Improving Data Reuse in NPU On-chip Memory with Interleaved Gradient Order for DNN Training

Jungwoo Kim, Seonjin Na, Sanghyeon Lee, Sunho Lee, and Jaehyuk Huh

56th IEEE/ACM International Symposium on Microarchitecture (MICRO)

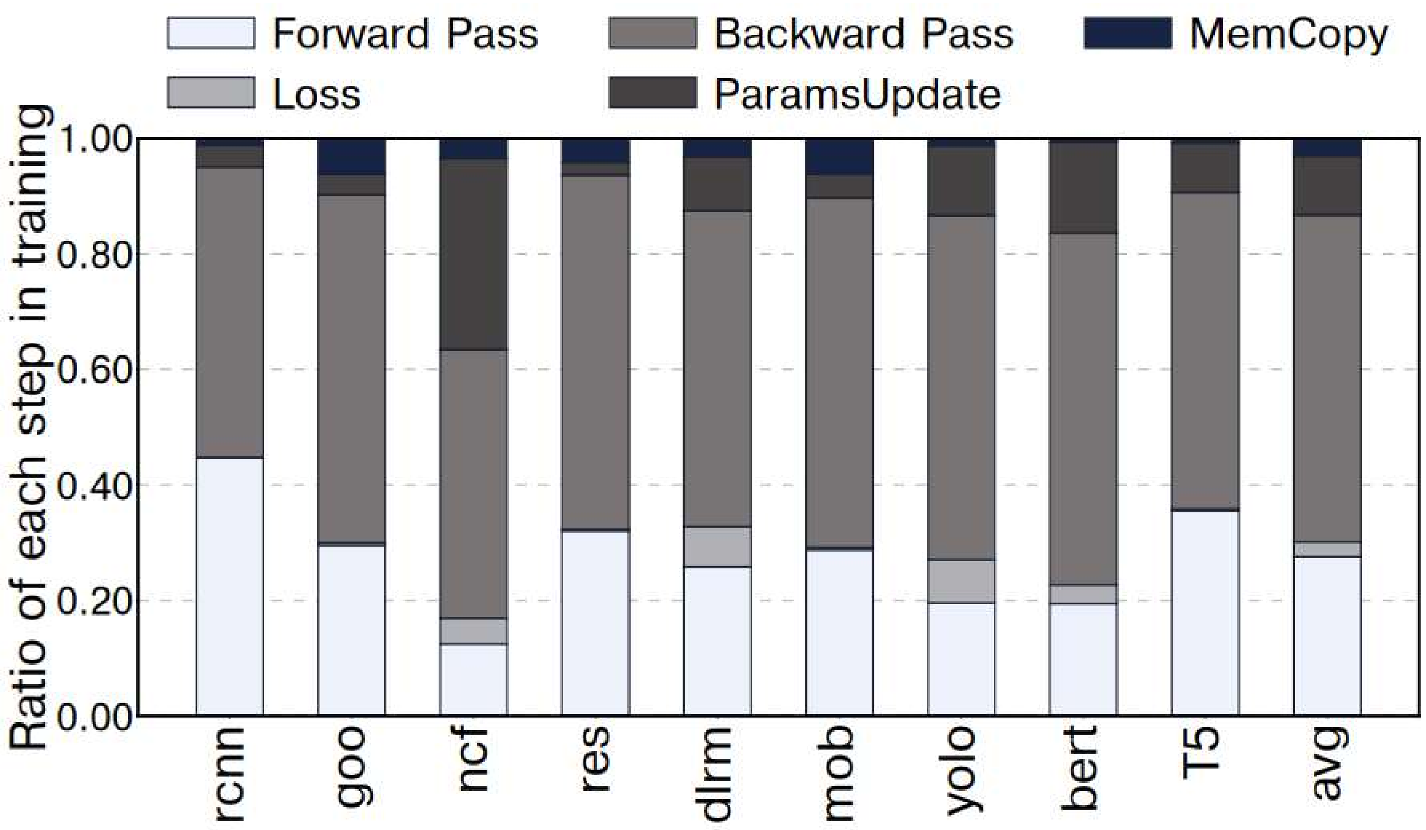
DNN Training Problem : Backward Pass!

- Computation of gradients in the backward pass accounts for majority of costs in model training.
- Backward pass has to compute input gradient & weight gradient both from output graident.
- This incurs off-chip memory accesses, the most time consuming part in training.

Prior Work

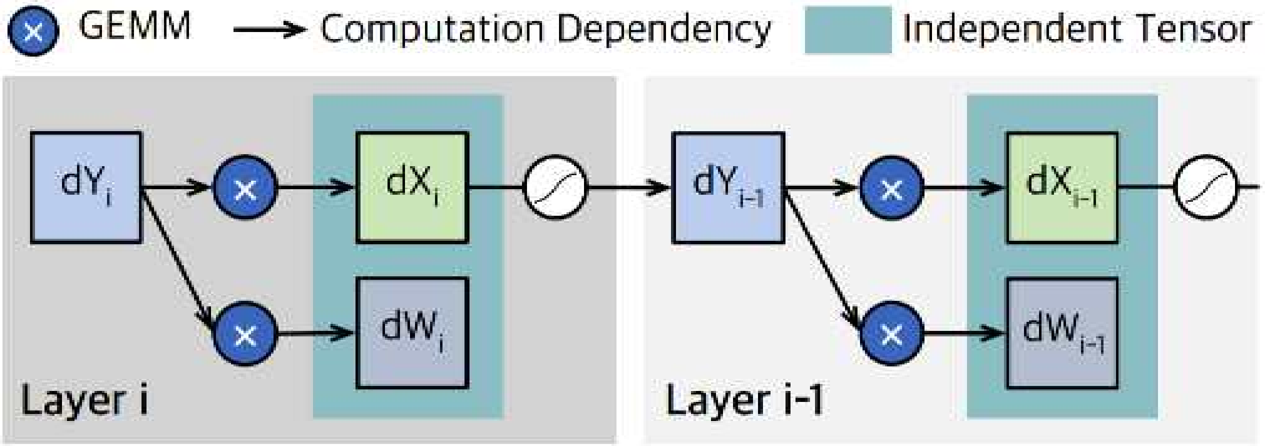
- Prior works focus on intra-operation data reuse.
- This cannot represent multiple independent operations, it can only represent single nested loop.
- Thus, they lose chance of performance improvement opportunity by data reuse.

