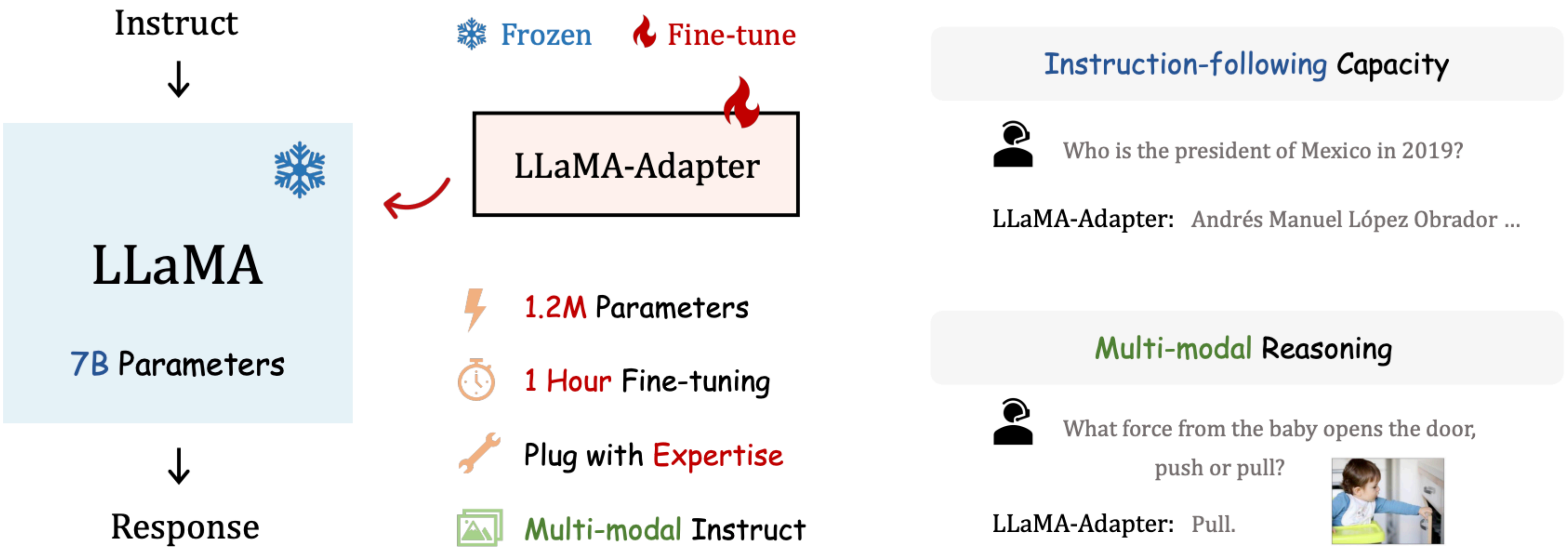
Parameter-Efficient Orthogonal Finetuning via Butterfly Factorization

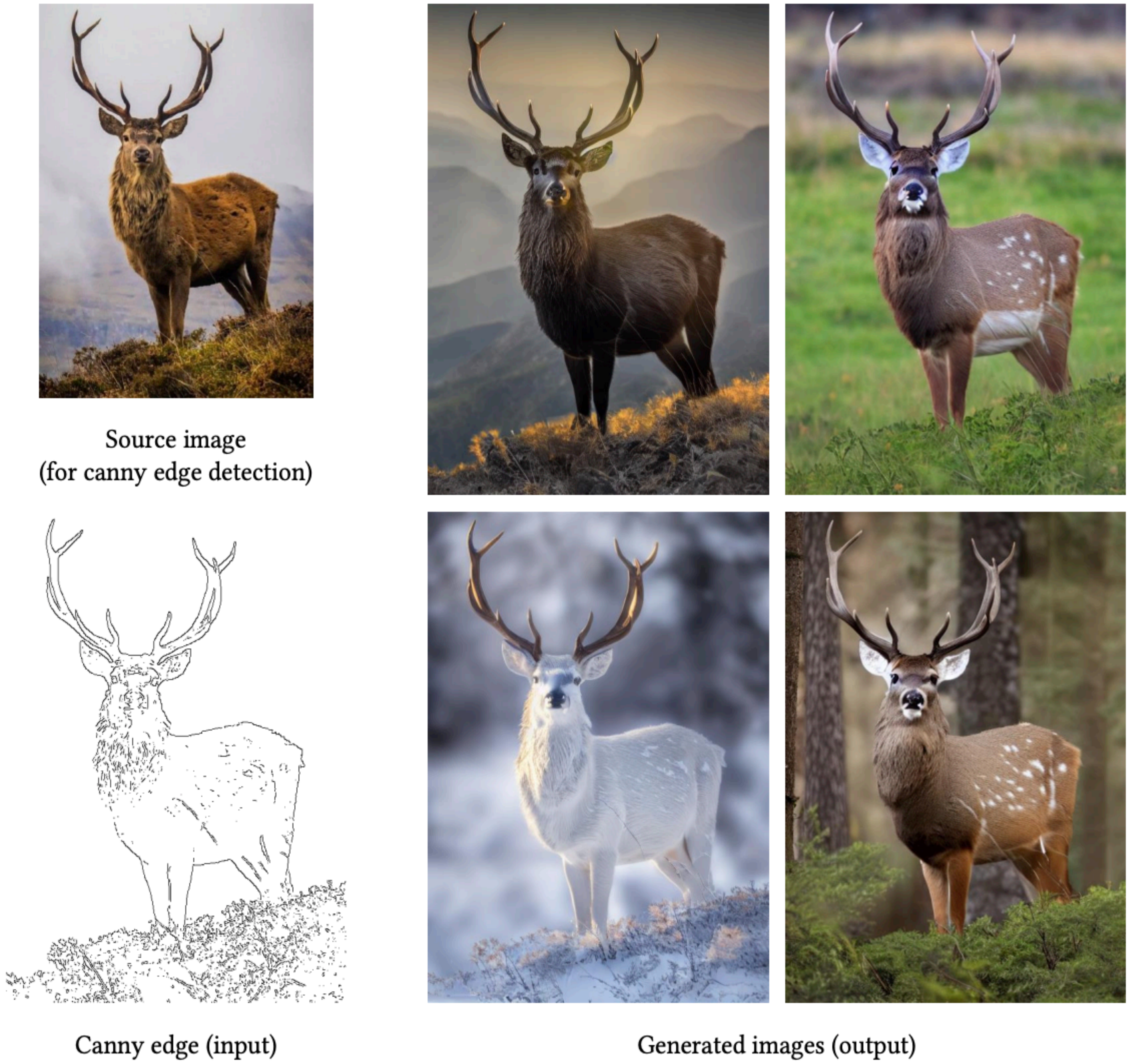
Weiyang Liu*, Zeju Qiu*, Yao Feng**, Yuliang Xiu**, Yuxuan Xue**, Longhui Yu**, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, Bernhard Schölkopf

