PermLLM: Learnable Channel Permutation for N:M Sparse Large Language Models

Lancheng Zou Shuo Yin Zehua Pei Tsung-Yi Ho Farzan Farnia Bei Yu
The Chinese University of Hong Kong

Introduction

Background: N:M sparsity (N out of M consecutive weights are zero) enables efficient LLM deployment with hardware acceleration. Channel permutation is crucial for maintaining model performance by reordering channels to group important weights together before pruning.

Challenges:

- **Combinatorial Explosion:** Finding optimal channel order is NP-hard with $C!$ possible permutations
- **Handcrafted Metrics:** Existing methods use heuristics that don't correlate with actual pruning impact
- **Non-differentiable:** Discrete permutation matrices prevent end-to-end optimization

Our Solution: PermLLM introduces **learnable channel permutation (LCP)** that directly minimizes output errors between dense and sparse models.

- First learnable channel permutation framework for N:M sparsity
- Sinkhorn normalization for differentiable permutation learning
- Block-wise strategy for computational efficiency
- Compatible with existing pruning methods (Wanda, RIA)

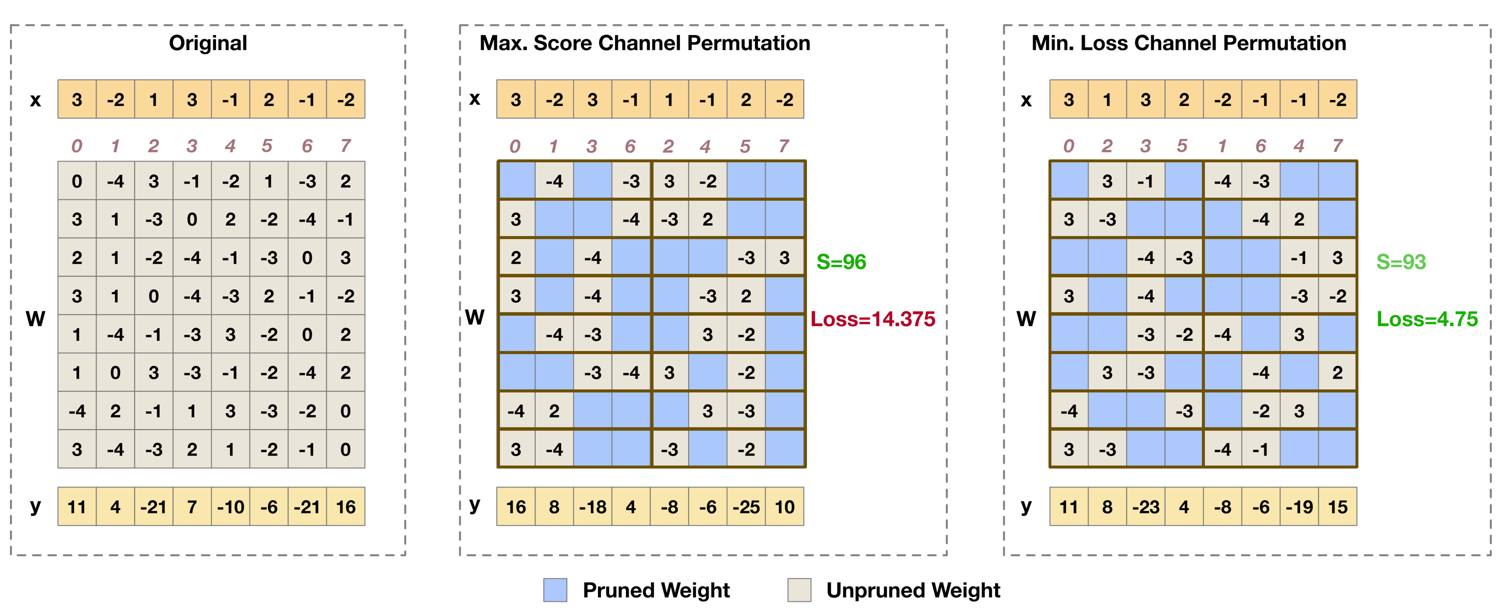

Figure 1. Effects of different channel permutation strategies on the outputs.

Learnable Channel Permutation

Leveraging Sinkhorn normalization to obtain a soft permutation matrix:

$$S^0(\mathbf{X}) = \exp(\mathbf{X}), \tag{1}$$

$$S^i(\mathbf{X}) = \mathcal{T}_c(\mathcal{T}_r(S^{i-1}(\mathbf{X}))), \tag{2}$$

$$S(\mathbf{X}) = \lim_{l\to\infty} S^l(\mathbf{X}), \tag{3}$$

$$\widehat{\mathbf{P}} = S^L(\mathbf{W}_P/\tau) \tag{4}$$

Harden the soft permutation matrix into an exact permutation matrix:

$$\mathbf{P} = \arg\max_{\mathbf{P}\in\mathcal{P}} \mathrm{Tr}(\mathbf{P}^\top \widehat{\mathbf{P}}), \tag{5}$$

Block-wise Permutation

To reduce the high computational cost and vast solution space, we employ a block-wise permutation to achieve higher efficiency.

