LLM COMPRESSION BY BLOCK REMOVAL WITH CONSTRAINED BINARY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Abstract: We formulate the compression of large language models (LLMs) by deleting Transformer blocks ("block removal") as a constrained binary optimization (CBO) problem. This problem can be mapped to a physical system (Ising glass), whose energies we show provide a useful proxy for downstream model performance. Here, we focus on the lowest-energy state of the proxy CBO problem as the block combination to be removed, but in fact, other states in the low-energy spectrum also yield high-quality, non-trivial solutions that go beyond simple consecutive regions. Our technique outperforms other methods in 13 out of 20 model-benchmark configurations. Notably, we obtain particularly strong performance on MMLU: for example, for 37.5 and 50% compression of Llama-3.3-70B-Instruct and Llama-3.1-8B-Instruct respectively, we score 21.5 and 22.9 percentage points better than the second-best method. Interestingly, we never achieve the highest score on HellaSwag in our tested configurations. However, no single competing method consistently achieves the best results across all evaluated use cases, while our method obtains the *highest average* score in all but one test case (e.g. 11.2 points better average for the 50% compression of Llama-3.3-70B-Instruct and 6.1 points better for 35.5% compression of Llama-3.1-8B-Instruct). In general, our approach is computationally efficient and requires only forward and backward passes on a calibration dataset to compute the gradient with respect to N variables (where N is the number of transformer blocks). Additionally, we demonstrate that using good heuristic solvers for the CBO problem provides solutions that perform well on downstream tasks within seconds, even when it becomes infeasible to solve the CBO exactly.

1 INTRODUCTION

Large language models (LLMs) have demonstrated impressive capabilities in writing, summarization, and problem-solving (OpenAI, 2023), and in the last few years, the number of released models has grown rapidly (Llama Team, 2024; Qwen Team, 2025; OpenAI, 2025). However, these frontier models, built primarily on the transformer architecture (Vaswani et al., 2017), usually contain billions or even trillions of parameters, making them expensive to train and deploy, which greatly limits their use on resource-constrained devices. Consequently, substantial research effort has focused on compressing LLMs while preserving performance.

LLM compression methods take different approaches to reduce the model size, such as pruning (Sun et al., 2023; Muralidharan et al., 2024; Guo et al., 2025; Rosenberg et al., 2024), factorization (Yuan et al., 2023; Wang et al., 2025; Rausch et al., 2025), and block removal (Ding et al., 2025; Men et al., 2025; Mistral.AI, 2026). The latter consists of removing whole transformer blocks and is a particularly attractive compression method because, apart from memory savings, it also achieves predictable substantial gains in inference speed due to a shorter model and is compatible with the other compression techniques (cf. App. C). However, as models grow deeper, block pruning becomes challenging: the effect of removing a given block depends strongly on which other blocks are removed, which makes the task a hard combinatorial problem.

In this work, we introduce an algorithm for selecting combinations of transformer blocks to remove. We associate a binary block-selection variable with each block (0: keep, 1: remove), construct a second-order approximation of the loss by Taylor-expanding it with respect to these variables and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

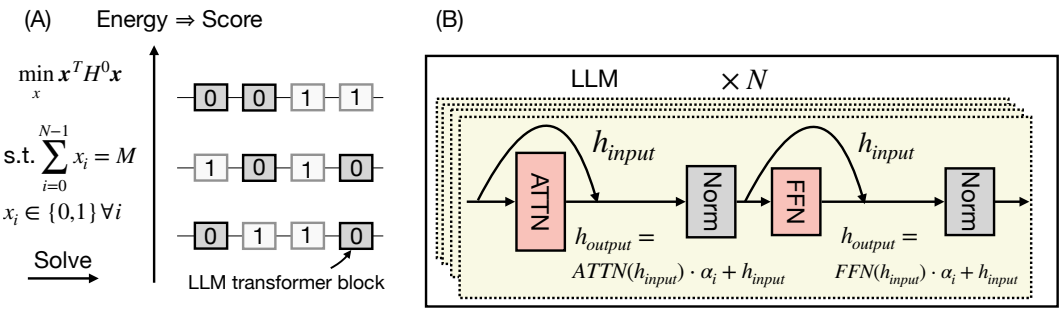


Figure 1: (A) Sketch of our method. We formulate block pruning as a constrained binary optimization (CBO) problem. Each feasible solution specifies which M out of N blocks are removed from the LLM. The optimization is constructed such that low-energy solutions correspond to pruned models with high performance across multiple benchmarks. (B) Illustration of the modification we make to the forward pass of the LLM described in Sec. 3.

compute an approximate Hessian. This maps the original task to a constrained binary optimization (CBO) problem. Although solving the CBO is exponentially complex in the number of blocks, it is substantially cheaper to compute a proxy metric for a certain configuration (its energy) than, for example, to benchmark the full model. Furthermore, as demonstrated in Sec. 4.2, heuristic solvers can often find the optimal (or close to optimal) solution in realistic scenarios within seconds.

