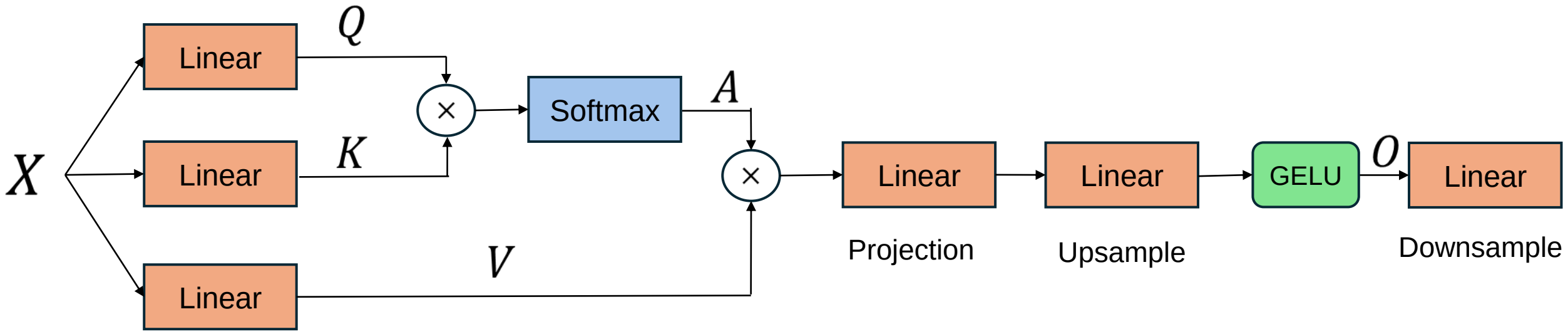
SLoPe:
Double-Pruned Sparse Plus Lazy Low-Rank Adapter Pretraining of LLMs

Mohammad Mozaffari[1], Amir Yazdanbakhsh[2], Zhao Zhang[3], Maryam Mehri Dehnavi[1]

[1] University of Toronto, [2] Google DeepMind, [3]Rutgers University

bit.ly/slope-llm

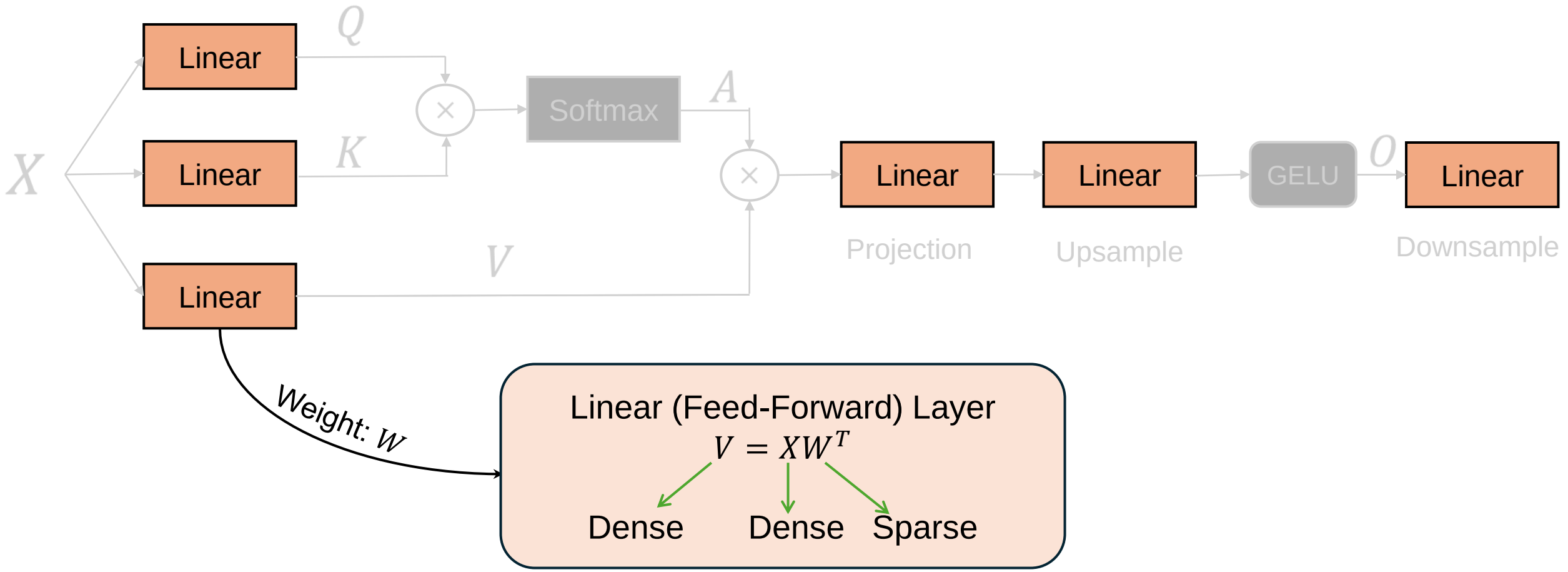
LLM Compute Graph



- Residual connections, layer norms, and other details of the compute graph are not illustrated.

