ICLR

PyTorch

NOLA: Compressing LoRA Using Linear Combination of Random Basis

Soroush Abbasi Koohpayegani* K L Navaneet*

Parsa Nooralinejad Soheil Kolouri Hamed Pirsiavash

* Equal contribution

UCDAVIS UNIVERSITY OF CALIFORNIA

VANDERBILT UNIVERSITY

Motivation

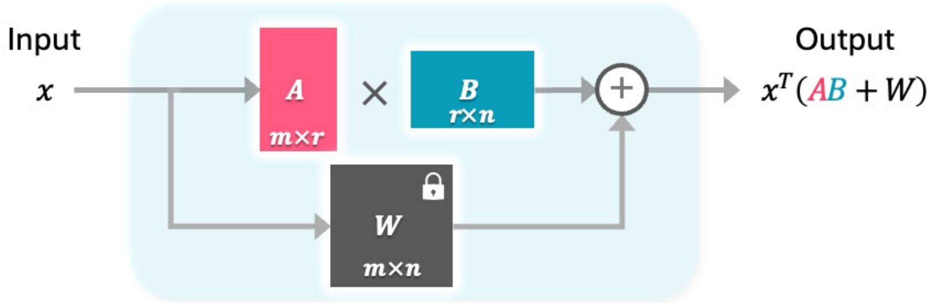
- Multiple Large Language Models (LLMs), each tailored for specific tasks.

Motivation

- Multiple Large Language Models (LLMs), each tailored for specific tasks.
- **Goal:** Reduce the model size for each LLM variation

Motivation

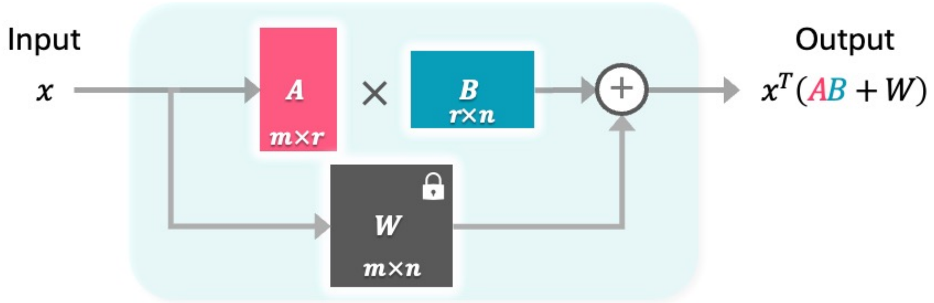
- Multiple Large Language Models (LLMs), each tailored for specific tasks.
- **Goal:** Reduce the model size for each LLM variation
- **Well-known solution:** LoRA [1]



[1]: Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. "LORA: Low-rank adaptation of large language models." ICLR 2022

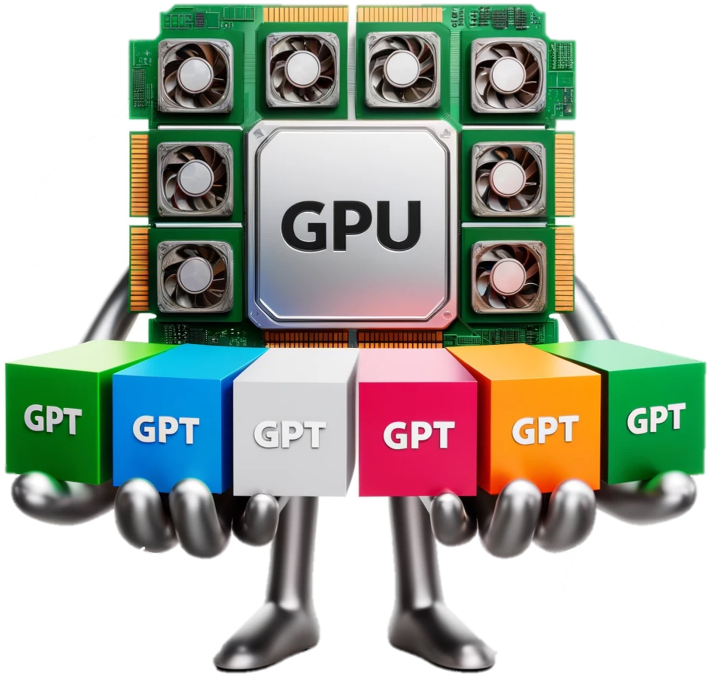
Motivation

- Multiple Large Language Models (LLMs), each tailored for specific tasks.
- **Goal:** Reduce the model size for each LLM variation
- **Well-known solution:** LoRA [1]
- We introduce **NOLA** that is more compact than the best LoRA can do.



