Monarch: Expressive Structured Matrices for Efficient and Accurate Training

Tri Dao, Beidi Chen, Nimit Sohoni, Arjun Desai, Michael Poli
Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, Christopher Ré

University at Buffalo
The State University of New York

Sparsity: A Natural Approach to Reduce Computation

Challenges with structured linear maps (low-rank, sparse, Fourier):

Sparsity: A Natural Approach to Reduce Computation

Challenges with structured linear maps (low-rank, sparse, Fourier):

Sparse end-to-end training

Efficiency-quality tradeoffs:

- **Efficiency**: on modern hardware (GPU)
- Quality: how **expressive** are the weight matrices (can they represent commonly used transforms)

Sparsity: A Natural Approach to Reduce Computation

Challenges with structured linear maps (low-rank, sparse, Fourier):

Sparse end-to-end training

Efficiency-quality tradeoffs:

- **Efficiency**: on modern hardware (GPU)
- Quality: how **expressive** are the weight matrices (can they represent commonly used transforms)

Dense-to-sparse finetuning

- **Projection**: How to find a sparse/structured matrix closest to a pretrained dense weight matrix

Sparsity: A Natural Approach to Reduce Computation

Challenges with structured linear maps (low-rank, sparse, Fourier):

Sparse end-to-end training

Efficiency-quality tradeoffs:
- **Efficiency**: on modern hardware (GPU)
- Quality: how **expressive** are the weight matrices (can they represent commonly used transforms)

Dense-to-sparse finetuning

- **Projection**: How to find a sparse/structured matrix closest to a pretrained dense weight matrix

Monarch: one of the first sparse training methods to achieve wall-clock speedup while maintaining quality.

Outline

Part 1 Monarch matrices

- Hardware-efficiency
- Expressiveness
- Tractable projection from dense weight matrices

Part 2 Ways to use sparse models

- Sparse end-to-end (E2E) training
- Sparse-to-dense (S2D) training (reverse sparsification)

Part 3 Applications

- Language modeling, computer vision, PDEs & MRI

Outline

Part 1 **Monarch matrices**

Hardware-efficiency
Expressiveness
Tractable projection from dense weight matrices

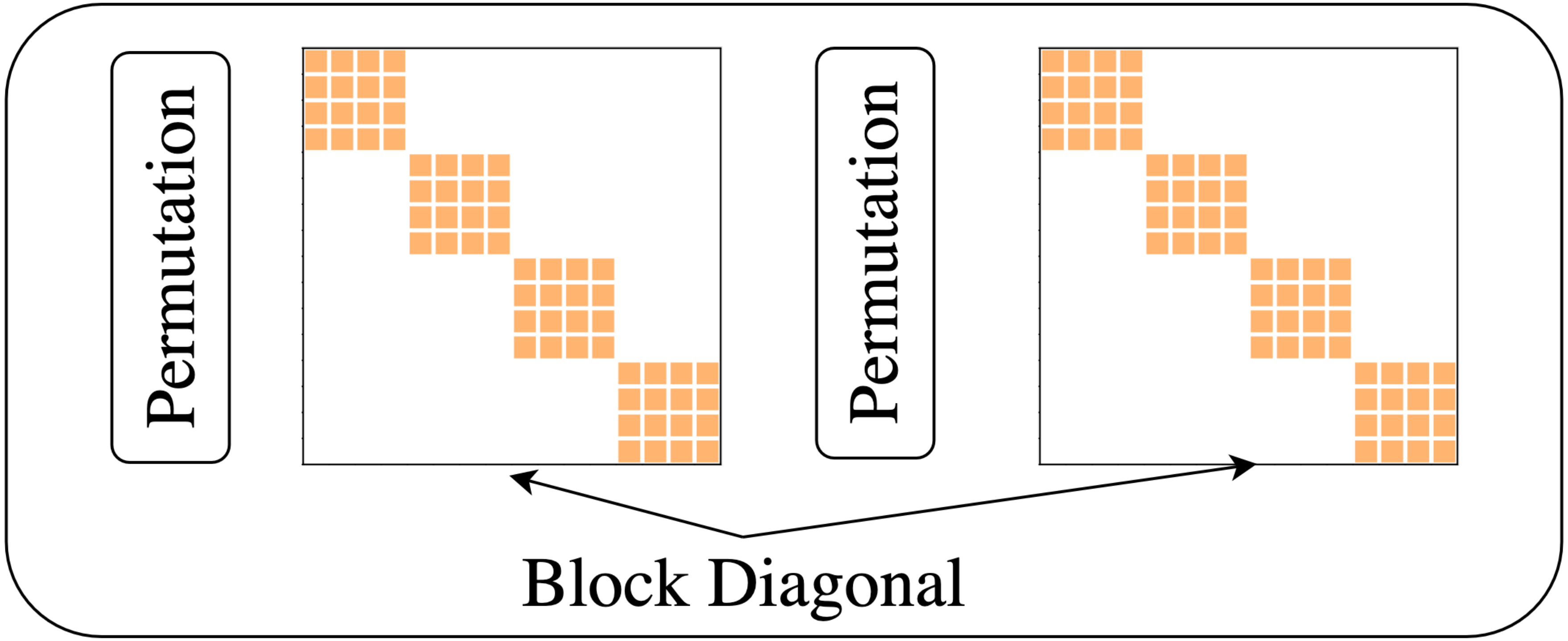
Part 2 **Three ways to use sparse models**

Sparse end-to-end (E2E) training
Sparse-to-dense (S2D) training (reverse sparsification)