Motivation : DNN Training Ratio



-> Backward pass occupies most of time on training process, indeed.

Motivation : Redundant Access

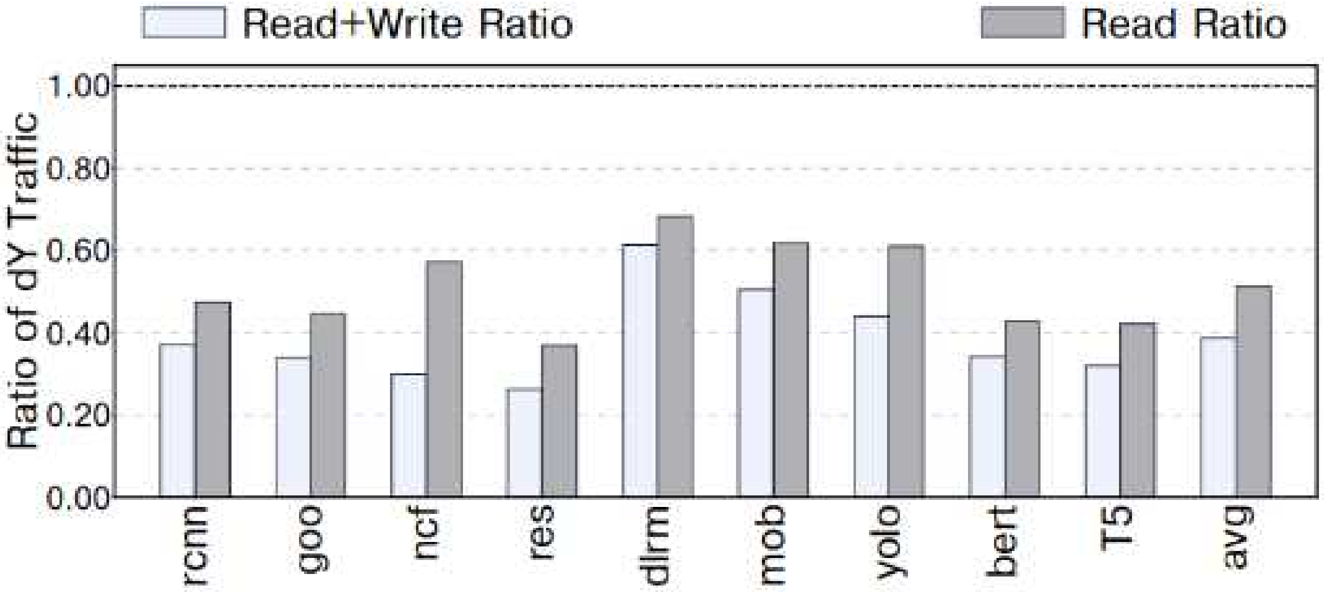


$$dX_i = \frac{\partial \mathcal{L}}{\partial X_i} = \frac{\partial \mathcal{L}}{\partial Y_i}\frac{\partial Y_i}{\partial X_i} = dY_i \times {W_i}^T$$

$$dW_i = \frac{\partial \mathcal{L}}{\partial W_i} = \frac{\partial Y_i}{\partial W_i}\frac{\partial \mathcal{L}}{\partial Y_i} = {X_i}^T \times dY_i$$

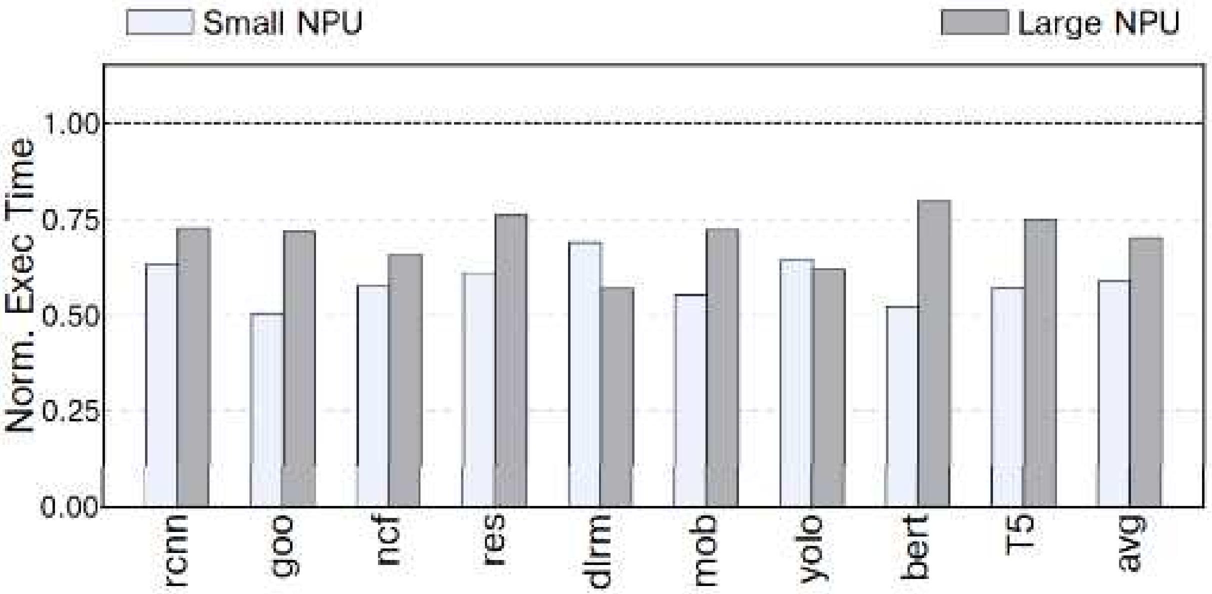
- In the figure above, dXi and dWi are computed by using the same dYi as an operand.
- As these are independent, it is possible to reorder or even fuse two operations in an interleaved manner.
- Additionally, redundant accesses to the output gradient (dYi) can be potentially reduced by such fusion.