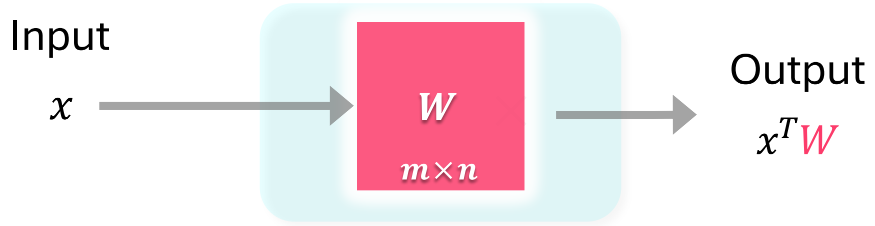
[1]: Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. "LORA: Low-rank adaptation of large language models." ICLR 2022

How many **variations of LLaMA2-70B** can we fit and run on a single 48GB GPU?



Background: Finetuning all parameters

Fine-Tuning All Parameters



Number of Optimized Parameters:

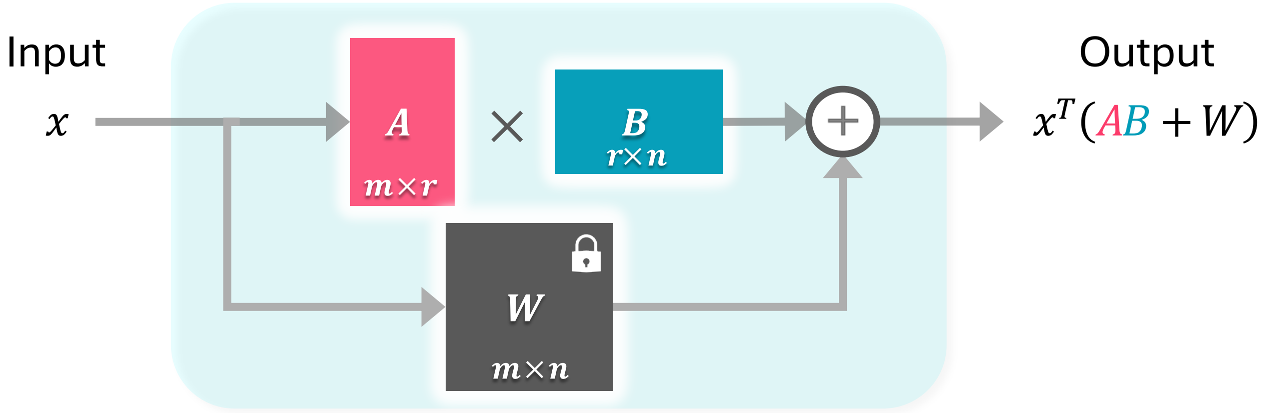
$\boldsymbol{mn}$

Overhead in Inference Time:

$\mathbf{0}$

Background: LoRA

Fine-Tuning Low-Rank Adapter
LoRA [1]



Number of Optimized Parameters:

$$\boldsymbol{mr + rn = r(m + n)}$$

If $\boldsymbol{r}$ is small, we are happy

Overhead in Inference Time:

$$\mathbf{0}$$

We can merge AB and W

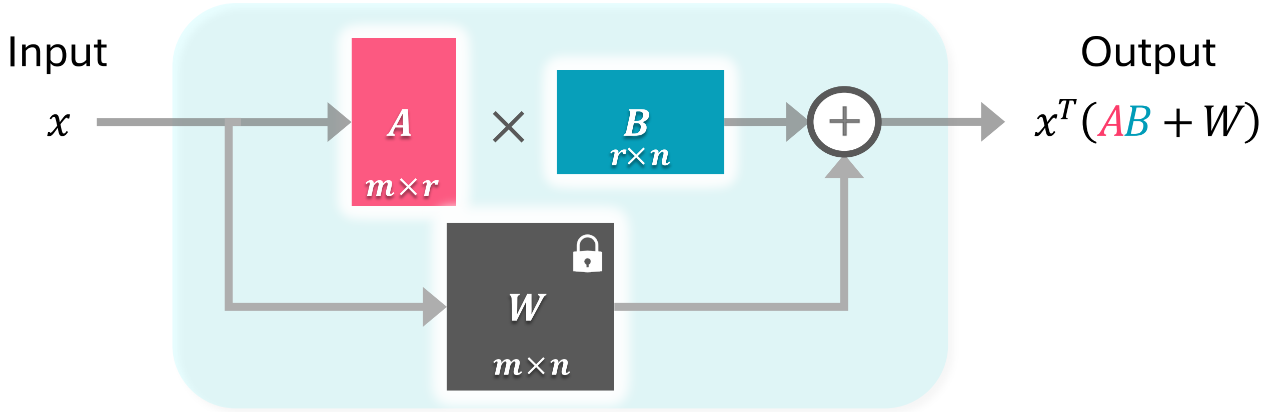
[1]: Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. "LORA: Low-rank adaptation of large language models." ICLR 2022

