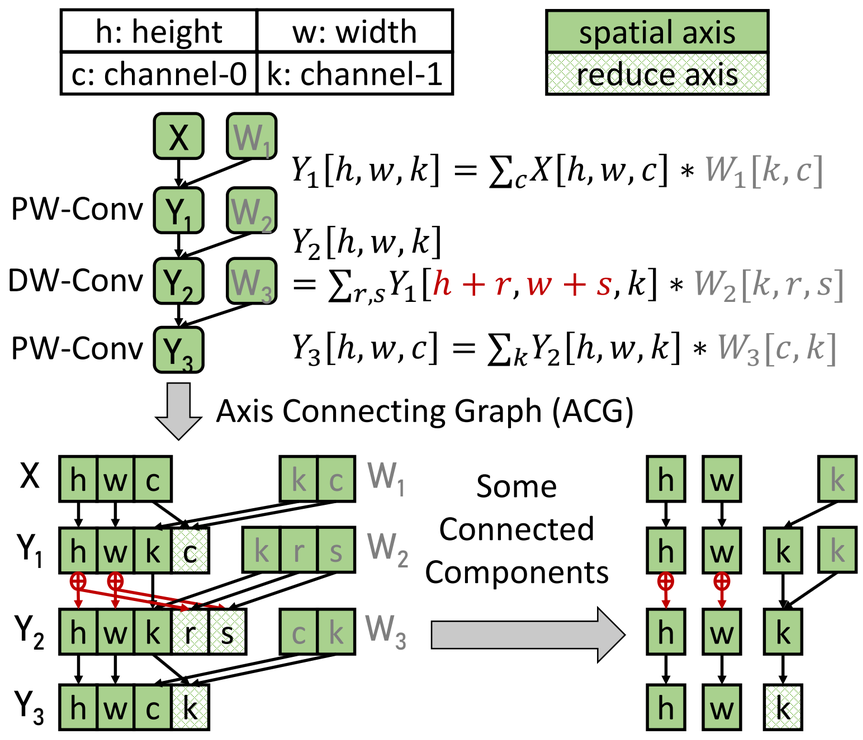
2.1. Basic Insights

- It is inefficient to consider the operator partition separately for each operator
  - We need to consider the partition and scheduling of operators at graph level.
- It is important to focus on memory bottlenecks for graph partitioning.
  - Only partitioning memory bottlenecks tensors can optimize peak memory.

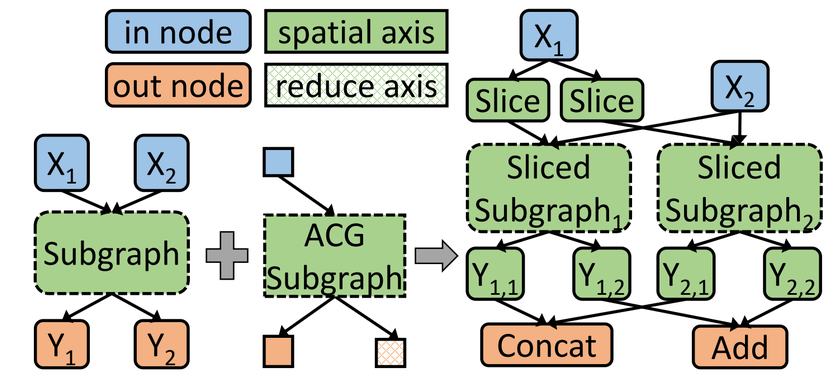
2.2. Workflow Overview



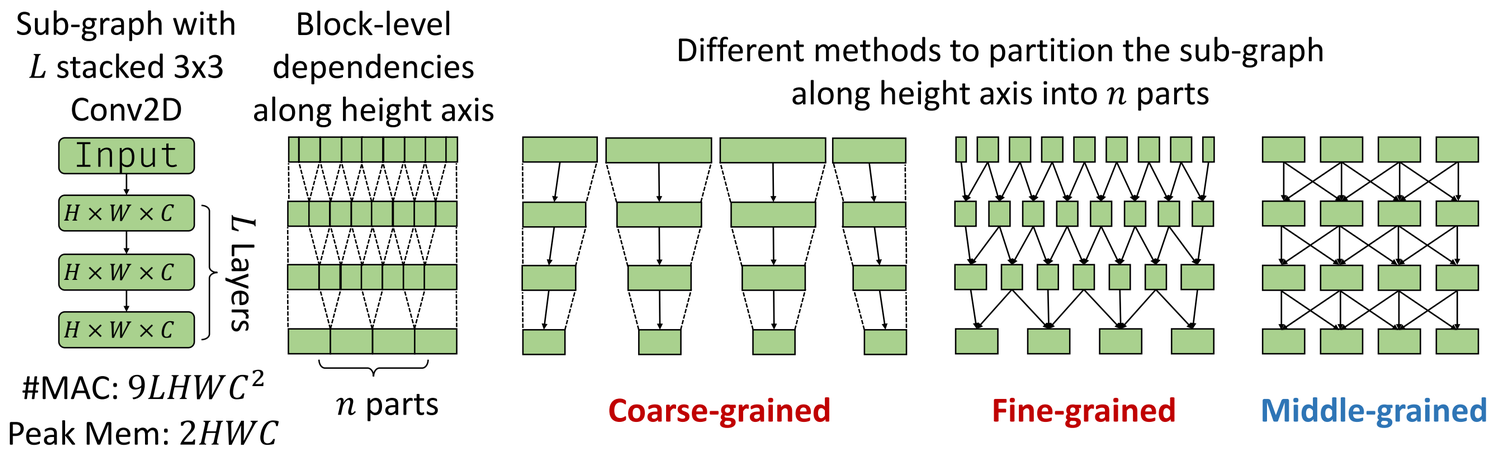
2.3. Axis Connecting Graph (ACG)



