Pixelated Butterfly:

Simple and Efficient Sparse Training for Neural Network Models

Presenter: Beidi Chen

Collaborators: Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, Christopher Ré

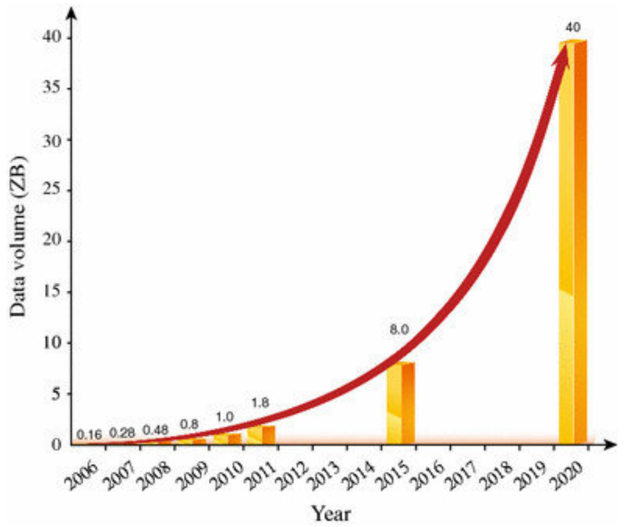
Stanford University

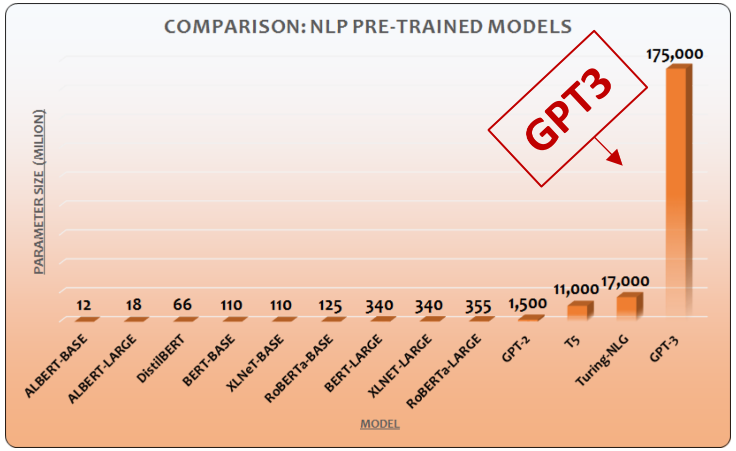
Neural Network (NN) Training Bottlenecks

Exponentially Growing Data Volume

Larger Model Size

More Resources





Training large-scale models imposes challenges on computational and memory resources.

middle plot by Kirk Borne on Twitter

A Simple & Popular Direction: Sparsify Models

Sparsity is not new!

- has a long history in machine learning (Lecun et al. 90)
- stats (Tibshirani et al. 96), neuroscience (foldiak et al. 03), signal processing (Candes et al. 05)

Existing approaches:

- Pruning: Deep Comp (Han et al. 16), Lottery Tickets (frankle et al. 18), RigL (Evci et al. 20) …
- Approx. matmul: Reformer (Kitaev et al. 20), Kaleidoscope (Dao et al. 20)…
  SLIDE, Mongoose, Scatterbrain (Chen et al. 20 & 21a & 21b)

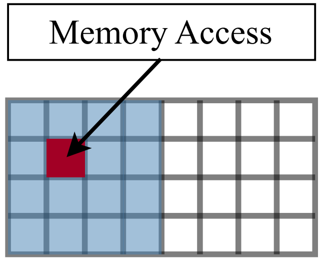
It is still hard to speed up training without degrading accuracy on the available hardware for DL.

We'll show how simple & static sparsity can speed up GPT-2, ViT and MLP-Mixer training by 2.5x in wall-clock time with no drop in accuracy.

Challenges & Goals

Challenges

- Dynamic sparsity can maintain accuracy but slow down training time
  - SOTA require up to 5x more epochs (Evci et al., 20)
- Unstructured Sparsity is not hardware-efficient (Hooker et al. 20)



- Sparse Attention target one module and thus does not speed up all layers
  - In many applications the MLP layers are the training bottleneck (Wu et al., 20)