Adaptation of foundation models is ubiquitous

DreamBooth: subject-driven generation



Instruction following

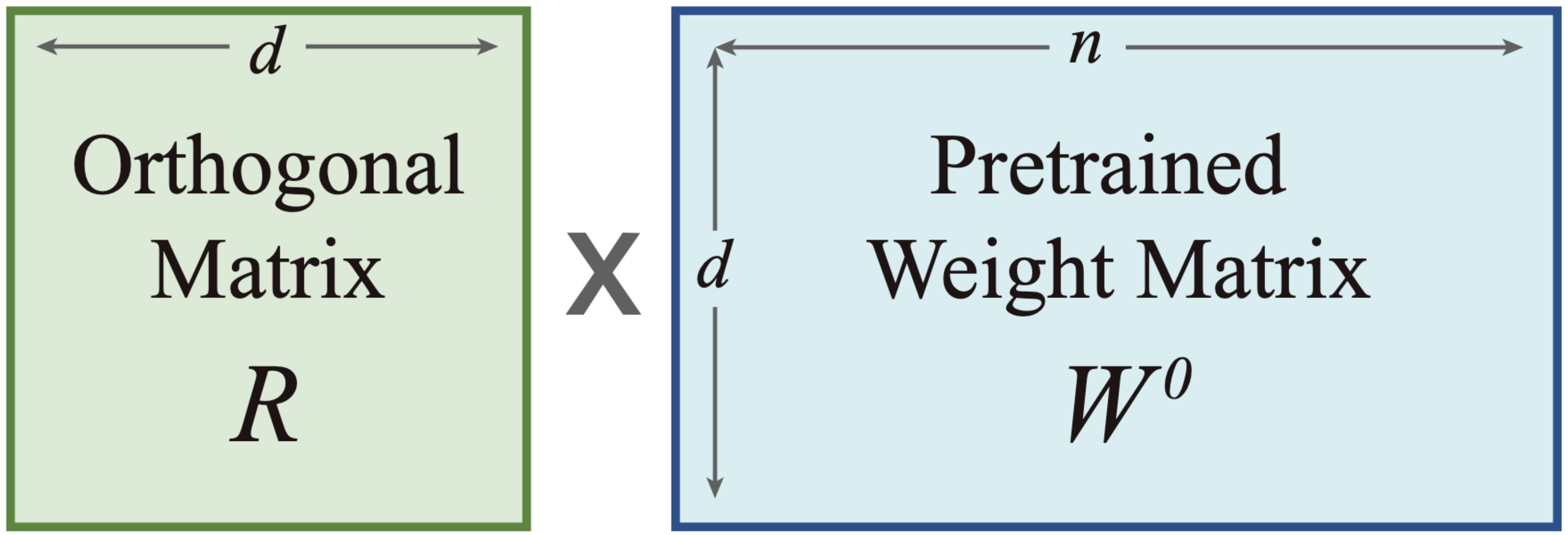


ControlNet: controllable generation

An effective way of finetuning foundation models is very important!

Orthogonal Finetuning

- **Key idea**: Angular information in neurons preserves semantics, so finetuning should preserve the angles between neurons.
- **Method**: Learn orthogonal multiplicative weight update

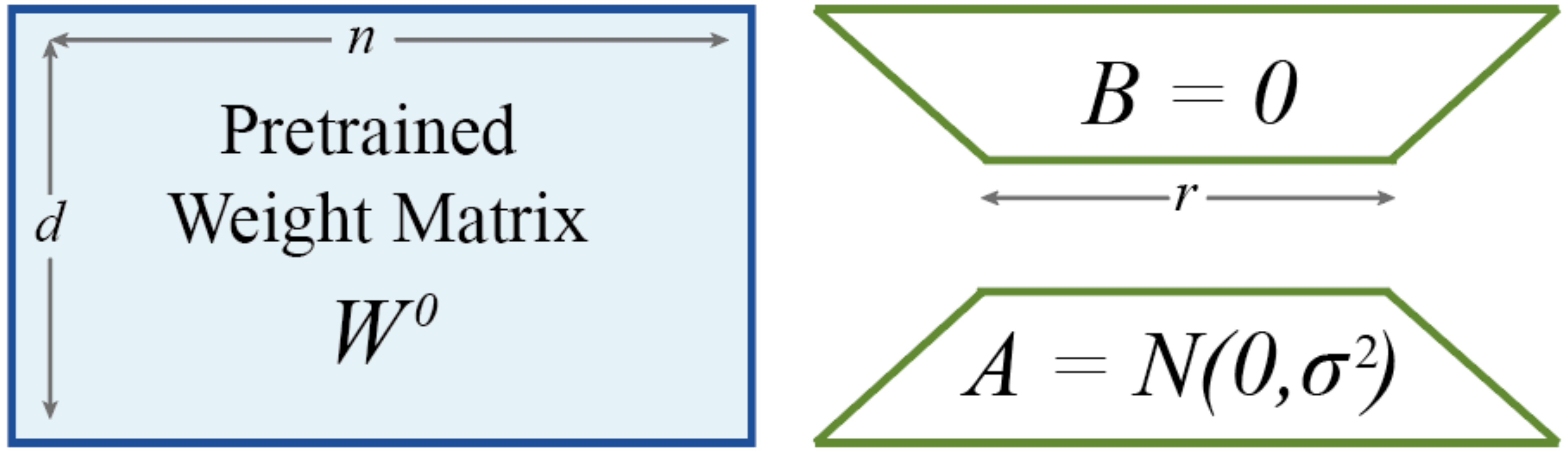


The hyperspherical energy does not change under the orthogonal transformation!

