A Multi-Neural Network Acceleration Architecture

2020 ISCA

Conventional Accelerator

Baseline

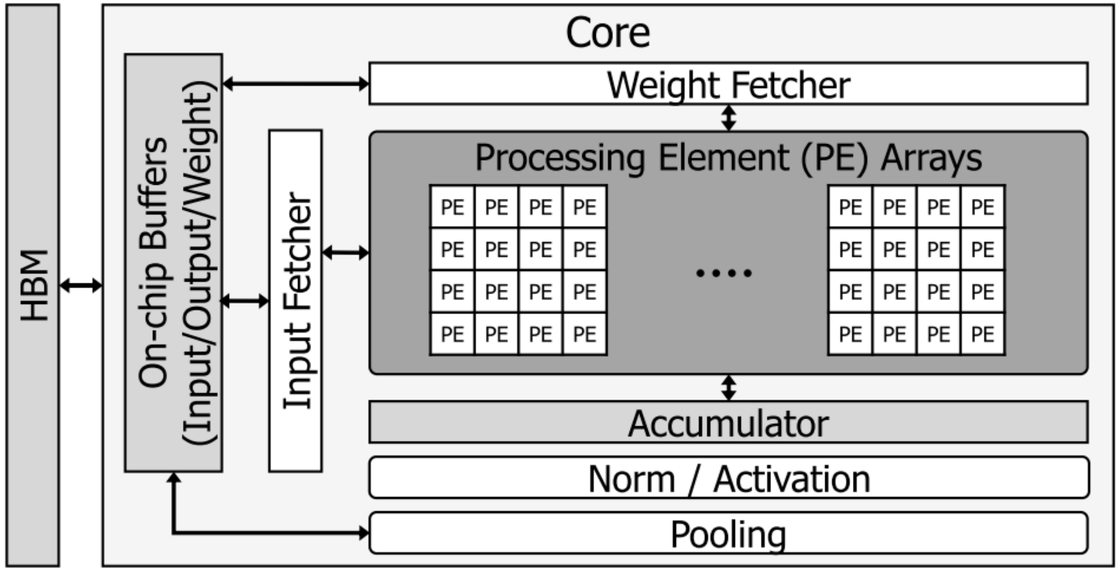


Fig. 2: Baseline neural network accelerator architecture

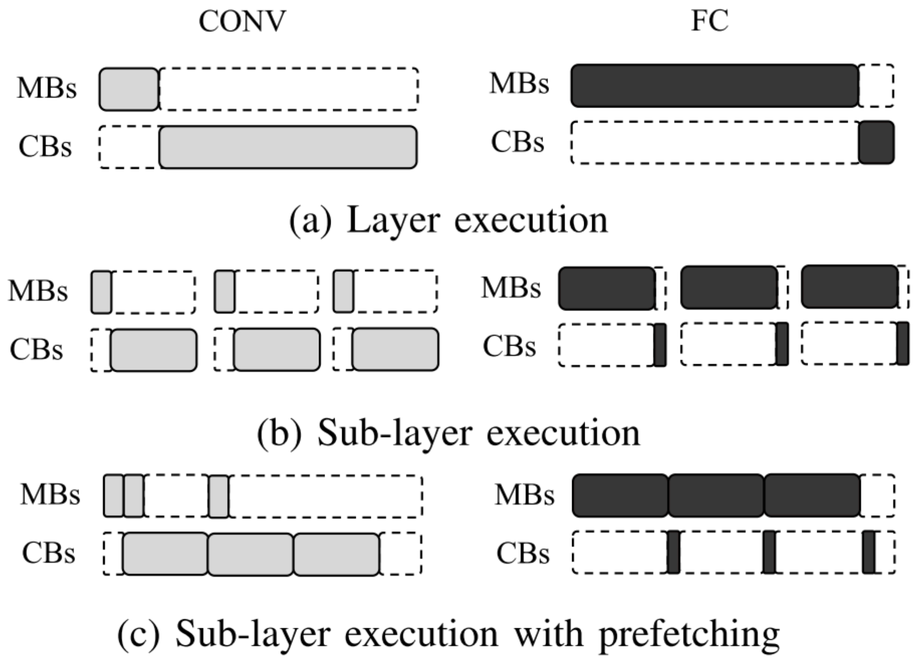


Fig. 4: Layer and sub-layer granularity executions

Motivation