- Reduces parameters from C_{in}^2 to $C_{in} \times B$ (block size B)
- Reduces Hungarian algorithm complexity from $O(C_{in}^3)$ to $O(C_{in} \times B^2)$
- Block-wise permutation: $\mathbf{P} = \mathrm{BlockDiag}(\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_K)$

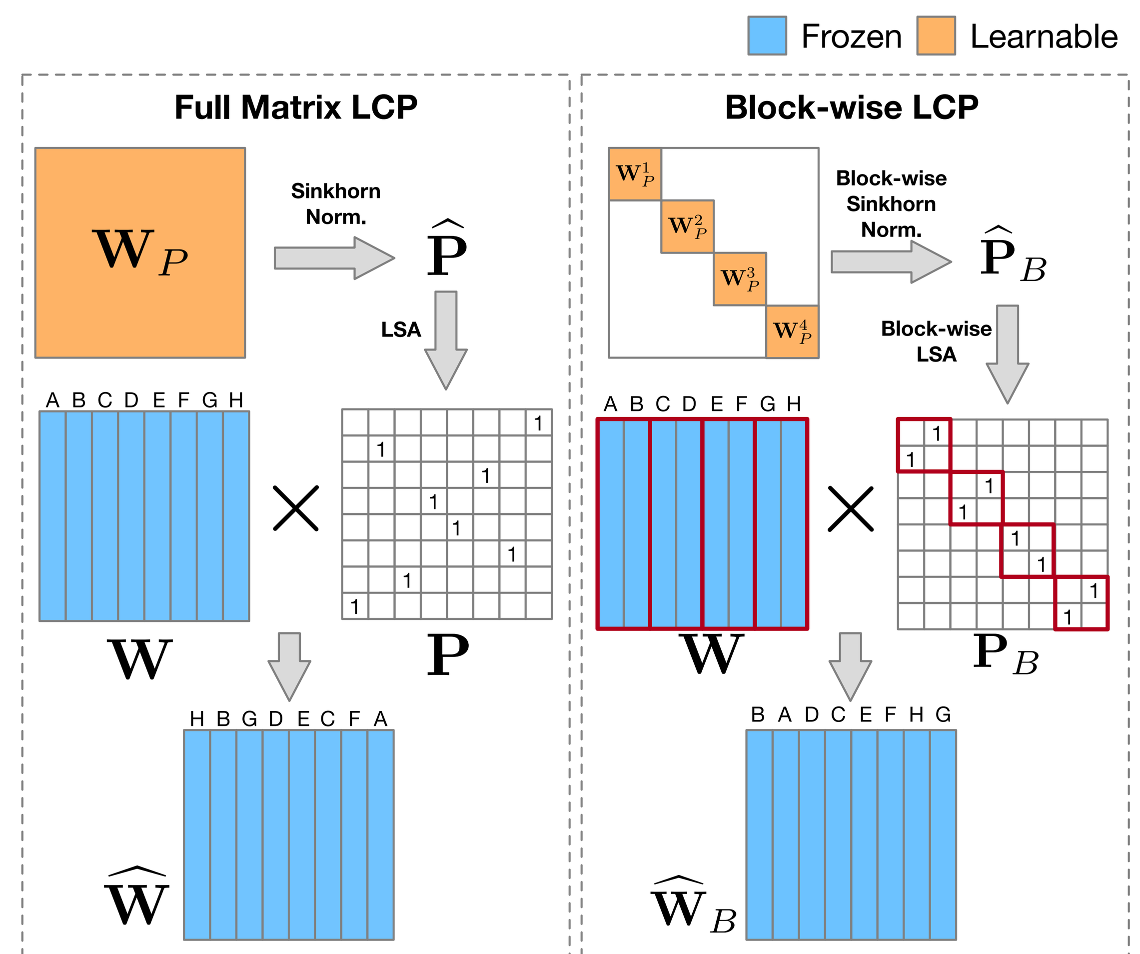

Figure 2. Illustration of learnable channel permuta- tion with different granularity: (a) full matrix LCP; (b) block-wise LCP.

PermLLM Framework

Objective: Minimize output discrepancy between dense and sparse models with the proposed block-wise learnable channel permutation.