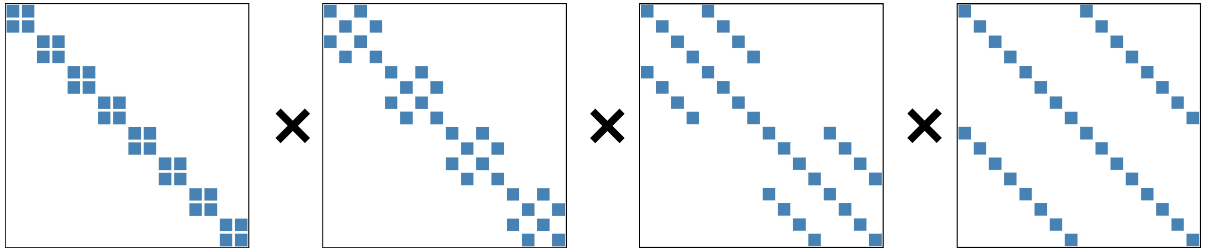
Ideal Sparsity

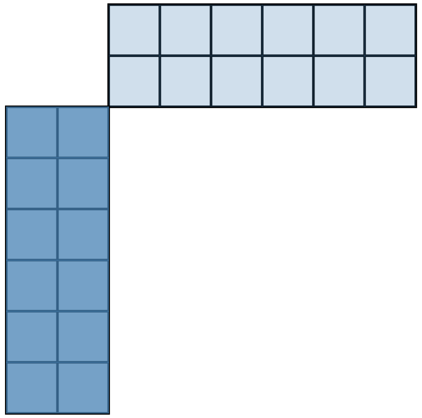
- Static, simple yet accurate
- Aligned with available hardware
- Applied to most NN layers

Observation: Butterfly + Low-rank is a simple & effective fixed sparsity pattern

Sparse + Low-rank

Captures global and local information (Candes et al., 2009, Chen et al., 2021)





Butterfly (cooley&Tukey 1965, Dao et al., 2019)

Low-rank (Hotelling et al., 1933, udell 2019)

Outline

Part 1 **Background & Observation**

Sparse + low-rank approx. to attention matrices, butterfly matrices
Observation: Butterfly + low-rank is an effective fixed sparsity pattern

Part 2 **Pixelated Butterfly**

Flat & block butterfly matrices
Analysis: Retain expressiveness & global convergence

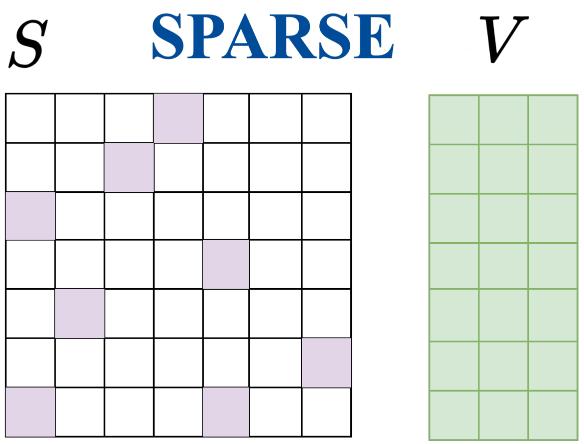
Part 3 **Applications**

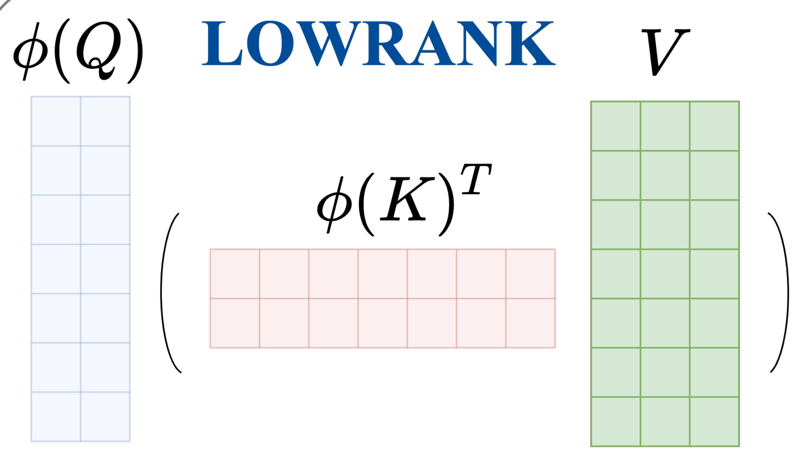
End-to-end training, downstream evaluation, empirical Neural Tangent Kernel
Experiments: performance on a wide range of vision and language tasks

Attention approximation: Sparse or Low-rank

Attention approximation → trade accuracy for efficiency

Sparse Transformer
(Child et al. 19)
Reformer
(Kitaev et al. 20)
Routing Transformer
(Roy et al. 20)