Motivation : Data Traffic of dYi



- The ratio of dYi traffic compared to all read and write data, which is 39.0% of total data traffic on average.
- Furthermore, dYi occupies 51.4% of read data on average.
- This shows opportunities to improve performance by reducing the dYi traffic.

Motivation : Potential Performance



- How much is the performance potential of eliminating redundant reads for the output gradient?

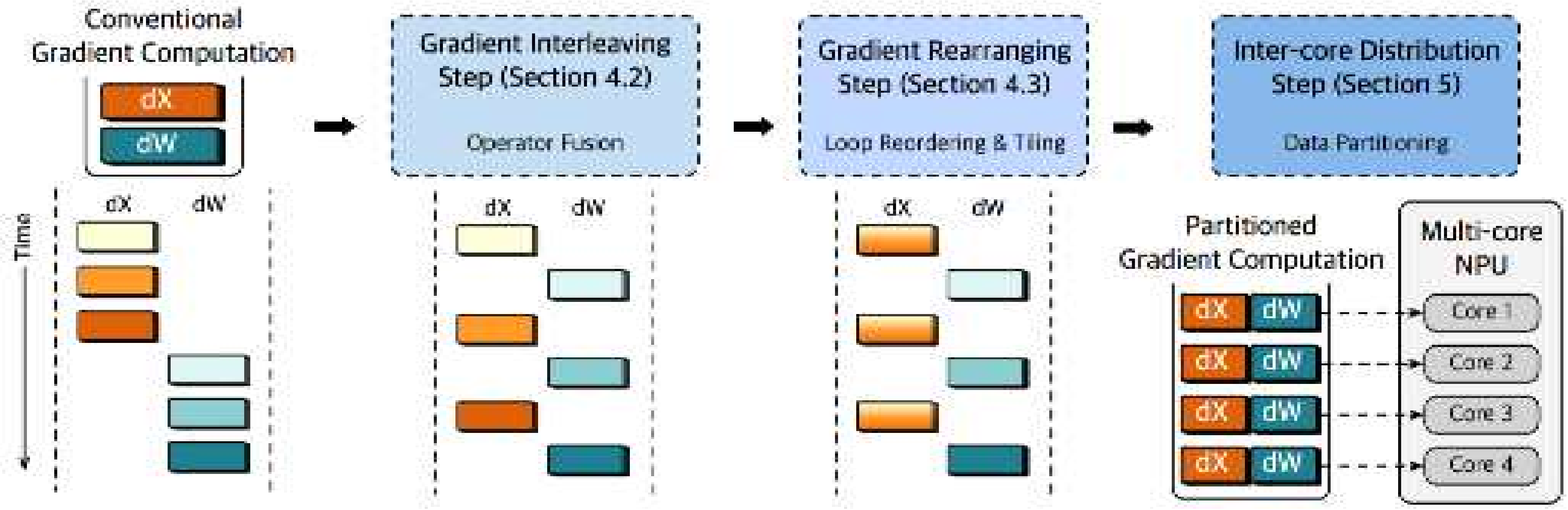
- Researchers simulated with the elimination one of two accesses for dY.

- As a result, the speed up against the baseline is 1.43x in Large NPU and 1.70x in Small NPU.

Data Reuse Opportunity

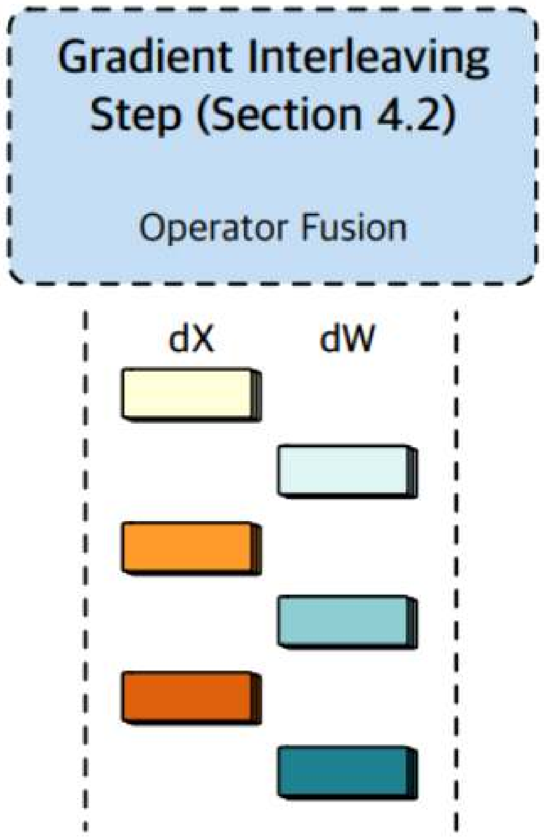
- Off-chip memory access is a problem -> we need on-chip SPM, Scratchpad Memory.
- This paper introduces a novel code transformation technique aimed at removing unnecessary memory accesses for the output gradient.
- This is called interleaved gradient order, which enables data reuse on SPM.

Overview



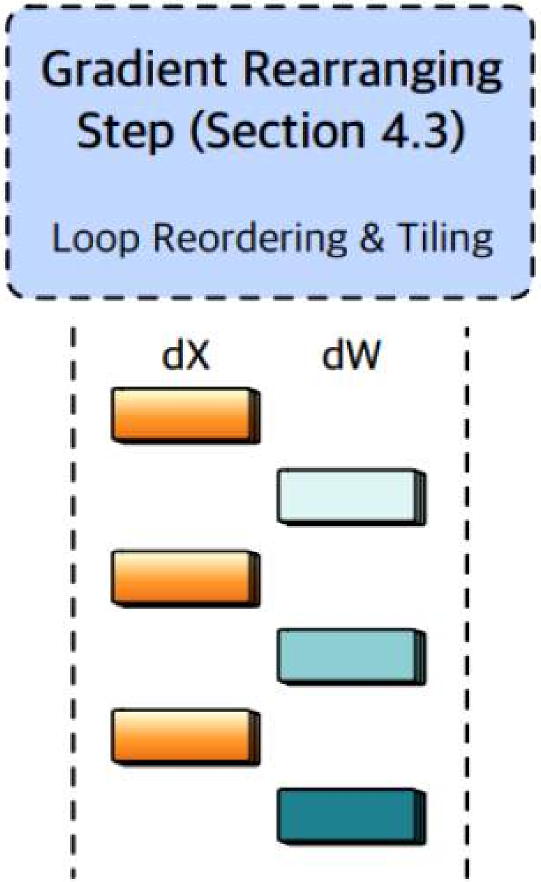
1. Gradient Interleaving Step
2. Gradient Rearranging Step
3. Inter-core Distribution Step