Mask Selection:

$$\arg\max_{\mathbf{M}} \sum_{i=0}^{C_{out}} \sum_{k=0}^{C_{in}/M} \Sigma(\mathbf{M} \odot \widehat{\mathbf{S}})_{i,kM:(k+1)M}, \quad \text{s.t. } \|\mathbf{M}_{i,kM:(k+1)M}\|_0 = M - N, \tag{6}$$

where $\widehat{\mathbf{S}} = \mathbf{S}\mathbf{P}_B$.

Soft Mask:

$$\widetilde{\mathbf{M}}_{i,kM:(k+1)M} = \mathrm{Softmax}(\widehat{\mathbf{S}}_{i,kM:(k+1)M}), \tag{7}$$

Loss Function:

$$\mathcal{L}_{cosine}(\mathbf{y}, \tilde{\mathbf{y}}) = 1 - \frac{\mathbf{y} \cdot \tilde{\mathbf{y}}}{||\mathbf{y}|| \cdot ||\tilde{\mathbf{y}}||}$$

Training Process:

1. Apply Sinkhorn normalization to get soft permutations
2. Hardening via Hungarian algorithm for forward pass
3. STE approximation for gradient flow in backward pass
4. Update permutations to minimize cosine loss

Experimental Results

Table 1. Evaluation results across various models on zero-shot and language modeling tasks.

Model	Method	HellaSwag	ARC_E	ARC_C	OBQA	RTE	Average	Wikitext2
OPT 6.7B	Dense	50.46	65.49	30.12	26.80	55.23	45.62	10.86
	SparseGPT	43.40	**60.82**	26.62	**24.40**	52.71	41.59	14.33
	Wanda	41.56	57.62	24.83	23.00	53.43	40.09	16.29
	Wanda+CP	42.87	59.51	26.02	22.00	52.71	40.62	15.28
	PermLLM_{Wanda}	**44.27**	59.43	**27.22**	24.00	**54.15**	**41.81**	**14.27**
LLaMA 7B	Dense	56.95	75.38	41.89	34.80	65.34	54.87	5.68
	SparseGPT	43.55	61.78	27.90	22.80	58.12	42.83	11.19
	Wanda	42.33	61.57	28.07	23.60	51.26	41.37	11.59
	Wanda+CP	44.21	63.51	29.86	24.00	58.12	43.94	11.07
	PermLLM_{Wanda}	**47.03**	**63.30**	**30.55**	**25.00**	**62.45**	**45.67**	**9.41**
LLaMA-2 7B	Dense	57.13	76.30	43.26	31.60	62.45	54.15	5.47
	SparseGPT	44.11	64.14	**31.31**	24.20	58.84	44.52	11.12
	Wanda	41.59	61.74	30.20	24.00	53.07	42.12	12.16
	Wanda+CP	43.40	64.69	30.03	26.00	53.07	43.44	11.00
	PermLLM_{Wanda}	**46.60**	**65.49**	31.14	**26.20**	**63.54**	**46.59**	**9.39**
LLaMA-3.1 8B	Dense	60.06	81.48	51.28	33.40	70.04	59.25	6.24
	SparseGPT	44.25	**63.76**	30.55	**24.20**	**53.79**	43.31	16.62
	Wanda	38.45	58.00	26.37	19.40	52.35	38.91	23.42
	Wanda+CP	39.32	62.25	28.92	20.40	52.71	40.72	21.09
	PermLLM_{Wanda}	**45.33**	62.58	**30.97**	24.00	**53.79**	**43.33**	**14.03**
Qwen-2.5 7B	Dense	58.79	79.56	46.08	33.00	76.90	58.87	7.74
	SparseGPT	46.20	**71.13**	37.46	26.00	75.45	51.25	14.34
	Wanda	40.60	67.17	33.45	25.40	72.92	47.91	24.44
	Wanda+CP	42.92	70.50	36.09	25.20	72.20	49.38	18.76
	PermLLM_{Wanda}	**47.30**	70.58	**38.13**	**27.60**	**77.26**	**52.17**	**13.58**

Table 2. Ablation study on block size.

Block size	HellaSwag	ARC_E	ARC_C	OBQA	RTE	Average	Wikitext2	Time
32	46.13	64.39	29.69	24.60	53.07	43.58	9.50	2h
64	46.60	65.49	31.14	26.20	63.54	46.59	9.39	2.5h
128	46.47	66.08	32.08	27.40	64.43	47.09	9.07	6h

Larger block sizes provide a greater optimization space. However, this increased space comes at the cost of longer exploration and convergence times.

Key Insights

Why PermLLM Works:

- Direct optimization of model performance vs. proxy metrics
- End-to-end learning captures complex inter-layer interactions
- Adaptive permutation compensates for pruning errors

Compatibility: Seamlessly integrates with existing one-shot pruning methods

Discussions:

- Apply learnable channel permutation for quantization.
- Apply learnable channel permutation for sparse-quantization co-optimization.
- Combine PermLLM with rotation technique.