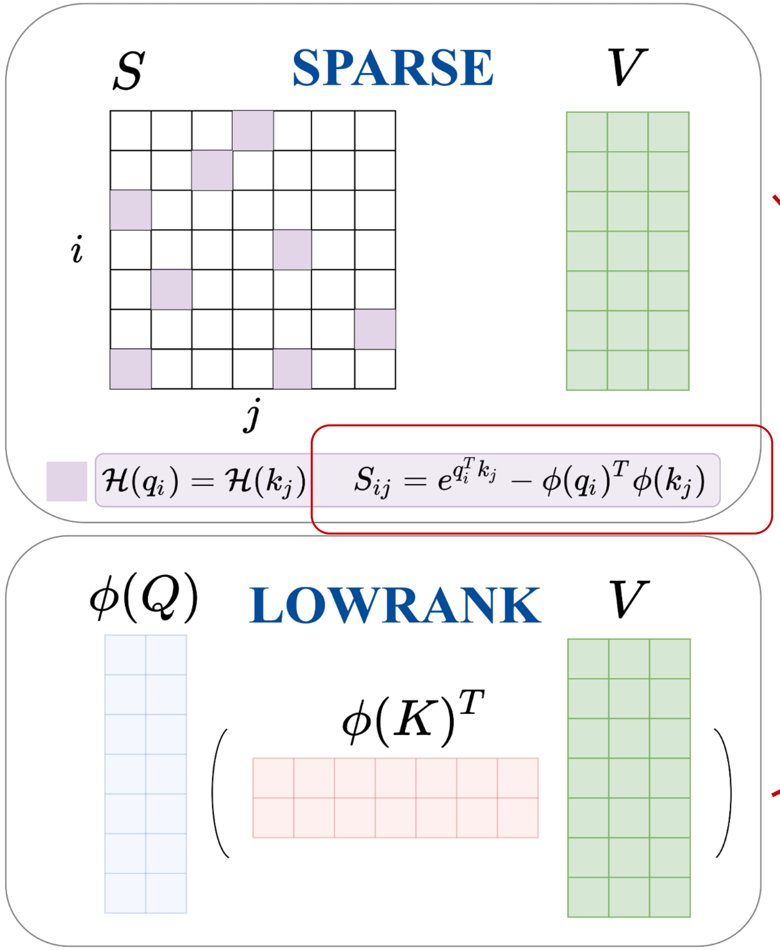




Linformer
(Wang et al. 20)
Linear Transformer
(Katharopoulos et al. 20)
Performer
(Choromanski et al. 20)

It is hard to find a robust approx. that performs well on a wide variety of tasks.

Sparse + Low-rank improves on either sparse / low-rank



Well-studied in stats and signal processing (Candes et al. 09)

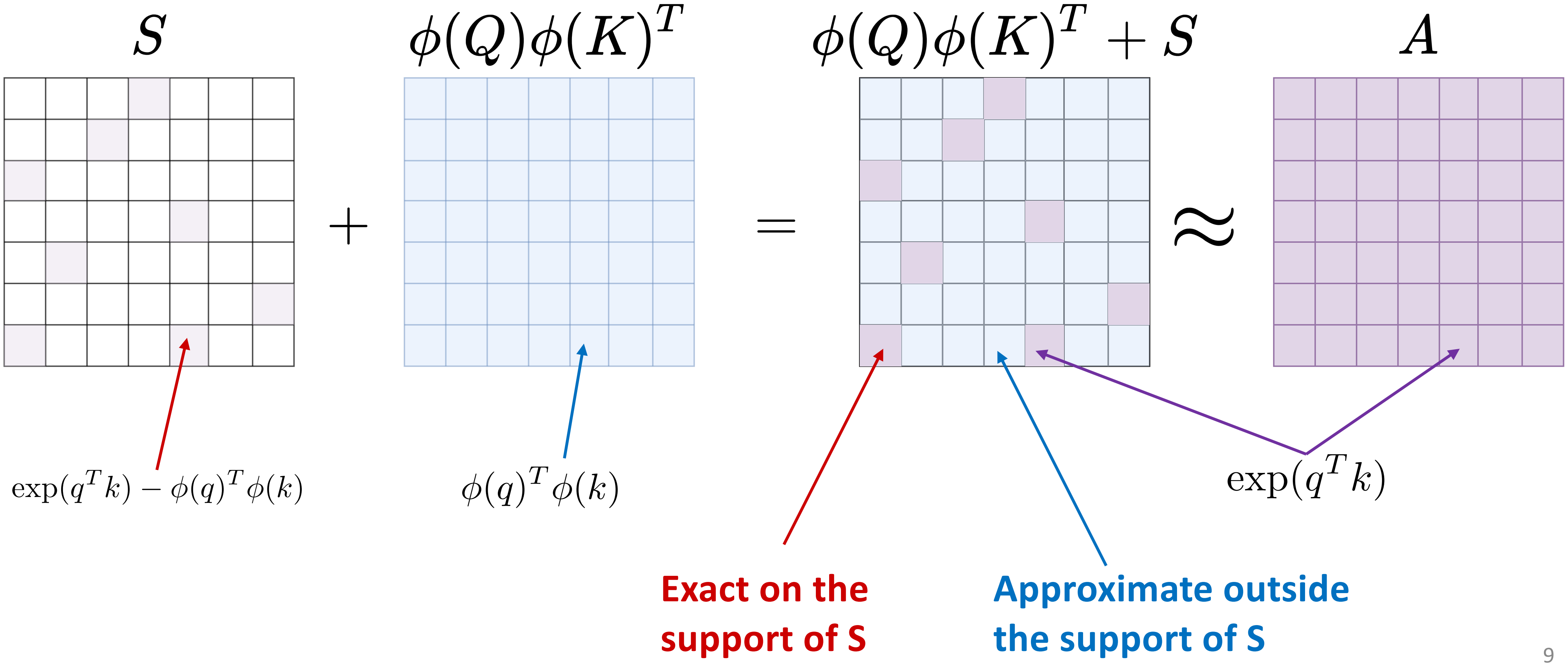
An example in (Chen et al. 21b):
Reformer (sparse) + Performer (low-rank)

