GOSIM

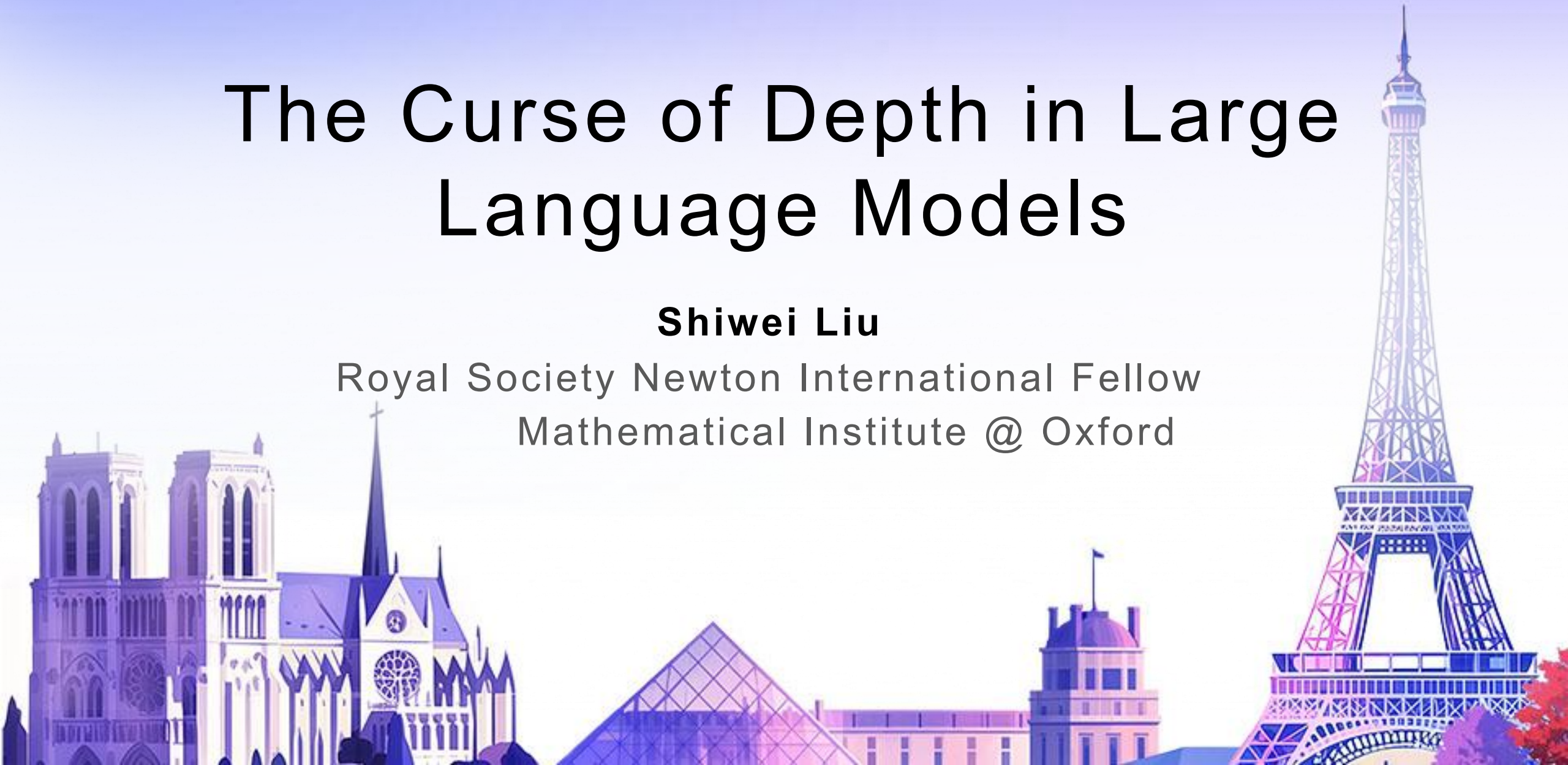
GOSIM AI Paris 2025

The Curse of Depth in Large Language Models

Shiwei Liu

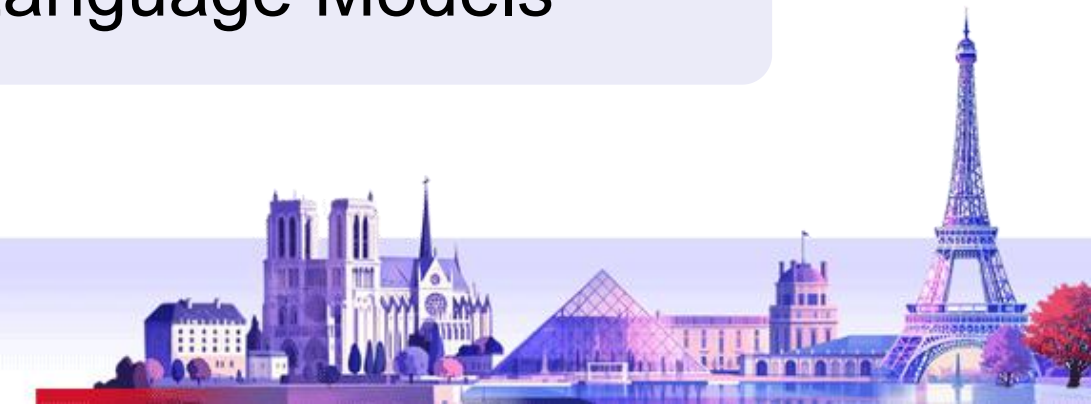
Royal Society Newton International Fellow

Mathematical Institute @ Oxford



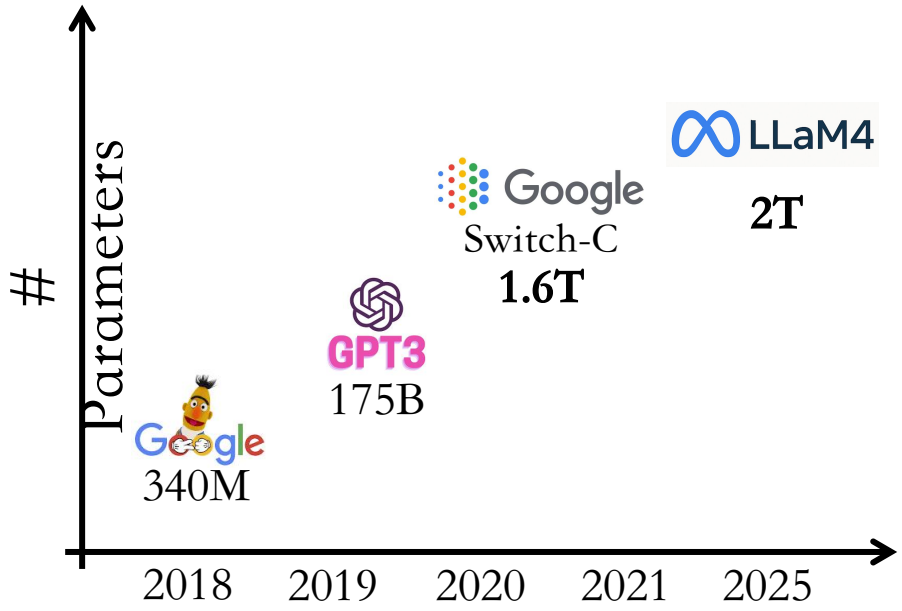
Outline

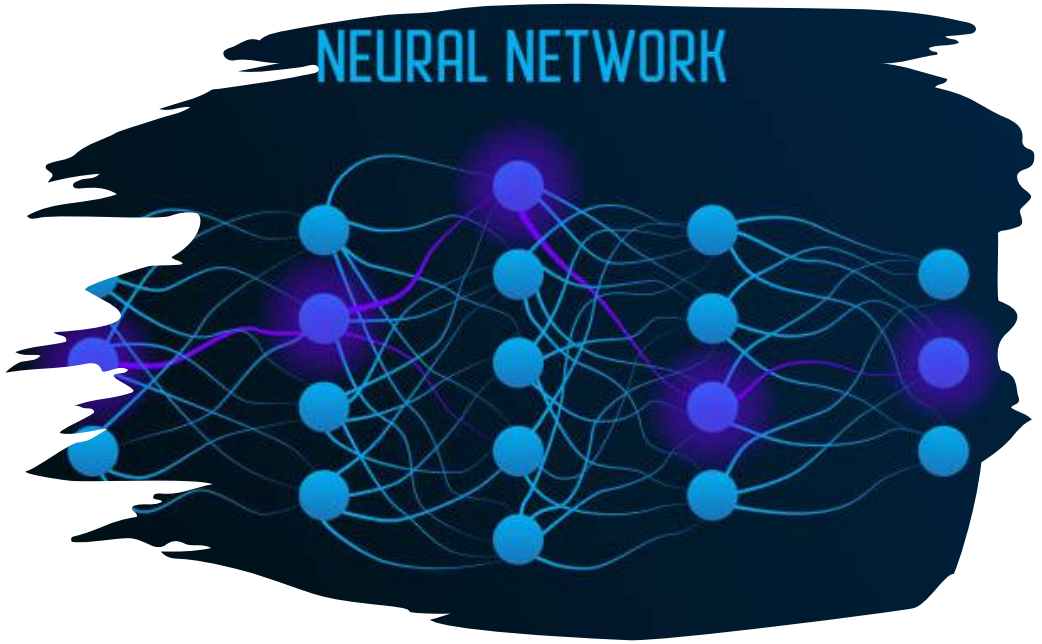
- ❖ Background of Efficient LLM Inference
- ❖ Leverage the Ineffective Layers for Efficient LLMs
- ❖ The Curse of Depth in Large Language Models



AI Is Powerful But Very Expensive

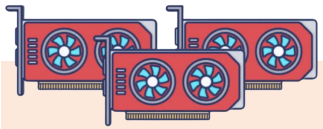
GOSIM





Training

Costs >$10M



Inference

175B model >5x A100

Inference Bottlenecks of LLMs

GOSIM

- **Massive Model Size**
  - High GPU memory requirements to load the model
    e.g., GPT-175B requires at least 320GB, ⩾ 5x A100/H100-80G
  - Full-parameter fine-tuning is not affordable
    Backpropagation needs 3x more memory than model parameter

- **Key-Value (KV) Cache in Long-context Scenarios**
  - High GPU usage to store attention Key and Value during generation
    A 30B model with 128 batch size and 1024 sequence length requires
    180GB
  - Retrieval Traffic
    Reading the cached pairs from memory to cores of chips

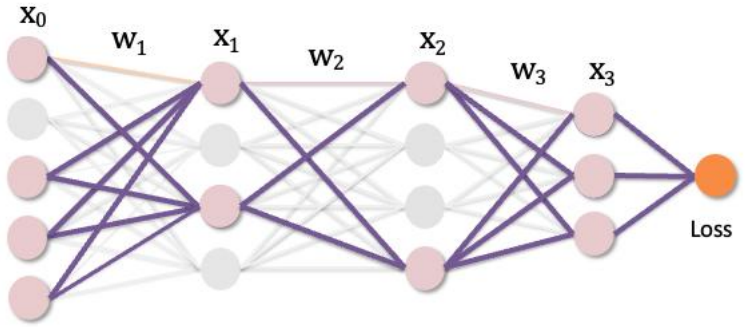
- **Quadratic Cost of Self-Attention**
  - Computing cost quadratically increases with sequence length