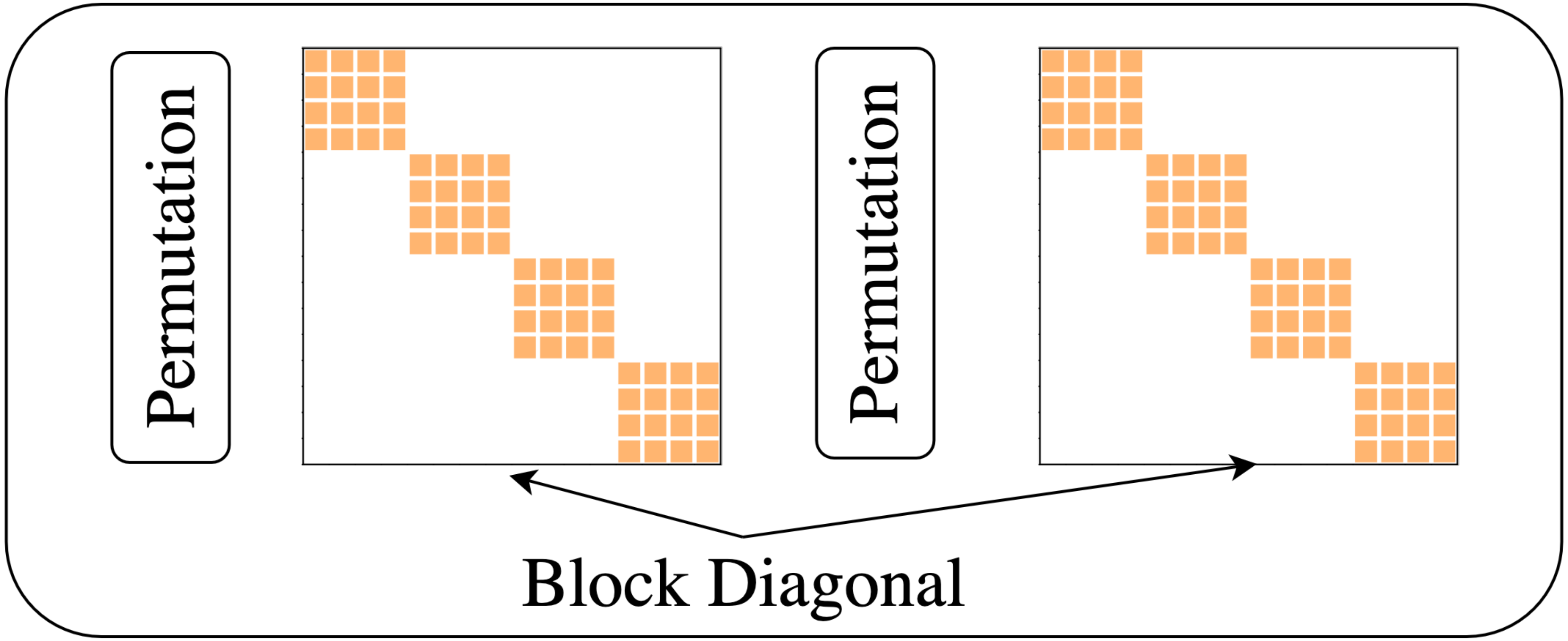
Part 3 **Applications**

Language modeling, computer vision, PDEs & MRI

Monarch Matrices: Efficient and Expressive

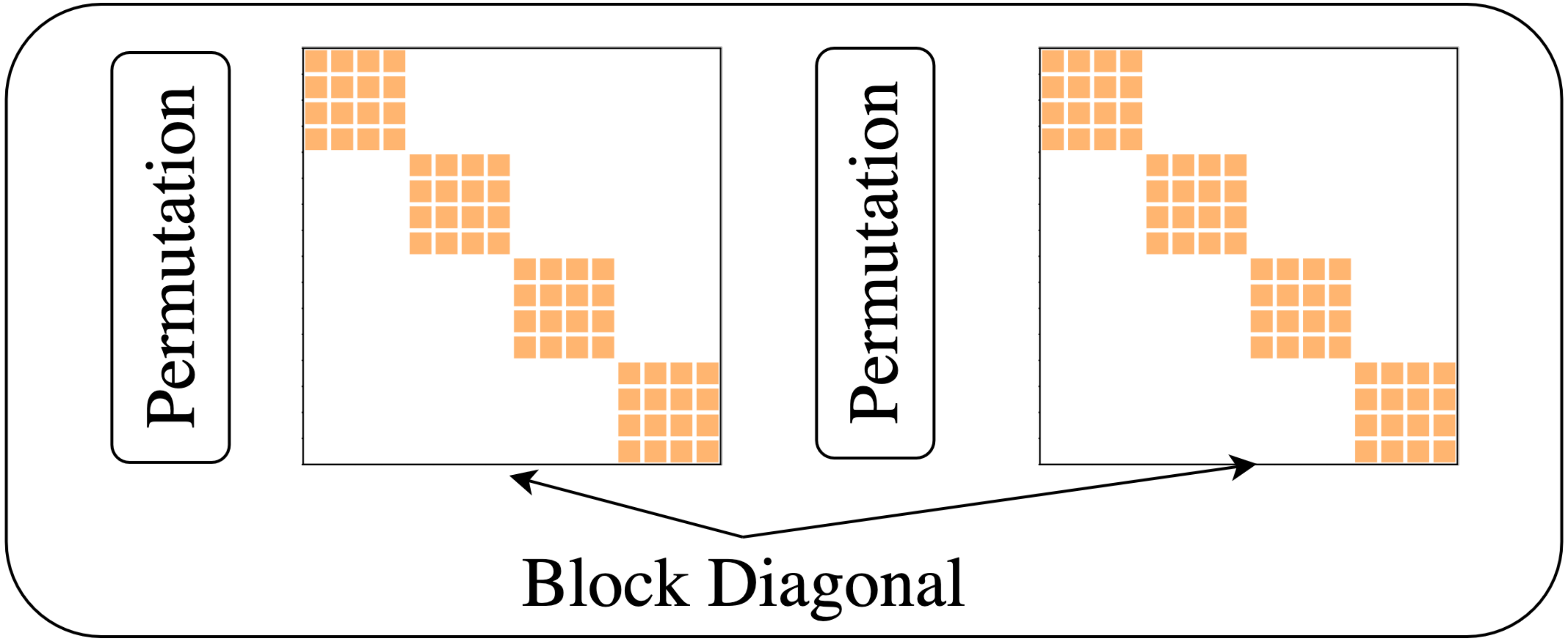


Monarch Matrices: Efficient and Expressive



Motivation: divide-and-conquer structure
(e.g., Cooley-Tukey FFT algorithm)

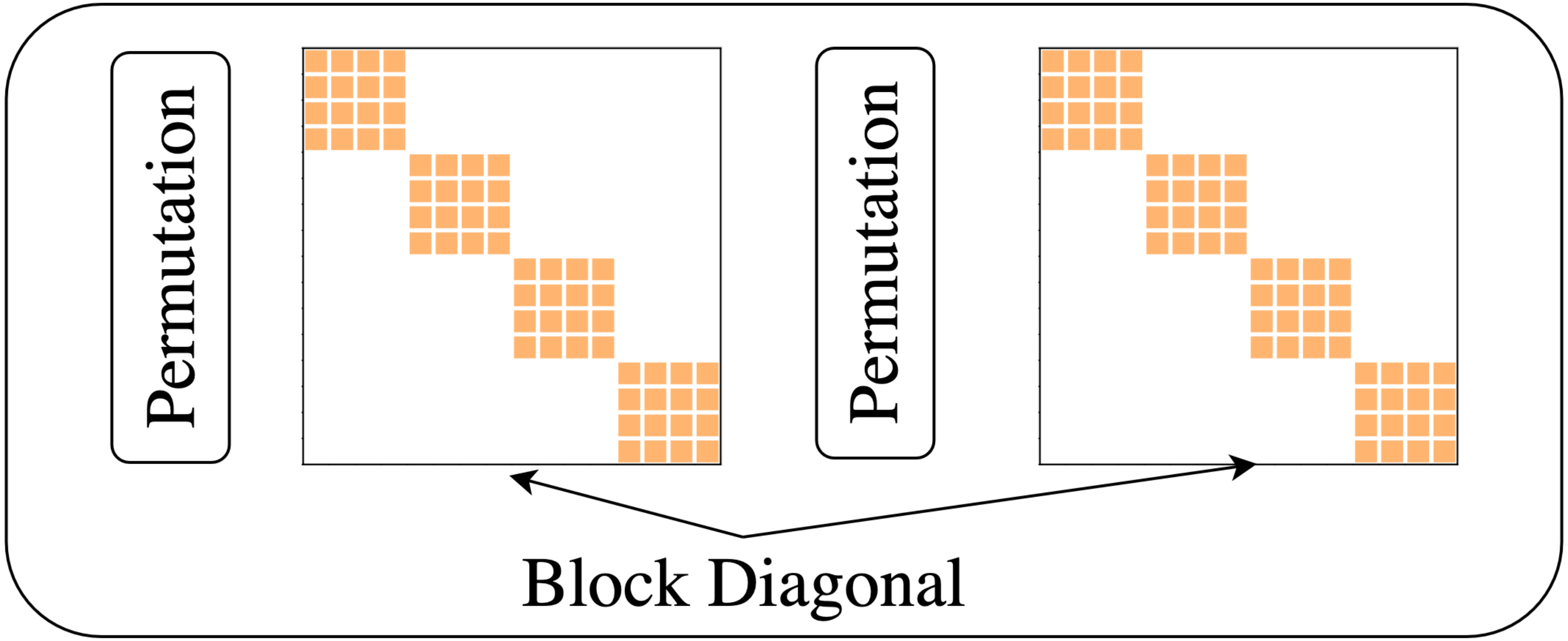
Monarch Matrices: Efficient and Expressive



(1) Hardware-efficient
(2) Expressive
(3) Tractable projection

Motivation: divide-and-conquer structure
(e.g., Cooley-Tukey FFT algorithm)

Monarch Matrices: Efficient and Expressive

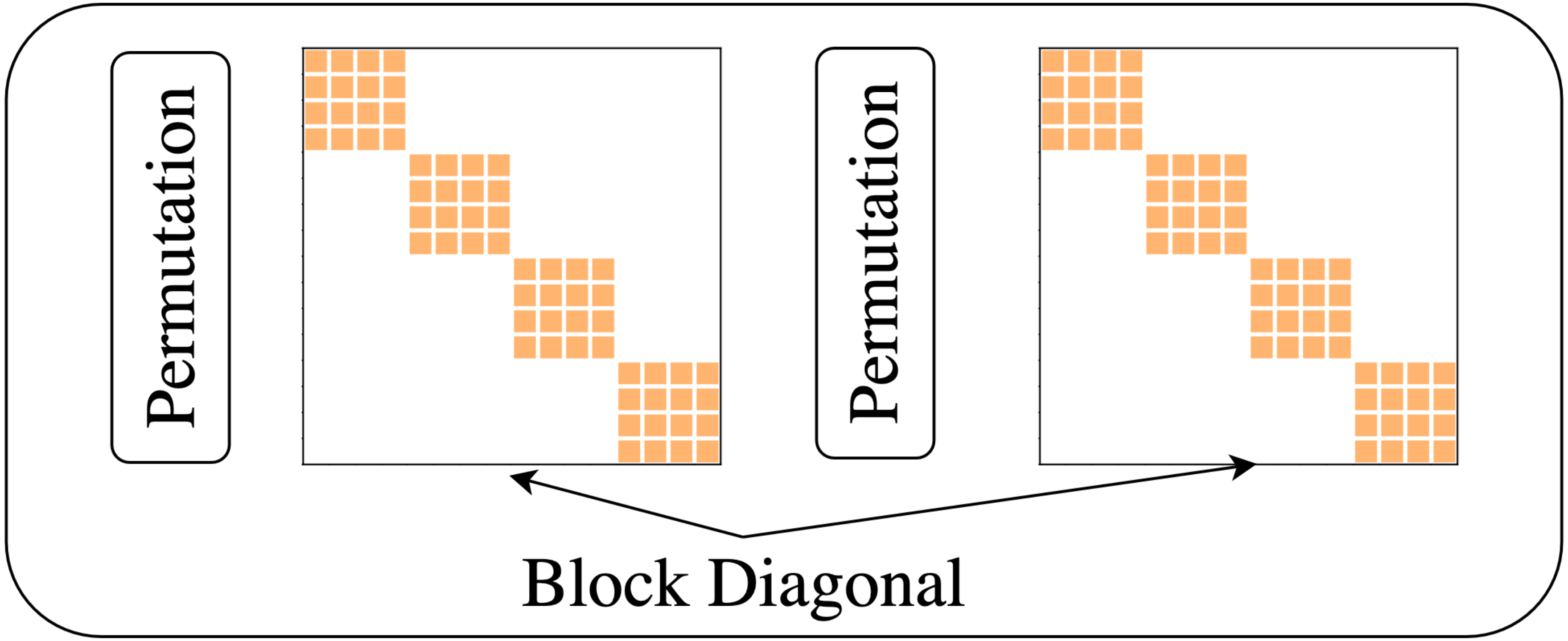


Focus of this talk

(1) Hardware-efficient
(2) Expressive
(3) Tractable projection

Motivation: divide-and-conquer structure (e.g., Cooley-Tukey FFT algorithm)