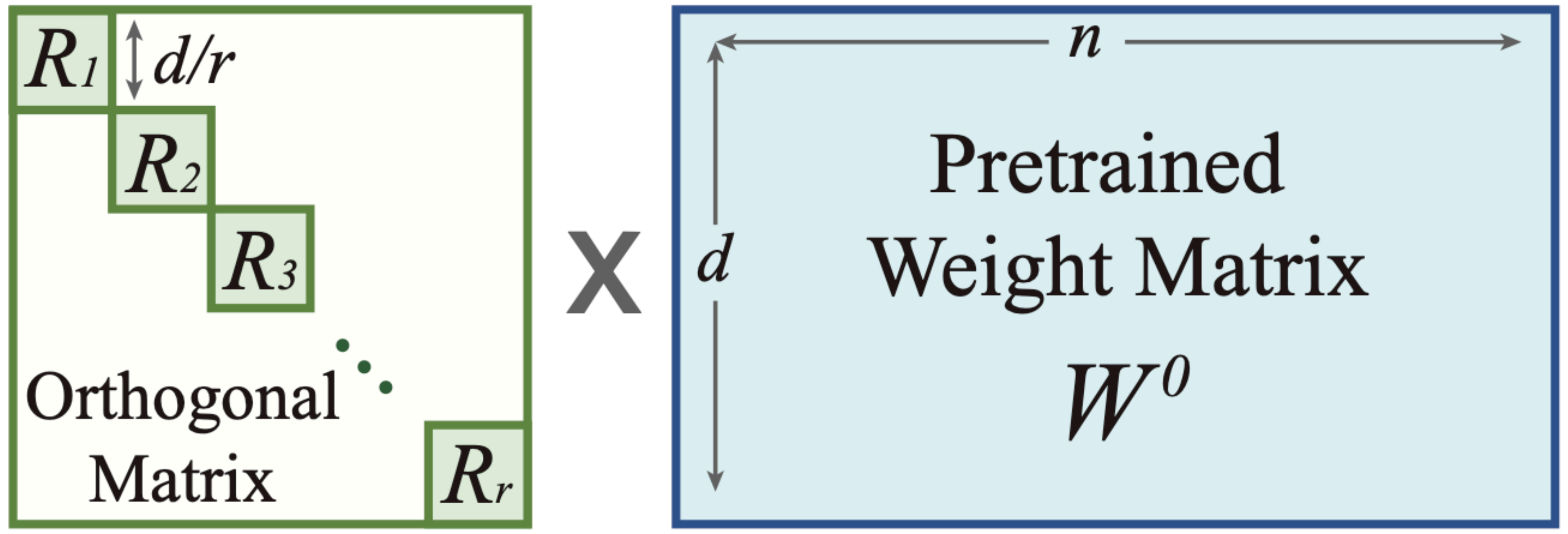
Qiu*, ***Liu****, et al. **Controlling Text-to-Image Diffusion by Orthogonal Finetuning**, NeurIPS 2023

Comparison to Low-Rank Adaptation (LoRA)

- LoRA uses a low-rank additive weight update:

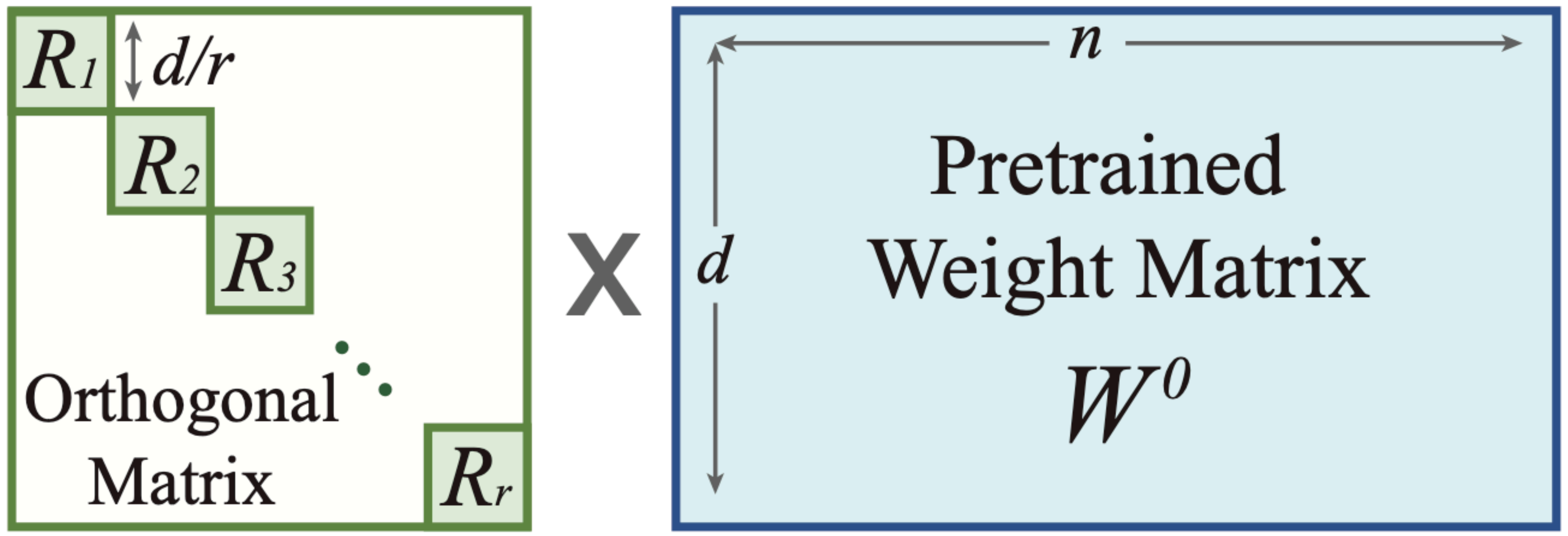


- The block diagonal structure in OFT acts like the low-rank structure in LoRA



Hu, et al. **LoRA: Low-Rank Adaptation of Large Language Models**, ICLR 2022

Revisit OFT's Parameter-efficiency



Sparse orthogonal matrix

Why the block-diagonal structure?

What about other sparsity pattern?

How to improve the expressiveness?

We need a dense orthogonal matrix!

Parameter-efficiency vs. Dense connectivity

The Problem

- Orthogonal transformation happens separately in different blocks.
  - Makes no sense to group dimensions in advance
  - Less flexible and expressive for finetuning
- To address this problem, we have to produce a dense orthogonal matrix.

Parameter-efficiency vs. Dense connectivity

It seems impossible to have the best of both world.

Can we have a way to parameterize a dense orthogonal matrix while making it parameter-efficient?

Factorize into multiple sparse orthogonal matrices!