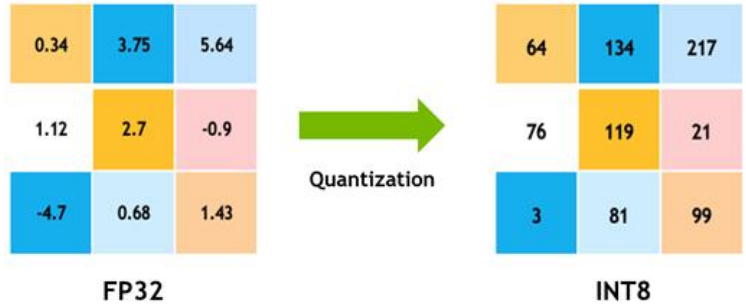
Traditional Model Compression

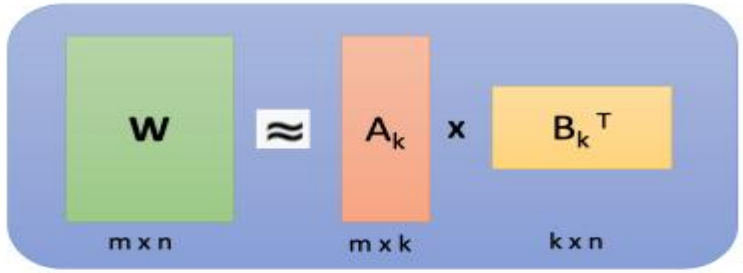
- **Sparsification**
  - Removing unimportant components such as weights, attention heads, layers, KV cache.

- **Quantization**
  - Mapping high-precision floating-point numbers to low precisions, e.g., 16bit, 8bit, ..., even 1bit.

- **Low-rank Approximation**
  - Approximating weight matrix of LLMs with products of smaller, low-rank matrices.







I. Leverage the Ineffective Layers for Efficient LLMs

GOSIM

- **My one-trick pony in efficient LLMs**:
  - Compressing more the layers that are less effective.

- **Research Question:**
  - Ho to find those less effectives layers in LLMs?
  - How to measure the effectiveness of layers in LLMs?

GOSIM

Outlier Weighed Layerwise Sparsity (OWL 🦉): A Missing Secret Sauce for Pruning LLMs to High Sparsity

Lu Yin [1 2 3] You Wu [3] Zhenyu Zhang [4] Cheng-Yu Hsieh [5] Yaqing Wang [3] Yiling Jia [3] Gen Li [6] Ajay Jaiswal [4]
Mykola Pechenizkiy [2] Yi Liang [3] Michael Bendersky [3] Zhangyang Wang [4] Shiwei Liu [7 2]

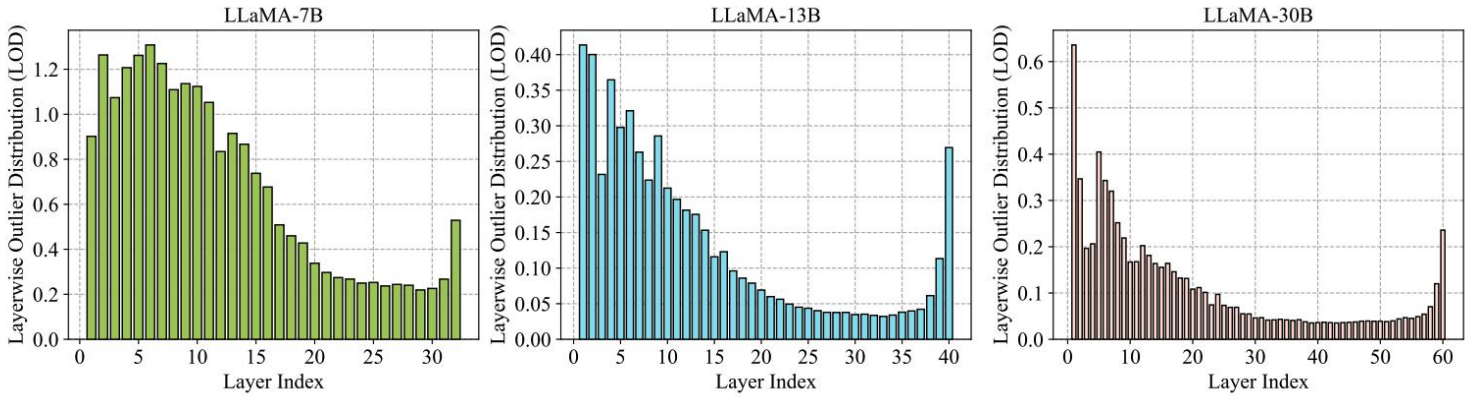


Figure 1: Layerwise Outlier Distribution (LOD) (%) of dense LLaMA-7B, 13B, and 30B.

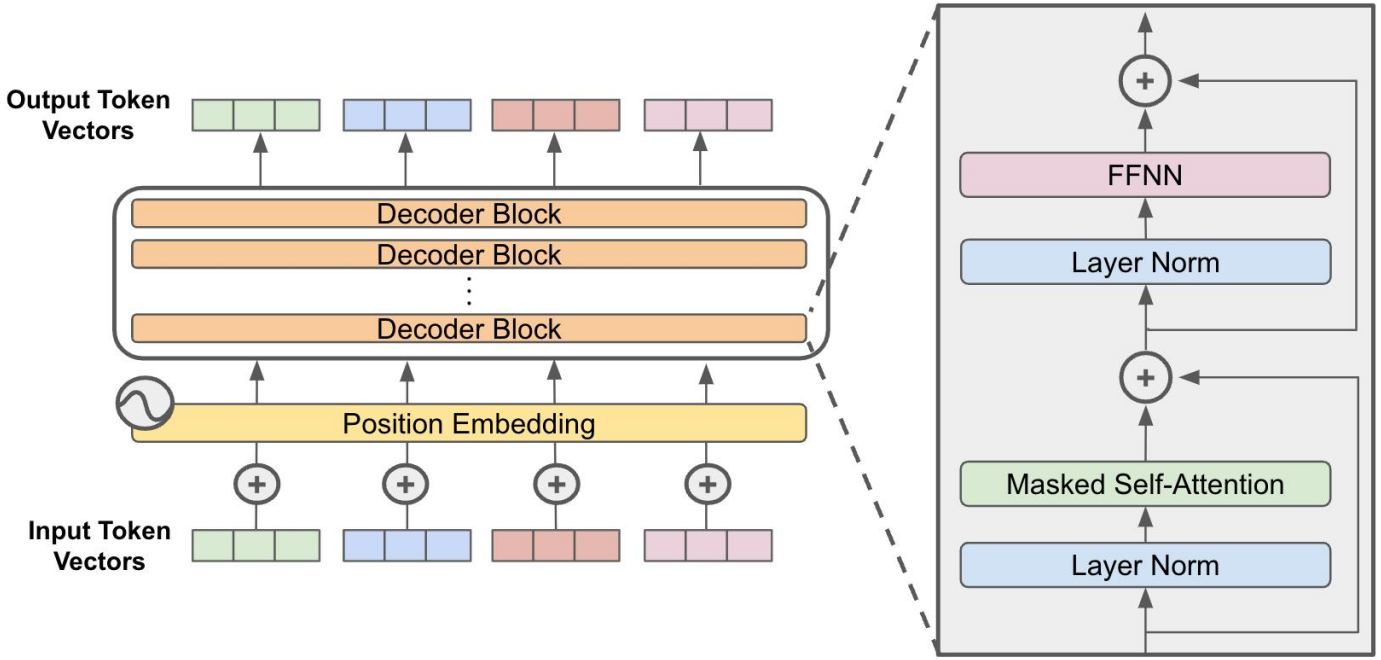
ICML 2024

GOSIM AI Paris 2025

❖ Notation of talk

- **Input** $X \in \mathbb{R}^{MxN}$
- **Weight** W with W_{ij} connects its j^{th} input to i^{th} output,
- **Sparsity** $= 1 - \frac{\|w\|_0}{MN}$
- **Perplexity** $= \exp\left(-\frac{1}{T}\sum_{i=1}^{T} \log P(X_i|X_1, \ldots, X_{i-1})\right)$

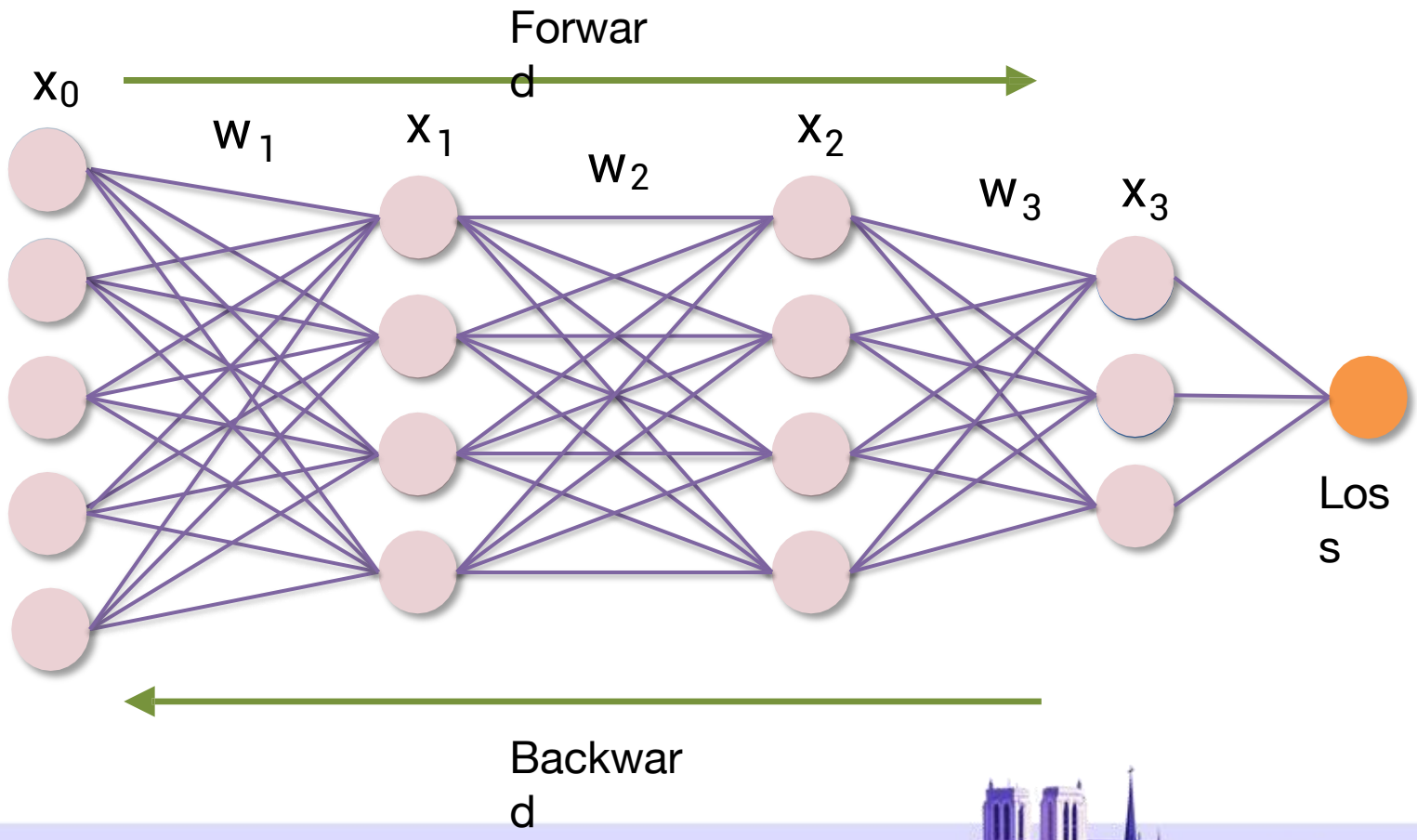
❖ Architecture: Decoder-only Transformer



What is pruning?