LLM Compute Graph | Weight Sparsity

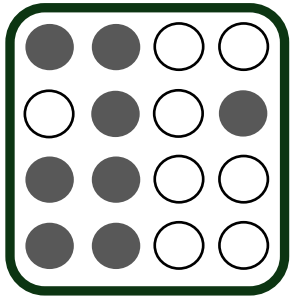
V: Output
X: Input
W: Weight



SLoPe | Double-Pruned Backward Pass

☑ Sparse Tensor Cores

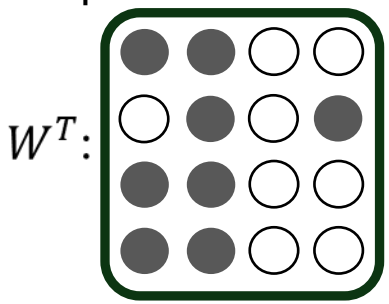
W^T:



Forward Pass: $Y = XW^T$

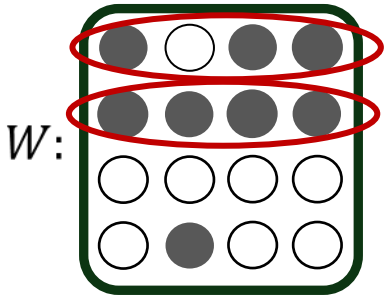
SLoPe | Double-Pruned Backward Pass

☑ Sparse Tensor Cores



Transpose

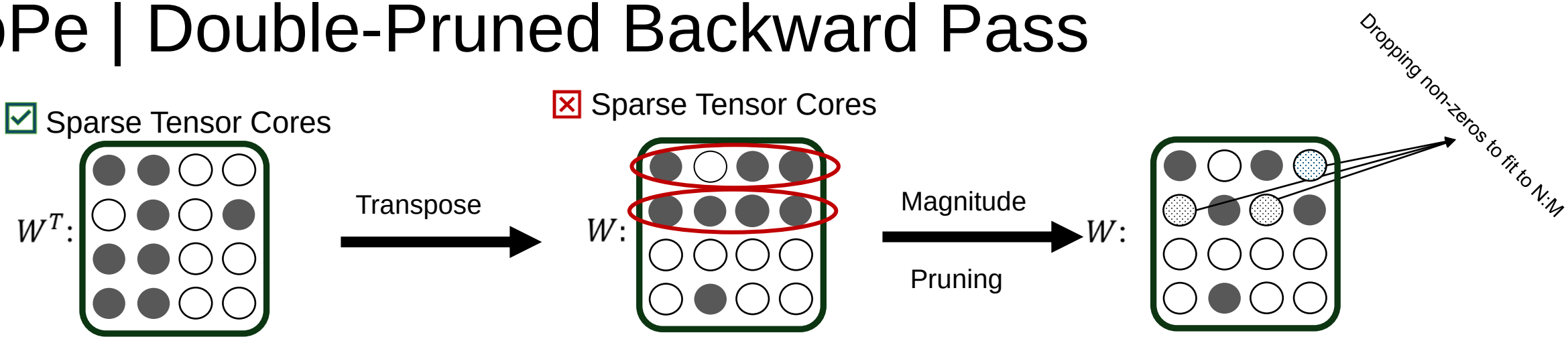
☒ Sparse Tensor Cores



Forward Pass: $Y = XW^T$

Backward Pass: $\nabla_X L = \nabla_Y LW$

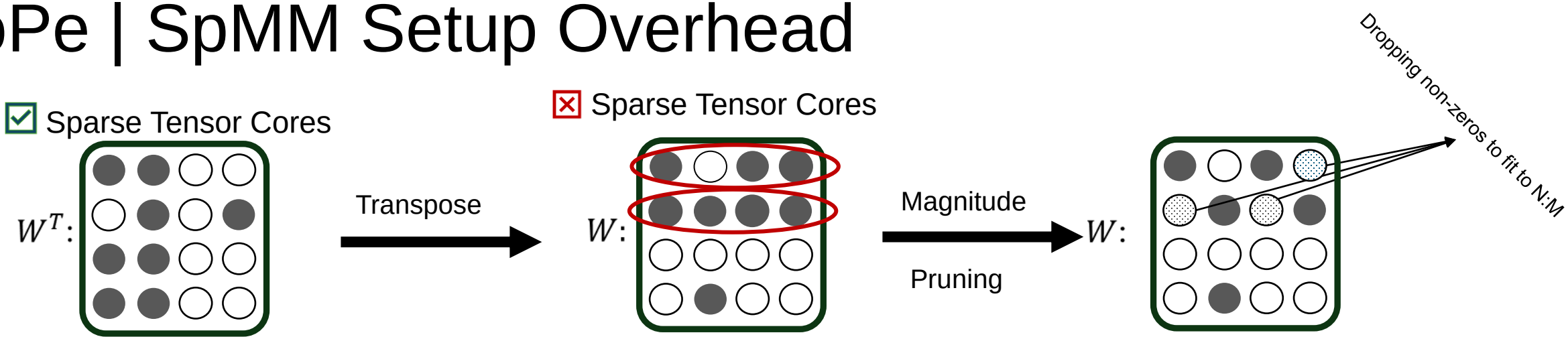
SLoPe | Double-Pruned Backward Pass



Forward Pass: $Y = XW^T$

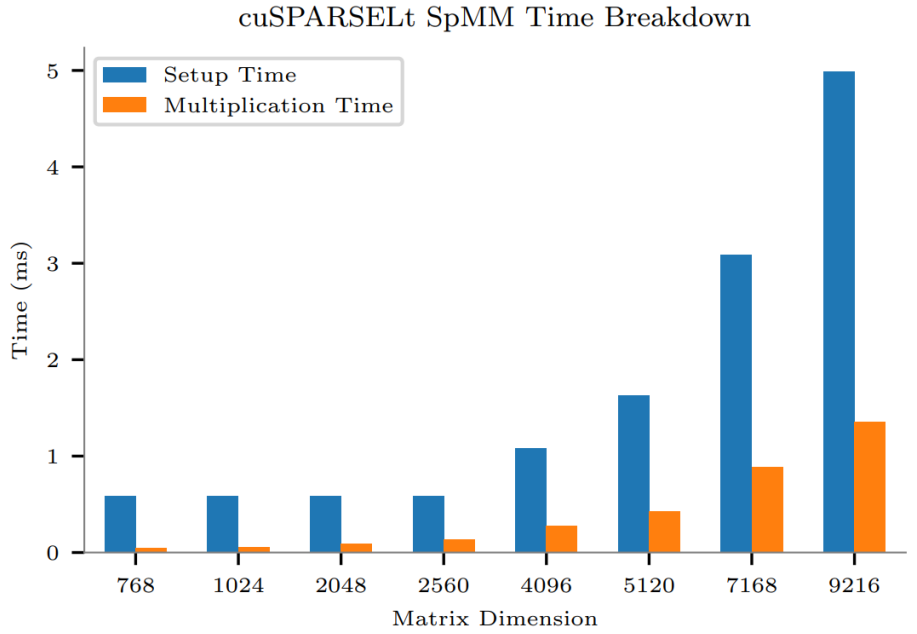
Backward Pass: $\nabla_X L = \nabla_Y LW$