In this work, we do a thorough testing of our method by comparing it to other state-of-the-art block-removal methods (Ding et al., 2025; Men et al., 2025; Mistral.AI, 2026; Sieberling et al., 2025) on the benchmarks MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), the ARC Challenge (Clark et al., 2018), and BBH (Suzgun et al., 2022) and the models Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct Llama Team (2024), Qwen3-14B (Qwen Team, 2025), and Falcon3-7B-Instruct (Technology Innovation Institute, 2024a). When averaged over the complete benchmark set, our method scores the highest in all test cases except for one (where the diagonal approximation of the same problem score the highest). In 13 out of 20 configurations (benchmarks and models), it gets the highest score, while the second-best method obtains the highest score in 3 ((Sieberling et al., 2025), counting instances using two different datasets for calibration). Interestingly, we observe that our method scores systematically lower than the best alternative on HellaSwag on all test cases, indicating that it is hard to find one method for all purposes. Among the methods that we have tried, there is also not one single method that consistently scores the highest in that benchmark. Note that the nature of the problem is such that large chunks of weights are removed without compensation, so that a block removal combination which is good across all benchmarks may simply not exist.

In summary, our main contributions are:

- We formulate transformer block pruning as a CBO problem by explicitly modeling second-order interactions between blocks, with tractable computational cost for models of practical depth.
- We show that the energy of the CBO problem is a useful proxy of the quality of the block-pruned model, in the sense that it correlates well with benchmark scores.
- We show that approximate solvers also deliver high-quality candidate solutions, eliminating the need for an exact CBO solver for large models.
- We demonstrate that this formulation outperforms other block-removal techniques across several benchmarks and finds a higher average score for 3 out of the 4 test cases.

2 Related work

Existing LLM compression approaches can broadly be categorized into pruning methods, factorization algorithms, accuracy-reduction methods, and structural approaches that remove entire transformer blocks ("depth pruning"). The method presented in this work falls into the last category.

In pruning, one can distinguish between structured ("width pruning") and unstructured pruning. Whereas the former removes complete rows and columns from the weight matrices, the latter focuses on removing individual weights. In both cases, the decisions are based on magnitude or sensitivity metrics (Sun et al., 2023; Muralidharan et al., 2024; Guo et al., 2025; Rosenberg et al., 2024). Factorization algorithms, on the other hand, are often based on the singular value decomposition, approximating weight matrices as the products of two lower-rank matrices (Yuan et al., 2023; Wang et al., 2025; Rausch et al., 2025). Lastly, quantization (Frantar et al., 2022; Lin et al., 2023) reduces the numerical precision of the weights and decreases their size without changing the number of parameters, and can be combined with the other techniques.

Removing entire transformer blocks directly reduces the model depth and boosts inference speed. Prior work in this area includes analyzing how input representations evolve across blocks (Chen et al., 2025; Men et al., 2025; Gromov et al., 2025; Mistral.AI, 2026), estimating block importance via various heuristic or sensitivity-based metrics (Kim et al., 2024), iteratively removing blocks under a global objective recomputed after each removal (Song et al., 2024; Zhong et al., 2024), and incorporating block merging strategies (Ding et al., 2025). While often effective, these methods differ in how they treat interactions between blocks. Methods based on local profiling do not explicitly take interactions between blocks into account when selecting a global subset to remove, and might therefore be called "mean-field methods" in physical terms. This, for example, involves scoring the blocks independently and removing the k lowest ranked blocks. Greedy algorithms do capture the influence of removing different blocks, but are restricted to exploring a single chain of removed blocks without backtracking. Another approach is to restrict the search to combinations of consecutive blocks only (Gromov et al., 2025), which goes beyond mean-field, but greatly reduces the combinatorial space. This approach also has an implicit assumption of continuity across the model blocks. As will be shown below, however, we find that high-quality solutions in fact do not always exhibit one contiguous removal region. Recently, a genetic algorithm was proposed to solve the block-removal problem (Sieberling et al., 2025). This is one possible approach that goes beyond mean-field, but whose success depends on whether better removal combinations can be iteratively bred from worse ones.

Our approach is inspired by second-order pruning techniques that rely on a Taylor expansion of the loss function. Such methods were first introduced in the context of weight pruning through a technique called *Optimal Brain Damage* (LeCun et al., 1989), and were later extended to combinatorial formulations, such as the *Combinatorial Brain Surgeon* (Yu et al., 2022), with further scalability improvements via iterative schemes (Rosenberg et al., 2024). However, these works focus on removing individual weights or channels rather than entire transformer blocks. In contrast, we introduce auxiliary variables that govern the selection of whole blocks. The second-order term of the loss function is then used to explicitly consider interactions between blocks, which goes beyond the previous mean-field approaches and respects the full combinatorial nature of the problem.