1) Multi-neural network 2) Resource Idleness 3) Limited SRAM Capacity

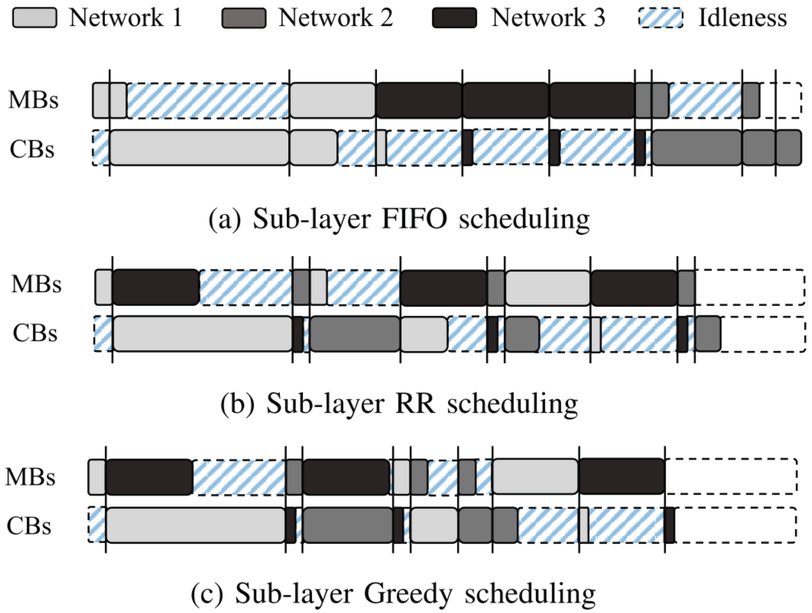


Fig. 6: Multi-neural network execution examples

Lack efficient scheduling

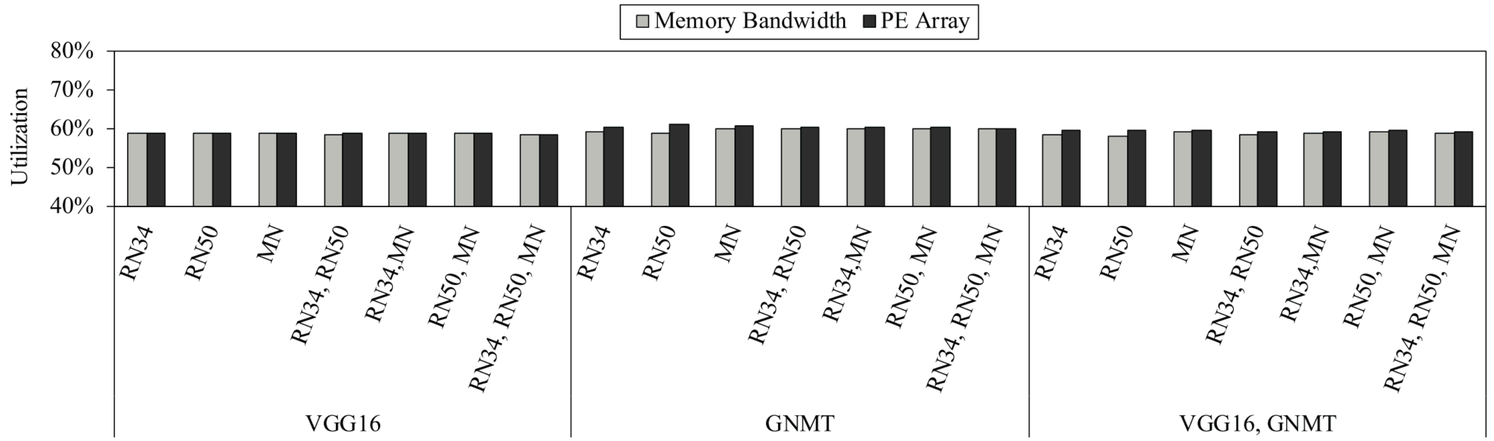


Fig. 7: Compute and memory bandwidth utilization under the round-robin scheduling algorithm (RN34: ResNet34, RN50: ResNet50, MN: MobileNet)