SLoPe | SpMM Setup Overhead

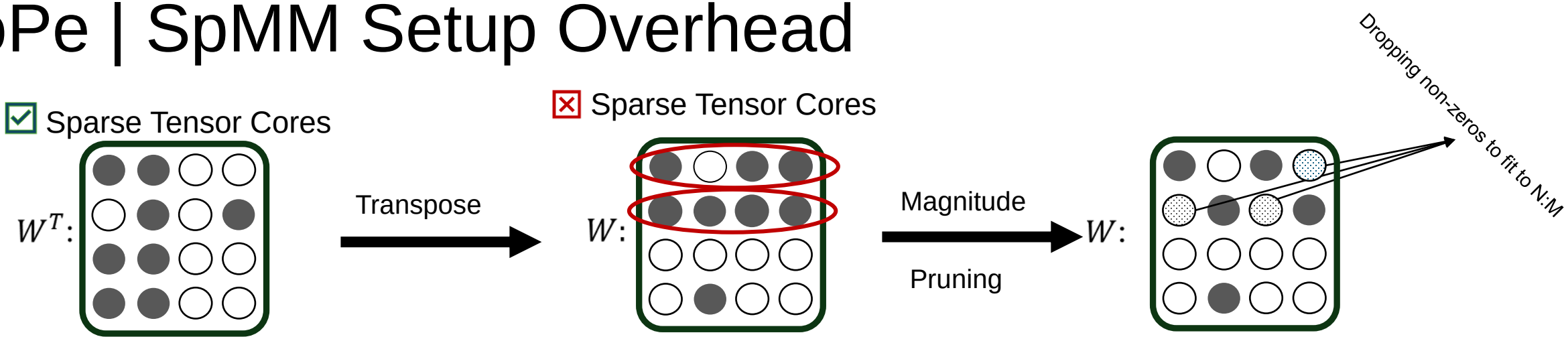


Forward Pass: $Y = XW^T$

Backward Pass: $\nabla_X L = \nabla_Y LW$



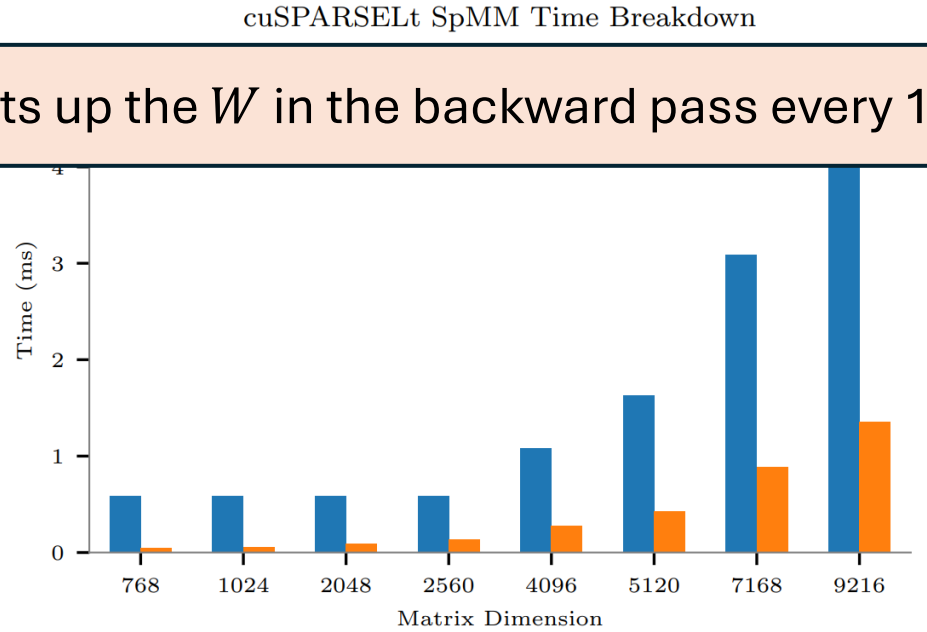
SLoPe | SpMM Setup Overhead



Forward Pass: $Y = XW^T$

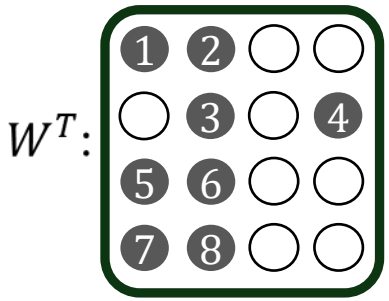
Backward Pass: $\nabla_X L = \nabla_Y LW$

SLoPe prunes and sets up the W in the backward pass every 100 iterations!