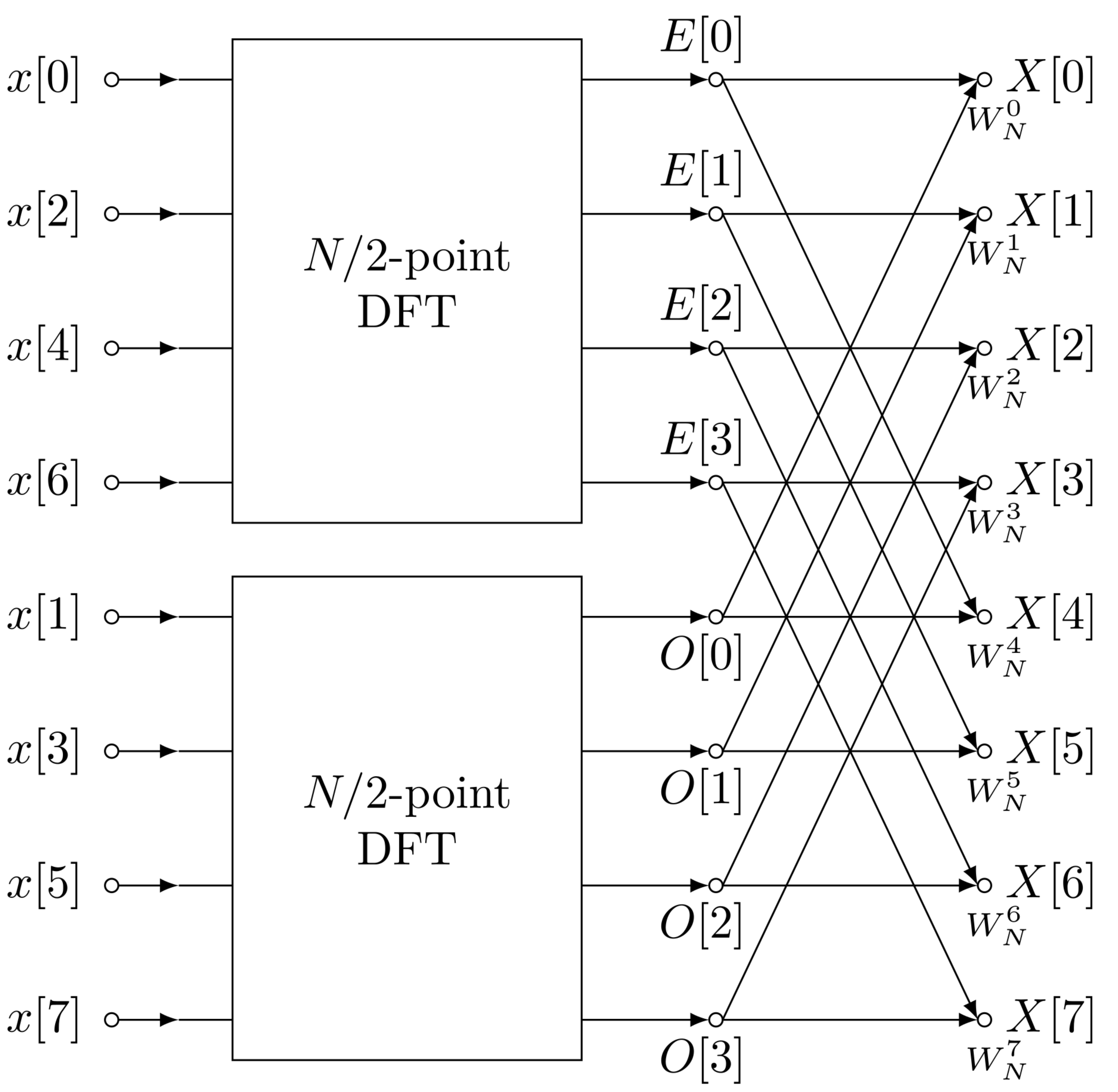
Inspiration

- Consider the fast Fourier transform algorithm:
  - recursive, divide-and-conquer

$$F_N x = \begin{bmatrix} F_{N/2} x_{\text{even}} + \Omega_{N/2} F_{N/2} x_{\text{odd}} \\ F_{N/2} x_{\text{even}} - \Omega_{N/2} F_{N/2} x_{\text{odd}} \end{bmatrix}$$

$$\downarrow$$

$$F_N = \begin{bmatrix} I_{N/2} & \Omega_{N/2} \\ I_{N/2} & -\Omega_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \begin{bmatrix} \text{Sort the even} \\ \text{and odd indices} \end{bmatrix}$$



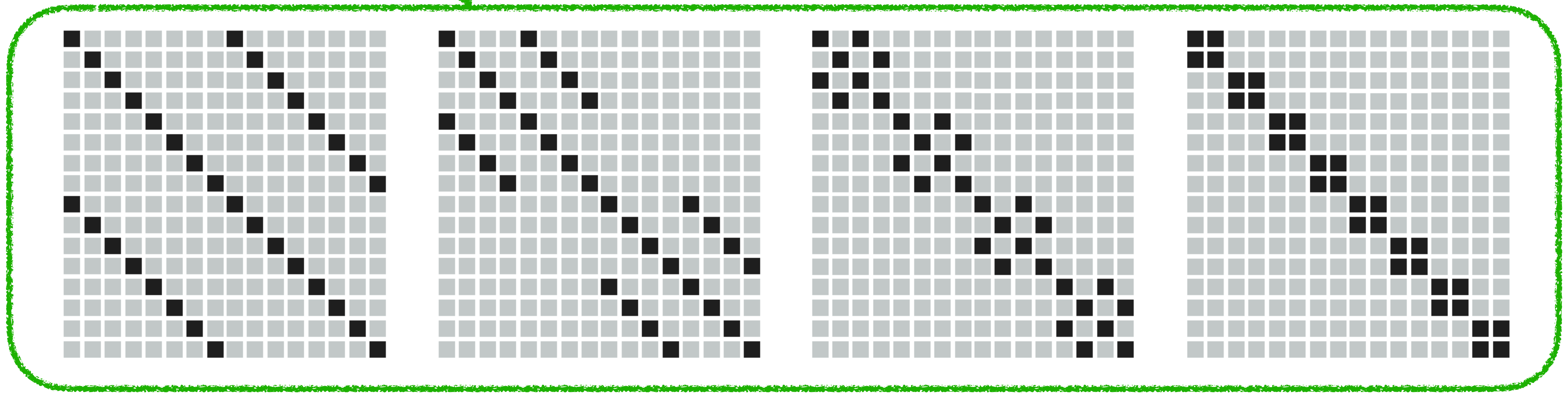
An efficient way to parameterize orthogonal matrices

$$
\begin{aligned}
F_N &= B_N \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} P_N \\
&= B_N \begin{bmatrix} B_{N/2} & 0 \\ 0 & B_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/4} & 0 & 0 & 0 \\ 0 & F_{N/4} & 0 & 0 \\ 0 & 0 & F_{N/4} & 0 \\ 0 & 0 & 0 & F_{N/4} \end{bmatrix} \begin{bmatrix} P_{N/2} & 0 \\ 0 & P_{N/2} \end{bmatrix} P_N \\
&= \cdots \\
&= \left(B_N \ldots \begin{bmatrix} B_2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & B_2 \end{bmatrix} \right) \left(\begin{bmatrix} P_2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & P_2 \end{bmatrix} \ldots P_N \right).
\end{aligned}
$$