$$\boxed{SV} + \boxed{\phi(Q)(\phi(K^T)V)} \approx \mathrm{softmax}(QK^T)V$$

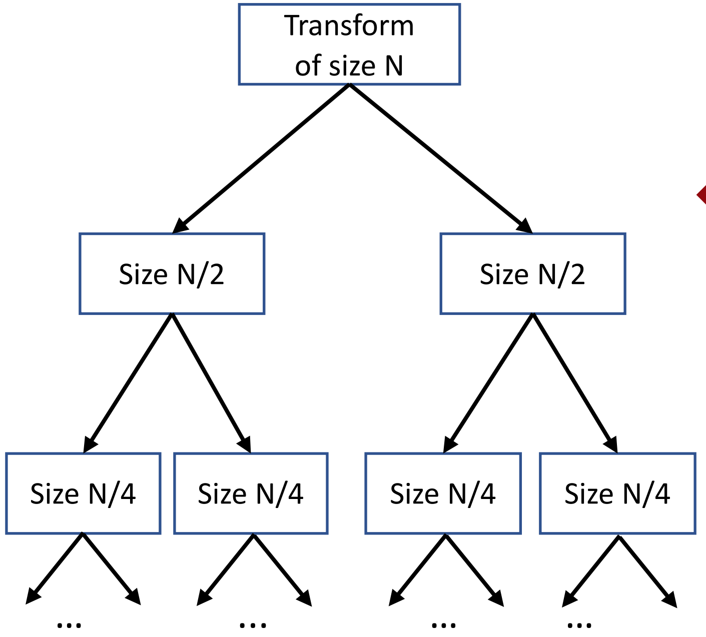
More examples:
Long-short Transformer (Zhu et al. 21)
Scalable Optimal Transport(Klicpera et al. 21)

Scatterbrain: combine Sparse and Low-rank Attention

Simple insight: discount low-rank contribution at sparse locations



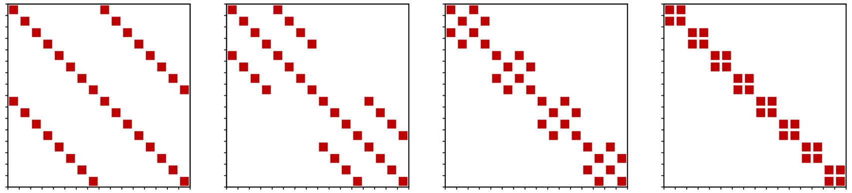
Butterfly matrices: Divide-and-Conquer



Recursive divide-and-conquer

(De Sa et al., 18)

$$\left(B_N \begin{bmatrix} B_{N/2} & 0 \\ 0 & B_{N/2} \end{bmatrix} \cdots \begin{bmatrix} B_2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & B_2 \end{bmatrix} \right)$$