GOSIM

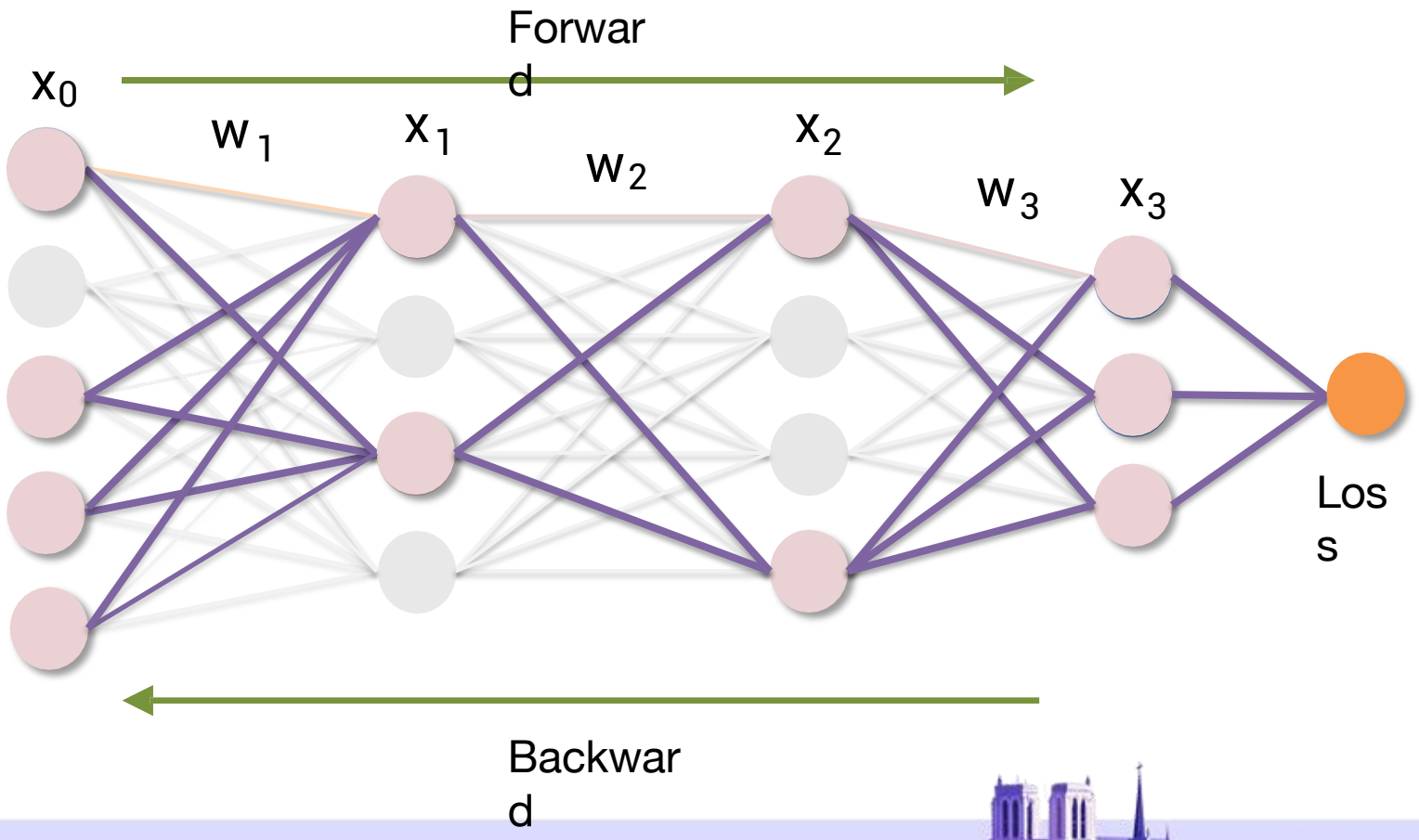
Dense model



What is pruning?

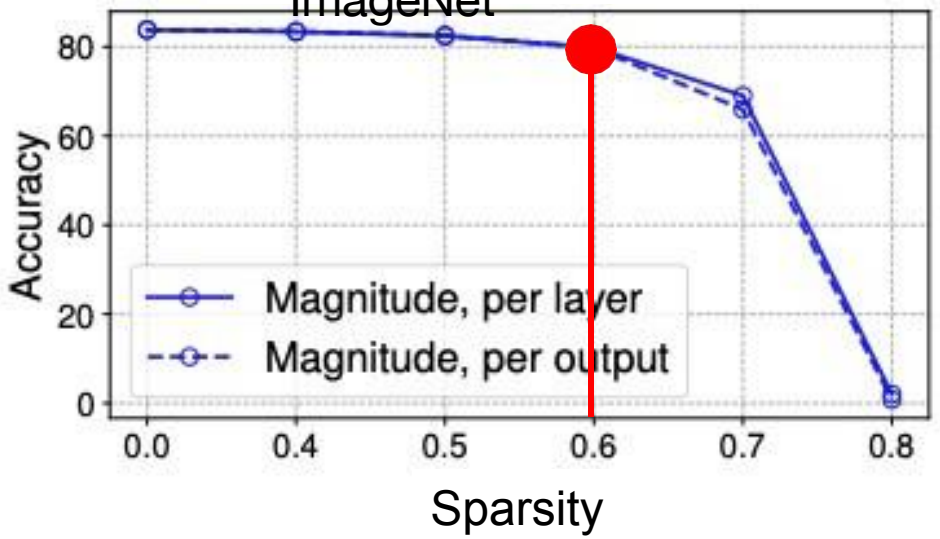
GOSIM

Sparse model



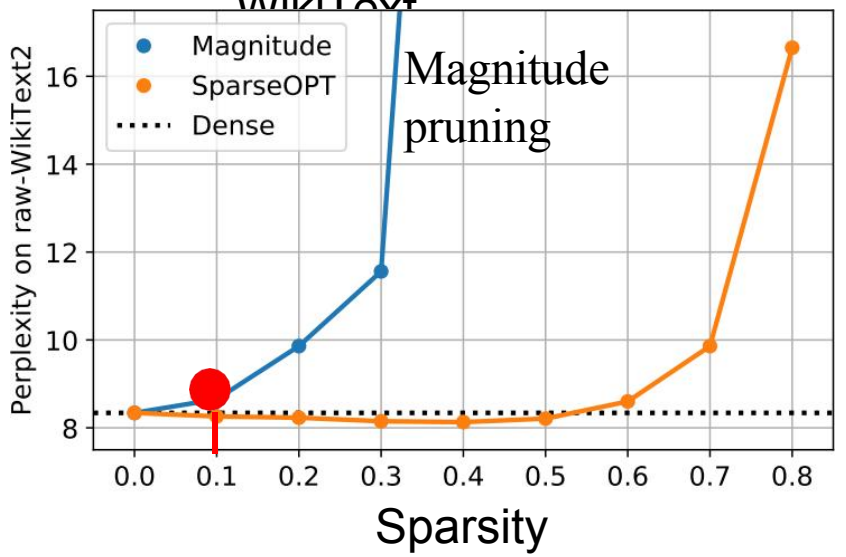
Magnitude Pruning Fails in LLMs

ConvNeXt on ImageNet



Wanda (Sun et al. 2023)

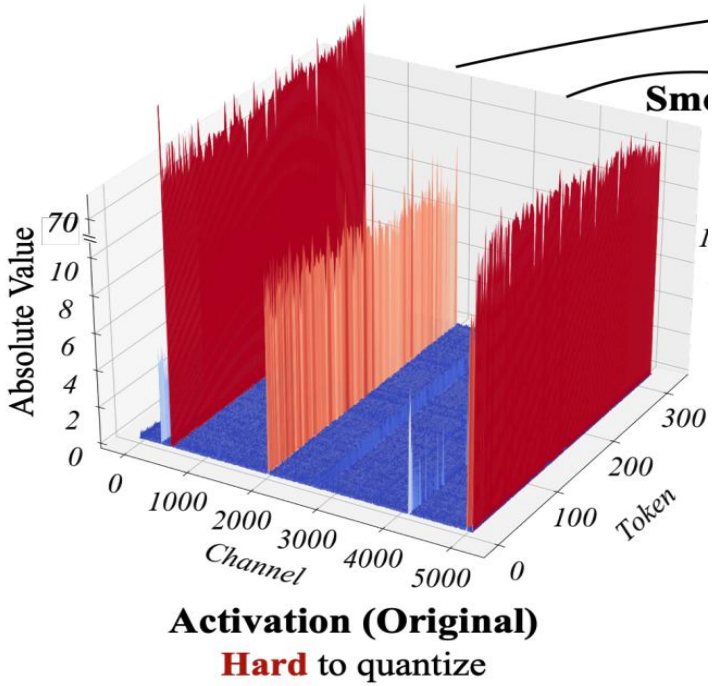
OPT-175B on WikiText



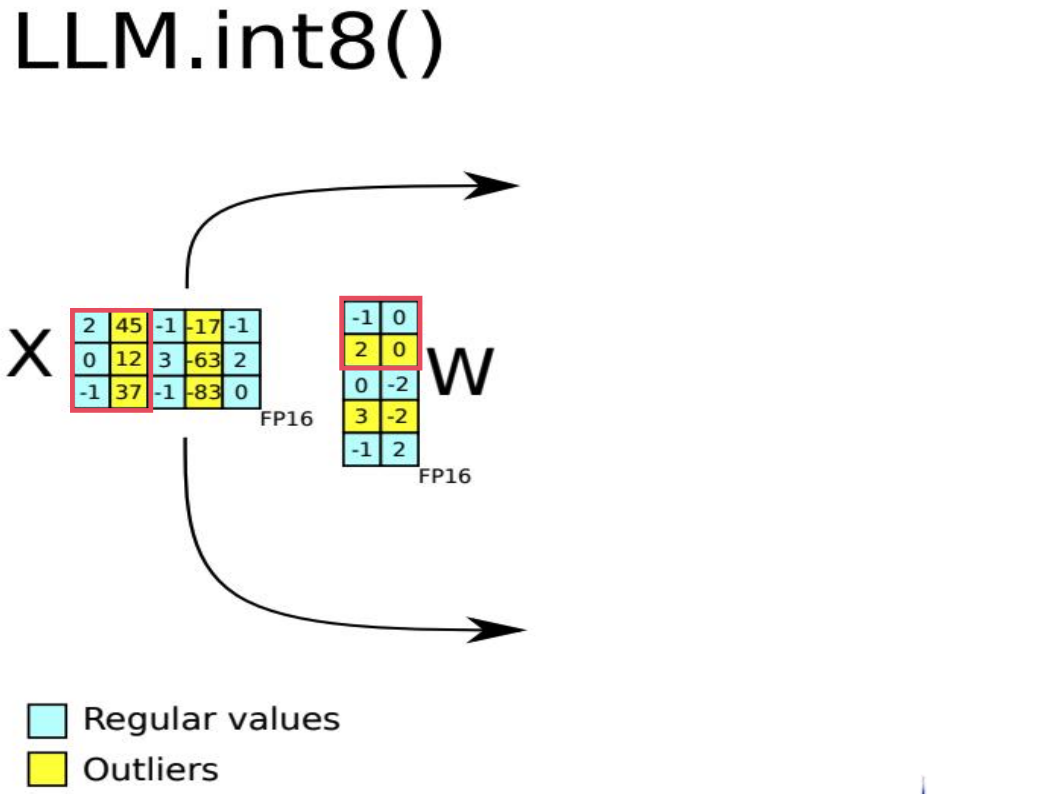
SparseGPT (Frantar & Dan et al. 2023)