Limitations of LoRA

- The model size depends on:
  the **architecture** $(\boldsymbol{m}+\boldsymbol{n})$ and the **rank** $(\boldsymbol{r})$

- Number of parameters is lower bounded by rank one, which is: $(\boldsymbol{m}+\boldsymbol{n})$

Fine-Tuning Low-Rank Adapter
LoRA [1]



[1]: Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. "LORA: Low-rank adaptation of large language models." ICLR 2022

Our Key Idea

Decouple the model size from the **architecture** and **rank**

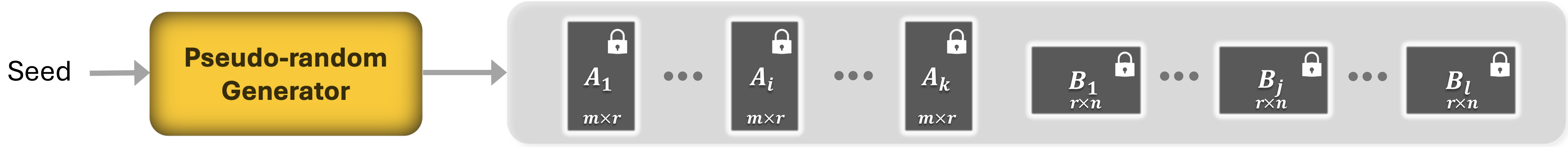
NOLA

- Reparametrize A and B.
- We construct A and B as **linear combination of random matrices** (basis).
  - Inspired by PRANC [1]
- Optimize only the **coefficients** for the basis.

$$A = \sum_{i=1}^{k} \boldsymbol{\alpha_i} A_i \qquad B = \sum_{j=1}^{l} \boldsymbol{\beta_j} B_j$$

[1] Nooralinejad, P., Abbasi, A., Abbasi Koohpayegani, S., Pourahmadi Meibodi, K., Khan, R. M. S., Kolouri, S., & Pirsiavash, H. “PRANC: Pseudo RAndom Networks for Compacting deep models.” ICCV 2023