Overview : Gradient Interleaving Step



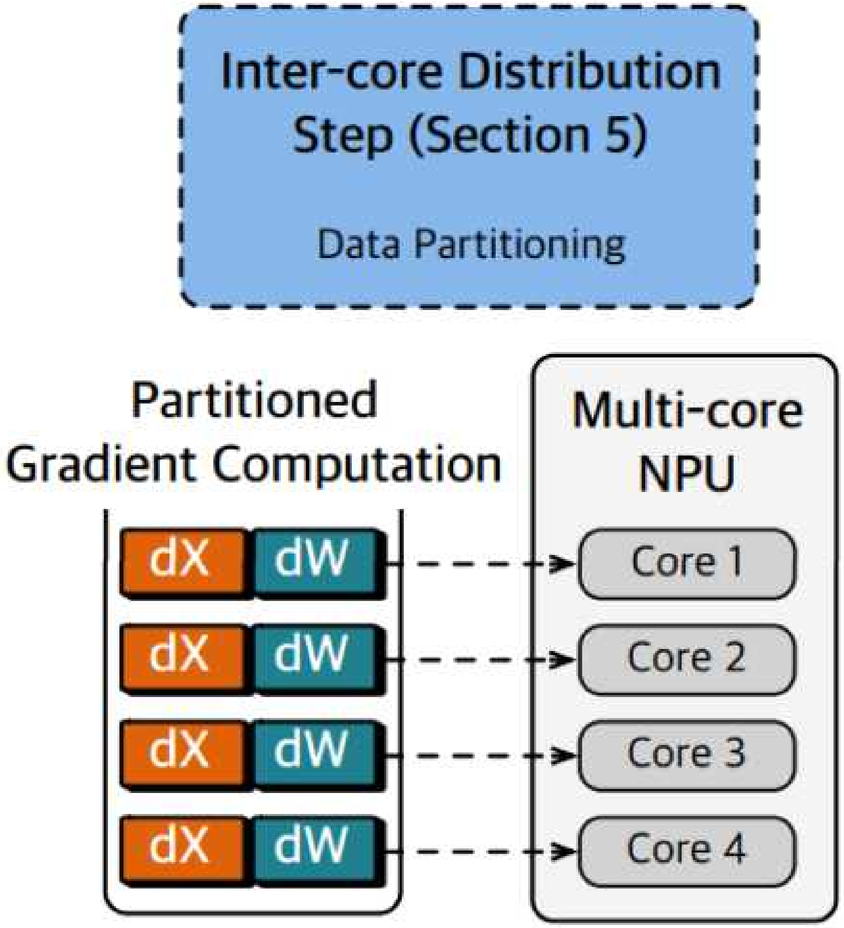
- This step combines two computations for dW and dX.
- Interleaves tiled operations for them.

Overview : Gradient Rearranging Step



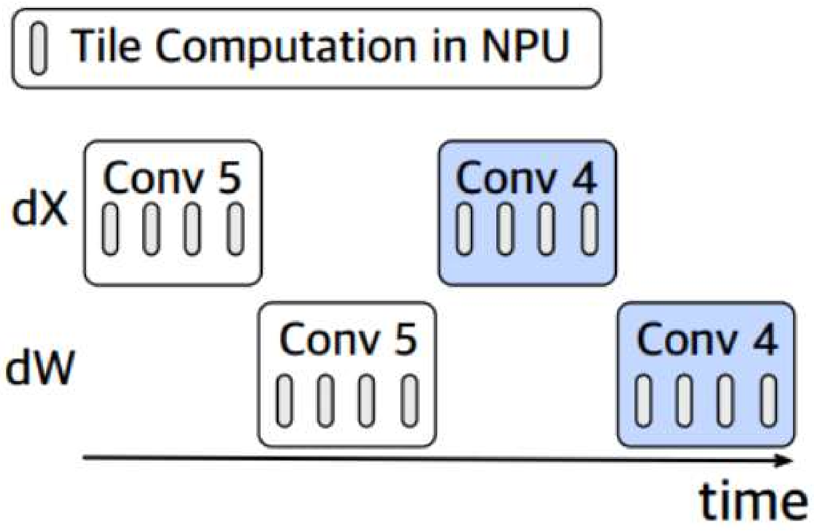
- Effectiveness of tile reuses can vary depending on the operand dimensions.
- Thus, gradient rearranging step reorganizes interleaved dX and dW computations.

Overview : Inter-core Distribution Step



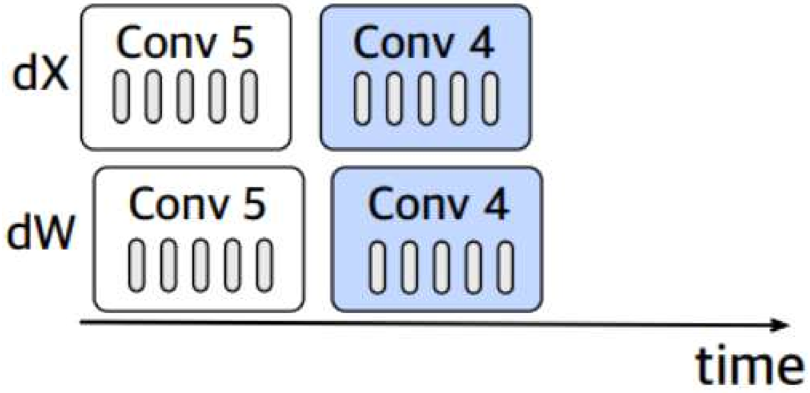
- For multi-core NPUs, operands are decomposed and assigned to different cores to maximize dY reuse.
- This step determines the optimal methods for operand decomposition.
- Finally, each segment is allocated to an NPU core.