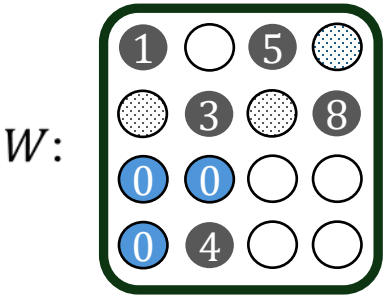
SLoPe | Running Example

☑ Sparse Tensor Cores

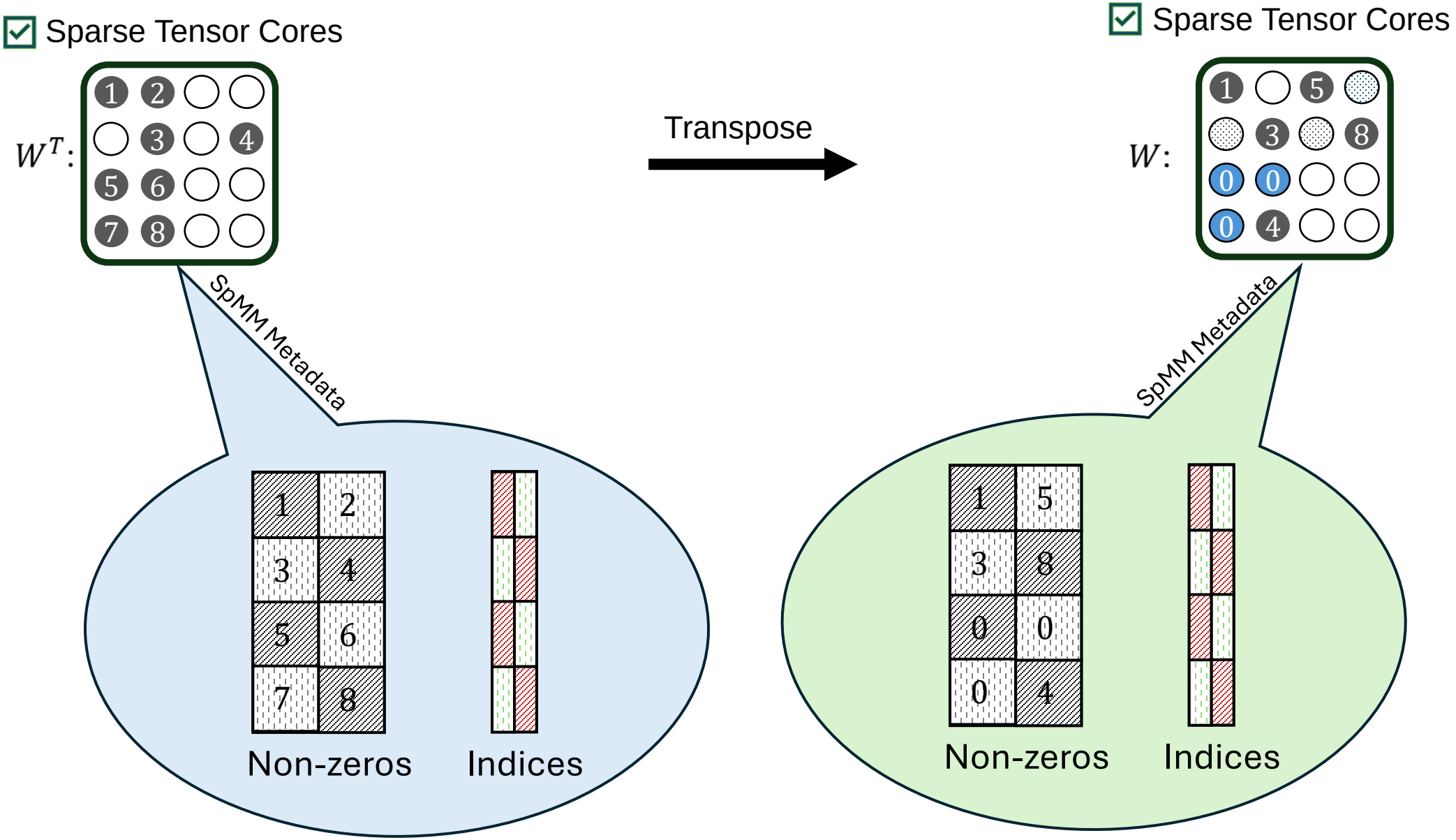


Transpose

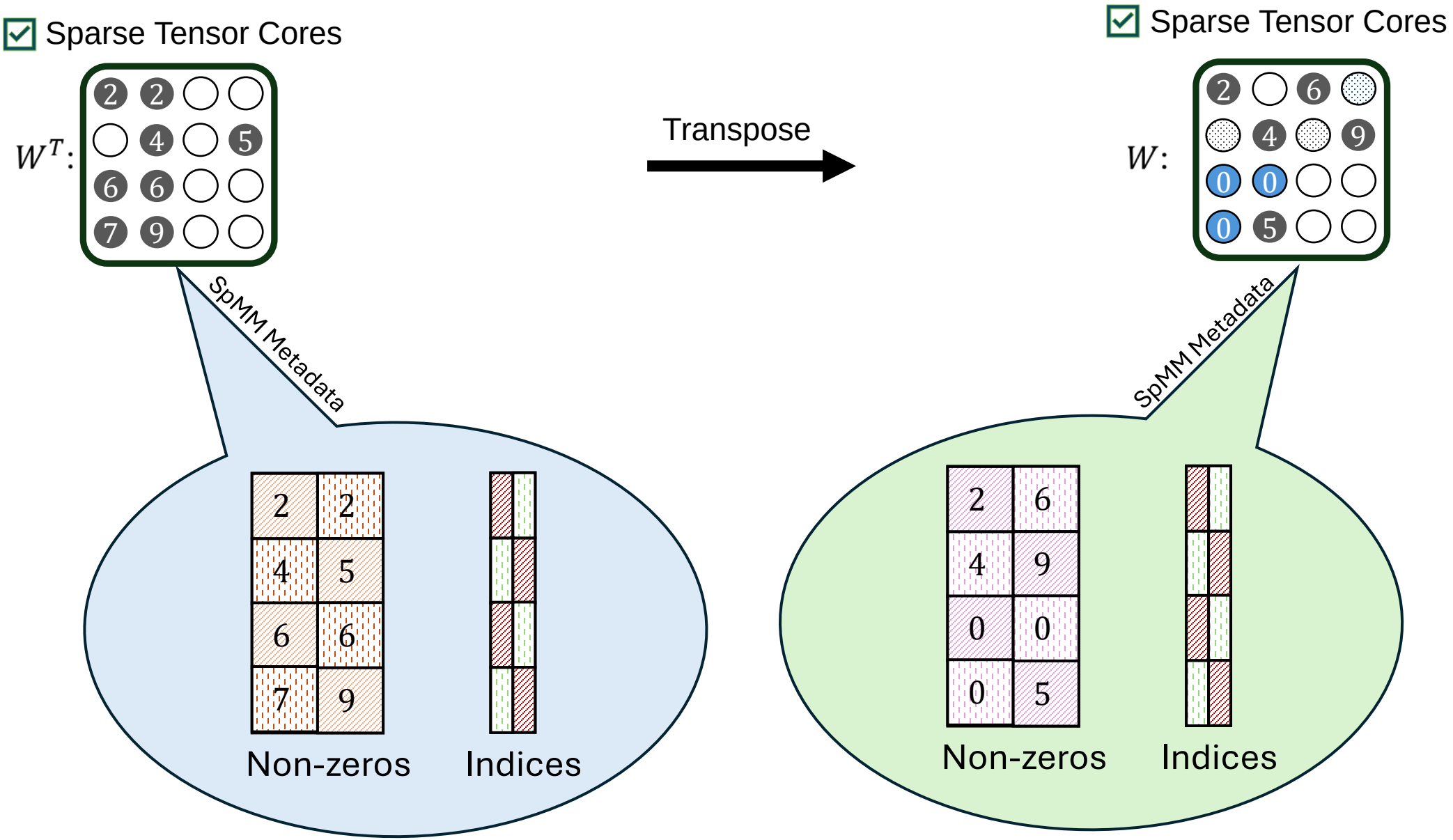
☑ Sparse Tensor Cores