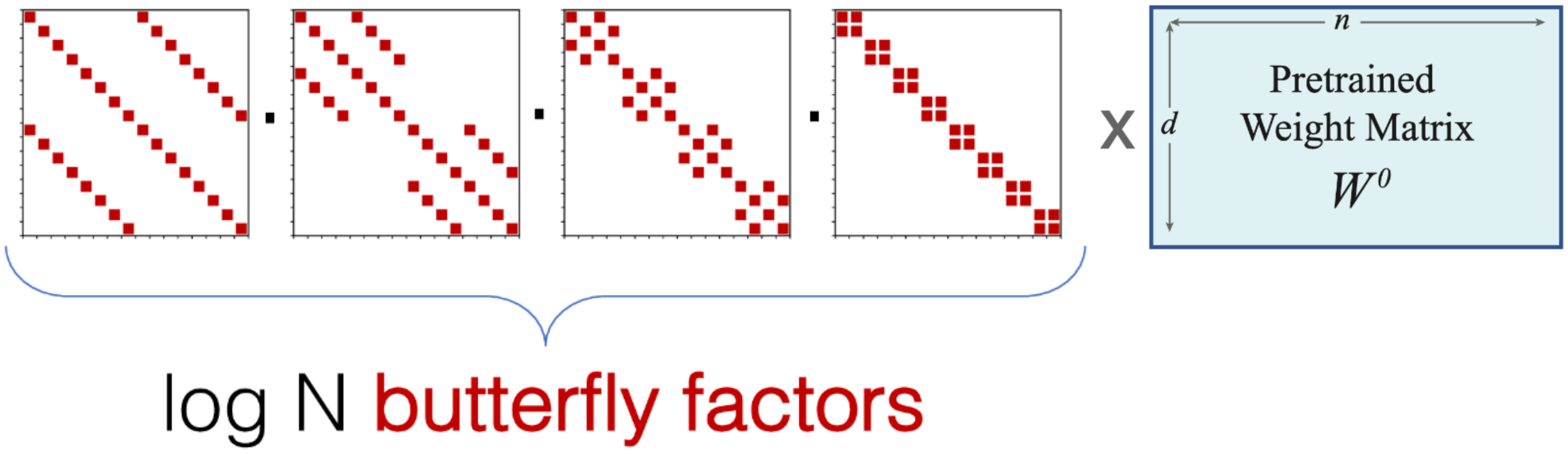
Butterfly matrix

Bit-reversal permutation

Sparsity pattern



Orthogonal Butterfly (BOFT)