Monarch Matrices: Efficient and Expressive



Focus of this talk

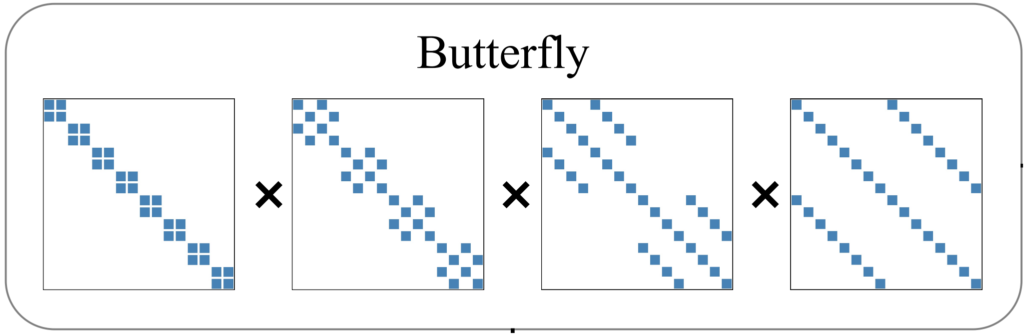
(1) Hardware-efficient
(2) Expressive

Nice and deep theory

(3) Tractable projection

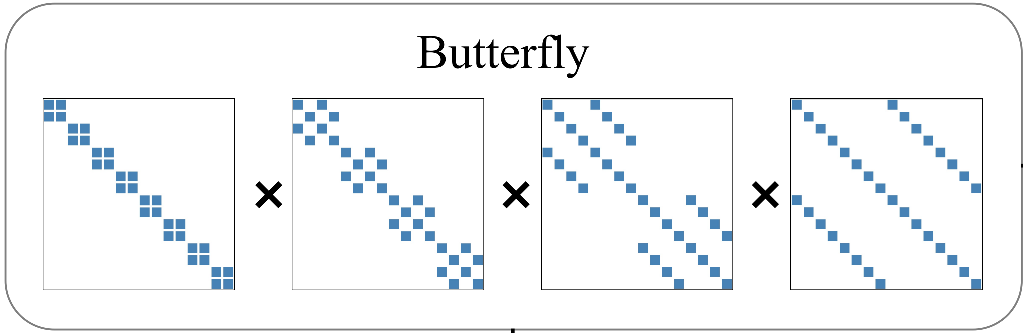
Motivation: divide-and-conquer structure (e.g., Cooley-Tukey FFT algorithm)

Hardware-Efficiency



[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency

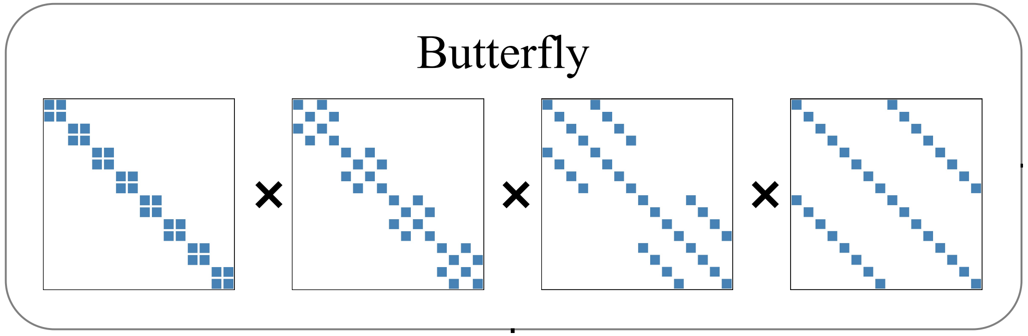


Fast in theory (O(N log N) runtime & parameters

Expressive: Can represent any structure (e.g., sparsity) almost optimally

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency

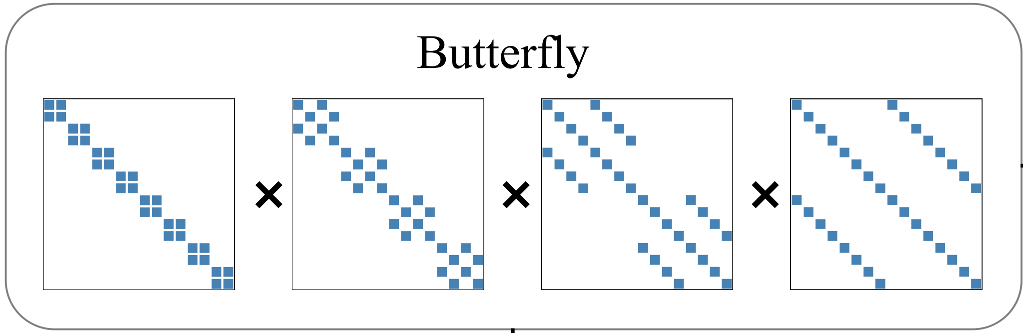


Fast in theory (O(N log N) runtime & parameters
Expressive: Can represent any structure (e.g., sparsity) almost optimally

Problem 1: Not block-aligned

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency



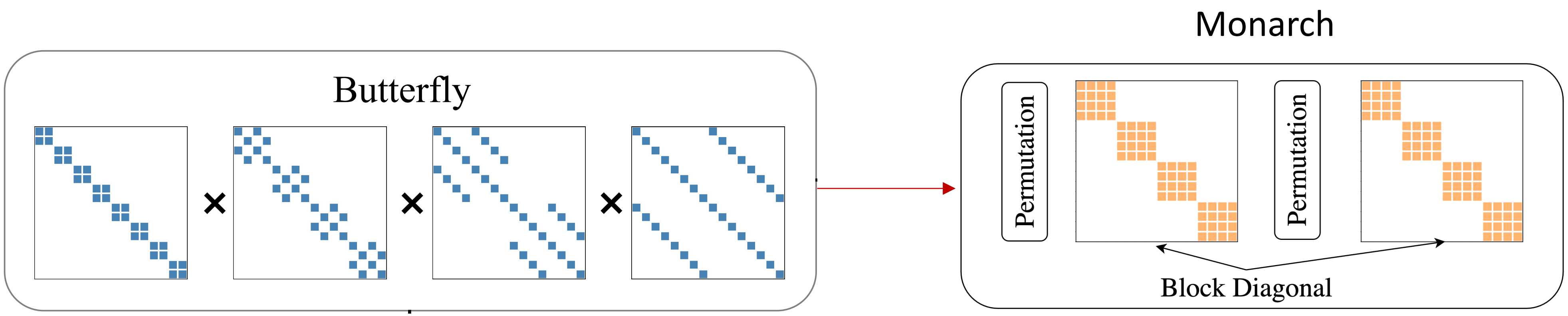
Fast in theory (O(N log N) runtime & parameters
Expressive: Can represent any structure (e.g., sparsity) almost optimally

Problem 1: Not block-aligned

Problem 2: Hard to parallelize the product of many factors

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency



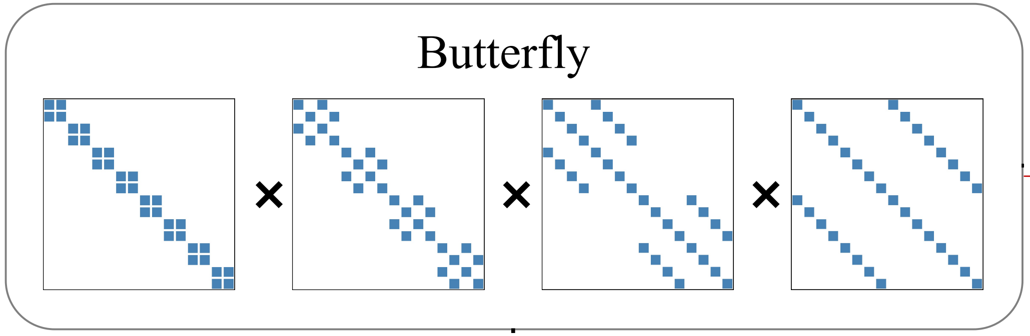
Fast in theory (O(N log N) runtime & parameters
Expressive: Can represent any structure (e.g., sparsity) almost optimally