Gradient Interleaving Step



Baseline approach

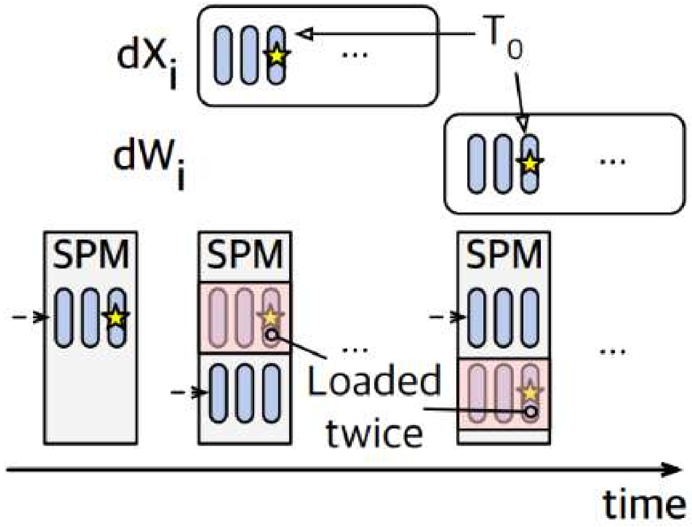
- dX and dW are calculated in a sequential manner.
- dY can be transferred to SPM twice due to prior loaded tile being evicted.



Interleaved computation

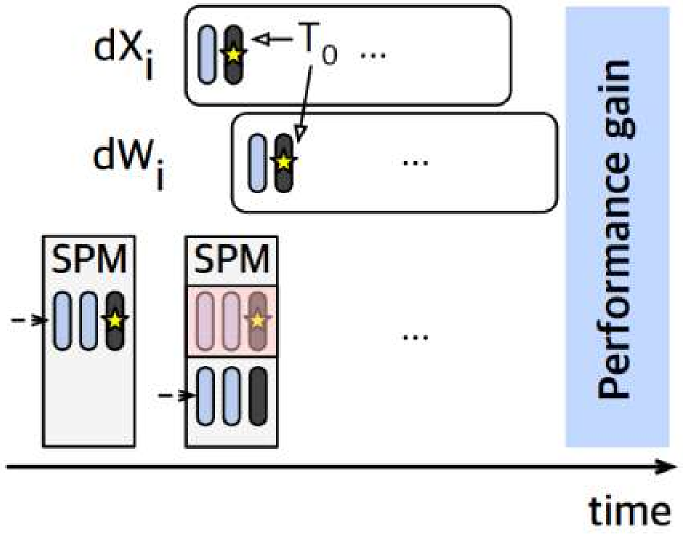
- dX and dW computations are interleaved (no dependency between them).
- Transformed code eliminates redundant accesses to dY tiles.

Gradient Interleaving Step



Baseline approach

- Yellow starred dYi loaded twice.
- Because distance between dXi and dWi calculations exceeds the capacity of SPM.



Interleaved computation

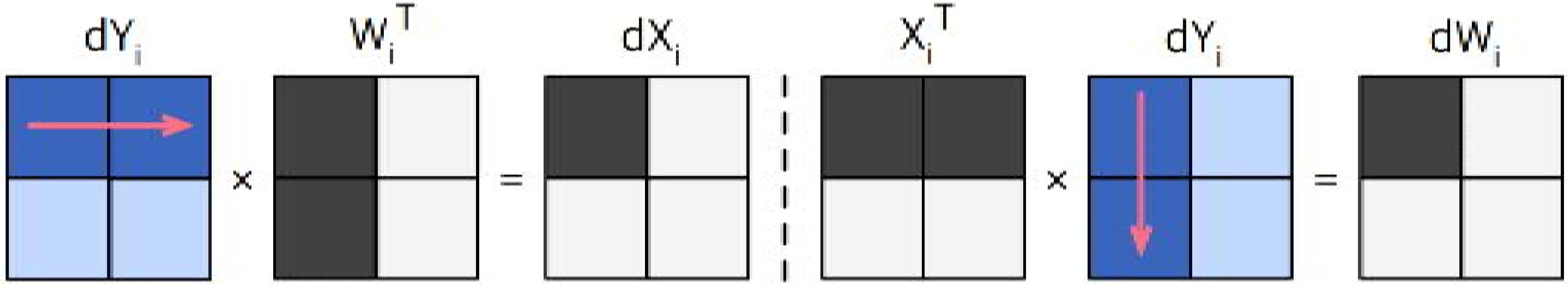
- Yellow starred dYi can be reused just one loading when interleaving is applied.
- Thus, interleaving significantly reduces data traiffc and boosts the utilization of SPM.

Gradient Interleaving Step : Problems

- Interleaving method does not significantly improve performance in certain layers that contain non-square tensors.

- Why? The access patterns for dY_i differ between dX_i and dW_i computations.

