Hopscotch: Discovering and Skipping Redundancies in Language Models

Mustafa Eyceoz
Red Hat AI Innovation
meyceoz@redhat.com

Nikhil Nayak
Red Hat AI Innovation
nnayak@redhat.com

Hao Wang
Red Hat AI Innovation
hao-wang@redhat.com

Ligong Han
Red Hat AI Innovation
lihan@redhat.com

Akash Srivastava
IBM Core AI
akash.srivastava@ibm.com

EMNLP 2025 Suzhou, China 中国苏州

In a Nutshell

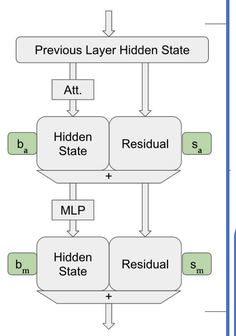
Attention blocks in language models contain highly redundant information w.r.t. specific tasks. These blocks can actually be identified and entirely removed from the model with negligible performance drop, after adding lightweight scaling parameters

Iterative Greedy Block Removal

To select an attention block for removal, we estimate the impact of removing each block. Rather than compute once and remove greedily, we must iteratively remove and recompute, since we know blocks are redundant, and therefore **information loss is not an independent consideration**

Adding Scaling Parameters

When skipping one attention block, the input distribution to the next layer changes.
To compensate the change, for each layer, we introduce **four trainable scalar factors**: one each for the outputs of the attention block, the MLP block, and the two residual connections.

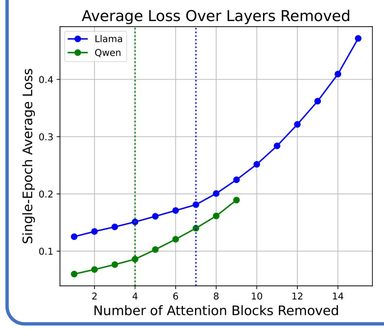


$$\boldsymbol{x}_1^{(l)} = b_{\text{att}}^{(l)}\,\text{Attention}(\text{Norm}(\boldsymbol{x}_{\text{in}}^{(l)})) + s_{\text{att}}^{(l)}\,\boldsymbol{x}_{\text{in}}^{(l)}$$

$$\boldsymbol{x}_{\text{out}}^{(l)} = b_{\text{mlp}}^{(l)}\,\text{MLP}(\text{Norm}(\boldsymbol{x}_1^{(l)})) + s_{\text{mlp}}^{(l)}\,\boldsymbol{x}_1^{(l)}.$$

Loss and Critical Break Points

$$\theta = \left\{b_{\text{att}}^{(l)}, s_{\text{att}}^{(l)}, b_{\text{mlp}}^{(l)}, s_{\text{mlp}}^{(l)}\right\}_{l=1}^{L} \qquad \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\sum_{t=1}^{T_i}\log P_\theta(y_t^{(i)} \mid y_{<t}^{(i)}, x^{(i)})\right)$$



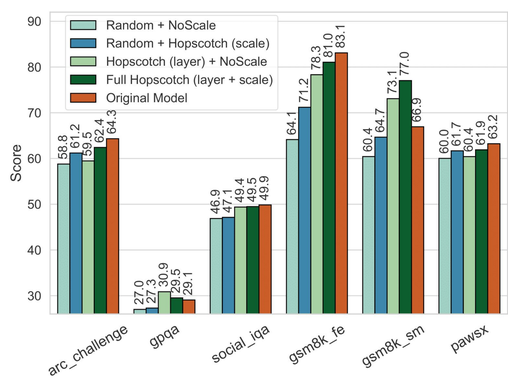
The removal of initial attention blocks results in relatively small, approximately linear increases in loss up to the seventh block for Llama 3.1 8B and the fourth for Qwen 2.5 7B.

Experimental Results with Llama-3.1-8B-Instruct

(see our paper for more experiments)

Method		gsm8k FE	gsm8k SM	arc_challenge	gpqa	social_iqa	pawsx
Baseline		83.10	66.94	64.33	29.07	49.85	63.26
4 Blocks	Hopscotch	**81.05**	**77.03**	**62.42**	**29.53**	**49.49**	**61.90**
	ShortGPT	62.33	58.45	57.00	28.44	47.59	59.49
	FinerCut	75.80	61.00	53.58	27.77	46.93	54.98
7 Blocks	Hopscotch	**79.38**	**75.82**	**61.17**	**29.39**	**48.93**	**60.44**
	ShortGPT	1.90	1.29	46.42	27.35	43.76	57.62
	FinerCut	42.50	6.80	52.65	28.10	46.47	54.87
10 Blocks	Hopscotch	**67.93**	**62.58**	**57.80**	**29.83**	**48.23**	**61.06**

Hopscotch **outperforms prior works across the board**, and retains model sanity with more blocks removed. With four blocks, we also show the impacts of Layer Selection and Scaling Parameters individually.

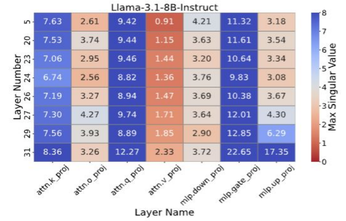


Model Efficiency Gains

Self-attention dominates wall-clock time in LLMs (~66% of forward pass). Each attention block in Llama 3.1 8B accounts for ~2.06% of total latency, skipping 7 blocks cuts inference time by ~15% while retaining 98% accuracy. Removing these blocks reduces memory by ~4–7%, with further proportional savings in KV-cache and optimizer states.

Attn. Blocks Removed	Inference Time Reduction
1	2.06%
4	8.24%
7	14.42%

Why It Works: Spectral and Distributional Analysis



Hopscotch remains effective **even when every block contributes** to model behavior. In Llama 3.1 8B, layerwise spectral analysis shows uniformly high singular values ($\|W\|_2$) indicating information-bearing transformations rather than redundant layers.
Hopscotch's learned scaling compensates for hidden-state distribution shifts, reducing Maximum Mean Discrepancy between modified and baseline activations.

Compression Compatibility

Method	Strict Match	Flexible Extract
Baseline (Instruct)	67.0	83.9
GPTQ	68.0	83.9
GPTQ + Hopscotch (4)	77.3	79.4
GPTQ + Hopscotch (7)	75.1	78.5
Hopscotch (4) + GPTQ	77.9	79.2
Hopscotch (7) + GPTQ	73.8	78.5
AWQ	65.0	81.2
AWQ + Hopscotch (4)	71.4	75.1
AWQ + Hopscotch (7)	69.5	73.8
Hopscotch (4) + AWQ	75.2	78.3
Hopscotch (7) + AWQ	73.1	79.2

Take away

With a simple effective method for skipping attention blocks, we show leading results in reducing block-level redundancies in LLMs.