3 METHOD

Our method extends the ideas from the *Combinatorial Brain Surgeon* (Yu et al., 2022) to block removal. In that work, the authors describe how to identify which weights to prune by solving a combinatorial optimization problem. To adapt this approach to block-level pruning, we introduce a coupling variable α_i for each transformer block i. The residual connections in the forward pass of the attention block $ATTN$ and the feed-forward neural network block FNN are modified as follows:

$$h_{\text{output}} = ATTN(h_{\text{input}}) \cdot \alpha_i + h_{\text{input}} \,, \tag{1}$$

$$h_{\text{output}} = FFN(h_{\text{input}}) \cdot \alpha_i + h_{\text{input}} \,. \tag{2}$$

This modification is illustrated in Fig. 1 (B). We initialize $\alpha_i = 1$ for all i, which recovers the original model.

Next, we pass a calibration dataset batch through the model and consider the change in the cost function $\mathcal{L}(\boldsymbol{\alpha})$ (in our case always the cross-entropy) under perturbations $\delta\boldsymbol{\alpha} = \boldsymbol{\alpha} - \boldsymbol{\alpha}^0$ by performing a Taylor expansion (with the assumptions that the weights are close to optimal and the first order term is zero) (LeCun et al., 1989; Hassibi et al., 1993; Kurtic et al., 2022; Rosenberg et al., 2024):

$$\delta\mathcal{L}(\boldsymbol{\alpha}) = \mathcal{L}(\boldsymbol{\alpha}) - \mathcal{L}(\boldsymbol{\alpha}^0) \simeq \frac{1}{2}\delta\boldsymbol{\alpha}^\top H^0 \delta\boldsymbol{\alpha} + \mathcal{O}(\delta\boldsymbol{\alpha}^3) \,, \tag{3}$$

where $H^0 = \nabla^2 \mathcal{L}(\boldsymbol{\alpha}^0)$ is the Hessian matrix.

Similar to Ref. Yu et al. (2022), we introduce binary variables x_i with the meaning $x_i = 0$: block is retained, $x_i = 1$: block is removed, and write

$$\delta\alpha_i = \alpha_i - \alpha_i^0 = (1 - x_i)\alpha_i^0 - \alpha_i^0 = -x_i\alpha_i^0 \,. \tag{4}$$

We are then left with the following constrained binary optimization problem:

$$\min_{\boldsymbol{x}\in\{0,1\}^N} \boldsymbol{x}^\top H^0 \boldsymbol{x} \quad \text{subject to} \quad \sum_{i=0}^{N-1} x_i = M \,. \tag{5}$$

Here, M denotes the targeted number of blocks to be removed, and N the total number of blocks in the model.

Computing the exact Hessian requires second derivatives of the loss and can be considerably more expensive than computing ordinary gradients. We therefore replace H^0 by an outer-product-of-gradients approximation, closely related to the Fisher information and generalized Gauss–Newton curvature matrices commonly used in second-order optimization (Amari, 1998; Schraudolph, 2002; Martens, 2020). Let

$$\ell_s(\boldsymbol{\alpha}) \tag{6}$$

denote the loss associated with calibration sample s, such that

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{m}\sum_{s=1}^{m} \ell_s(\boldsymbol{\alpha}), \tag{7}$$

and define the corresponding per-sample gradient with respect to the coupling variables as

$$\boldsymbol{g}_s = \nabla_{\boldsymbol{\alpha}} \ell_s(\boldsymbol{\alpha}^0) \in \mathbb{R}^N. \tag{8}$$

Collecting these gradients row-wise gives

$$\mathcal{A} = \begin{bmatrix} \boldsymbol{g}_1^\top \\ \boldsymbol{g}_2^\top \\ \vdots \\ \boldsymbol{g}_m^\top \end{bmatrix} \in \mathbb{R}^{m\times N}. \tag{9}$$

We then approximate the curvature matrix as

$$H^0 \approx \widehat{F}_{\text{emp}} = \frac{1}{m}\sum_{s=1}^{m} \boldsymbol{g}_s \boldsymbol{g}_s^\top = \frac{1}{m}\mathcal{A}^\top\mathcal{A}. \tag{10}$$

For a negative log-likelihood objective such as cross-entropy, this quantity is commonly referred to as the *empirical Fisher information matrix* (Martens, 2020). It can be obtained using only first-order backpropagation and is positive semidefinite by construction. The latter property is particularly convenient in our setting, since it provides a well-behaved quadratic measure of the joint sensitivity of a set of blocks. Similar empirical-Fisher approximations have also been successfully employed in second-order neural-network pruning methods (Singh & Alistarh, 2020).