Problem 1: Not block-aligned

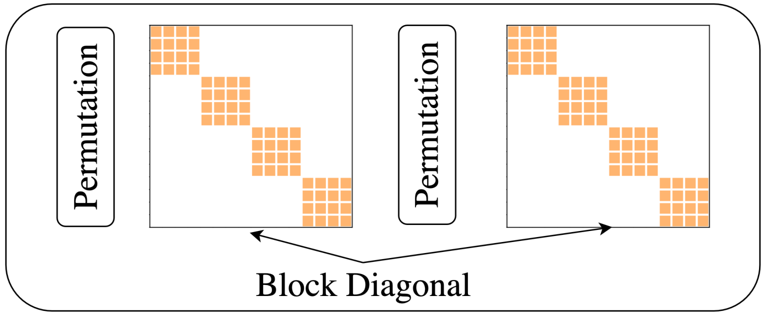
Problem 2: Hard to parallelize the product of many factors

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency



Monarch



Fast in theory (O(N log N) runtime & parameters
Expressive: Can represent any structure (e.g., sparsity) almost optimally

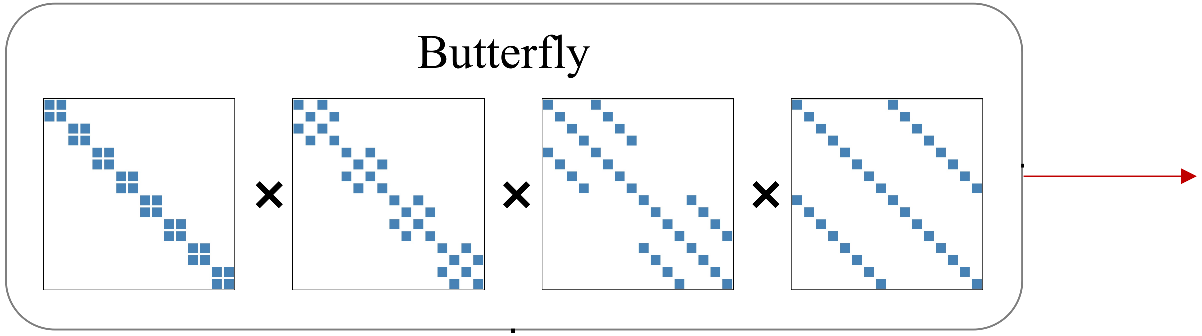
3-5x faster on GPUs
Same expressivity

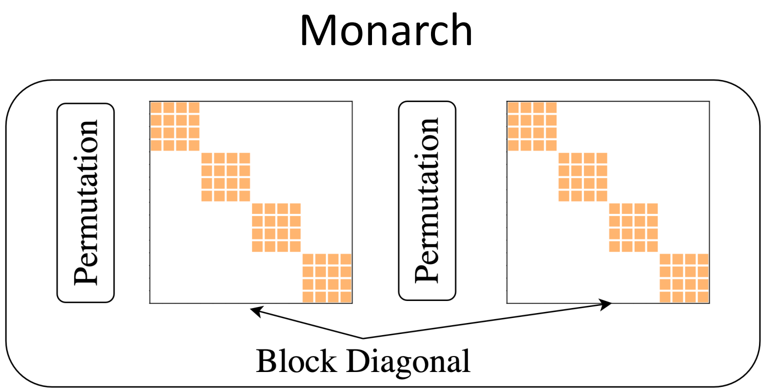
Problem 1: Not block-aligned

Problem 2: Hard to parallelize the product of many factors

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Hardware-Efficiency





Fast in theory (O(N log N) runtime & parameters
Expressive: Can represent any structure (e.g., sparsity) almost optimally

3-5x faster on GPUs
Same expressivity

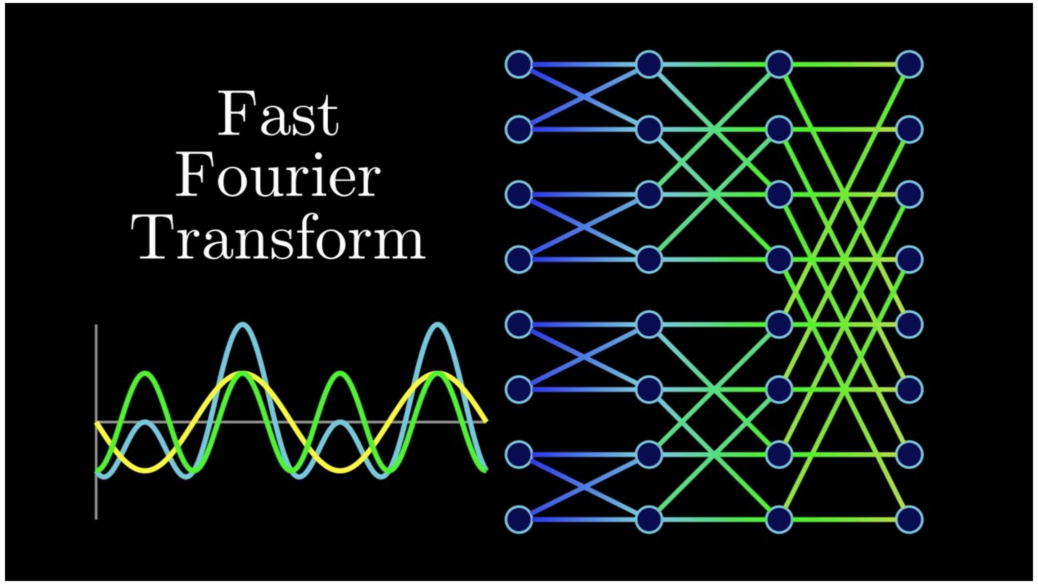
Problem 1: Not block-aligned

Problem 2: Hard to parallelize the product of many factors

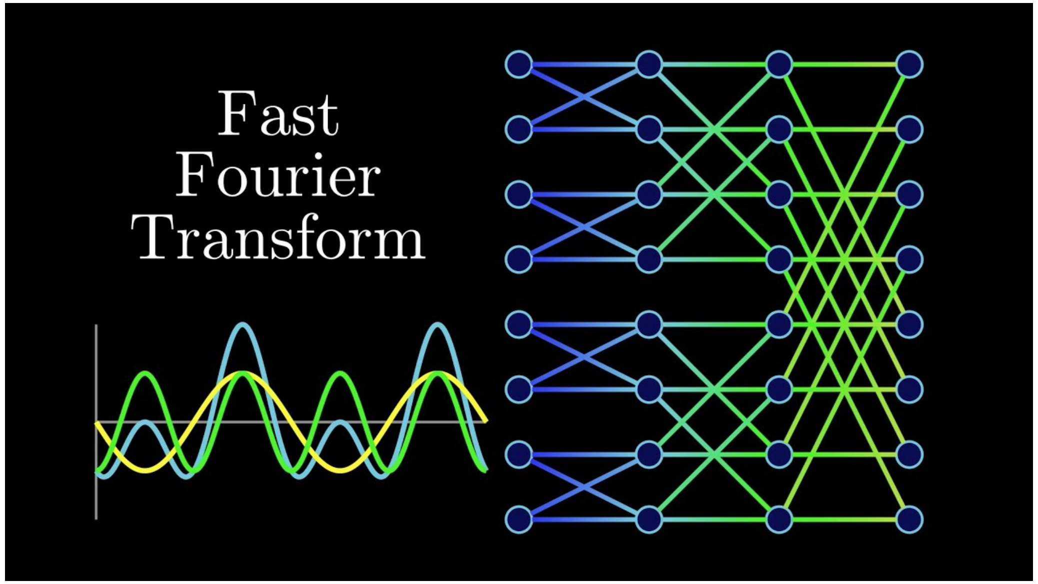
Block-diagonal leverages efficient batch-matrix-multiply on GPUs

[Beneš, 65; *Parker, 95; Matthieu & LeCun, 14; De Sa et al., 18, Dao et al., 19]*

Expressiveness: Monarch Can Represent Many Structured Matrices

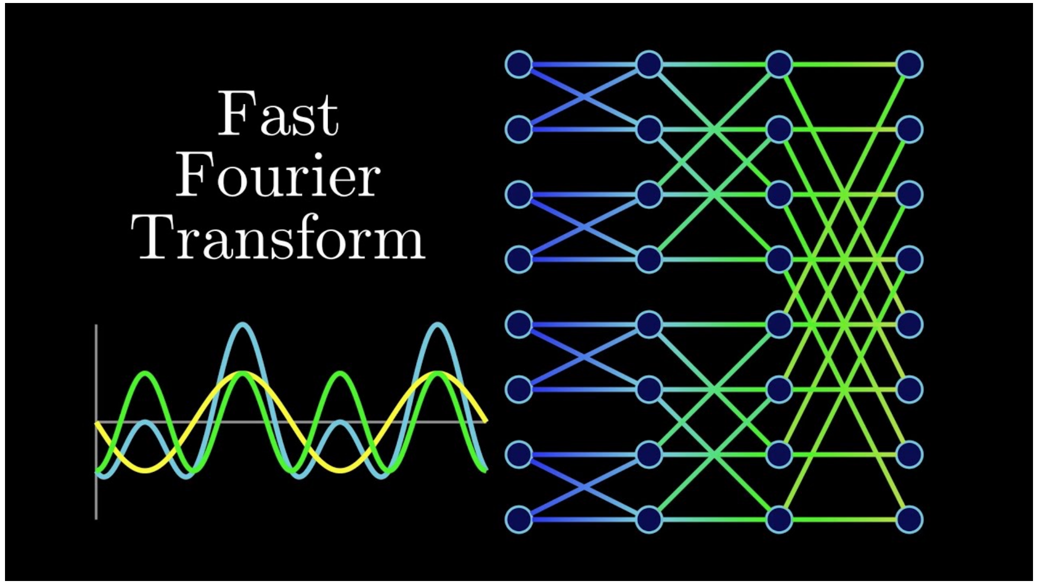


Expressiveness: Monarch Can Represent Many Structured Matrices



(Elementwise) sparsity & low-rank can't represent most of these structures

Expressiveness: Monarch Can Represent Many Structured Matrices



(Elementwise) sparsity & low-rank can’t represent most of these structures

Monarch can represent & learn these structures

Outline

Part 1 **Monarch matrices**

Hardware-efficiency
Expressiveness
Tractable projection from dense weight matrices