Resource underutilization

AI-MultiTasking Achitecture

Overview

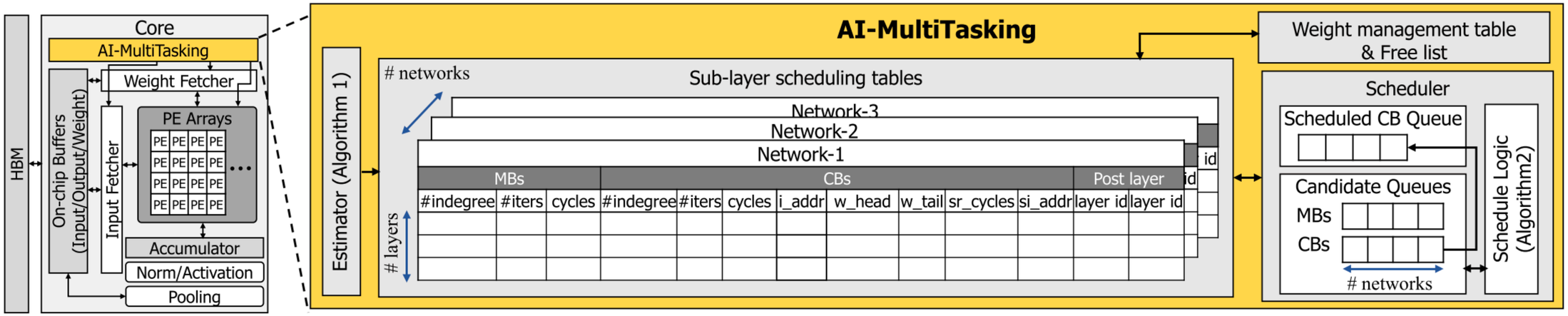


Fig. 11: Overview of AI-MT

-

- Estimator: estimates latency
- Scheduler: uses #indegree, layer_id to track dependency of layers
- Candidate Queues: uses #iters to track dependency sub-layers in the same layer
- Weight management table: w_head and w_tail indicate the start and end block id of the prefetched weight blocks for the layer’s execution

AI-MultiTasking Architecture

Load Balancing Scheduling

- Memory Block Prefetching: prefetches MBs whenever the remained SRAM capacity is enough to serve them regardless of the sub-layer boundary

- Compute Block Merging: first selects the next MB, and then schedules multiple CBs until the total cycles taken by the scheduled CBs become larger than the scheduled MB. This mechanism keeps the PE arrays busy during the selected MB's execution