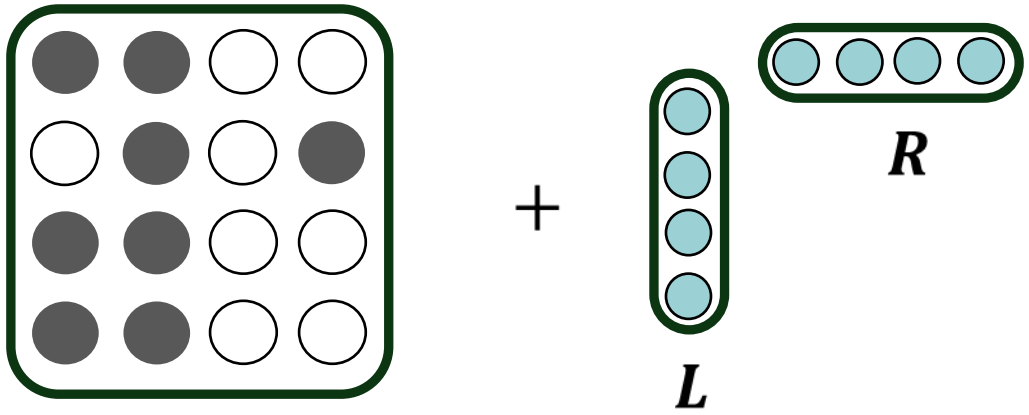
SLoPe | Running Example



SLoPe | Running Example | Weight Update



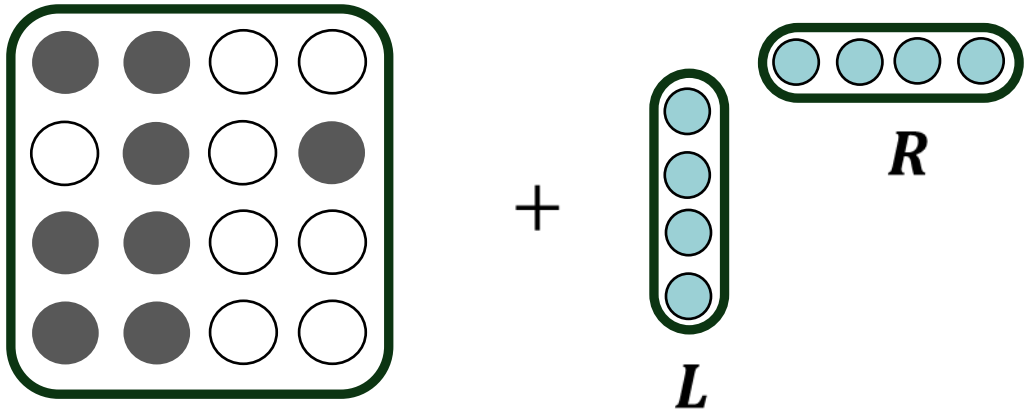
SLoPe | Lazy Low-rank Adapters



$$W = W_{sparse} + LR$$

SLoPe | Lazy Low-rank Adapters