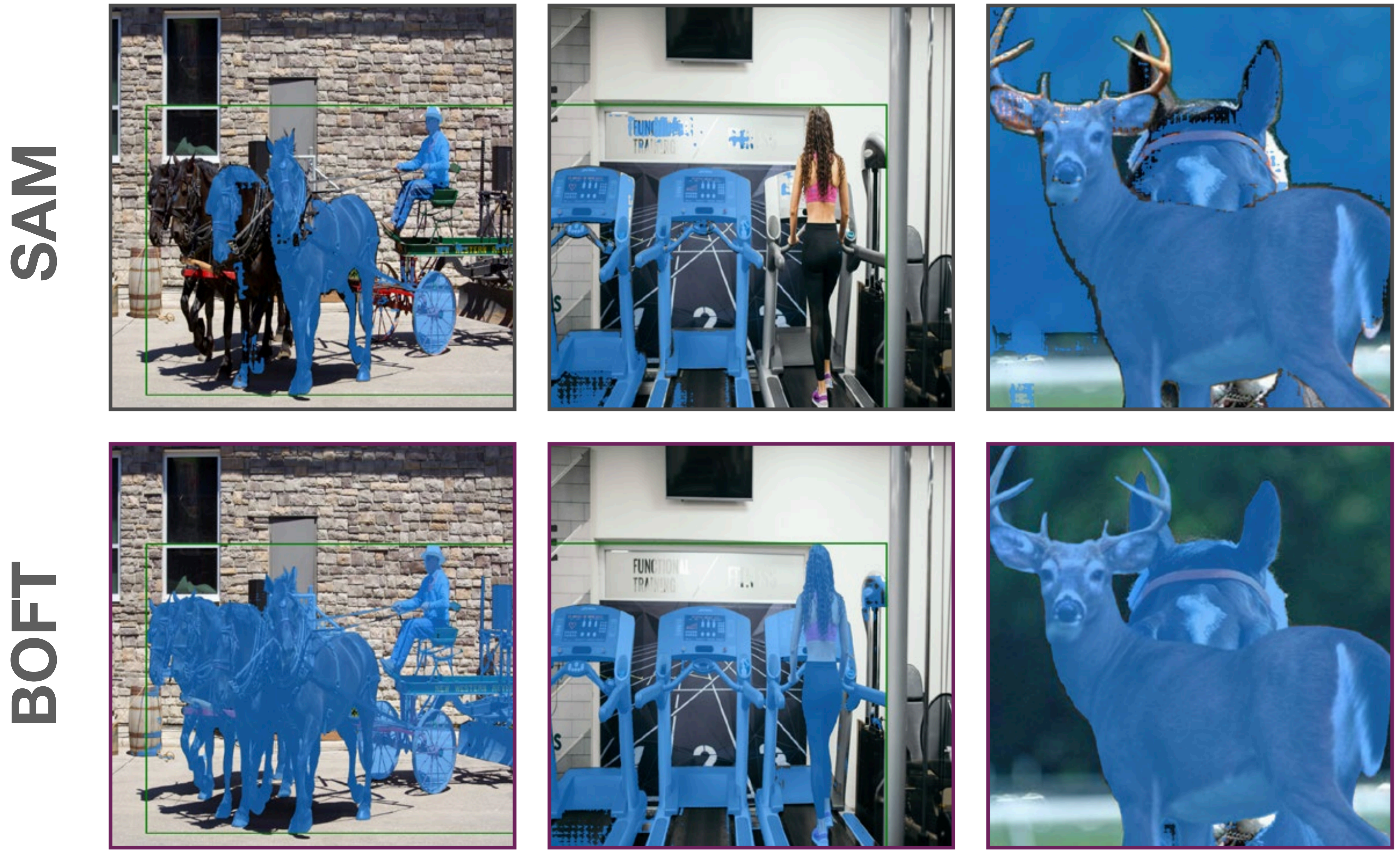


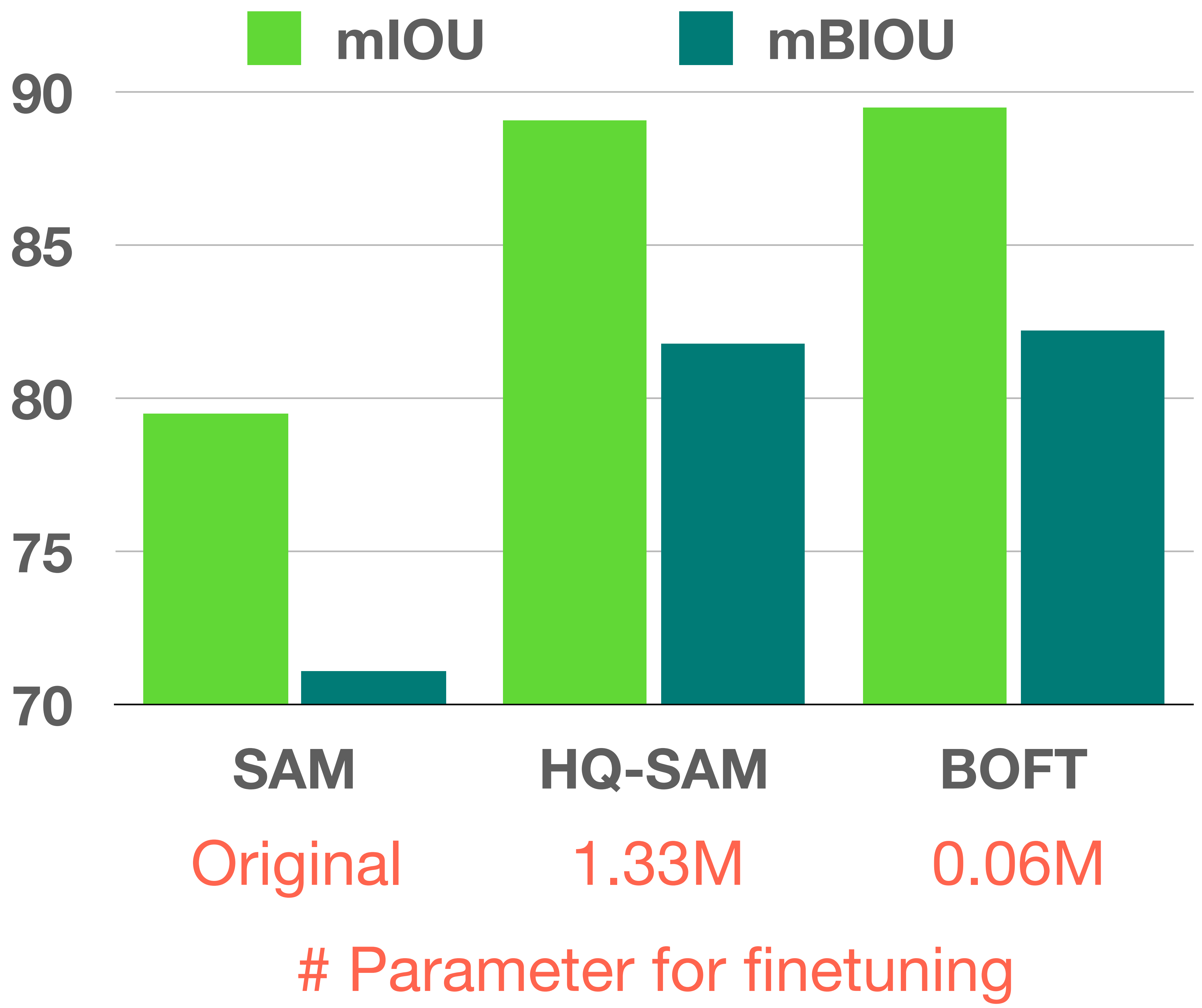
- Ensure each butterfly factor to be orthogonal
  - We simply need to ensure each 2x2 block is orthogonal!
- A more efficient parameterization
  - From ***O(d^2)*** to ***O(d* log *d)***

Liu*, et al. **Parameter-Efficient Orthogonal Finetuning via Butterfly Factorization**, ICLR 2024

Orthogonal Butterfly for Vision Tasks

- Finetuning Segment Anything (SAM):





Orthogonal Butterfly for NLP Tasks

- Finetuning Llama-2-7B on the Alpaca dataset and test on MMLU

Method	# Param	MMLU (5-shot)					MMLU (0-shot)				
		Hums.	STEM	Social	Other	Avg.	Hums.	STEM	Social	Other	Avg.
Llama-2-7B	-	43.0	36.9	51.6	52.1	45.7	38.8	33.3	46.8	45.0	40.8
LoRA$_{r=16}$	0.125%	42.9	38.5	54.5	53.8	47.0	42.5	37.1	51.5	52.3	45.5
LoRA$_{r=32}$	0.25%	42.9	38.7	**54.6**	54.7	47.3	42.5	36.7	52.8	52.7	45.9
OFT $_{b=16}$	0.13%	44.0	38.9	54.2	54.3	47.5	44.0	36.7	52.9	52.0	46.2
BOFT $^{m=2}_{b=8}$	0.12%	**44.5**	**39.0**	54.4	**55.1**	**47.9**	**44.3**	**37.4**	**53.1**	**52.8**	**46.7**

- Finetuning Llama-2-7B on MetaMathQA-40K and test on math questions

Method	# Param	GSM8K	MATH
Llama-2-7B	-	14.6	2.5
LoRA$_{r=32}$	0.25%	50.2	7.8
OFT $_{b=16}$	0.13%	50.1	8.4
BOFT $^{m=2}_{b=8}$	0.12%	**50.6**	**8.6**

Orthogonal Butterfly for NLP Tasks

- Finetuning Llama-2-7B on MetaMathQA-40K and test on math questions

Method	# Param	GSM8K	MATH
Llama-2-7B	-	14.6	2.5
LoRA$_{r=32}$	0.25%	50.2	7.8
OFT $_{b=16}$	0.13%	50.1	8.4
BOFT $^{m=2}_{b=8}$	0.12%	**50.6**	**8.6**