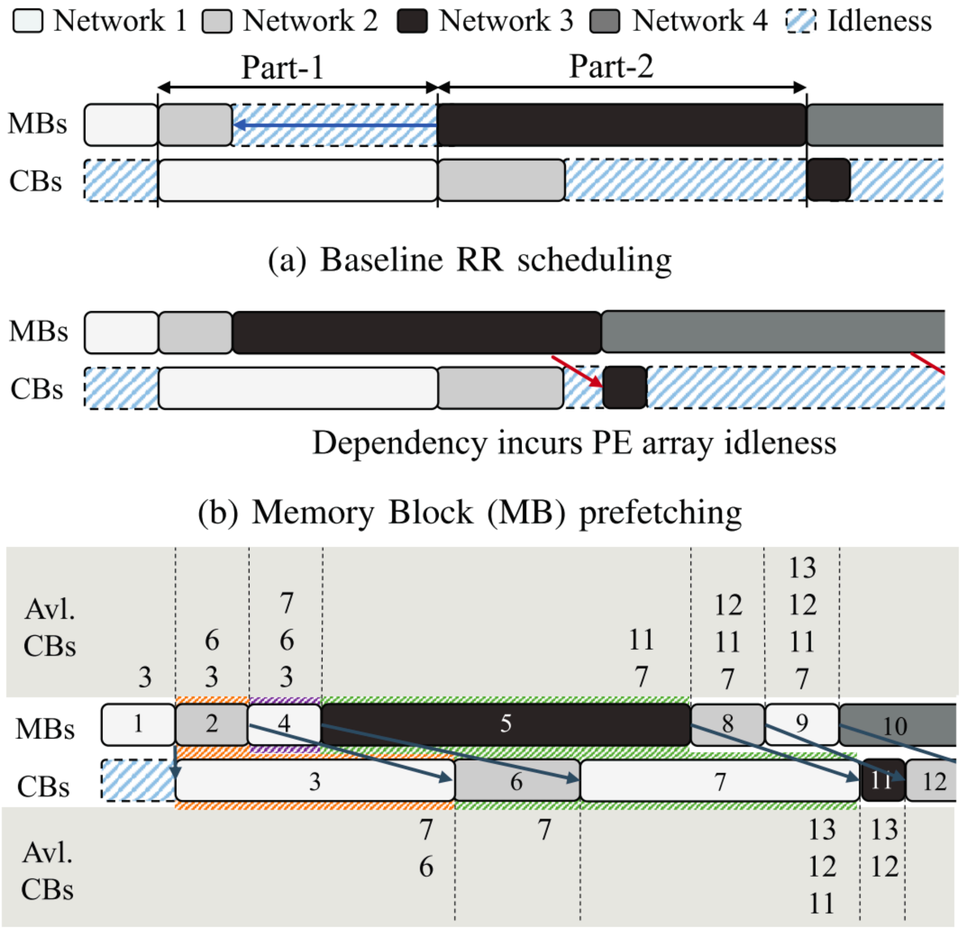


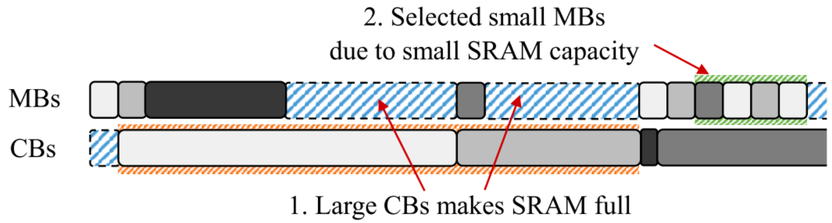
Fig. 12: Examples of load balancing scheduling

AI-MultiTasking Architecture

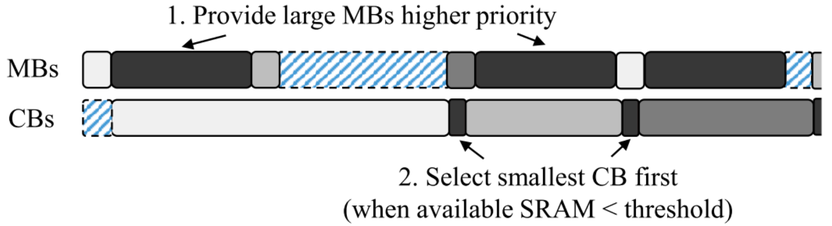
SRAM capacity aware scheduling

1 Give the high priority to the SRAM capacity-critical MBs whose cycles are larger than the cycles of the corresponding CBs.

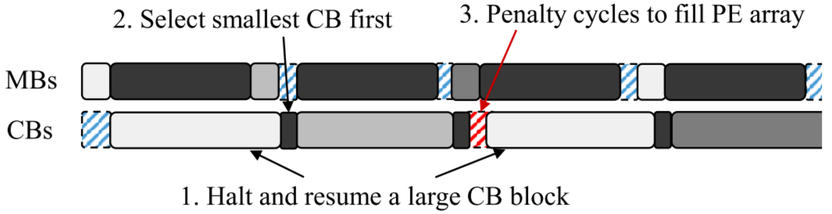
2 Early MB eviction halts the current long CB execution and schedules smaller CBs first to recover SRAM capacity quickly rather than waiting until the executing large CB finished