Magnitude pruning quicks fail in LLMs at mild sparsity i.e., 10%.

Outliers in LLMs



SmoothQuant (Xiao et al 2022)



LLM.int8 (Dettmers et al 2022)

Outliers Weighed Layerwise Importance (OWL

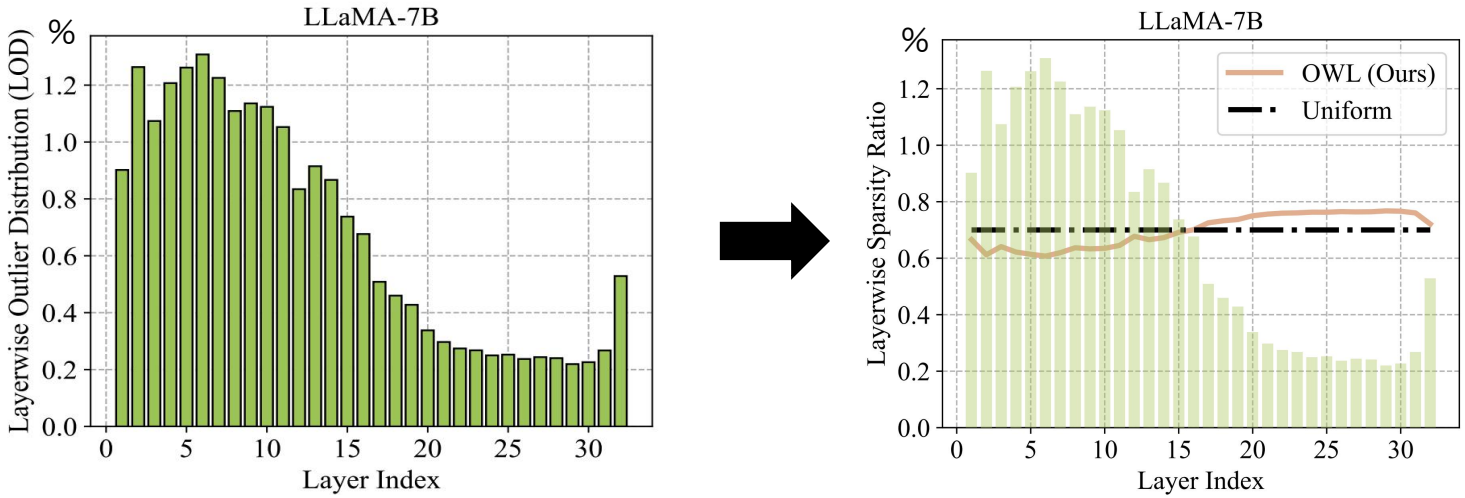
❖ **Rationale: Layers with higher outlier ratios are more importance.**

Step 1: Assign an outlier score for each weight:

$$A^l_{ij} = \|X^l_j\|_2 |W^l_{ij}|$$

Step 2: Calculating the ratio of outlier weights in each layer, τ controls the magnitude of outlier

$$D^l = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(A^l_{ij} > \tau \bar{A}^l)}{NM}, \quad \bar{A}^l = mean(Al)$$



Evaluation

One-shot pruning without fine-tuning

Table 3: WikiText validation perplexity of pruning methods for LLaMA-V1 family and OPT-6.7B at 70% sparsity. The best performance method is indicated in **bold**, and the gain in perplexity achieved by OWL is highlighted in blue.

Method	Layerwise Sparsity	Weight Update	LLaMA-V1 7B	LLaMA-V1 13B	LLaMA-V1 30B	LLaMA-V1 65B	OPT 6.7B
Dense	-	-	5.68	5.09	4.10	4.77	10.13
Magnitude	Uniform	✗	48419.12	84539.45	977.73	46.89	290985.03
Wanda	Uniform	✗	85.77	55.90	17.37	15.23	162.92
OWL *w.* Wanda	Non-Uni	✗	**24.55** (-61.22)	**17.17** (-38.73)	**10.75** (-6.62)	**8.61** (-6.62)	**40.22** (-120.70)
SparseGPT	Uniform	✓	26.30	19.24	12.56	10.45	**20.29**
OWL *w.* SparseGPT	Non-Uni	✓	**19.49** (-6.81)	**14.55** (-4.69)	**10.28** (-2.28)	**8.28** (-0.64)	22.48 (2.19)

Real Speed-up

Table 7: End-to-end decode latency speedup of LLaMA-V2-7B-chat-hf using OWL with the DeepSparse (DeepSparse, 2021) inference engine.

Sparsity	Dense	10%	20%	30%	40%	50%	60%	70%	80%	90%
Latency (ms)	213.83	216.86	221.62	218.01	167.54	121.25	101.41	81.89	64.57	54.24
Throughput (tokens/sec)	4.68	4.61	4.51	4.59	5.97	8.25	9.86	12.21	15.48	18.43
Speedup	1.0x	1.0x	1.0x	1.0x	1.3x	1.8x	2.1x	2.6x	3.3x	3.9x

OWL can deliver 2.6x end-to-end inference speedup with 70% sparsity.

Universal Layerwise Importance Score

Low-rank Decomposition

Table 10: WikiText Perplexity of LLaMA-V1-7B Model.

Rank Reduction Ratio	0%	20%	30%	40%	50%	60%
Uniform - SVD	5.68	8.48	17.23	1909.34	13627.03	34354.9
OWL - SVD	5.68	**8.25**	**12.92**	**43.02**	**10707.41**	**20118.7**

Jaiswal, A., Yin, L., Zhang, Z., Liu, S., Zhao, J., Tian, Y. and Wang, Z., 2024. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. ICML 2025.

Quantization

Table 14: Perplexity of mixed-precision quantization with LLaMA-7B on WikiText.

Method	Precision	Perplexity
Same Bit-width	2 Bit	104151.84
Same Bit-width	3 Bit	25.82
Same Bit-width	4 Bit	6.29
Select with random	Mixed 3/4 Bit	12.04
Select with L_1 norm	Mixed 3/4 Bit	14.61
Select with OWL	Mixed 3/4 Bit	**9.09**

Wang, Haoyu, Bei Liu, Hang Shao, Bo Xiao, Ke Zeng, Guanglu Wan, and Yanmin Qian. "CLAQ: Pushing the Limits of Low-Bit Post-Training Quantization for LLMs." *arXiv preprint arXiv:2405.17233* (2024).

Structured Pruning

Table 13: Perplexity of Structure Pruning with LLaMA-7B on WikiText and PTB.

Dataset	Pruning Method	Layerwise Sparsity	20%	40%	60%	80%
WikiText	LLM Pruner	Uniform	19.09	30.39	90.02	1228.17
WikiText	LLM Pruner	OWL	**18.57**	**28.65**	**76.99**	**321.64**
PTB	LLM Pruner	Uniform	29.51	66.90	192.06	1691.87
PTB	LLM Pruner	OWL	**28.82**	**53.22**	**150.16**	**502.07**

Zhang, Stephen, and Vardan Papyan. “Oats: Outlier-aware pruning through sparse and low rank decomposition.” ICLR 2025

Parameter Efficient Fine-tuning

Table 7: Fine-tuning performance of LLaMa2-7B with various approaches on MMLU benchmark.

Method	Humanities	STEM	Social Sciences	Other	Avg.
Full-FT	49.9	41.7	57.5	57.0	51.5
LoRA	46.1	40.8	56.6	56.2	49.9
GaLore	45.4	41.7	55.8	56.0	49.7
LISA	44.9	41.2	54.7	57.6	49.6
OwLore (Full-Rank)	49.1	41.3	58.8	59.1	52.1
OwLore	49.8	42.1	58.6	59.7	**52.6**

Li, Pengxiang, Lu Yin, Xiaowei Gao, and Shiwei Liu. "Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm fine-tuning." *arXiv preprint arXiv:2405.18380* (2024).

II. The Curse of Depth in Large Language Models

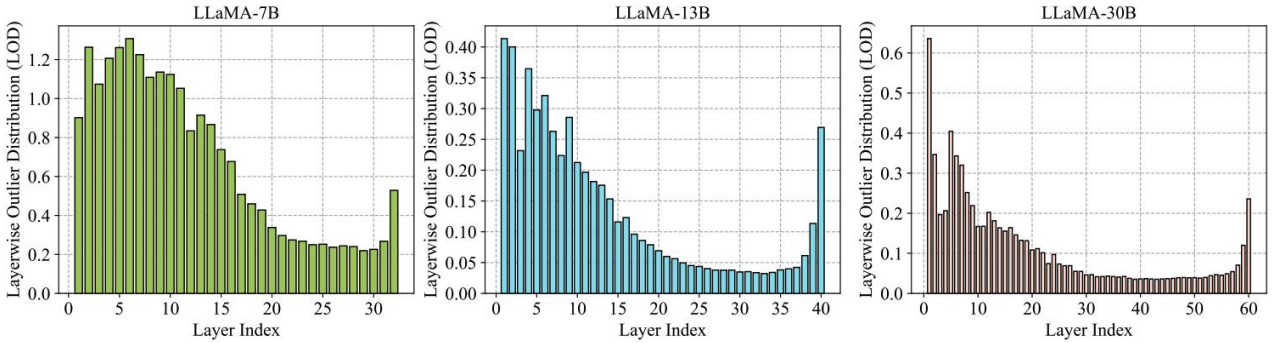


Figure 1: Layerwise Outlier Distribution (LOD) (%) of dense LLaMA-7B, 13B, and 30B.