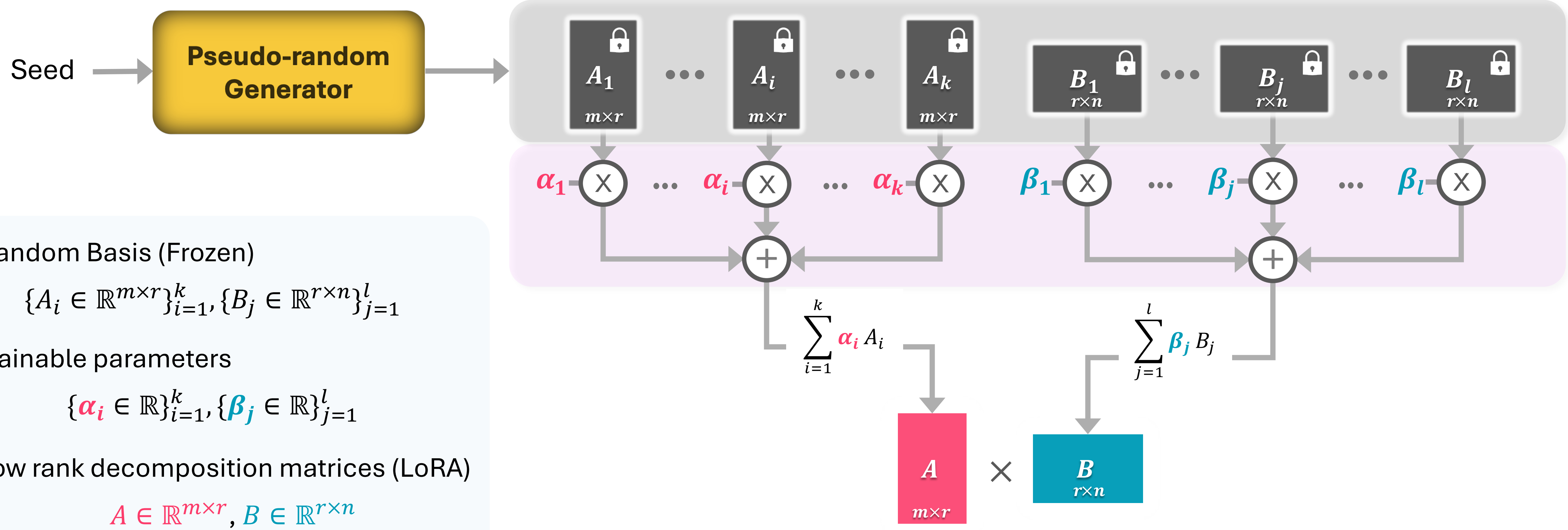
NOLA



Random Basis (Frozen)

$$\{A_i \in \mathbb{R}^{m \times r}\}_{i=1}^{k}, \{B_j \in \mathbb{R}^{r \times n}\}_{j=1}^{l}$$

NOLA



Random Basis (Frozen)

$$\{A_i \in \mathbb{R}^{m \times r}\}_{i=1}^{k}, \{B_j \in \mathbb{R}^{r \times n}\}_{j=1}^{l}$$