d: Hidden Dimension
r: Adapter Rank
b: Batch Size

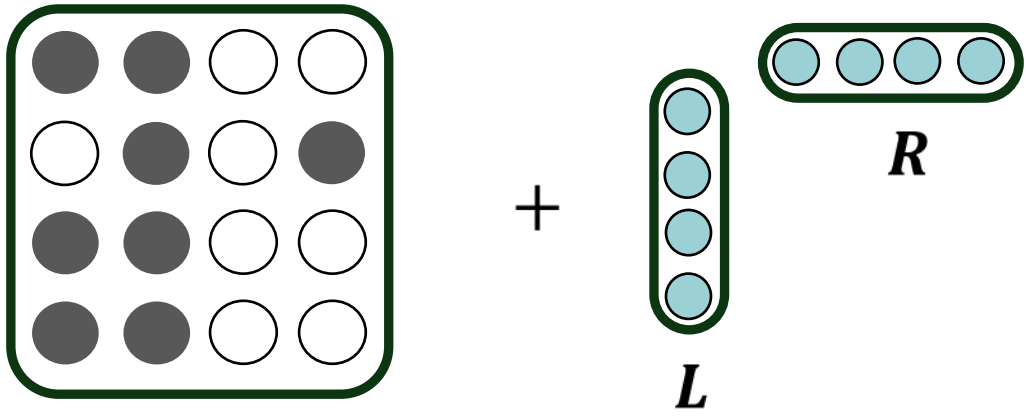


$$W = W_{sparse} + LR$$

Complexity	Sparse + Low-rank	Dense
Memory	$\frac{d^2}{2} + 2rd$	d^2
Compute	$\frac{bd^2}{2} + 2brd$	bd^2