NVIDIA.

2024-12-10

LLM Pruning and Distillation in Practice: The Minitron Approach

Sharath Turuvekere Sreenivas*, Saurav Muralidharan*, Raviraj Joshi, Marcin Chochowski, Ameya Sunil Mahabaleshwarkar, Gerald Shen, Jiaqi Zeng, Zijia Chen, Yoshi Suhara, Shizhe Diao, Chenhan Yu, Wei-Chun Chen, Hayley Ross, Oluwatobi Olabiyi, Ashwath Aithal, Oleksii Kuchaiev, Daniel Korzekwa, Pavlo Molchanov, Mostofa Patwary, Mohammad Shoeybi, Jan Kautz, and Bryan Catanzaro

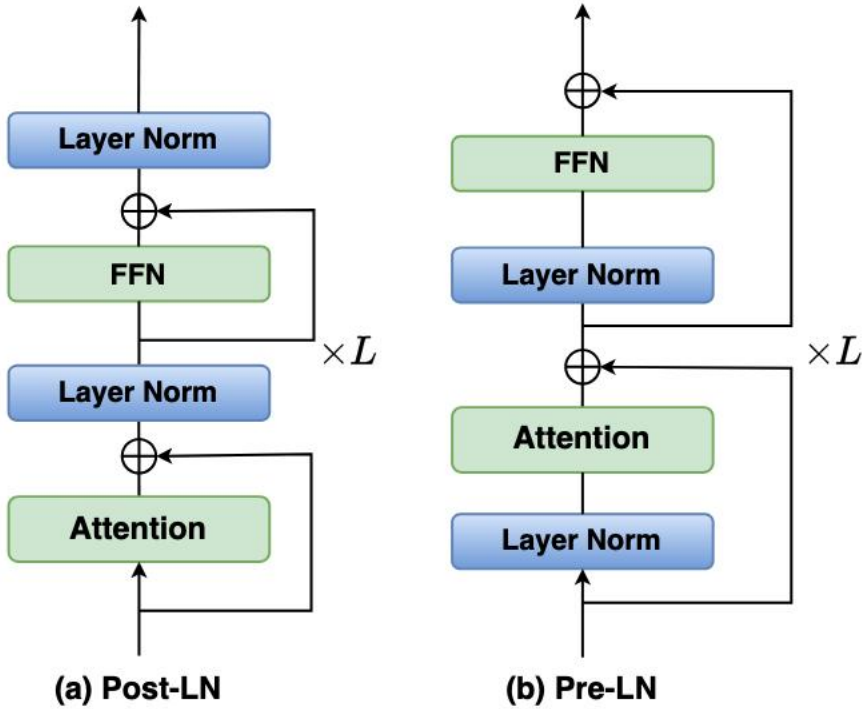
II. The Curse of Depth in Large Language Models

GOSIM

- **Curse of Depth in LLMs :** Deep layers in LLMs contribute significantly less (but not nothing) to learning and representation compared to earlier layers.
- **My thoughts:**
  - Ideally, all layers in a model should be well-trained, with sufficient diversity in features from layer to layer, to maximize the utility of the network's parameters.

What is the root cause of Ineffectiveness of deep layers in LLMs?

Transformer
BERT
T5



GPT
LLaMa
Mistral

Our Hypothesis

The inefficiency of deeper layers in LLMs primarily stems from the choice of Layer Normalization. Specifically,

- **Pre-LN** tends to produce smaller gradients in deeper layers, thereby having **more effective shallower layers**;
- **Post-LN** results in larger gradients in deeper layers and small gradients in earlier layers, thereby having **more effective deeper layers.**

Hypothesis Evaluation (Empirically)

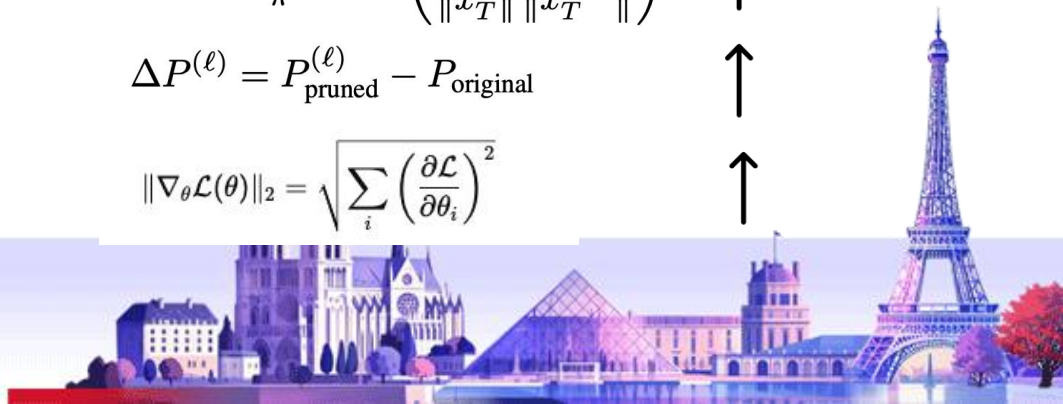
GOSIM

- **Methodology**:
  - **Pre-LN vs Post-LN:** compare layer importance across depths between Pre-LN LLM and Post-LN LLM.
- **Models**:
  - **Open-weight large-scale LLMs**: LLaMa2-7B (Pre-LN) vs. BERT-Large (Post-LN)
  - **In-house small-scale LLMs:** LLaMa-130M (Pre-LN) vs. LLaMa-130M (Post-LN)
- **Layer effectiveness metrics**:
  - **Angular Distance** between adjacent layers
  - **Performance Drop** of layer pruning
  - **Gradient Norm** of each layer

$$d(x^{\ell}, x^{\ell+n}) = \frac{1}{\pi} \arccos\left(\frac{x_T^{\ell} \cdot x_T^{\ell+n}}{\|x_T^{\ell}\| \, \|x_T^{\ell+n}\|} \right) \uparrow$$

$$\Delta P^{(\ell)} = P_{\text{pruned}}^{(\ell)} - P_{\text{original}} \uparrow$$

$$\|\nabla_{\theta} \mathcal{L}(\theta)\|_2 = \sqrt{\sum_i \left(\frac{\partial \mathcal{L}}{\partial \theta_i} \right)^2} \uparrow$$



Open-weight large-scale LLMs

BERT-Large (Post-LN) vs. LLaMa2-7B (Pre-LN)

- BERT's early layers are more similar to their adjacent layers (yellow color)
- Pruning entire early layers leads to much smaller performance drop.

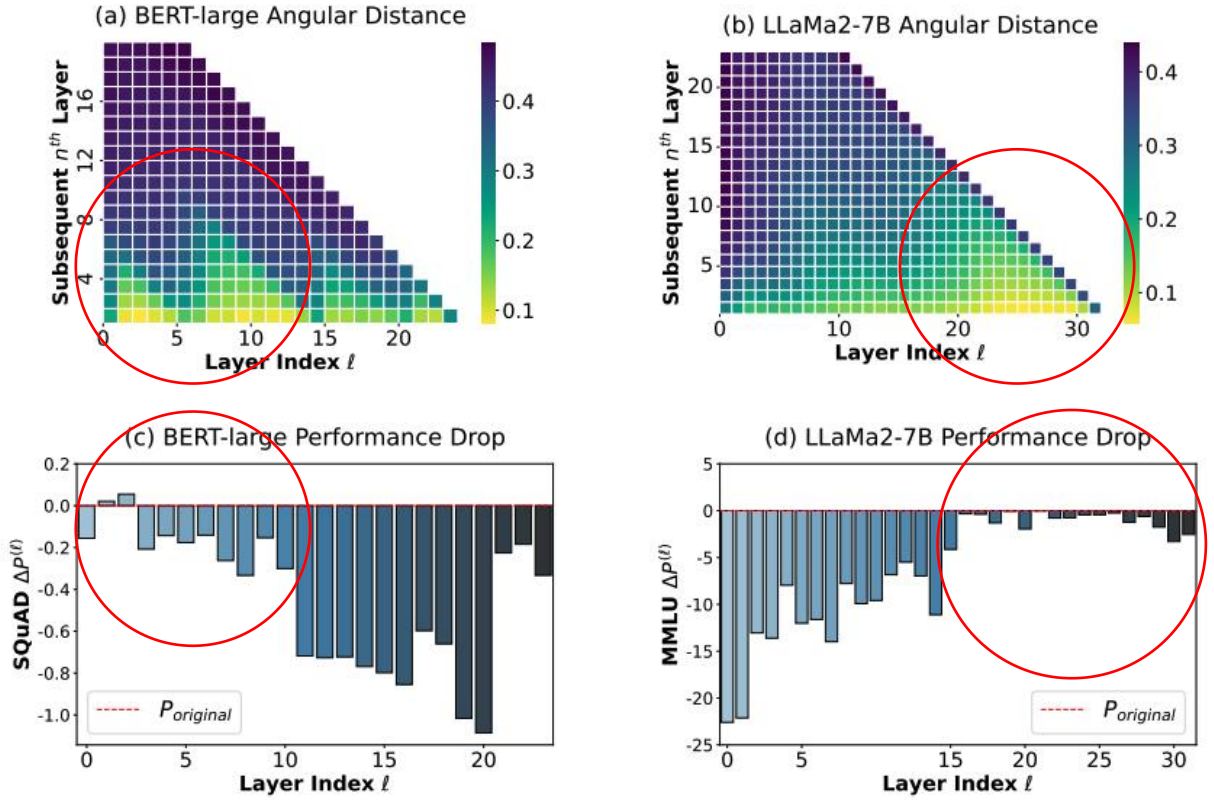


Figure 2: Results of open-weight large-scale LLMs. **Angular Distance (a, b)**: Each column represents the angular distance from the initial layer ℓ (x-axis) and its subsequent n^{th} layer (y-axis). The distance is scaled to the range [0, 1], where yellow indicates smaller distances and purple indicates larger distances. **Performance Drop (c, d)**: (c): SQuAD v1.1 performance drop of removing each single layer from BERT-large; (d): MMLU accuracy drop of removing each single layer from LLaMa2-7B.

- LLaMa2-7B's deeper layers are more similar to their adjacent layers (yellow color)
- Pruning entire deep layers leads to marginal performance drop.