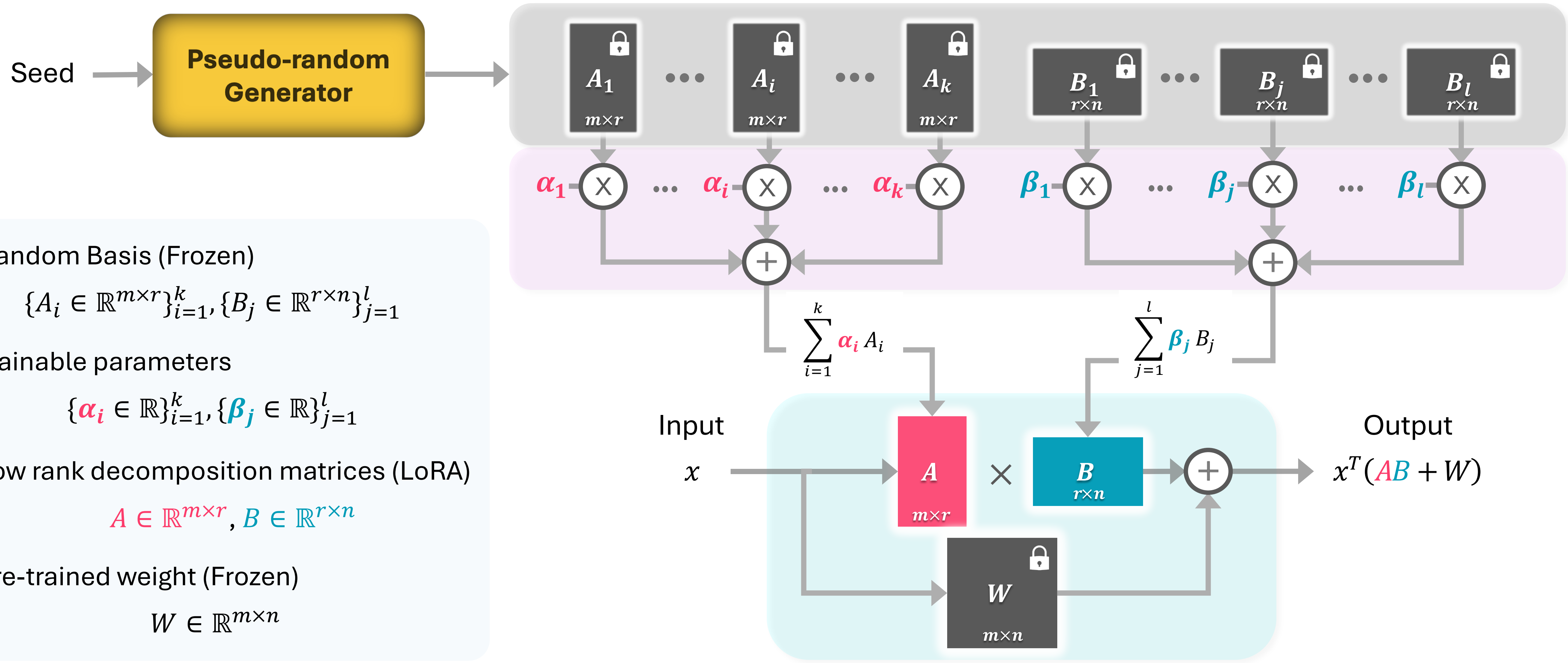
Trainable parameters

$$\{\alpha_i \in \mathbb{R}\}_{i=1}^{k}, \{\beta_j \in \mathbb{R}\}_{j=1}^{l}$$

Low rank decomposition matrices (LoRA)

$$A \in \mathbb{R}^{m \times r}, B \in \mathbb{R}^{r \times n}$$

NOLA



Random Basis (Frozen)

$$\{A_i \in \mathbb{R}^{m\times r}\}_{i=1}^{k}, \{B_j \in \mathbb{R}^{r\times n}\}_{j=1}^{l}$$

Trainable parameters

$$\{\alpha_i \in \mathbb{R}\}_{i=1}^{k}, \{\beta_j \in \mathbb{R}\}_{j=1}^{l}$$

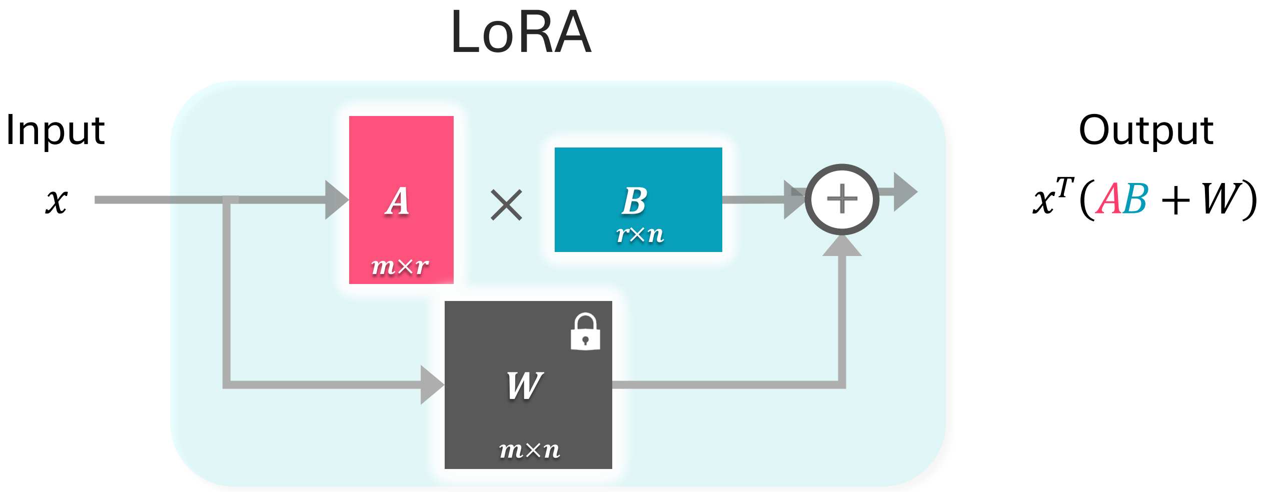
Low rank decomposition matrices (LoRA)