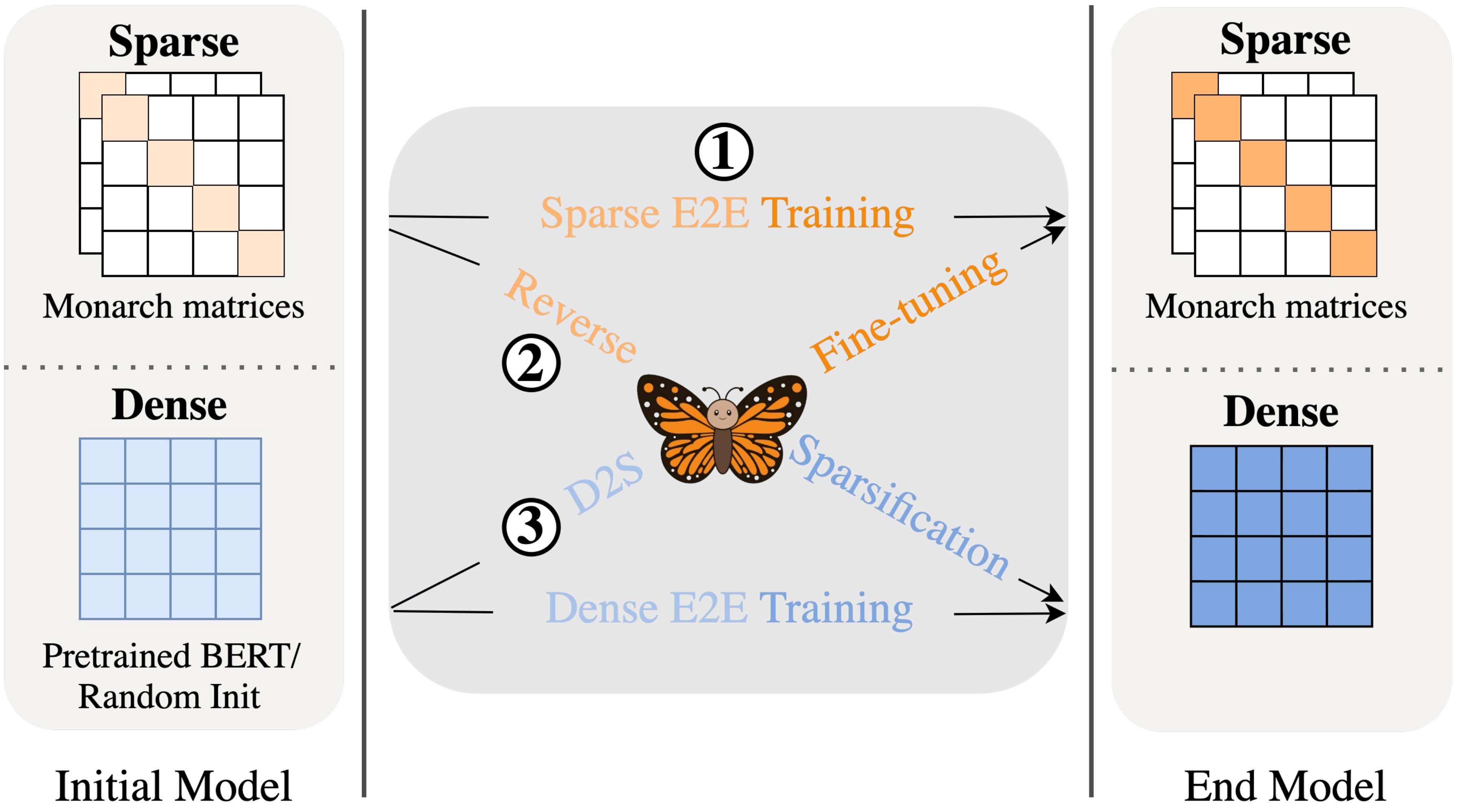
Part 2 **Ways to use sparse models**

Sparse end-to-end (E2E) training
Sparse-to-dense (S2D) training (reverse sparsification)