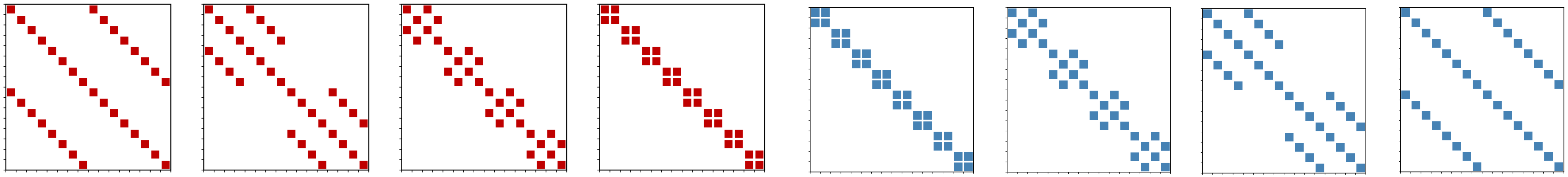
(Parker, 95; Matthieu & LeCun, 14;
Dao et al., 19, Roberts et al., 21)

Trainable with gradient descent on nonzero entries of butterfly matrix.

Captures recursive divide-and-conquer structure.

Butterfly matrices can represent ANY Sparse matrix

Deep composition of butterfly matrices: $B^{(1)} {B^{(2)}}^T B^{(3)} {B^{(4)}}^T B^{(5)} {B^{(6)}}^T \ldots$



$B^{(1)}$ ${B^{(2)}}^T$

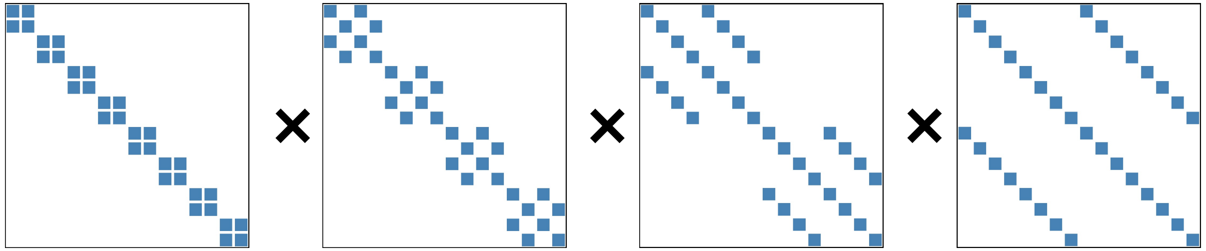
Butterfly matrix: **Fixed** sparsity

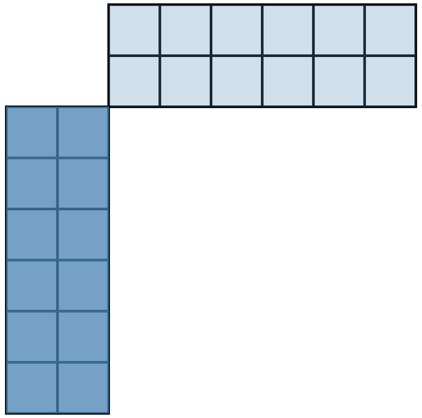
Provably capture any sparse matrix with near-optimal space and time complexity

Observation: Butterfly + Low-rank is a simple & effective fixed sparsity pattern

Sparse + Low-rank

Captures global and local information (Candes et al., 2009, Chen et al., 2021)





Butterfly (cooley&Tukey 1965, Dao et al., 2019)

Low-rank (Hotelling et al., 1933, udell 2019)

Butterfly + Low-rank can: (1) avoid dynamic overhead (3) apply to most matmul-based layers
(2) but it is not hardware-efficient

Outline

Part 1 **Background & Observation**

Sparse + low-rank approx. to attention matrices, butterfly matrices
Observation: Butterfly + low-rank is an effective fixed sparsity pattern

Part 2 **Pixelated Butterfly**

Flat & block butterfly matrices
Analysis: Retrain expressiveness & global convergence

Part 3 **Applications**

End-to-end training, downstream evaluation, empirical Neural Tangent Kernel
Experiments: performance on a wide range of vision and language tasks

Issues of Butterfly matrices

Issues

- Slow speed: Sparsity patterns are not block-aligned → not friendly to modern hardware.
- Difficulty of parallelization: They are products of many factors → sequential operations
- Reduced expressiveness: Flat Butterfly are necessarily high-rank → cannot represent low-rank matrices