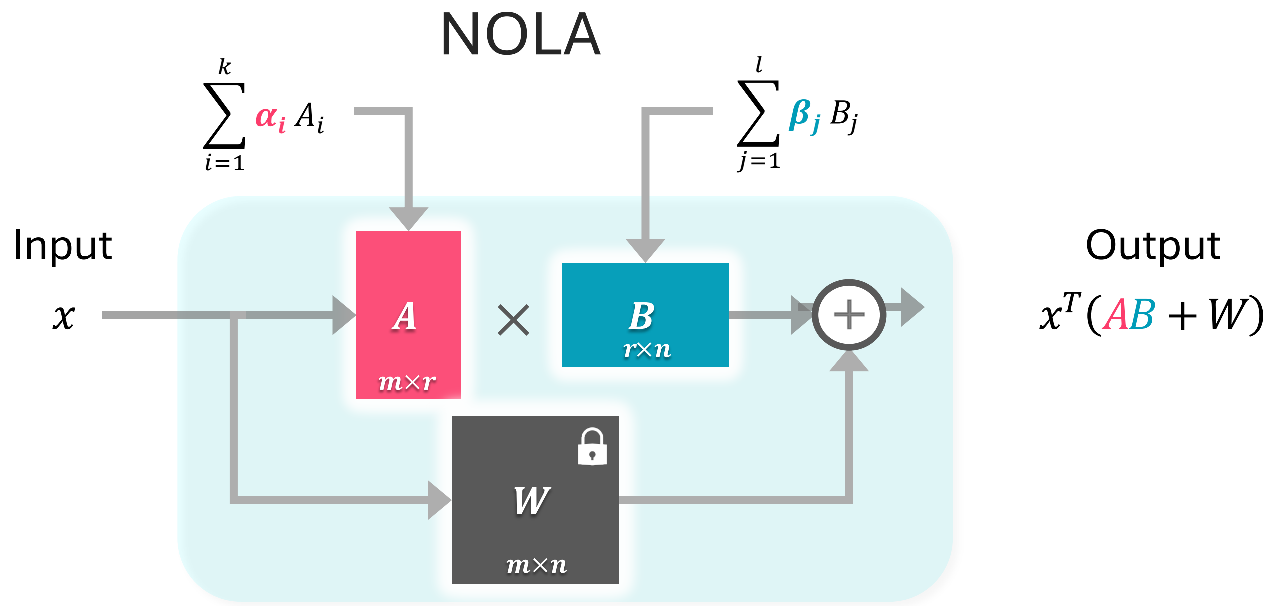
$$A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{r\times n}$$

Pre-trained weight (Frozen)

$$W \in \mathbb{R}^{m\times n}$$

NOLA vs LoRA





# Parameters	**Overhead in Inference Time**
$mr + rn = r(m + n)$ Depend on r, m, n	**0** We can merge AB and W
$k + l$ Decoupled from rank and architecture	**0** We can merge AB and W

Results – LLaMA 2

	LLaMA-2 - 7B (8-bit)			LLaMA-2 - 13B (8-bit)			LLaMA-2 - 70B (8-bit)		
	w/o Finetuning	LoRA	NOLA	w/o Finetuning	LoRA	NOLA	w/o Finetuning	LoRA	NOLA
Adapter Rank	-	1	16	-	1	16	-	1	16
Trainable Parameters	-	2.50M	0.06M (↓97%)	-	3.91M	0.14M (↓96%)	-	12.94M	0.57M (↓95%)
Train Loss	1.53	0.97	1.05	1.43	0.94	0.95	1.42	0.87	0.90
Val Loss	1.74	1.04	1.01	1.59	0.96	0.97	1.53	0.92	0.90
MMLU Acc	45.3	46.5	46.5	54.8	55.3	55.3	68.9	69.5	69.4

- NOLA uses **95%** fewer parameters compared to LoRA with rank one

- LLaMA-70B with 4-bit precision; 0.6 million parameters (FP16) per NOLA model:

Can fit and run more than **10,000 variations of LLaMA-70B (4-bit)** on a 48GB GPU

35GB (base model) + **1.3GB** (KV cache) + **5.7GB** (NOLA parameters)

Results – GPT-2

GPT-2 L						
Method	Adapted Layers	Adapter Rank	# Trainable Parameters	E2E NLG Challenge		
				BLEU	MET	ROUGE-L
Finetune	All Layers	-	774.030M	68.5	46.0	69.9
Adapter^{L}	Extra Layers	-	0.880M	69.1	46.3	71.4
Adapter^{L}	Extra Layers	-	23.000M	68.9	46.1	71.3
PreLayer	Extra Tokens	-	0.770M	70.3	46.2	**71.7**
LoRA	QV	4	0.770M	**70.4**	**46.8**	**72.0**
LoRA	QV	1	0.184M	69.9	**46.7**	71.6
NOLA (Ours)	QV	8	0.144M	**70.5**	**46.8**	**71.7**
NOLA (Ours)	QV	8	0.036M	70.1	**46.7**	**71.7**

- We can adjust the number of parameters while keeping the rank constant
- NOLA can achieve on-par performance to other PEFT baselines with fewer number of parameters