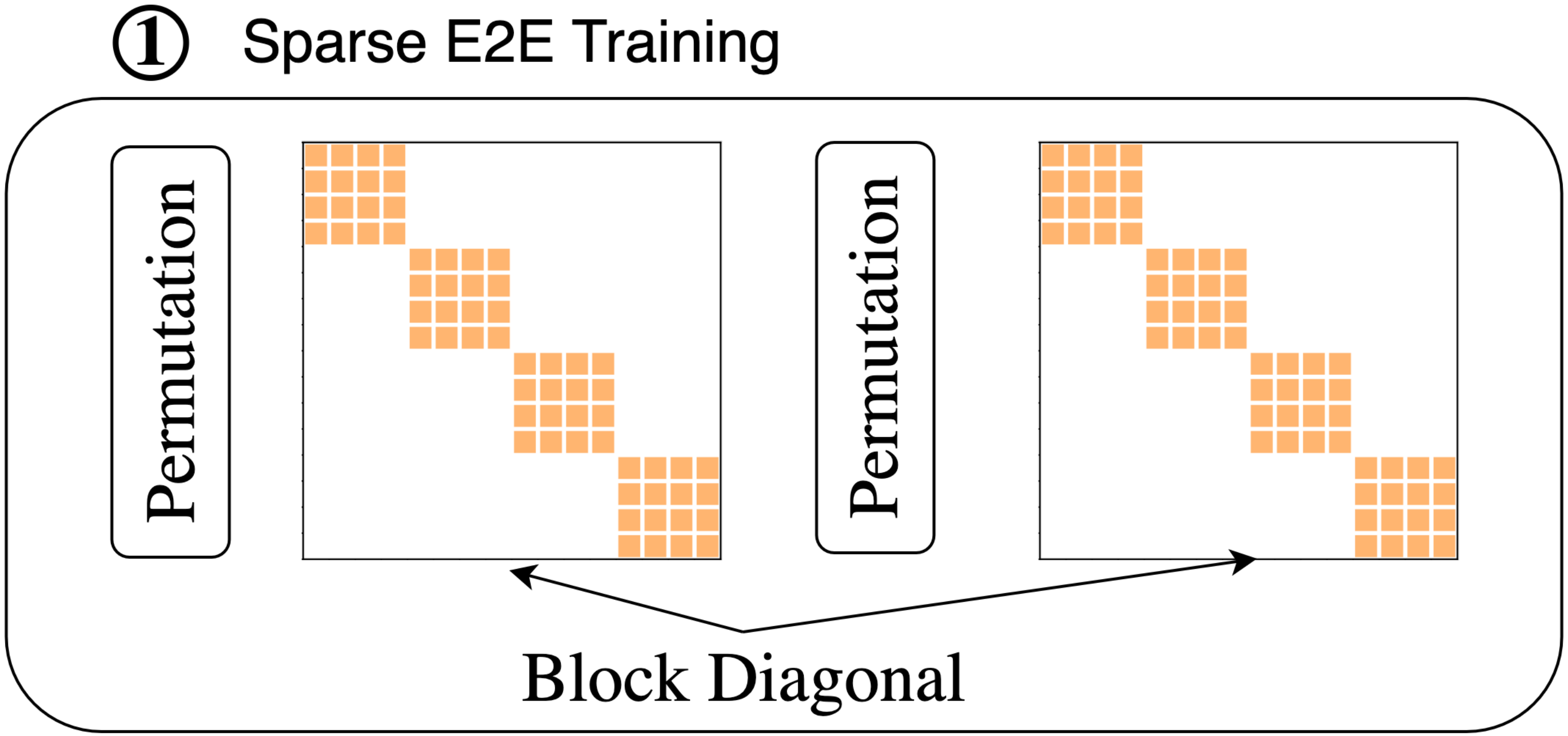
Part 3 **Applications**

Language modeling, computer vision, PDEs & MRI

Three Ways to Use Sparse Models

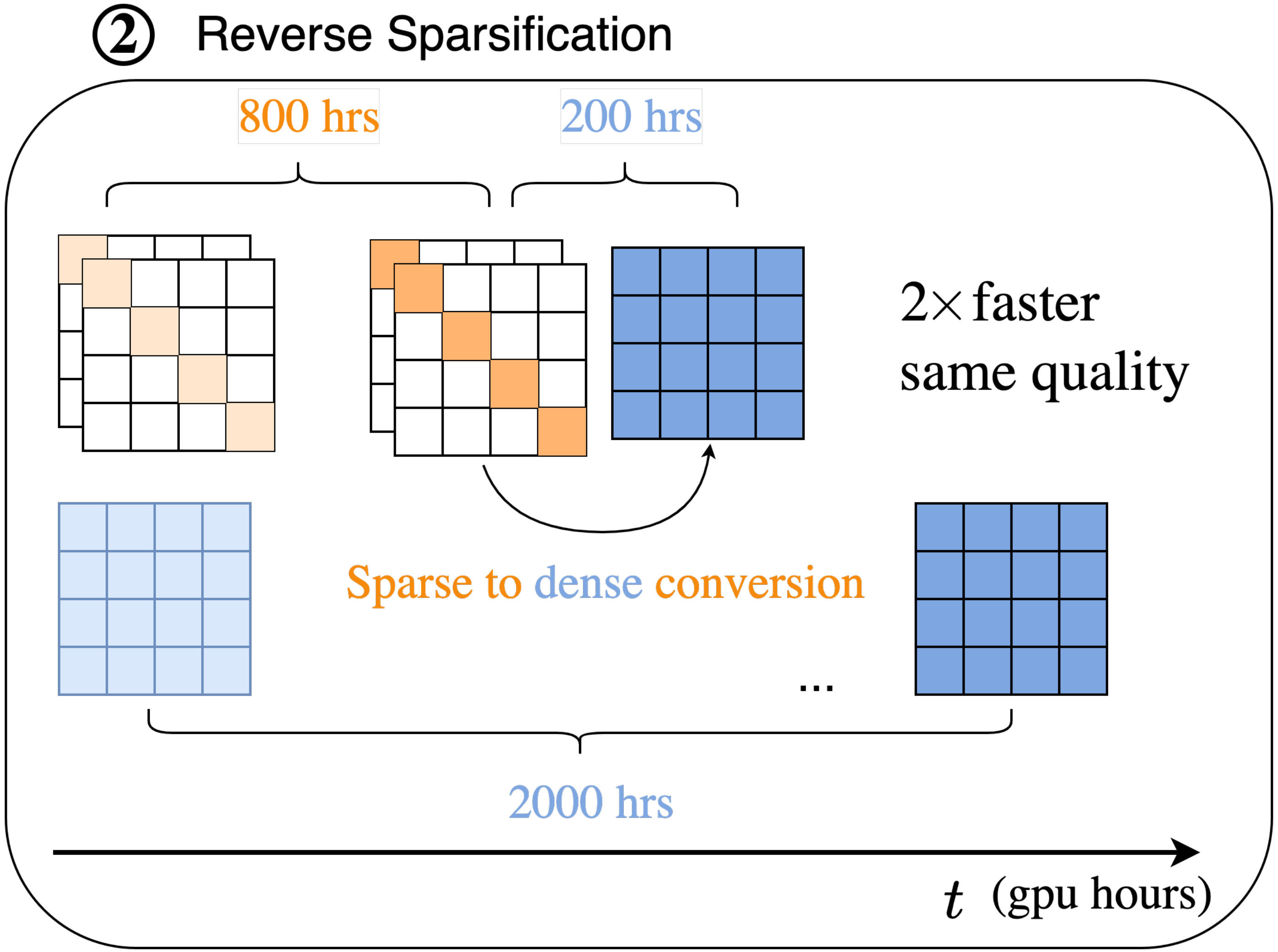


Sparse End-to-End Training



Replace dense weight matrices (e.g., attention & FFN) with Monarch matrices for efficiency

Sparse-to-Dense Training (reverse sparsification)



Outline

Part 1 **Monarch matrices**

Hardware-efficiency
Expressiveness
Tractable projection from dense weight matrices

Part 2 **Ways to use sparse models**

Sparse end-to-end (E2E) training
Sparse-to-dense (S2D) training (reverse sparsification)

Part 3 **Applications**

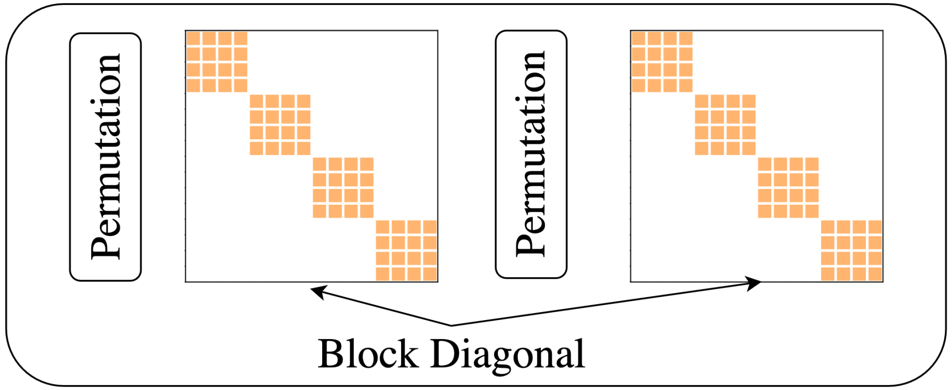
Language modeling, computer vision, PDEs & MRI

Validation: Sparse End-to-End Training

Benchmark tasks

Model	WikiText 103(ppl)	Speedup
GPT-2 Small	20.6	-
Monarch GPT-2-small	20.7	**1.8 x**

① Sparse E2E Training

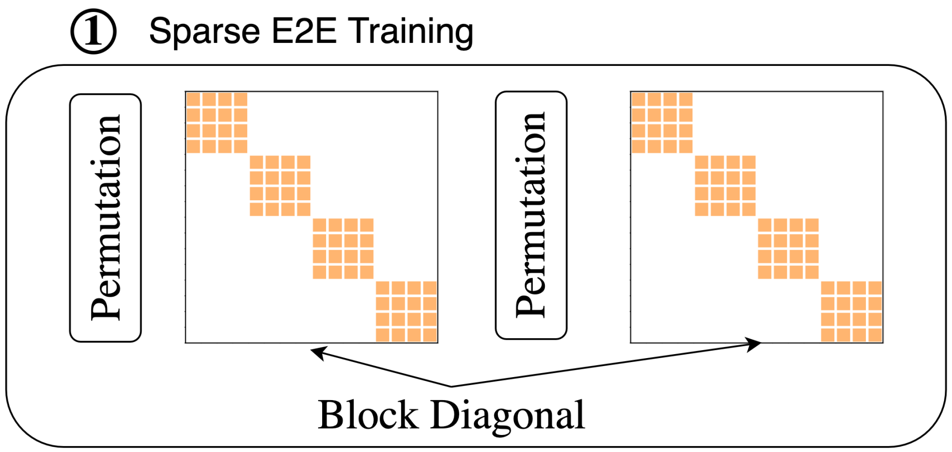


Validation: Sparse End-to-End Training

Benchmark tasks

Model	WikiText 103(ppl)	Speedup
GPT-2 Small	20.6	-
Monarch GPT-2-small	20.7	**1.8 x**

Model	ImageNet (acc)	Speedup
ViT-Base	78.5	-
Monarch ViT-Base	78.7	**2.0 x**

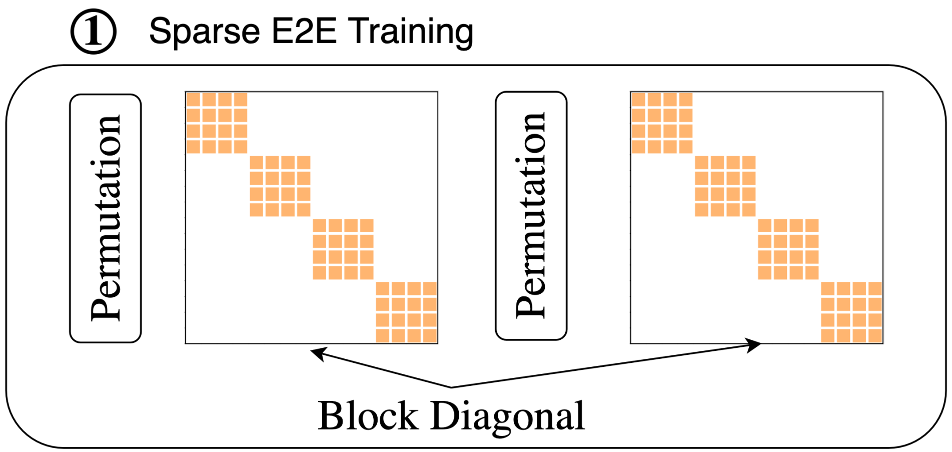


Validation: Sparse End-to-End Training

Benchmark tasks

Model	WikiText 103(ppl)	Speedup
GPT-2 Small	20.6	-
Monarch GPT-2-small	20.7	**1.8 x**