SLoPe | Lazy Low-rank Adapters

d: Hidden Dimension
r: Adapter Rank
b: Batch Size



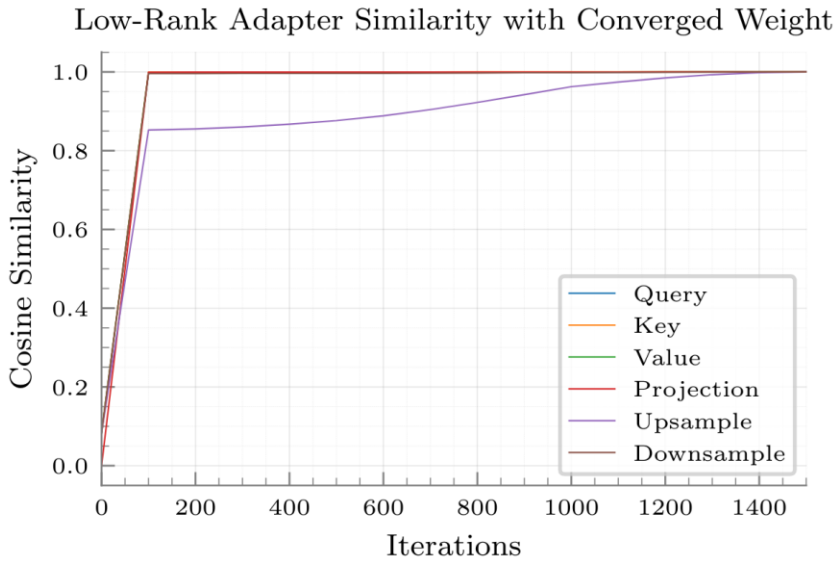
$$W = W_{sparse} + LR$$

Complexity	Sparse + Low-rank	Dense
Memory	$\frac{d^2}{2} + 2rd$	d^2
Compute	$\frac{bd^2}{2} + 2brd$	bd^2

Since $r \ll d$, the memory and compute overhead is negligible.

SLoPe | Lazy Low-rank Adapters | Convergence Rate

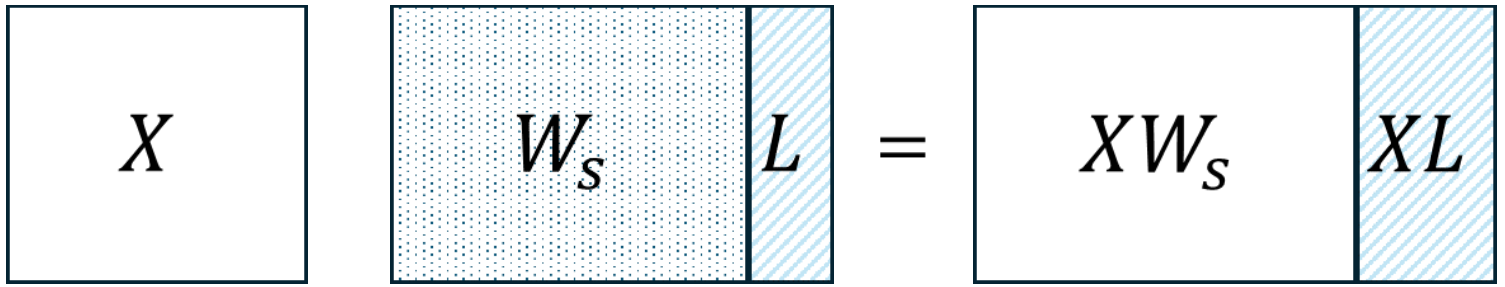
- We test the convergence rate of different layers in BERT-Large-Uncased.
  - Low-rank adapters converge after only 100 iterations.



- Due to their fast convergence rate, SLoPe adds low-rank adapters in the last 1% of training.

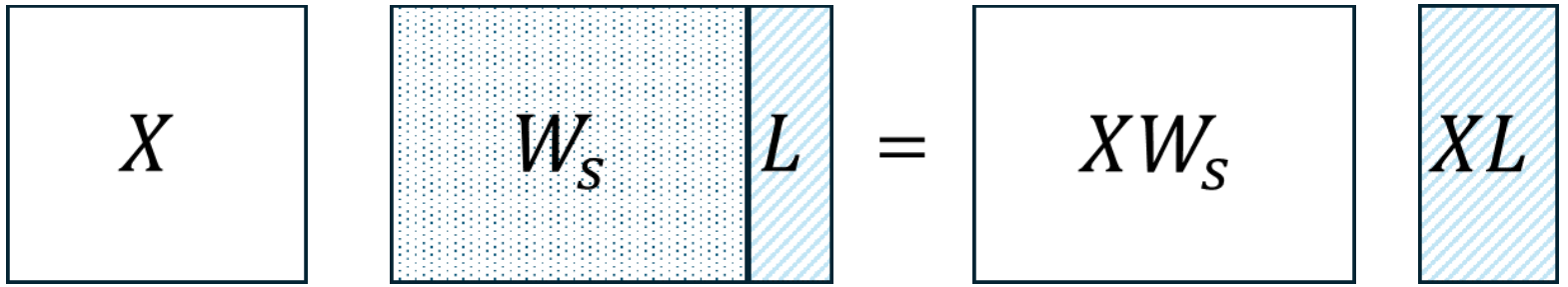
SLoPe | Combined SpMM and Low-rank Adapters