Pixelated Butterfly

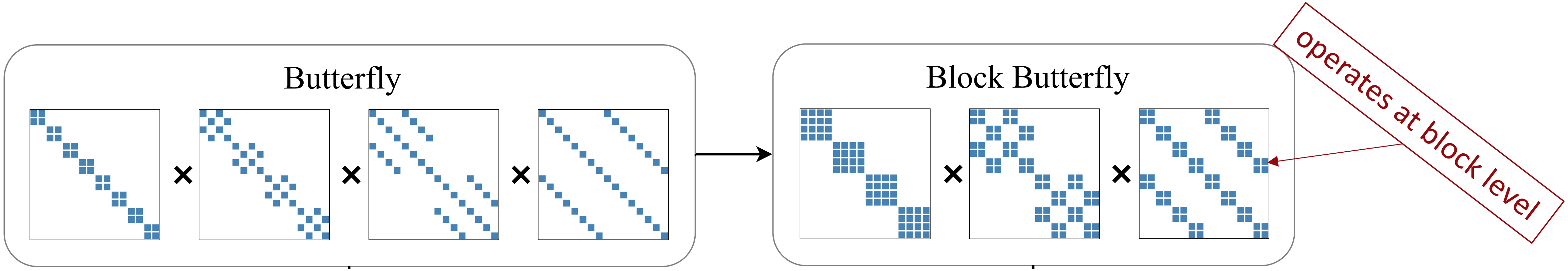
- Block Butterfly: Block-aligned sparsity pattern.
- Flat Butterfly: First order approximation of butterfly, turning product into sum.
- Low-rank term: Increase expressiveness of Flat Block Butterfly matrices.

Issues of Butterfly matrices

Issues	Pixelated Butterfly
Slow speed: Sparsity patterns are not block-aligned → not friendly to modern hardware.	Block Butterfly: Block-aligned sparsity pattern.
Difficulty of parallelization: They are products of many factors → sequential operations	Flat Butterfly: First order approximation of butterfly, turning product into sum.
Reduced expressiveness: Flat Butterfly are necessarily high-rank → cannot represent low-rank matrices	Low-rank term: Increase expressiveness of Flat Block Butterfly matrices.