2.4. Graph Partitioning with ACG



2.5. Handling Overlapped Sliding-Window



	Coarse-grained	Fine-grained	Middle-grained
Extra #MAC Ratio	$n(L-1)/H$	≈ 0	≈ 0
Peak Mem Ratio	$((1+2/n)H + 2(2L-1))/(2H)$	$((1+L/n)H + L(L-1))/(2H)$	$(n+L^2)/(2n)$
#Kernel-Launch	nL	$\approx \min\{n(n-1)(2^L-1), HL\}$	nL

2.6. Optimization Algorithm

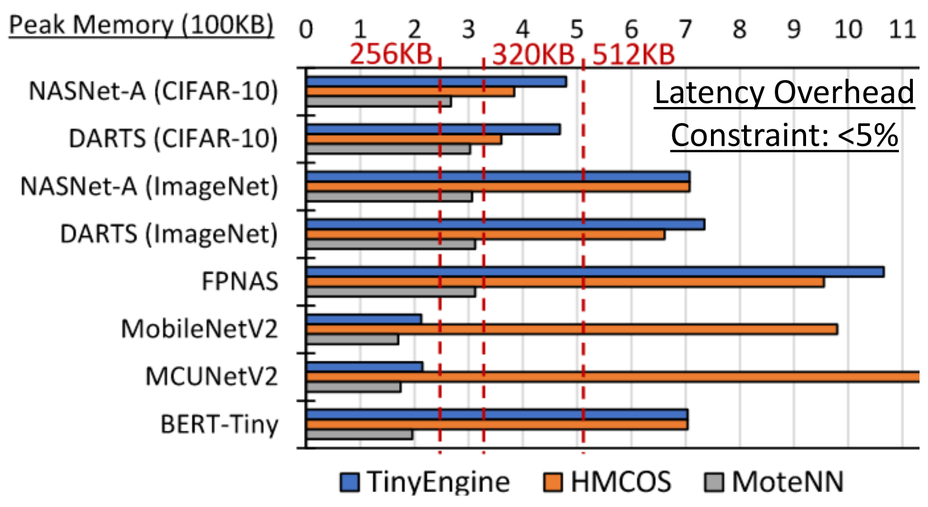
① Accept an original graph and a latency constraint ratio as input, searching for fine-grained graph and schedule to achieve lowest peak memory under the given latency constraint.
② In each iteration, select the cell that contributes most to peak memory usage and the Axis Connecting Graphs containing bottlenecks within the cell.
③ Generate partition schemes to build new fine-grained graphs and measures the new graphs to save best-β ones for next iteration.
④ After several iterations within a time budget, output the fine-grained graph and schedule achieving the lowest peak memory under the given latency constraint.

3. Evaluation

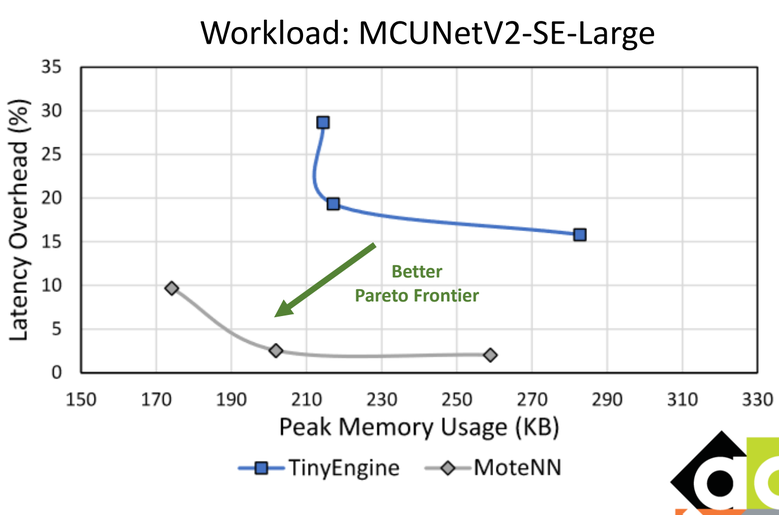
3.1. Experiment Setup

- Device: STM32H7A3ZI-Q MCU (1.4 MB SRAM)
- Baselines:
  - TinyEngine (NIPS'21)
    - Patch-based Inference for Simple CNN
  - HMCOS (DAC'22)
    - Coarse-grained Scheduling for Complex DNN
- Workloads:
  - Simple CNN: MobileNetV2, MCUNetV2
  - Complex CNN: NASNet, DARTS, FPNAS
  - Transformer: BERT-Tiny

3.2. Peak Memory Optimization



3.3. E2E Latency Overhead



THE CHIPS TO SYSTEMS CONFERENCE