- Low arithmetic intensity in low-rank adapters
  - **Solution:** Combines SpMM and Low-rank adapters



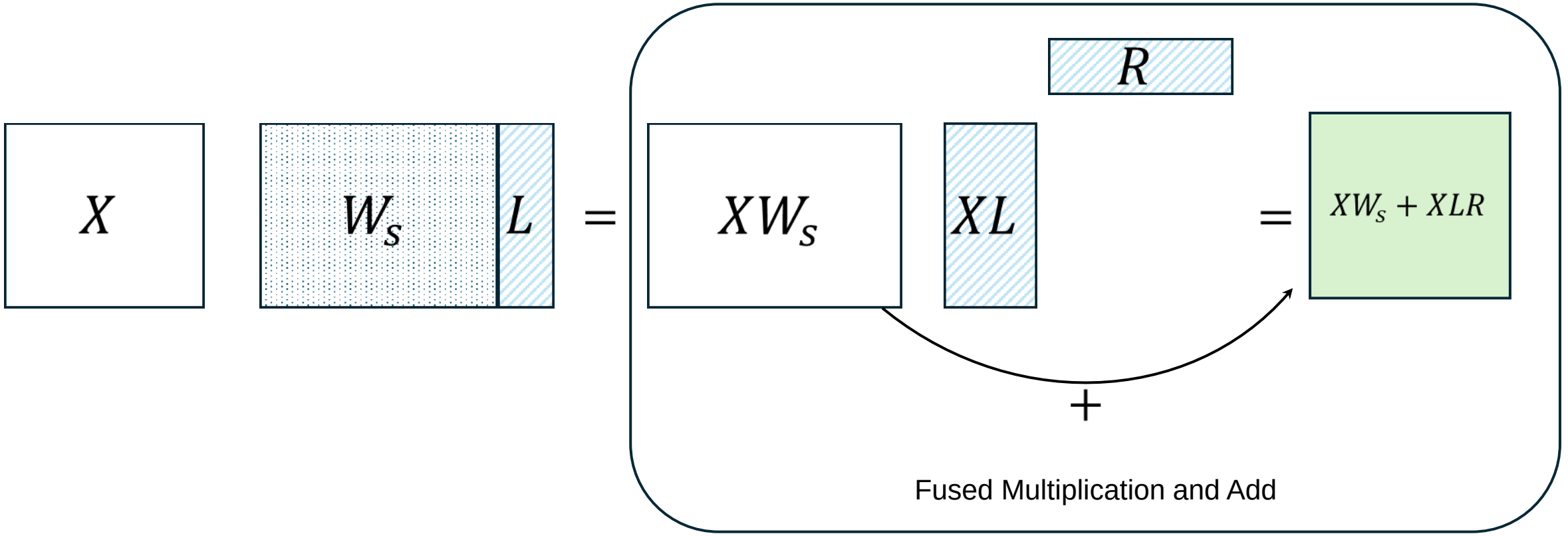
SLoPe | Combined SpMM and Low-rank Adapters

- Low arithmetic intensity in low-rank adapters
  - **Solution:** Combines SpMM and Low-rank adapters



SLoPe | Combined SpMM and Low-rank Adapters

- Low arithmetic intensity in low-rank adapters
  - **Solution:** Combines SpMM and Low-rank adapters



SLoPe | Results| Speedup and Memory Reduction

Speedup

SLoPe achieves up to

$1.25\times$ speedup in training!

$1.53\times$ speedup in inference!

Memory Reduction

SLoPe achieves up to

$0.63\times$ memory reduction in training!

$0.51\times$ memory reduction in inference!

SLoPe | Results | GPT2 Zero-shot Accuracy

Method	Adapter Rank	MMLU ↑	Arc Challenge↑	Open-BookQA↑	Wino-Grande↑	Hella-Swag↑	MathQA↑	PiQA↑	Race↑
Dense	N/A	22.9	20.7	16.2	50.6	28.5	21.8	59.8	28.4
SLoPe	2.1%	23.0	19.3	**16.4**	**50.8**	**27.5**	20.8	**57.6**	**27.2**
	0.05%	23.0	**19.4**	16.2	50.5	27.4	20.8	57.5	27.1
	0	23.0	19.3	16.0	50.1	27.5	20.8	57.4	27.1
Extended SR-STE	2.1%	**24.2**	18.3	14.2	47.5	26.9	**21.4**	55.2	24.2
	0.05%	24.1	18.4	14.2	47.5	26.8	21.2	54.5	24.2
	0	24.1	18.3	12.6	47.5	26.9	21.2	54.8	24.0

SLoPe outperforms state-of-the-art in 6 out of 8 tasks!

SLoPe | Results | BERT-Large-Uncased Accuracy

BERT-Large-Uncased

Dataset	Dense	$r = 0$	$r = 0.39\%$	$r = 1.56\%$	$r = 6.25\%$
SQuAD	90.4	89.1	89.1	89.2	89.5
GLUE	80.2	77.4	77.7	77.8	78.2

SLoPe achieves 0.9% and 2% accuracy gap on SQuAD v1.1 and GLUE dataset on BERT-Large-Uncased