More LLMs

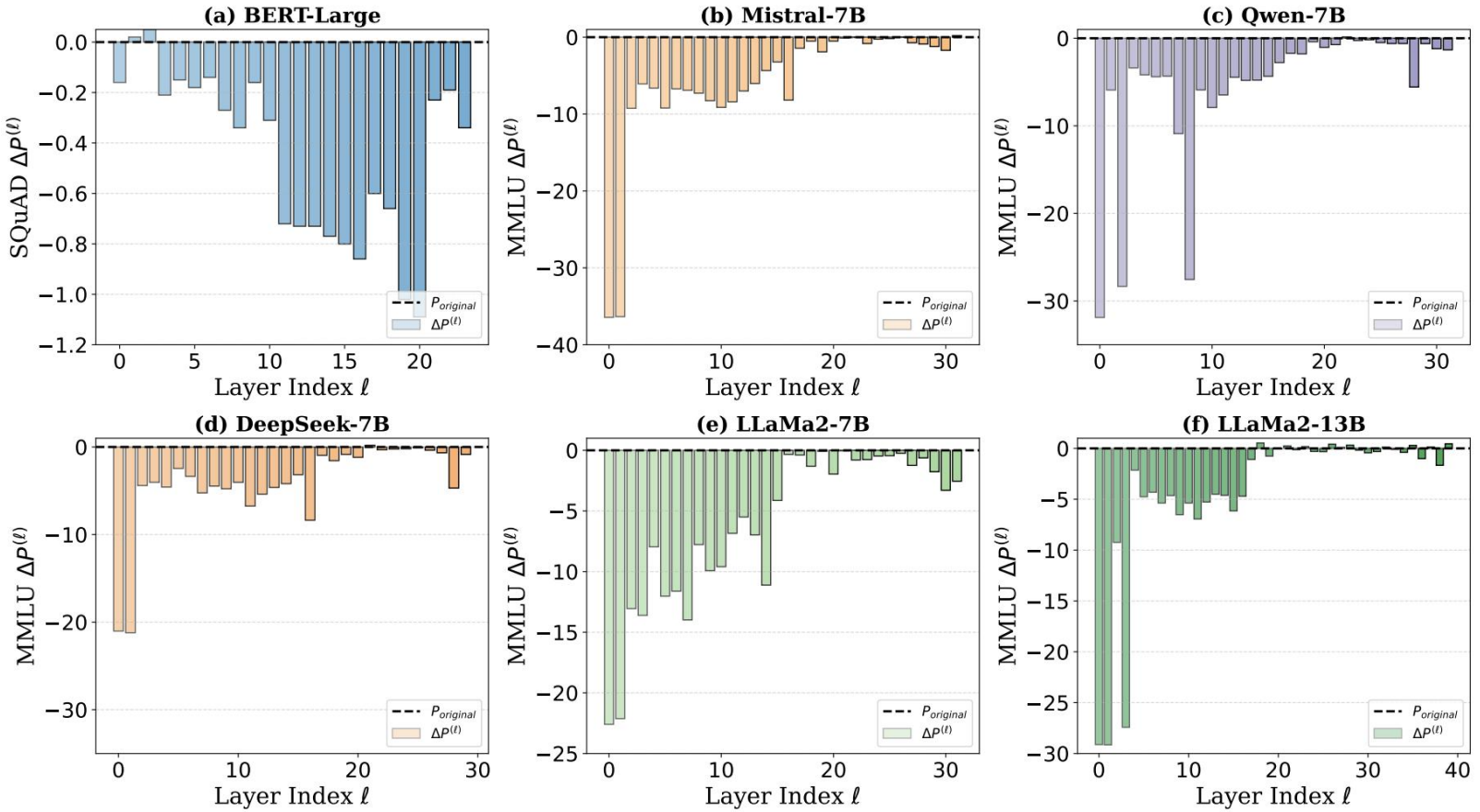
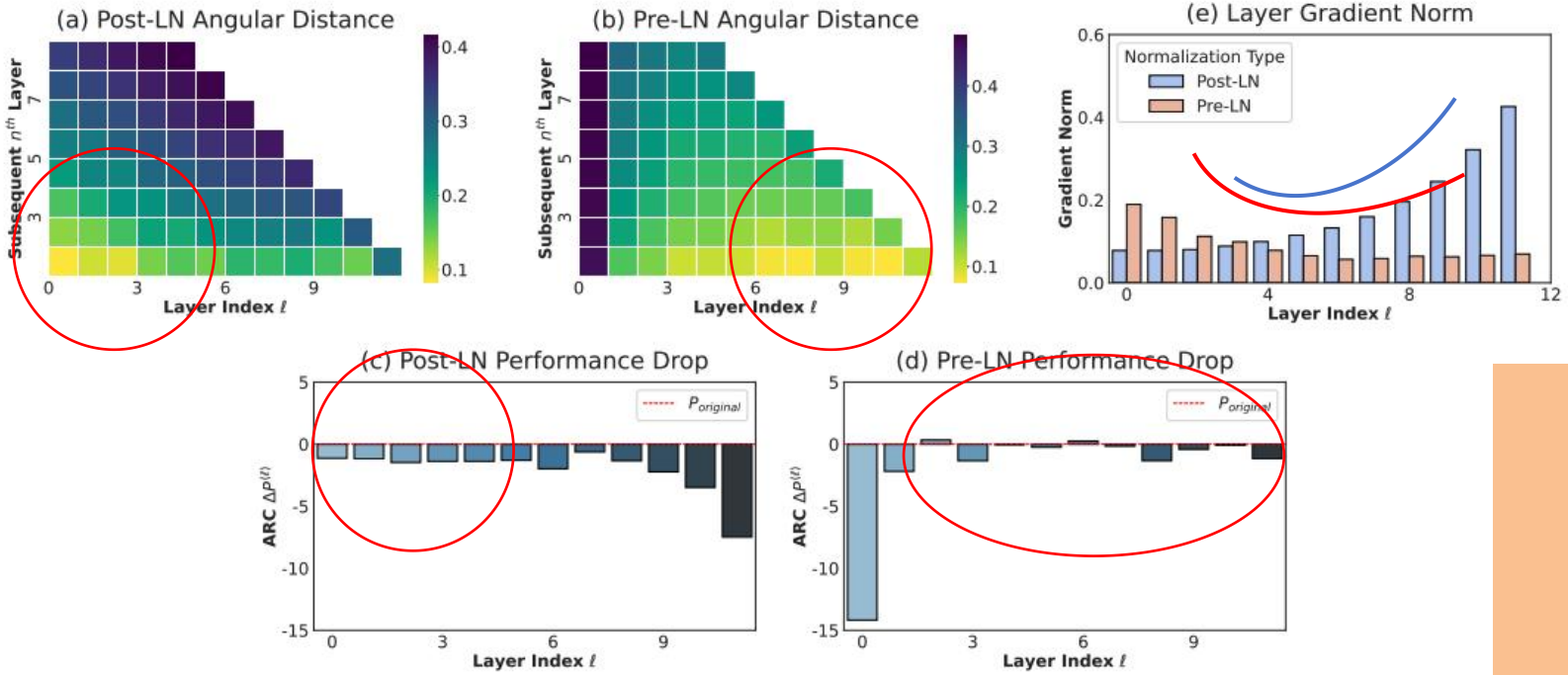


Figure 2. **Performance drop of layer pruning across different LLMs.** (a) BERT-Large (Post-LN), (b) Mistral-7B (Pre-LN), (c) Qwen-7B (Pre-LN), (d) DeepSeek-7B (Pre-LN), (e) LLaMA2-7B (Pre-LN), and (f) LLaMA2-13B (Pre-LN). The results show that Pre-LN models exhibit significant inefficiency in deeper layers, while Post-LN models maintain strong deep-layer contributions.

In-house small-scale LLMs

LLaMa-130M (Pre-LN) vs. LLaMa-130M (Post-LN)

- Post-LN leads to larger gradients in deeper layers
- Pre-LN maintains healthy gradient flow in early layers but diminishes in later layers.



- Post-LN's early layers are more similar to their adjacent layers (yellow color)
- Pruning entire early layers leads to much smaller performance drop.

- Pre-LN's deeper layers are more similar to their adjacent layers (yellow color)
- Pruning entire deep layers leads to marginal performance drop.

Figure 3: Results of in-house small-scale LLaMa-130M. **Angular Distance (a, b)**: Each column represents the angular distance from the initial layer ℓ (x-axis) and its subsequent n^{th} layer (y-axis). The distance is scaled to the range [0, 1], where yellow indicates smaller distances and purple indicates larger distances. **Performance Drop (c, d)**: ARC-e performance drop of removing each single layer from LLaMa-130M. **Gradient Norm (e)**: Gradient norm of each layer in LLaMa-130M.

Hypothesis Evaluation (Theoretically)

GOSIM

Post-LN applies $\mathrm{LN}(\cdot)$ after the residual addition:

$$\text{Post-LN}(x) = \mathrm{LN}(x + \mathcal{F}(x)).$$

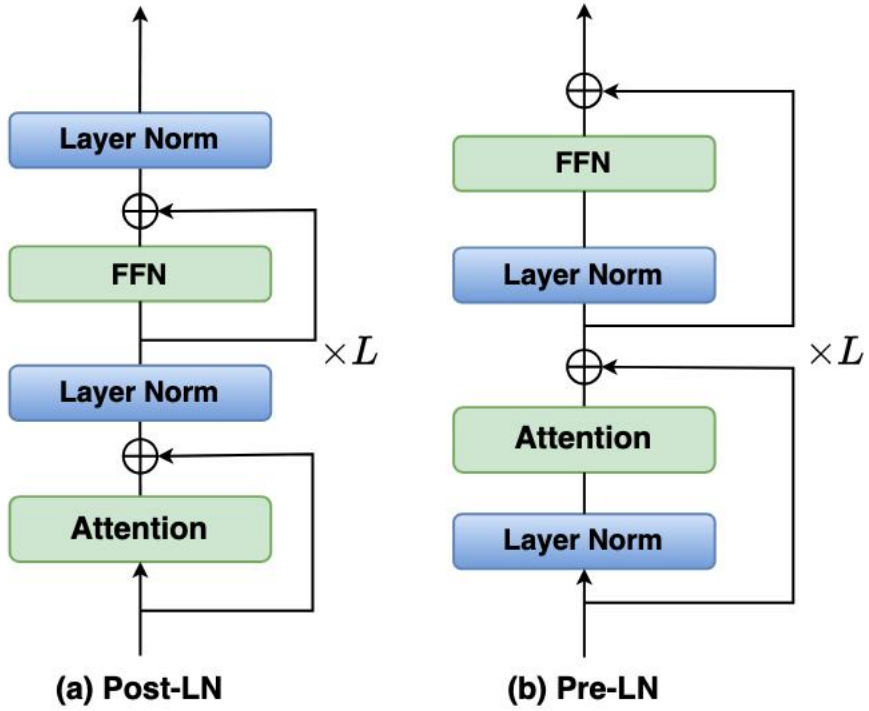
In contrast, Pre-LN applies $\mathrm{LN}(\cdot)$ before the residual addition:

$$\text{Pre-LN}(x) = x + \mathcal{F}(\mathrm{LN}(x)).$$

As shown in our submission, the derivative of Post-LN and Pre-LN are given:

$$\frac{\partial \text{Post-LN}(x)}{\partial x} = \frac{\partial \mathrm{LN}(x + \mathcal{F}(x))}{\partial (x + \mathcal{F}(x))} \left(I + \frac{\partial \mathcal{F}(x)}{\partial x} \right),$$