Example E.1: Case Study 1

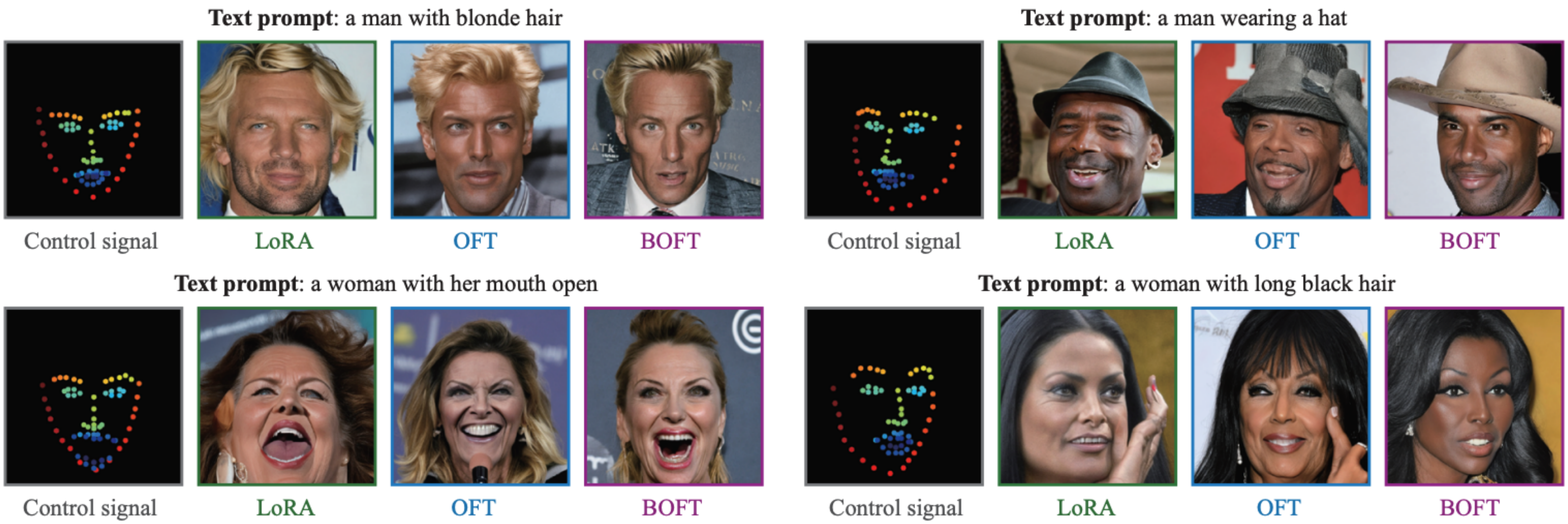
Question: James buys 5 packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. How much did he pay? **(Ground-truth answer is** 110)
Llama-2 Answer: James bought 5 packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. So he paid 5 * 4 * 5.50 = 137.50 dollars. The answer is 137.50.
BOFT Answer: James bought 5 packs of beef, and each pack is 4 pounds, so he bought a total of 5 * 4 = 20 pounds of beef. The price of beef is $5.50 per pound, so James paid 20 * $5.50 = $110. The answer is: 110

Example E.2: Case Study 2

Question: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week? **(Ground-truth answer is** 540)
Llama-2 Answer: James runs 60 meters each sprint. So he runs 60 * 3 = 180 meters each week. The answer is 180.
BOFT Answer: James runs 3 sprints 3 times a week, so he runs 3 sprints x 3 times = 9 sprints in a week. Each sprint is 60 meters, so James runs 9 sprints x 60 meters = 540 meters in a week. Therefore, James runs a total of 540 meters in a week. The answer is: 540

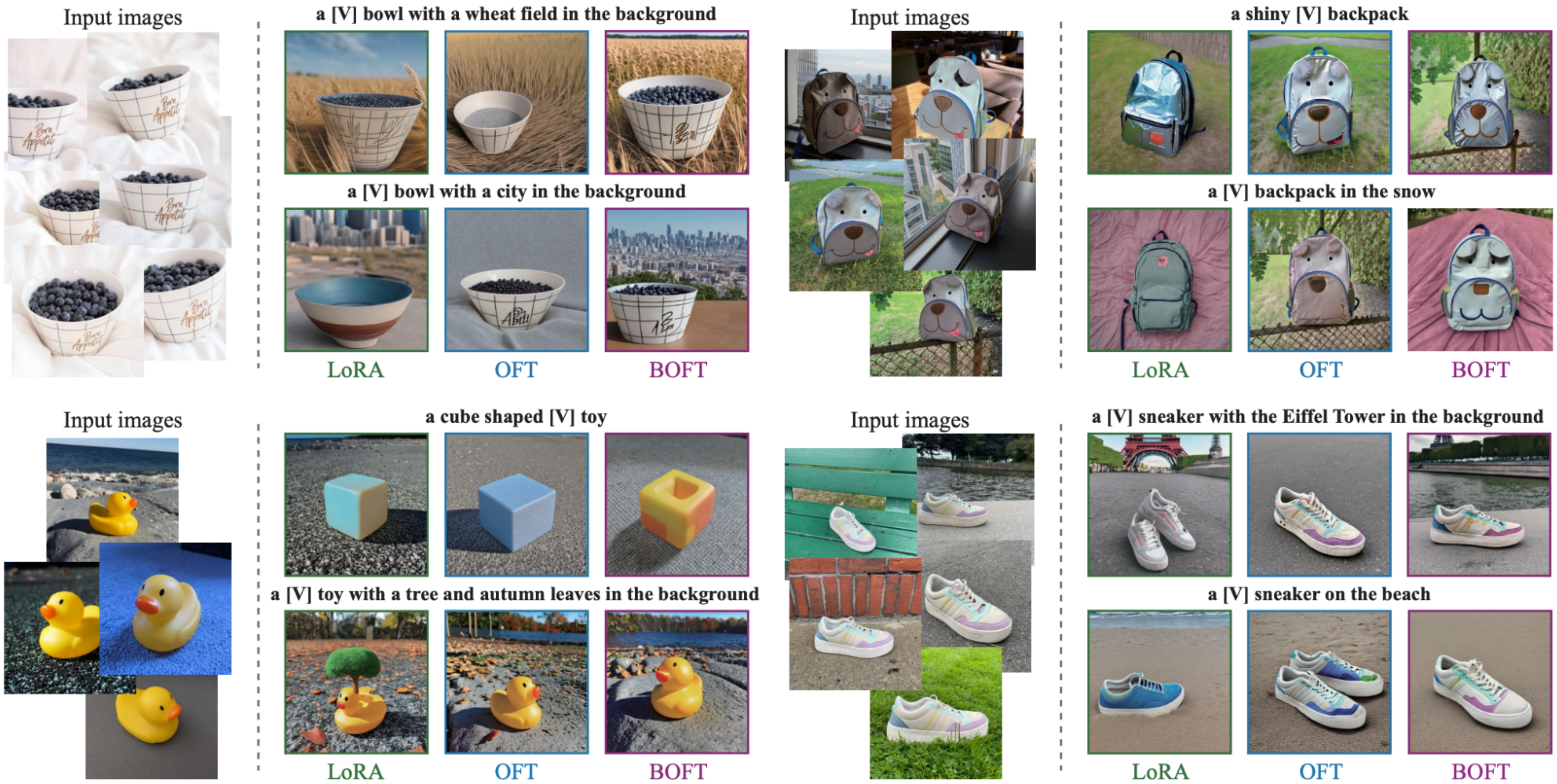
Orthogonal Butterfly for Text-to-image Tasks