Model	ImageNet (acc)	Speedup
ViT-Base	78.5	-
Monarch ViT-Base	78.7	**2.0 x**



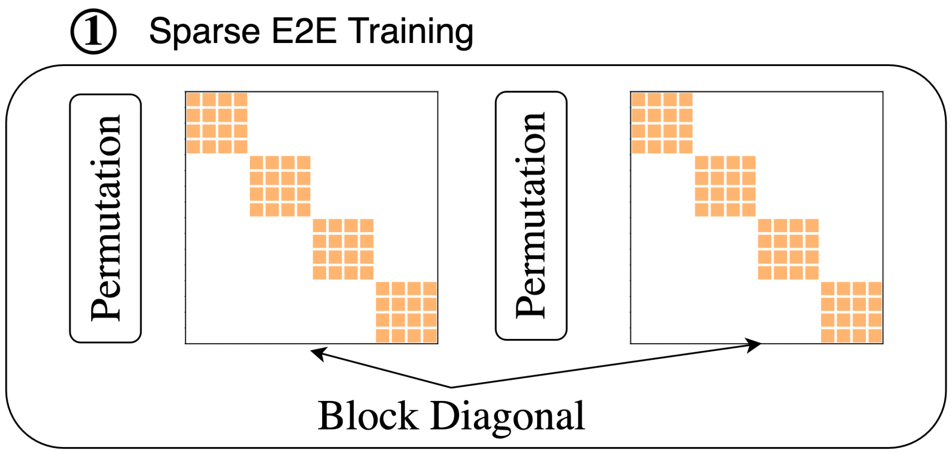
Other applications: PDEs solving, MRI reconstruction

Validation: Sparse End-to-End Training

Benchmark tasks

Model	WikiText 103(ppl)	Speedup
GPT-2 Small	20.6	-
Monarch GPT-2-small	20.7	**1.8 x**

Model	ImageNet (acc)	Speedup
ViT-Base	78.5	-
Monarch ViT-Base	78.7	**2.0 x**

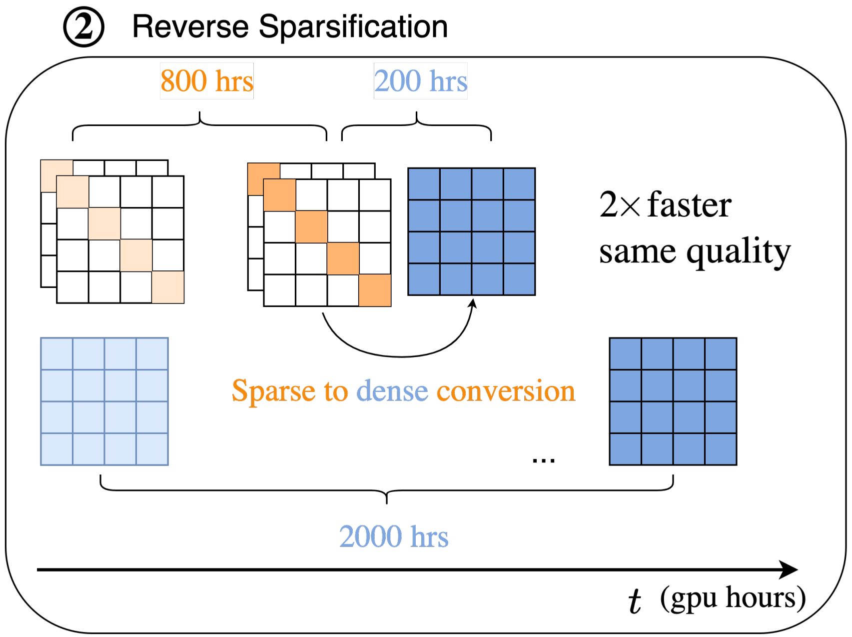


Other applications: PDEs solving, MRI reconstruction

Speeds up training without losing performance 🚀!

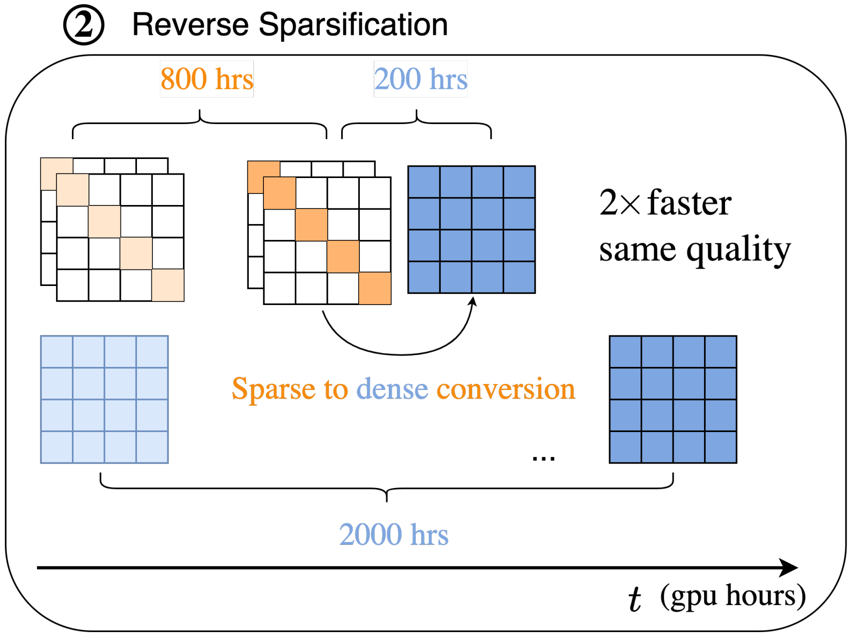
Validation: Sparse-to-Dense Training

BERT Implementation	Training time (h) to the same MLM accuracy
HuggingFace	84.5
Megatron	52.5
Nvidia MLPerf 1.1	30.2
Monarch	**23.8**



Validation: Sparse-to-Dense Training

BERT Implementation	Training time (h) to the same MLM accuracy
HuggingFace	84.5
Megatron	52.5
Nvidia MLPerf 1.1	30.2
Monarch	**23.8**

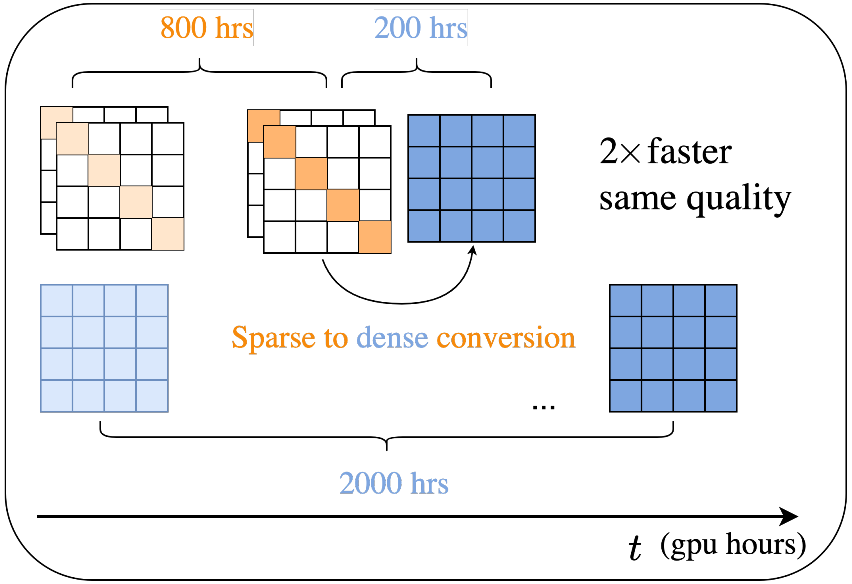


3.5x faster than HuggingFace, 23% faster than Nvidia's MLPerf

Validation: Sparse-to-Dense Training

BERT Implementation	Training time (h) to the same MLM accuracy
HuggingFace	84.5
Megatron	52.5
Nvidia MLPerf 1.1	30.2
Monarch	**23.8**
Monarch + FlashAttention	**21.5**

② Reverse Sparsification



Late breaking results: FlashAttention
Fast & mem-efficient exact attention
https://github.com/HazyResearch/flash-attention

3.5x faster than HuggingFace, 23% faster than Nvidia's MLPerf

Summary

Code: https://github.com/HazyResearch/monarch

Summary

Monarch: hardware-efficient, expressive matrices

Ways to use: sparse end-to-end training, sparse-to-dense training (reverse sparsification)

Upshot: wallclock-time speedup with sparse training, maintaining model quality

Code: https://github.com/HazyResearch/monarch

Summary

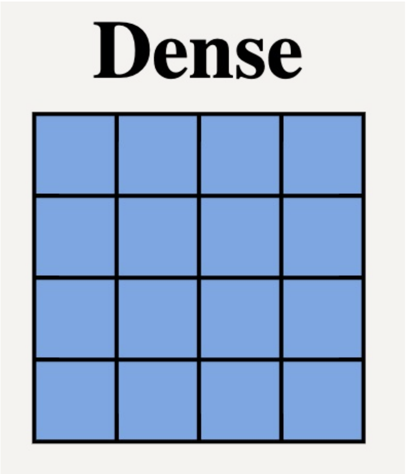
Monarch: hardware-efficient, expressive matrices