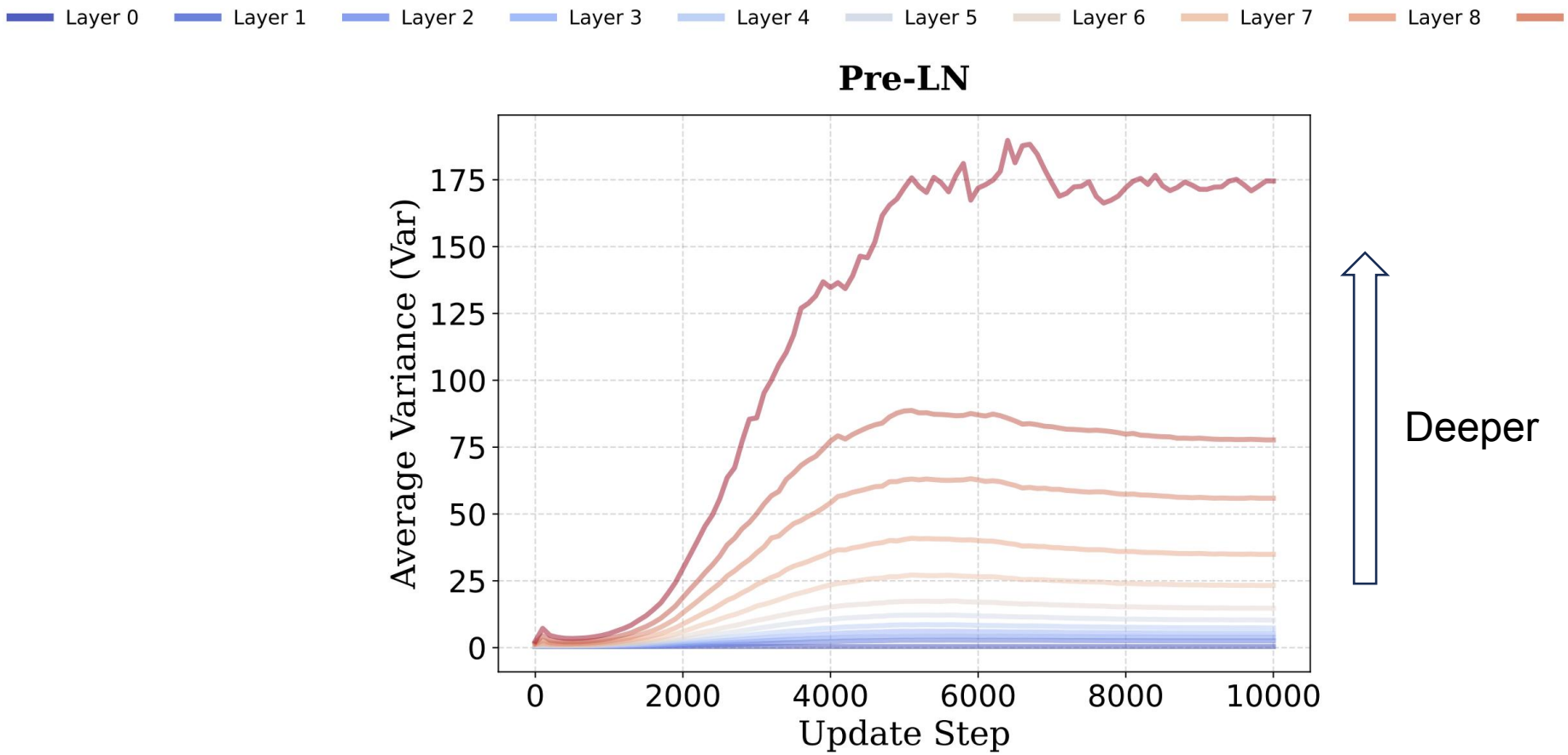
$$\frac{\partial \text{Pre-LN}(x)}{\partial x} = I + \frac{\partial \mathcal{F}(\mathrm{LN}(x))}{\partial \mathrm{LN}(x)} \frac{\partial \mathrm{LN}(x)}{\partial x},$$



Assuming x follows a normal distribution with a mean of 0, σ_x is variance of x, we have

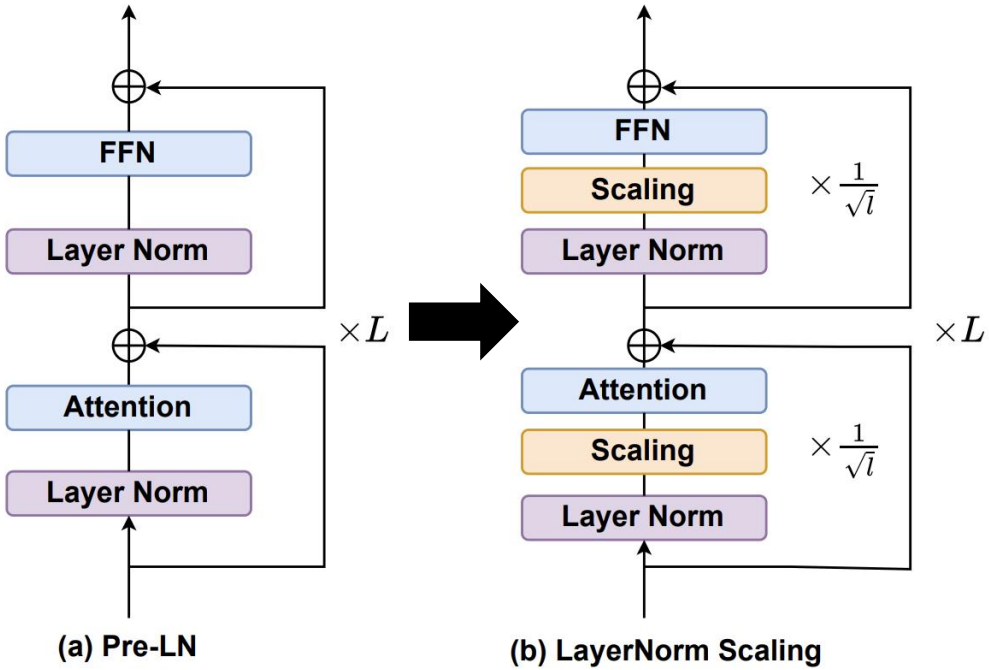
$$\frac{\partial \mathrm{LN}(x')}{\partial x'} \approx \frac{1}{\sigma_{x'}} I \approx 0.$$

Layerwise Output Variance



The Curse of Depth in Large Language Models

Wenfang Sun [*1] Xinyuan Song [*2] Pengxiang Li [3] Lu Yin [4] Yefeng Zheng [1] Shiwei Liu [5]



Layerwise Output Variance

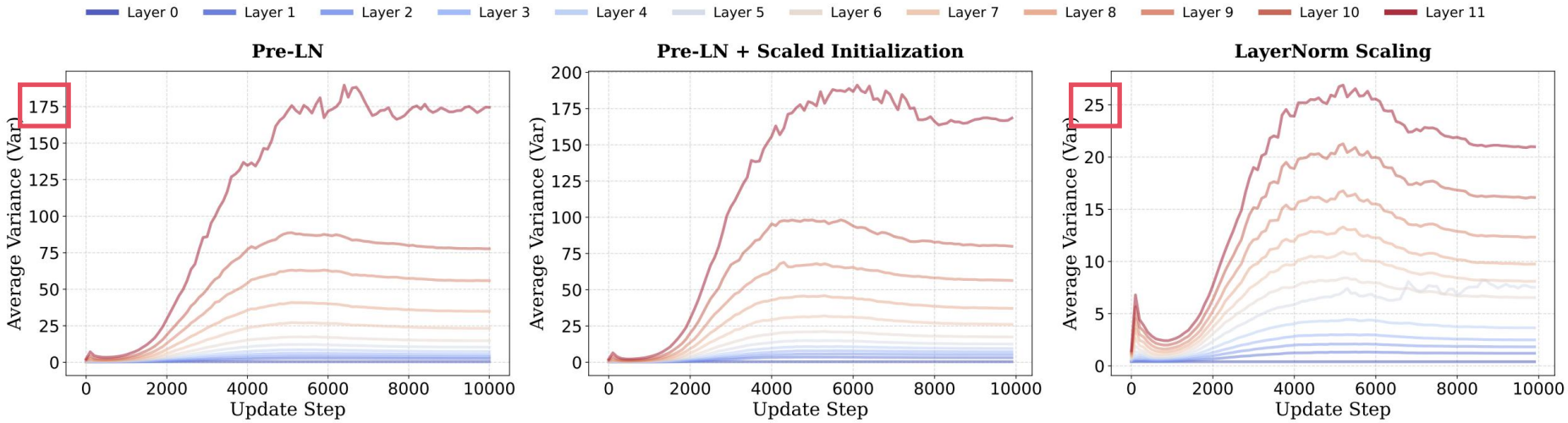


Figure 1. **Layerwise output variance.** This figure compares the output variance across various layers for different setups: (1) Pre-LN; (2) Pre-LN with Scaled Initialization; and (3) LayerNorm Scaling. The experiments are conducted on the LLaM-130M model trained for 10,000 steps. The proposed LayerNorm Scaling effectively controls the variance across layers.

Evaluation Results

LLM Pre-training Perplexity ↓

Table 1. Perplexity (↓) comparison of various layer normalization methods across various LLaMA sizes.

	LLaMA-130M	**LLaMA-250M**	**LLaMA-350M**	**LLaMA-1B**
Training Tokens	2.2B	3.9B	6.0B	8.9B
Post-LN (Ba, 2016)	26.95	1409.79	1368.33	1390.75
DeepNorm (Wang et al., 2024)	27.17	22.77	1362.59	1409.08
Mix-LN (Li et al., 2024b)	26.07	21.39	1363.21	1414.78
Pre-LN (Baevski and Auli, 2019)	26.73	21.92	19.58	17.02
Pre-LN + LayerNorm Scaling	**25.76**	**20.35**	**18.20**	**15.71**

Scaling to 7B/33B tokens

Table 2: Pre-training Perplexity of OLMO-1B and OLMO-7B.

Model	**#Params**	**Pre-LN**	**Pre-LN + LNS**
LLaMA-1B	1B	23.14	**21.99**
LLaMA-7B	7B	16.06	**14.98**

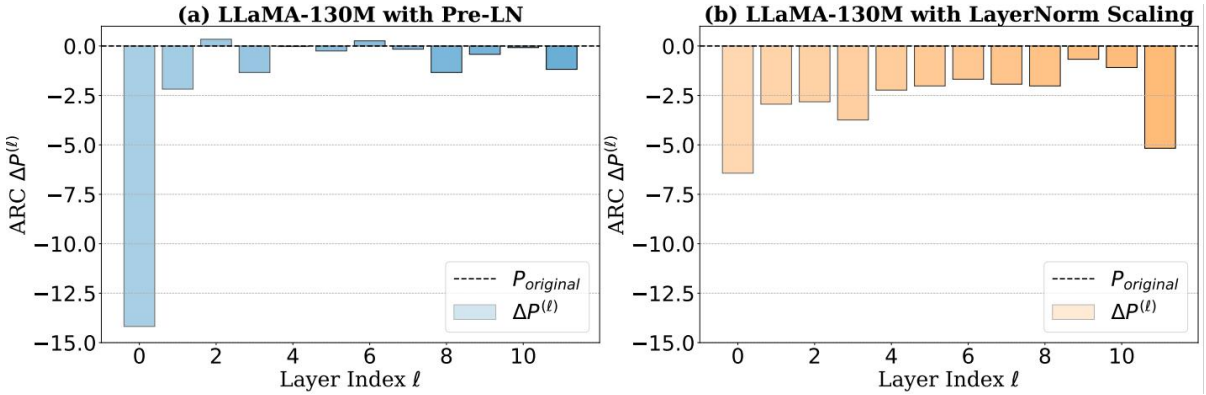
Scaling to Qwen2.5-0.5B

Table 3: Pre-training Perplexity of Qwen2.5-0.5B.