- Qualitative results (controllable generation)



Orthogonal Butterfly for Text-to-image Tasks

- Qualitative results (subject-driven generation)



BOFT comes with free weight interpolation

- BOFT with 6 butterfly components

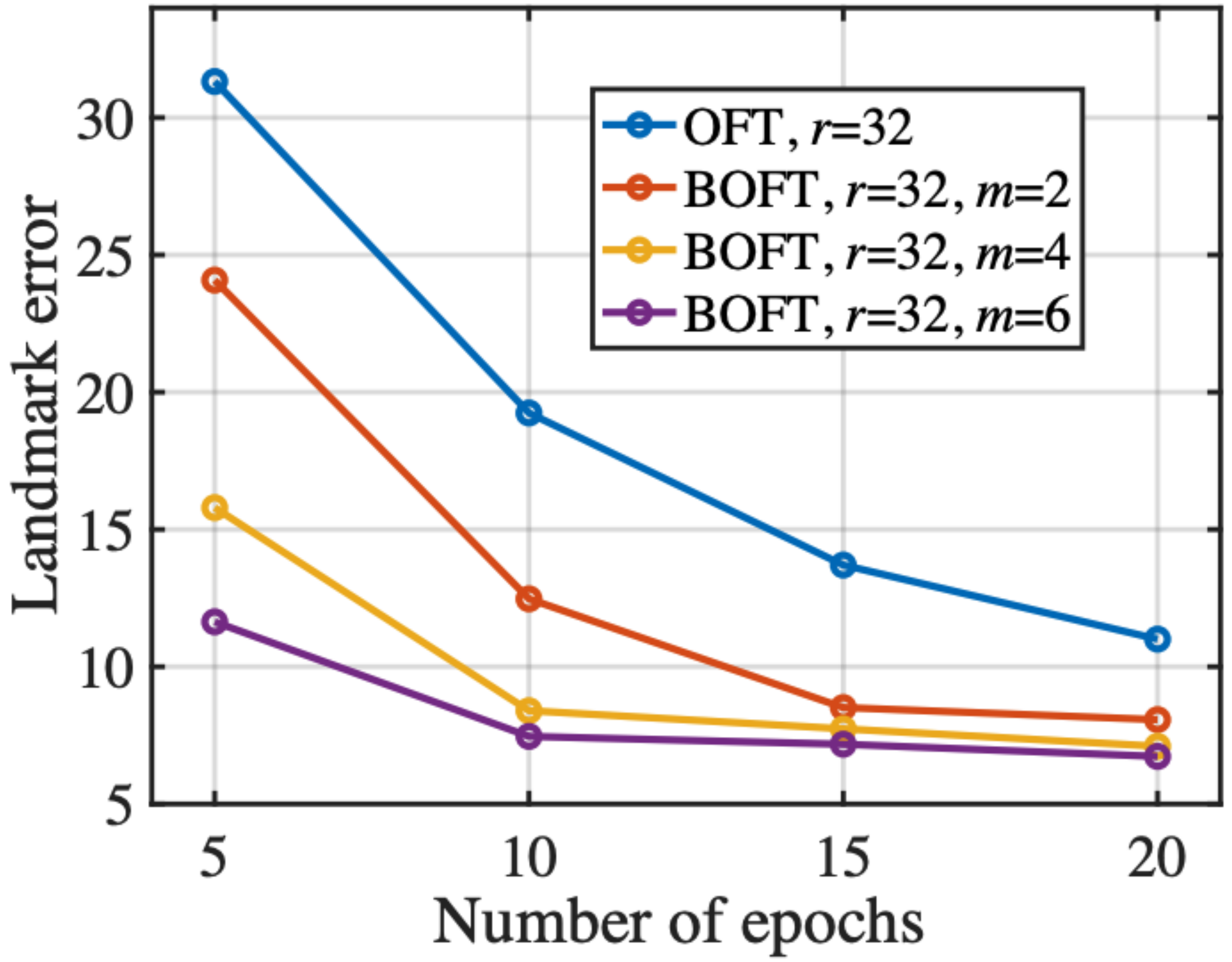
$$\boldsymbol{B}_6\ \boldsymbol{B}_5\ \boldsymbol{B}_4\ \boldsymbol{B}_3\ \boldsymbol{B}_2\ \boldsymbol{B}_1$$

Control signal | BOFT (6 matrices) | 5 matrices | 4 matrices | 3 matrices | 2 matrices | 1 matrix | SD* (0 matrix)

Orthogonal Butterfly for Text-to-image Tasks

- Quantitative results

Method	# Param	Error
$\mathrm{LoRA}_{r=128}$	20.17M	8.038
$\mathrm{LoRA}_{r=16}$	2.52M	8.878
$\mathrm{OFT}_{r=16}$	2.71M	8.876
$\mathrm{OFT}_{r=4}$	10.50M	6.537
$\mathrm{BOFT}^{m=2}_{r=32}$	2.66M	8.070
$\mathrm{BOFT}^{m=5}_{r=16}$	12.93M	6.387
$\mathrm{BOFT}^{m=4}_{r=8}$	20.76M	**5.667**



Thanks!

- Our project page: https://boft.wyliu.com/
- BOFT is integrated into the Hugging Face PEFT library.
  - https://huggingface.co/docs/peft/main/en/conceptual_guides/oft

Project page

Hugging Face PEFT