(a) Resource idleness under SRAM capacity shortage



(b) Priority mechanism of the early MB eviction



(c) Split mechanism of the early MB Eviction

Fig. 13: SRAM capacity aware scheduling

Evaluation

Set up

TABLE I: Hardware and architecture parameters

Parameter	Value
Processing Element Dimension	128x128
# Processing Element Array	16
Frequency	1 GHz
Memory Bandwidth	450 GB/s
On-Chip SRAM Size (Input/Output)	18 MB
On-Chip SRAM Size (Weight)	1 MB

TABLE II: Neural network workloads and their configurations

Name	Layers		Batch Size
	FC	CONV	
ResNet34	1	36	1-32
ResNet50	1	53	1-32
VGG16	3	13	1-32
MobileNet	1	27	1-32
GNMT	6		1-32

Evaluation

Speed-up

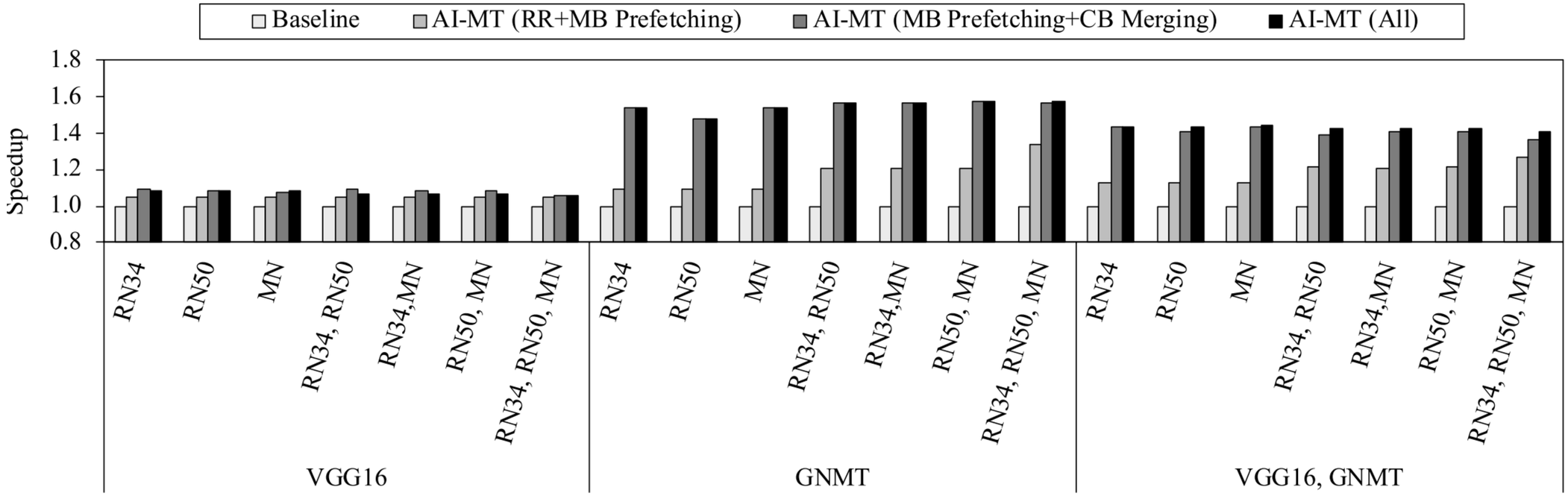


Fig. 14: Speedup of AI-MT over network-serial multi-neural network execution (Baseline)