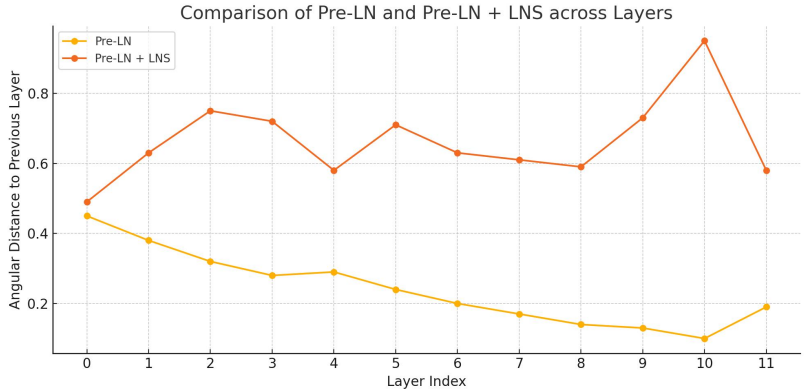
Model	**#Params**	**Pre-LN**	**Pre-LN + LNS**
Qwen2.5-0.5B	0.5B	20.62	**19.57**

Layerwise Effectiveness

Performance Drop of Layer Pruning



Layerwise Angular Distance



Conclusion

- The curse of depth in LLMs is ubiquitous
- Pre-LN is the root cause
- Scaling output of layernorm addresses this, leading to better LLMs

Future Direction

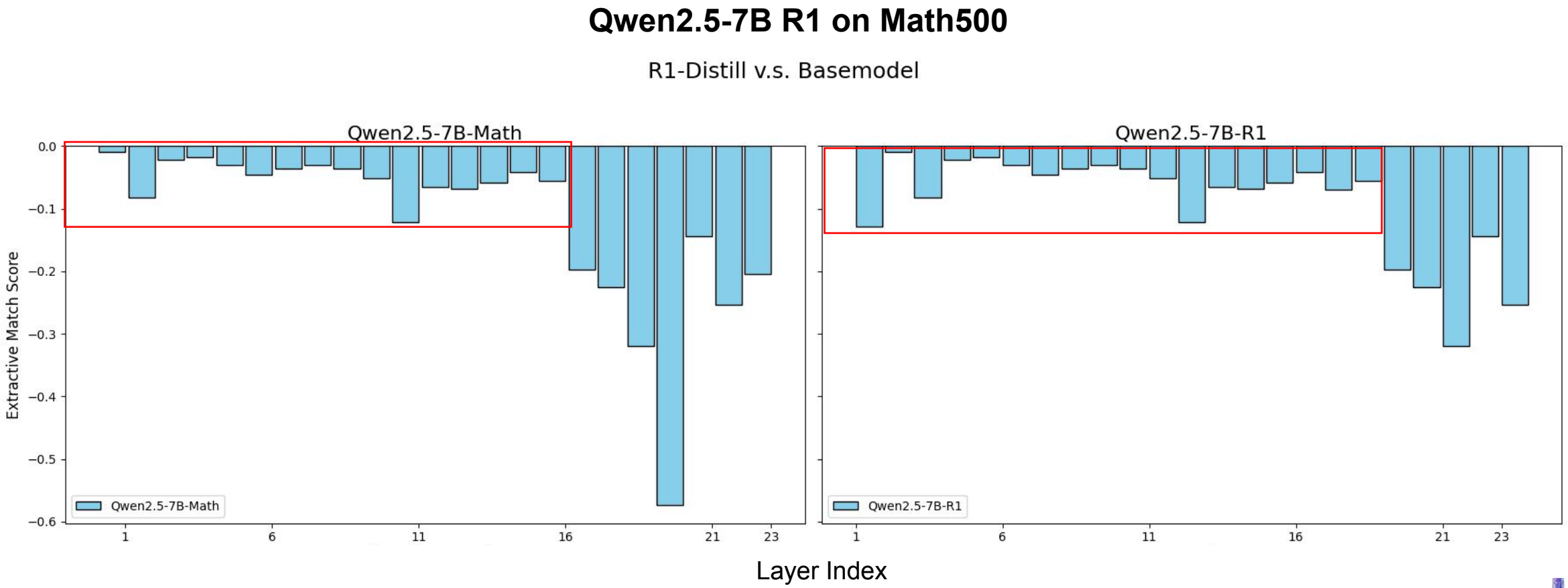
- How to leverage "Cuse of Depth" for LLM compression?
- Is it possible to address "the curse of depth" in post-training?
- Why vision transformers suffer less from Cuse of Depth?

GOSIM

GOSIM AI Paris 2025

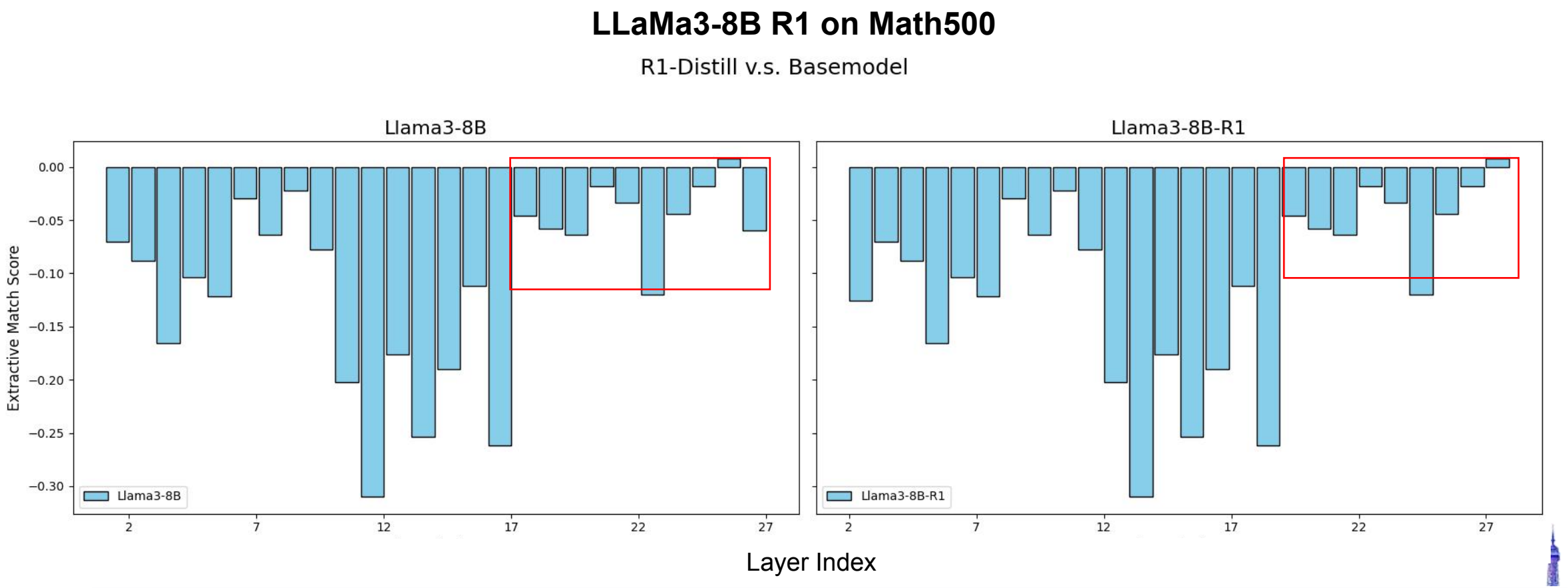
THANK YOU

How about Tasks That Involves Long Thinking?



- Qwen2.5 shows opposite trends: early layers are more important than later layers for Math500.
- R1 training does not change layer importance.

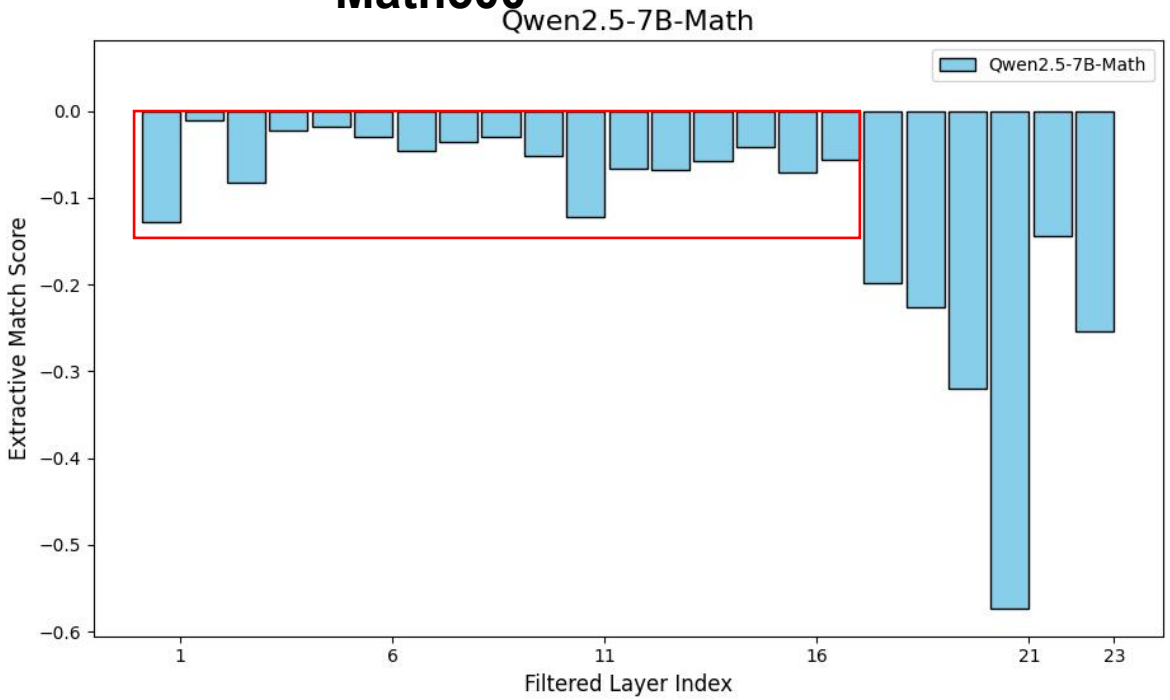
How about Tasks That Involves Long Thinking?



- ❖ Llama3: In the Math500 task, the later layers are less important than the early layers,
- ❖ But they play a more significant role compared to the MMLU task.
- ❖ R1 training does not change layer importance.

It is the Qwen Models That Make Difference

Qwen2.5-7B-Math on Math500



Qwen2-7B on Math500