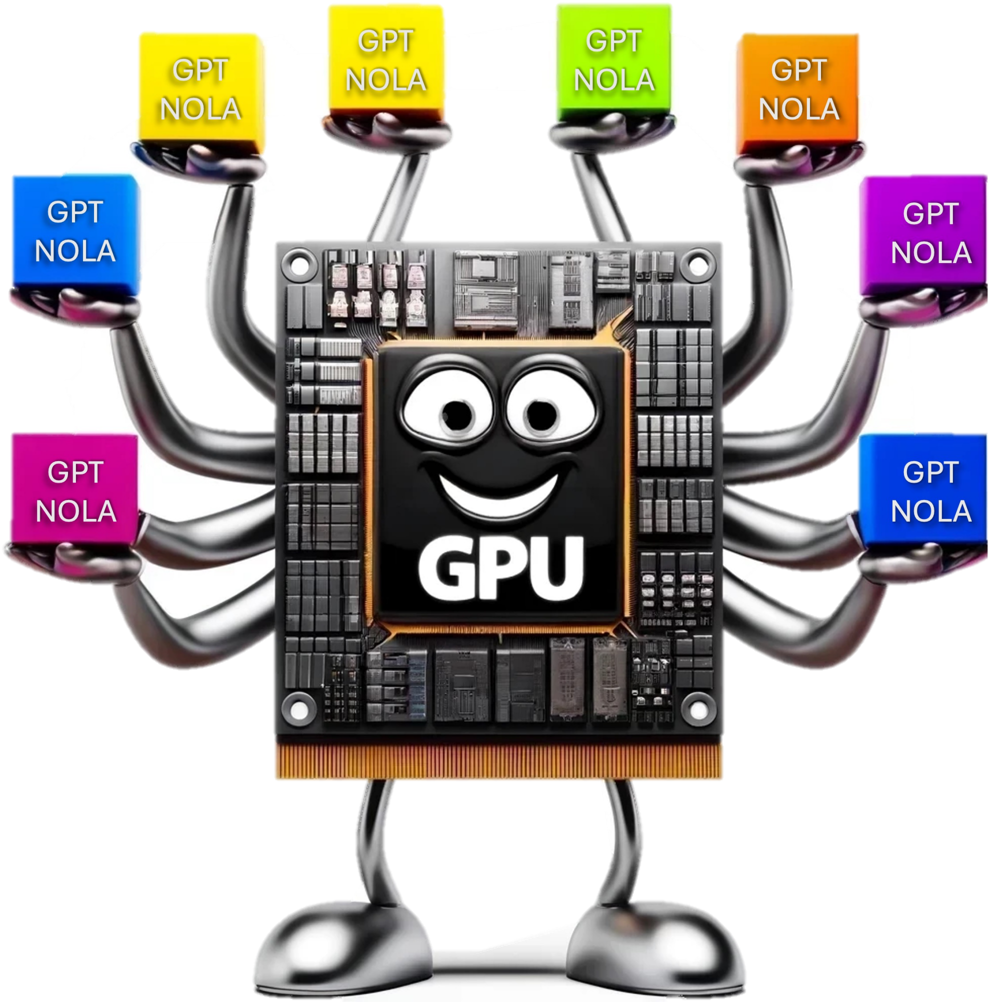
Results – Vision Transformer

Base Model		# Train Params	CIFAR-100 5	CIFAR-100 10	CUB-200-2011 5	CUB-200-2011 10	Caltech-101 5	Caltech-101 10
ViT-L	Nearest Neighbor		68.9	74.0	77.4	82.3	88.4	90.1
	Linear	0	63.7 (1.3)	70.6 (0.9)	73.7 (0.6)	79.2 (0.3)	87.6 (0.9)	89.9 (0.4)
	Full-FT	289M	74.0 (2.3)	86.2 (0.6)	73.3 (0.9)	83.9 (0.2)	88.7 (1.0)	91.3 (0.7)
	LoRA (r=4)	375K	82.9 (0.9)	87.6 (0.6)	**81.2** (0.4)	**85.3** (0.3)	89.3 (0.7)	91.3 (0.3)
	LoRA (r=1)	94K	82.2 (0.8)	85.6 (0.9)	80.6 (0.3)	85.2 (0.3)	89.9 (1.0)	91.6 (0.4)
	NOLA-MLP	94K	**83.6** (0.9)	**87.8** (0.6)	80.8 (0.6)	85.2 (0.2)	**90.0** (0.7)	**91.7** (0.3)
	NOLA-MLP	**47K**	81.2 (1.0)	87.1 (0.6)	80.7 (0.5)	85.0 (0.3)	89.8 (0.8)	91.5 (0.4)

- NOLA is on-par with LoRA with fewer parameters in vision transformers (ViT-B)

Takeaway Message

How many **variations of LLaMA2-70B** (4 bits) can fit and run on a single 48GB GPU?