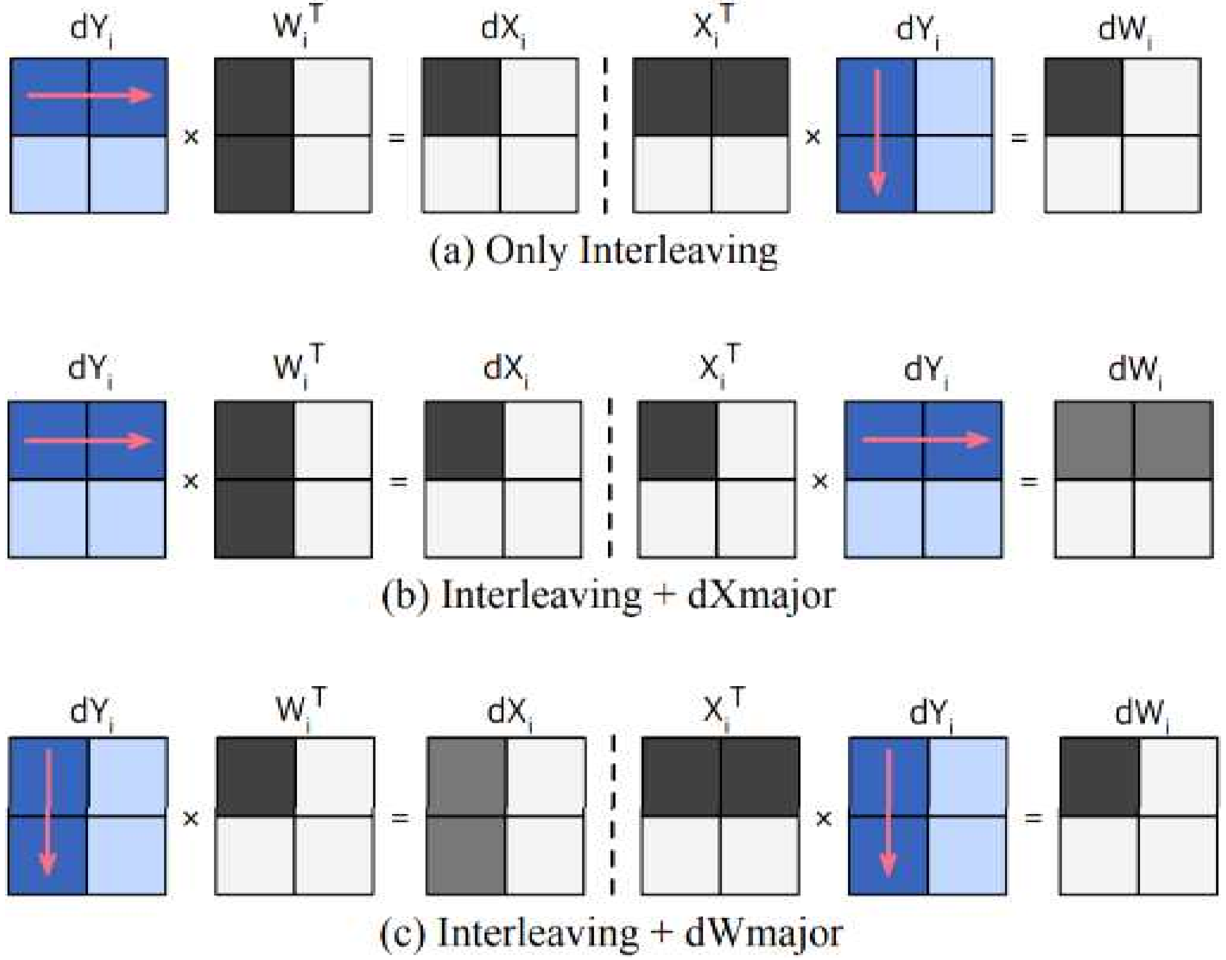
Gradient Interleaving Step : Problems



- When computing dXi, the access of dYi follows a row-major access order.

- When computing dWi, dYi is accessed in a column-major order.

Three Groups of Memory Access Orders



(a) Only Interleaving

(b) Interleaving + dXmajor

(c) Interleaving + dWmajor

- Traditional access order
- row-major access for both dXi and dWi
- column-major access for both dXi and dWi

Selection Algorithm

+ Data reuse of dY_i can be enhanced by appropriately altering the memory access order.

– Intermediate results can be generated in dWmajor and dXmajor due to the changed computation order.

-> Straightforward but fairly accurate algorithm is needed.

Selection Algorithm

GEMM in forward pass is $X_i(M, K) \times W_i(K, N) \rightarrow Y_i(M, N)$
if *AlmostSquareComputation()* **then**
    Use *Interleaving*
else if $K > N$ ***and*** $K > M$ **then**
    Use *Interleaving+dWmajor*
else
    Use *Interleaving+dXmajor*

- Algorithm selects appropriate memory access order based on the shape of tensors.
- Nearly square tensors by *AlmostSquareComputation()* -> *Only Interleaving*.
- Non-square tensor
  - Column dim of W_i > Row dim of W_i -> *Interleaving* + *dWmajor*
  - Otherwise -> *Interleaving* + *dXmajor*

Gradient Rearranging Step

- Gradient rearranging step is combination of:

1. Interleaved gradient computations

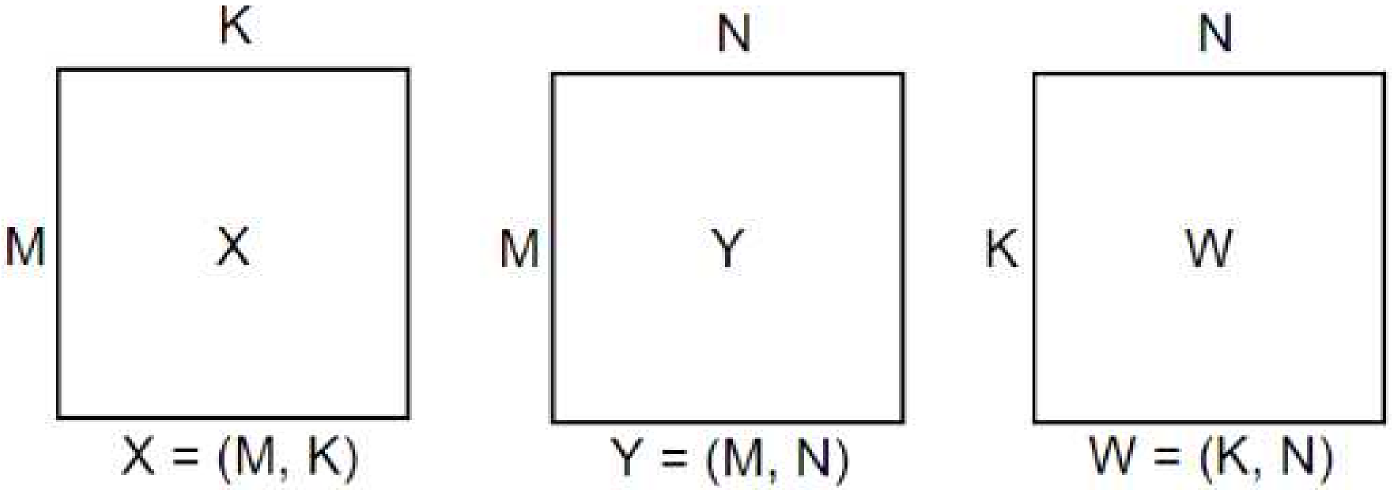
2. Selection of optimal tile access order

Data Partitioning

- Single GEMM divided into smaller partitioned GEMMs.
- Performance depends on the dimensions of the GEMM.
- Thus, we need new data partitioning schemes to better suit interleaving and rearrangement steps.