Our Approach: Flat & Block Butterfly

Problem 1: Not block-aligned



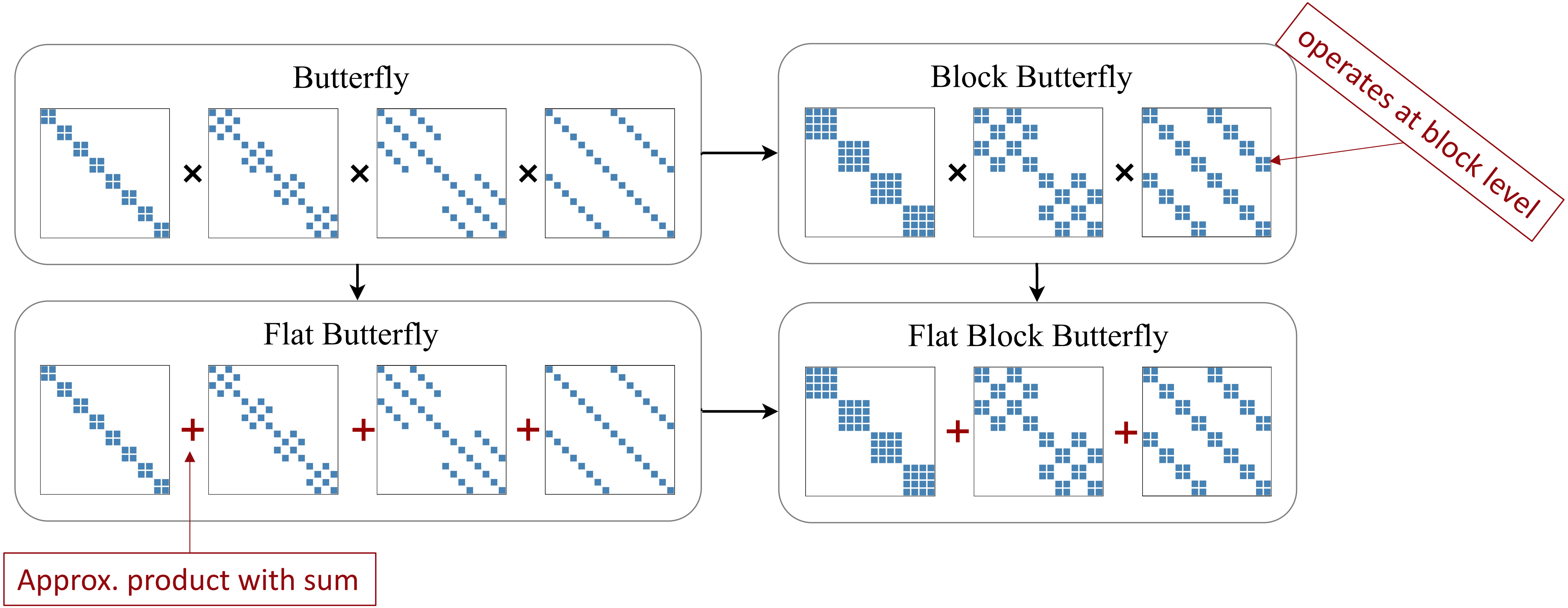
Issues of Butterfly matrices

Issues	Pixelated Butterfly
• Slow speed: Sparsity patterns are not block-aligned → not friendly to modern hardware.	• Block Butterfly: Block-aligned sparsity pattern.
• Difficulty of parallelization: They are products of many factors → sequential operations	• Flat Butterfly: First order approximation of butterfly, turning product into sum.
• Reduced expressiveness: Flat Butterfly are necessarily high-rank → cannot represent low-rank matrices	• Low-rank term: Increase expressiveness of Flat Block Butterfly matrices.

Our Approach: Flat & Block Butterfly

Problem 1: Not block-aligned

Problem 2: Hard to parallelize the product of many factors



Issues of Butterfly matrices

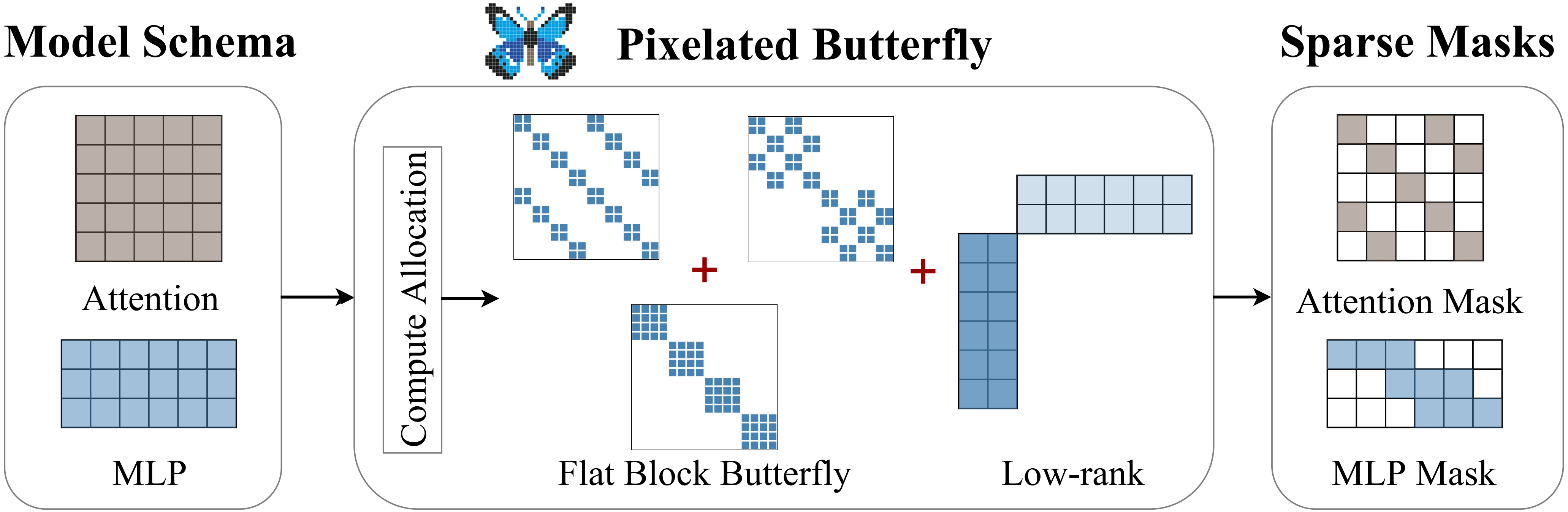
Issues

- Slow speed: Sparsity patterns are not block-aligned → not friendly to modern hardware.
- Difficulty of parallelization: They are products of many factors → sequential operations
- Reduced expressiveness: Flat Butterfly are necessarily high-rank → cannot represent low-rank matrices

Pixelated Butterfly

- Block Butterfly: Block-aligned sparsity pattern.
- Flat Butterfly: First order approximation of butterfly, turning product into sum.
- Low-rank term: Increase expressiveness of Flat Block Butterfly matrices.