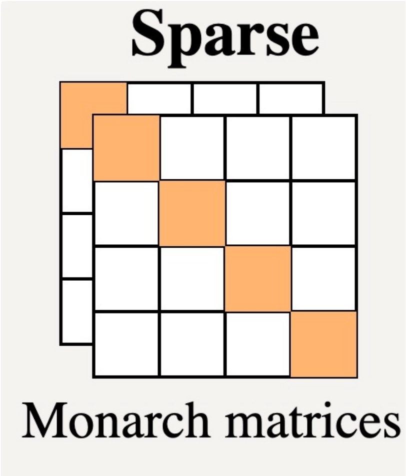
Ways to use: sparse end-to-end training, sparse-to-dense training (reverse sparsification)

Upshot: wallclock-time speedup with sparse training, maintaining model quality

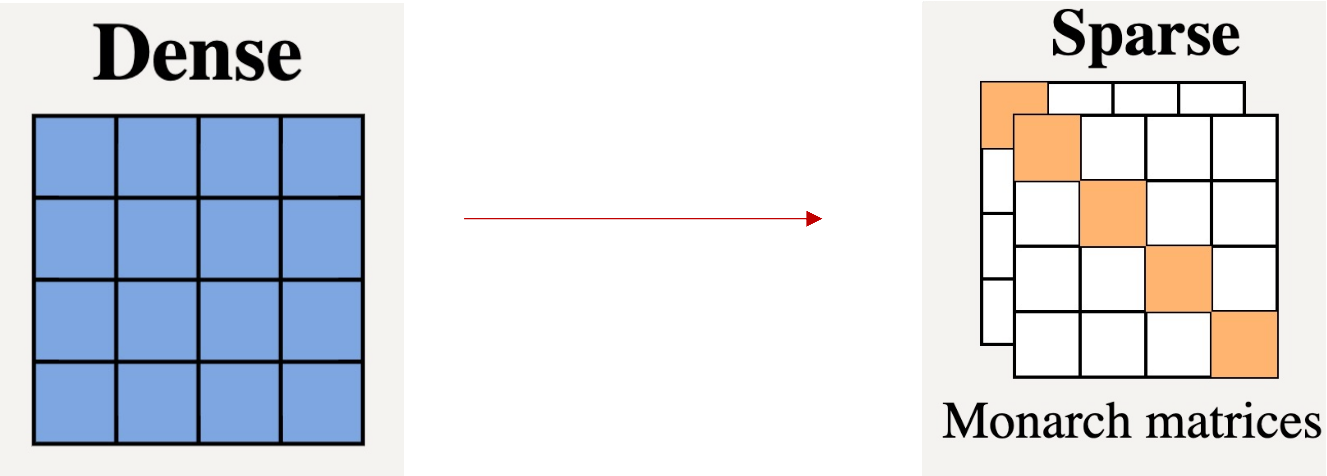
Code: https://github.com/HazyResearch/monarch

Tractable Projection



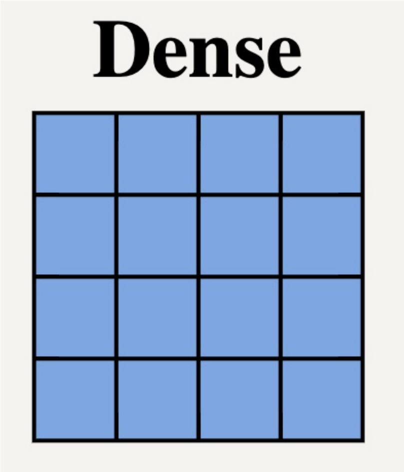


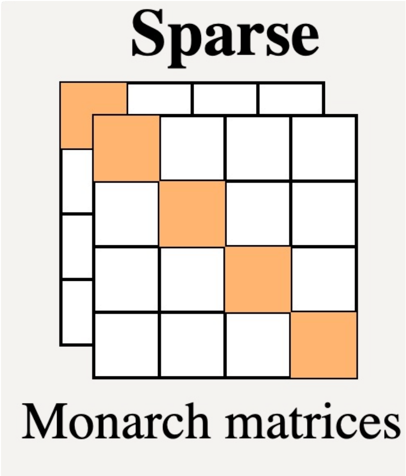
Tractable Projection



For a given dense matrix A, find $\underset{\mathbf{M}\in\mathcal{M}}{\operatorname{argmin}} \|\mathbf{A}-\mathbf{M}\|_F^2$

Tractable Projection

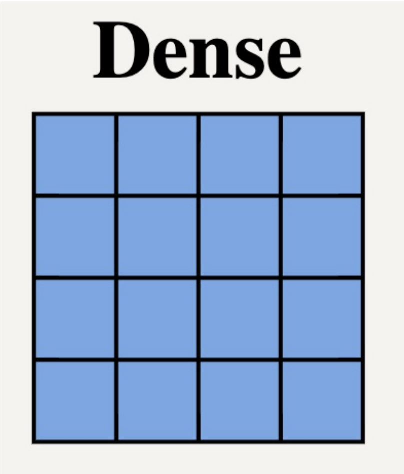


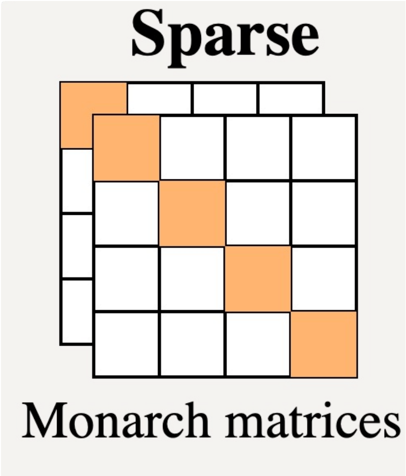


For a given dense matrix A, find $$\underset{\mathbf{M}\in\mathcal{M}}{\operatorname{argmin}} \|\mathbf{A}-\mathbf{M}\|_F^2$$

Hard problem for most classes of matrices

Tractable Projection



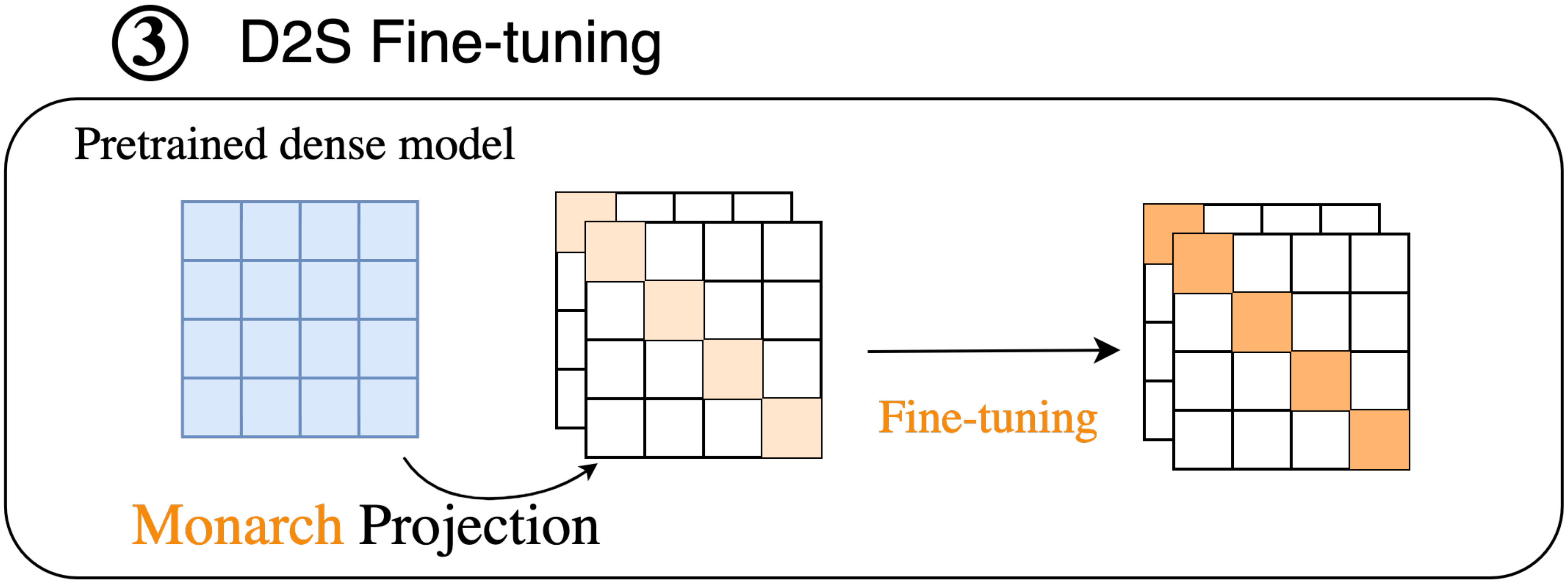


For a given dense matrix A, find $\underset{\mathbf{M} \in \mathcal{M}}{\operatorname{argmin}} \|\mathbf{A} - \mathbf{M}\|_F^2$

Hard problem for most classes of matrices

Monarch: tractable projection algorithm, analogous to the SVD

Dense-to-sparse finetuning



Validation: dense-to-sparse finetuning

Model	GLUE (avg)	Speedup
BERT-large	80.4	-
Monarch BERT-large	79.6	**1.7x**

Validation: dense-to-sparse finetuning

Model	GLUE (avg)	Speedup
BERT-large	80.4	-
Monarch BERT-large	79.6	**1.7x**

Speed up finetuning by trading off some model quality.