Data Partitioning

Three Tensors:
- X
- W
- Y



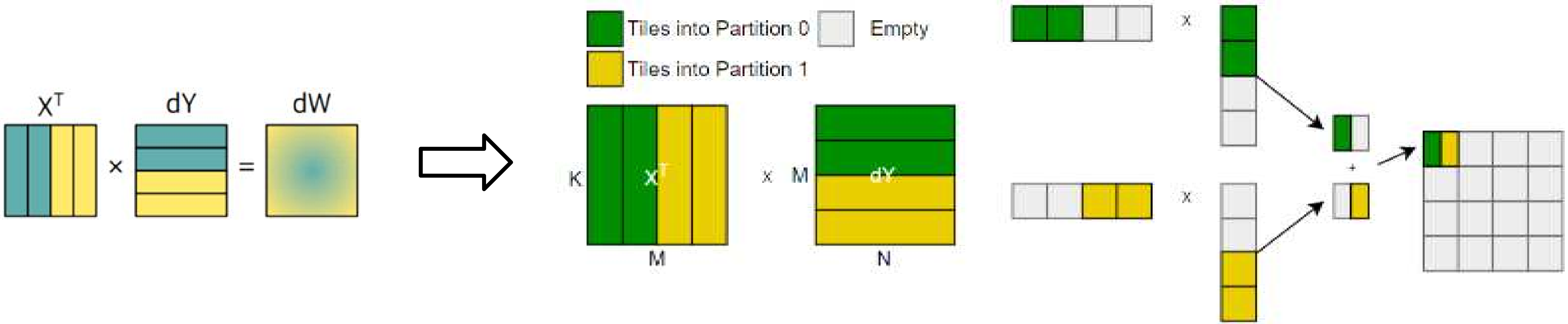
- It is common for GEMM to be partitioned on a batch basis (dimension M).

- dY and X are split for computing dX and dW.

Data Partitioning : Problem

- Partitioned on a batch basis has a problem.
  - dW: dY being right-hand-side operand
  - dX: dY being left-hand-side operand

-> This prevents a single partition from performing all the necessary calculations.
-> Also it might have to accumulate intermediate results from multiple partitions.
-> If M is not significant in GEMM, this is not an efficient way.

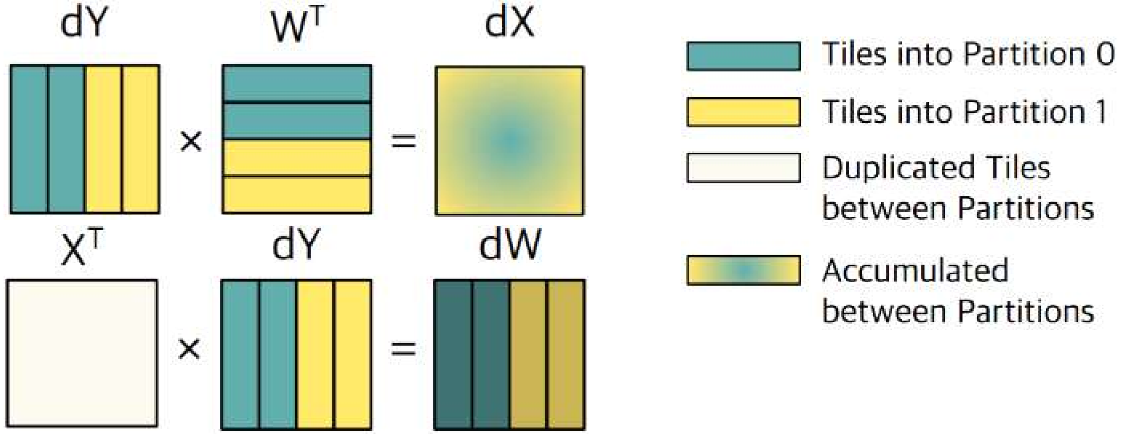


Data Partitioning in Another Dimensions

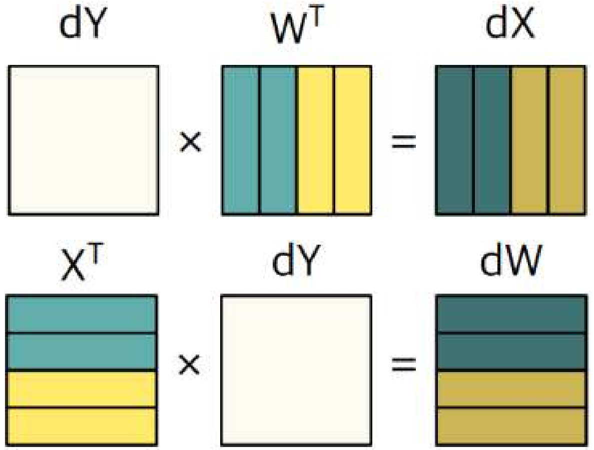
Partitioning GEMM in N or K dimension rather than M dimension could be more efficient in some layers.