Pixelated Butterfly Workflow



Compute Budget Allocation

Sparsity Mask Selection

Model Sparsification

Theoretical properties of Pixelated Butterfly

Theorem 1 [Informal]: Block butterfly retains the expressiveness of Butterfly and flat butterfly can accurately approximate the residual form of butterfly. (Dao et al. 20)

Theorem 2: Flat block butterfly + low-rank is more expressive than sparse or low-rank matrices alone. (Chen et al. 20)

Theorem 3: Training wide an sparse networks with gradient descent converges globally, similar to the result for wide dense networks. (dzps19, als19)

Intuition: Pixelated butterfly inherits all the nice properties of butterfly matrices, sparse + low-rank matrices, and sparse training.

Outline

Part 1 **Background & Observation**

Sparse + low-rank approx. to attention matrices, butterfly matrices
Observation: Butterfly + low-rank is an effective fixed sparsity pattern

Part 2 **Pixelated Butterfly**

Flat & block butterfly matrices
Analysis: Retrain expressiveness & global convergence

Part 3 **Applications**

End-to-end training, downstream evaluation, empirical Neural Tangent Kernel
Experiments: performance on a wide range of vision and language tasks

Evaluation* 1: Image Classification

Model	ImageNet (Top1 Acc)	CIFAR10	CIFAR100	Speedup
Mixer-B/16	75.6	87.6	59.5	-
Pixerfly-Mixer-B/16	**76.3**	**90.6**	**65.4**	**2.3 x**
ViT-B/16	78.5	89.9	61.9	-
Pixerfly-ViT-B/16	**78.6**	**92.2**	**65.1**	**2.0 x**

Pixelated butterfly is up to 2.3x faster (wall-clock) than dense MLP-Mixer and Vision Transformer models without accuracy loss.

*More results in paper

Evaluation 2: Language Modeling & Classificaiton

Model	WikiText103(ppl)	Speedup
GPT-2 Small	22.2	-
BigBird	23.3	0.96 x
Pixerfly-Small	22.5	**2.1 x**
GPT-2 Medium	21.5	**-**
BigBird Medium	21.5	1.1 x
Pixerfly-Medium	21.0	**2.5 x**

Model	Long Range Arena (avg Acc)	Speedup
Transformers	59.01	-
Reformer	53.9	0.8 x
Pixerfly	**59.86**	**5.2 x**

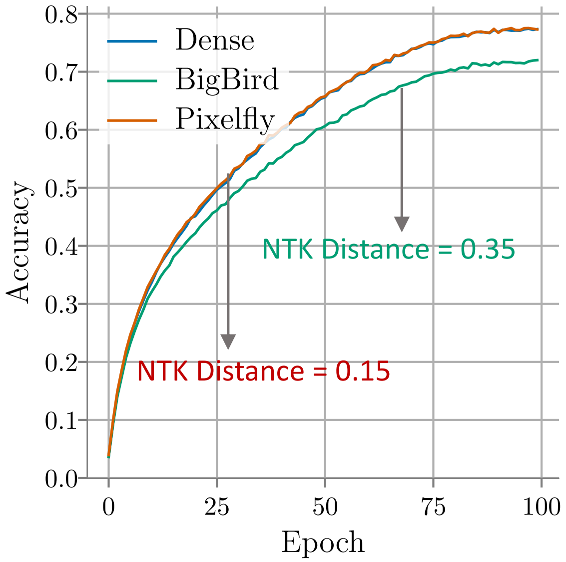
Pixelated butterfly is up to 2.5x faster (wall-clock) than dense GPT-2, 5x faster than vanilla Transformer without accuracy loss.

Extended Evaluations: Downstream Tasks & NTK

Upstream task: OpenWebText

Model	WikiText (ppl)[1]	Lambada (acc)[1]	Classification (avg acc)[2]
GPT-2 Medium	31.87	35.4	33.2
Pixerfly Medium	30.5	38.9	33.4

Empirical NTK: relative differences between kernels of models with different sparsity patterns & that of the dense one.



[1] https://github.com/EleutherAI/lm-evaluation-harness
[2] Zhao et al. 2021

Conclusion and Future Directions

Conclusion

Early Exploration: A simple pattern, butterfly + low-rank consistently performed among the best.

Proposal: Pixelated butterfly, a static and block sparsity pattern that aligns with modern hardware.

Result: Train GPT-2, ViT and MLP-Mixer up to 2.5x faster on GPU.

Future Directions

Pixelfly 2.0: Going beyond dense models in applications, e.g., PDE solving & MRI reconstruction.

Pixelfly-hardware Co-design: Efficient sparsity / butterfly on next generation of ML accelerators.

Pixelfly meets data sparsity: Find structures in data and speed up training from a different angle.

THANKS!

Paper: https://arxiv.org/abs/2112.00029
Code: https://github.com/HazyResearch/pixelfly

Contact: beidic@stanford.edu