Evaluation

Sensitivity test

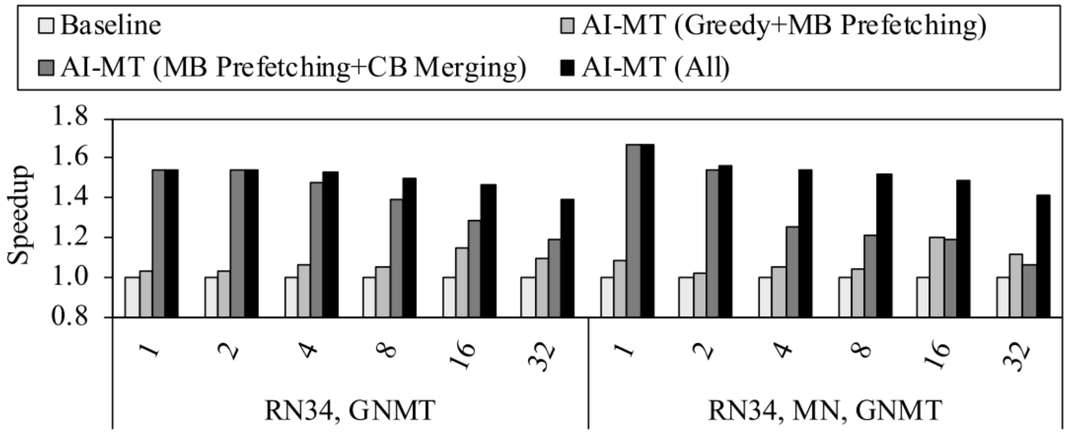


Fig. 15: Sensitivity test with different batch sizes

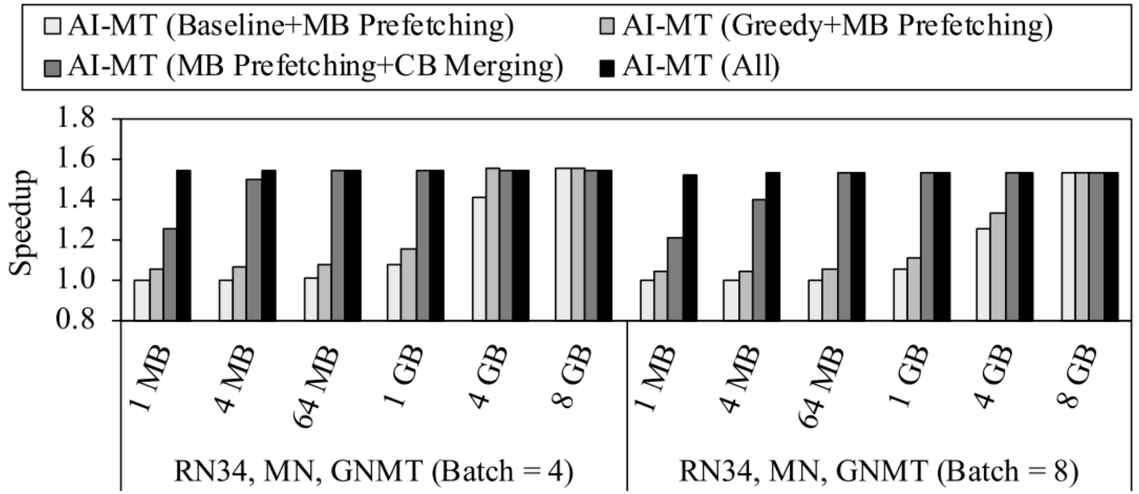


Fig. 16: Sensitivity test with different SRAM sizes

Increasing batch size makes the CBs latency longer

As it can prefetch all the weight values during a large amount of CBs, it achieves ideal performance when the SRAM capacity is infinitive.

Inspiration

- 使用了Unified Buffer，多个网络公用
- Latency 建模思想：根据指令建模

Algorithm 1: Latency estimation model

if *LayerType = CONV* **then**
  $MB.cycle = read\_cyc\_per\_array$
  $CB.cycle = \lceil \frac{ow*oh}{\#PE\_array} \rceil * batch + filling\_time$
  $\#iters = \lceil \frac{oc}{PE\_dim} \rceil * \lceil \frac{ic*k*k}{PE\_dim} \rceil$
else if *LayerType = FC* **then**
  $MB.cycle = read\_cyc\_per\_array * \#PE\_array$
  $CB.cycle = batch + filling\_time$
  $\#iters = \lceil \frac{oc}{PE\_dim*\#PE\_array} \rceil * \lceil \frac{ic*1*1}{PE\_dim} \rceil$

- Scheduling的本质是hide fetch latency和去掉了instruction synchronization