Data Partitioning in Another Dimensions

- Partitioning GEMM in N
- All X is duplicated.
- Requires another accumulation



- Partitioning GEMM in K
- All dY is duplicated.
- Requires no accumulation

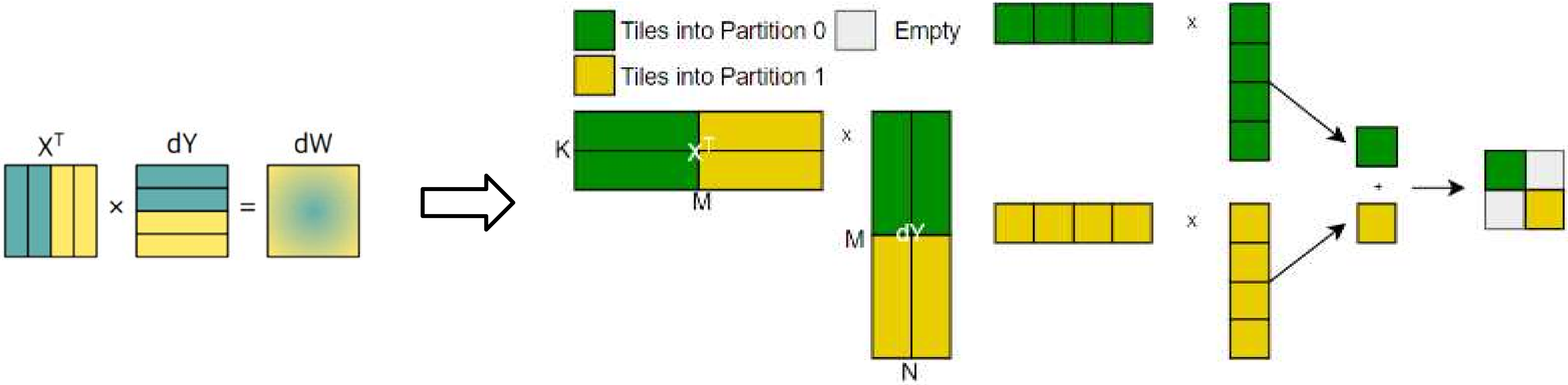


Data Partitioning : Selection

- Finally, overall performance depends on the dimensions of tensors.
- What if dimension M is siginificant rather than N and K in GEMM?

-> Systolic array can be fully utilized.

-> Because dimension M is wide enough. If it is not, other data partitioning scheme should be considered.

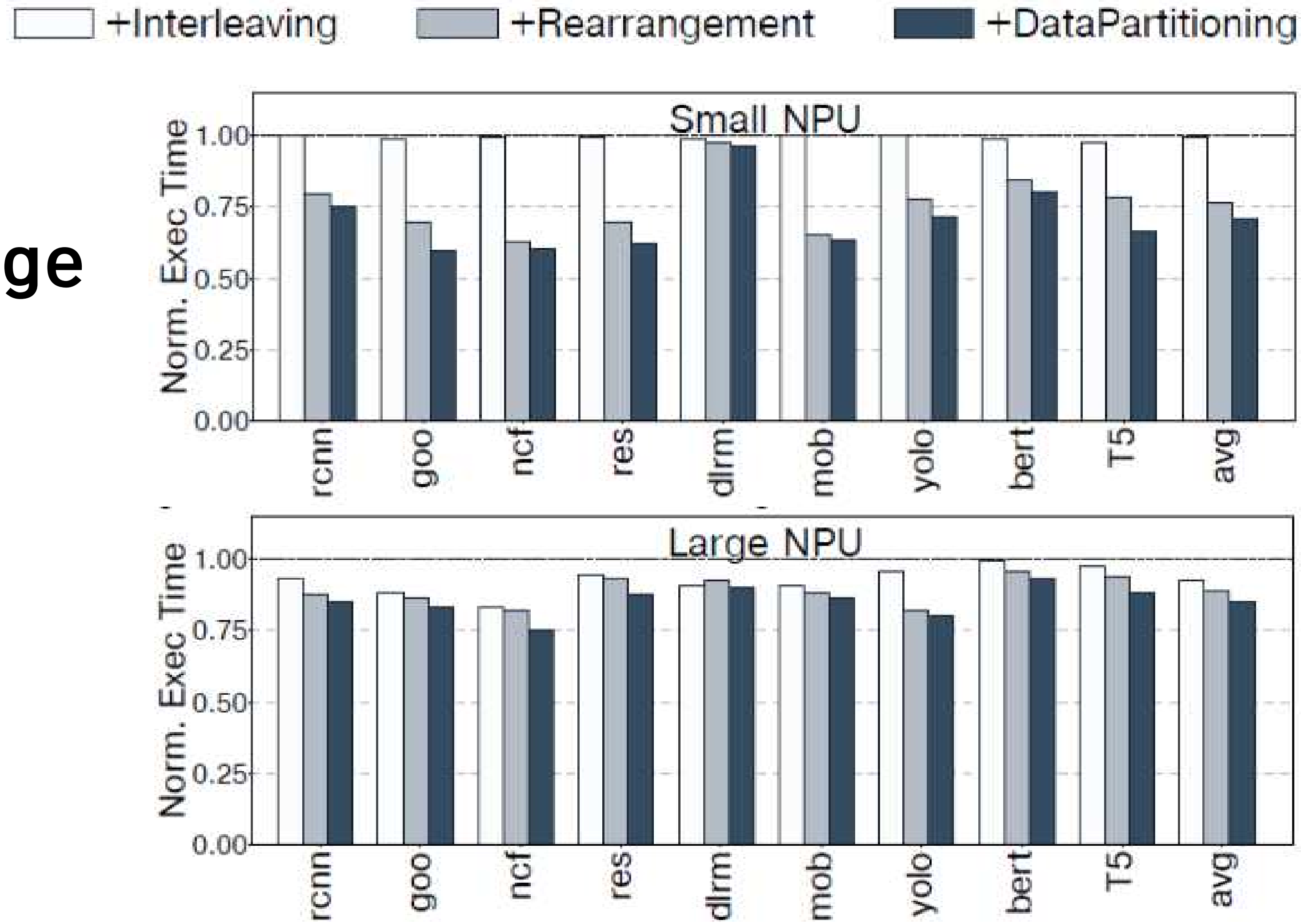


Data Partitioning : Selection

- We need to determine optimal data partitioning schemes for each layer.
- Thus, KNN algorithm is applied using the dimensions of dX, dW, dY as features.
- It has 91% accuracy and wrong predictions are negligible for performance.

Performance Evaluation

- Performance improvement on average
- Small NPU: 29.3%
- Large NPU : 14.5%



Performance Evaluation

- Small NPU has more performance improvement since small NPU has smaller SPM.

- Thus, on-chip data reuse by interleaved gradient order becomes more significant.

Summary

- Interleaved Gradient Order maximizes data reuse within SPM during the backward pass of DNN training.
- Improves performance by inter-operation data sharing, not just intra-operation as prior works.
- This has three key steps.
  1) Interleaves gradient computation
  2) Select optimal tile access order
  3) Data partitioning by selection algorithm
- Performance improvement of 29.3% for small NPU and 14.5% for large NPU.