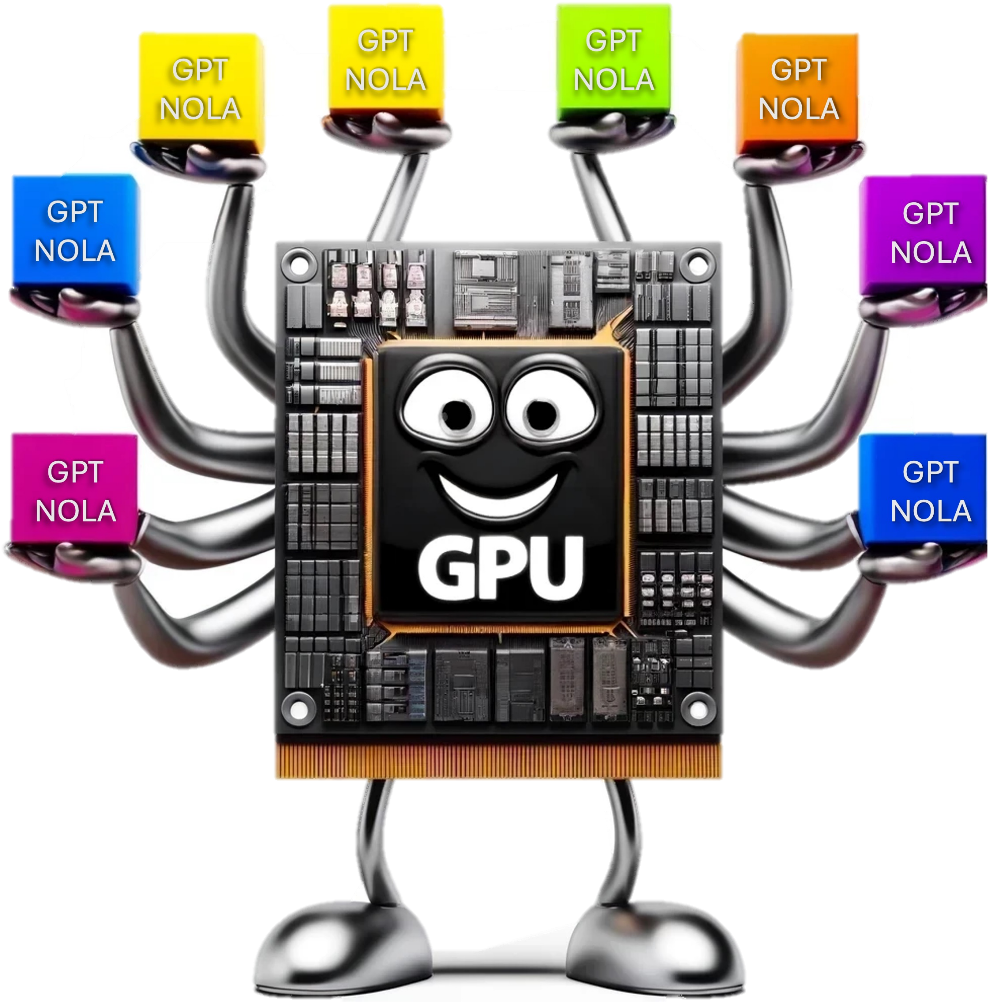
Takeaway Message

How many **variations of LLaMA2-70B (4 bits)** can fit and run on a single 48GB GPU?

10,000 😊

35GB (base model) + **1.3GB** (KV cache) + **5.7GB** (NOLA parameters)

NOLA offers flexibility in adjusting the number of parameters during fine-tuning by **decoupling** the number of parameters from the choice of **architecture** and **rank**



Thank You!

PyTorch Code: https://github.com/UCDvision/NOLA

Halle B
Tue 7 May 4:30 p.m. CEST — 6:30 p.m. CEST