We note the convenient property of the problem that the Hessian only needs to be computed once, after which Eq. (5) can be solved for a variety of different values of M (i.e. different compression rates), and with various solvers.

4 EXPERIMENTS

First, we want to experimentally demonstrate that the energy of the optimization problem is indeed a good proxy for downstream tasks. To do that, we remove 12/32 blocks from Llama-3.1-8B-Instruct (Llama Team, 2024) model. In all the following results, we use 2048 samples randomly drawn from OpenHermes-2.5 (Teknium, 2023), a calibration dataset for generating the Hessian (we study the effect of different datasets in App. A).

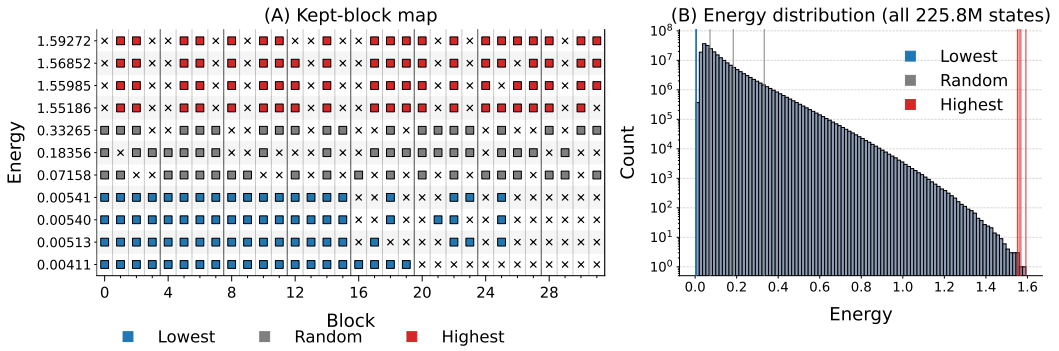


Figure 2: (A) State block configuration for the four lowest (blue), four highest (red) and three random energy states for Llama-3.1-8B-Instruct (12/32 blocks targeted for removal in total). Y-axis: state energy, x-axis: transformer block index, square: block was recommended for keeping, X: block is recommended for removal in that state. (B) Energy distribution for the same model. The red, grey, and blue horizontal lines show where the states from (A) were selected.

To get an idea of the performance on downstream tasks, we choose the benchmarks MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), the ARC Challenge (Clark et al., 2018), and BBH (Suzgun et al., 2022), all evaluated using lm-eval-harness (Gao et al., 2024).

The full workflow of the method is:

1. Forward pass of the calibration batch through the model with gradient computation that leads to the construction of the Hessian H^0 (Eq. (3)) from the gradient (Eq. (10)).
2. Solving the CBO problem Eq. (5) (minutes to hours for the exact brute-force approach, seconds using the tabu algorithm), see Sec. 4.2 for more details.
3. Remove the blocks according to the lowest-energy solution ("ground state") and benchmark.

4.1 Energy as a proxy for downstream performance

First, we empirically verify that the energy of the CBO problem is a good proxy for downstream performance. In Fig. 2, we (A) show a block removal map for the Llama-3.1-8B-Instruct model while removing 12/32 blocks, and (B), the energy distribution. In (A), the x-axis shows the block indices, and the y-axis the energies of the different states. Whenever a solution indicates the removal of a block, the corresponding square is marked with an "x"; and if the block is kept, the square is marked with a color. For example, the state with energy 0.00411 recommends removing blocks [20-31] while keeping the rest. In A, we show the configuration for the four lowest energy states (blue), three random states (grey), and the four highest excited states (red). We see that for this model, the low-lying states prefer removing blocks towards the end of the model, whereas the highly excited states choose blocks from different parts of the model. While in this case we observe that the lowest-energy state suggests the removal of the last consecutive 12 blocks, we note that this is not true in the general case (see App. B). In Fig. 2, we show the distribution of the energy spectrum. The blue, red, and grey lines illustrate where the configurations from (A) were taken from.

In Fig. 3 we show the benchmark scores as a function of energy for the same states as shown in Fig. 2. We see that whereas the 4 lowest energy states all have benchmarks above the random mean, the downstream pruned model quickly degrades for high-energy combinations. We note that the decrease of benchmark values is not monotonous for all benchmarks. For example, Winogrande goes from 60.22 to 58.56, and then rises again to 61.25, meaning that even though the lowest-energy state generally seems to provide the strongest results, it might be worth exploring other low-lying excited states if one is interested in specific benchmarks.

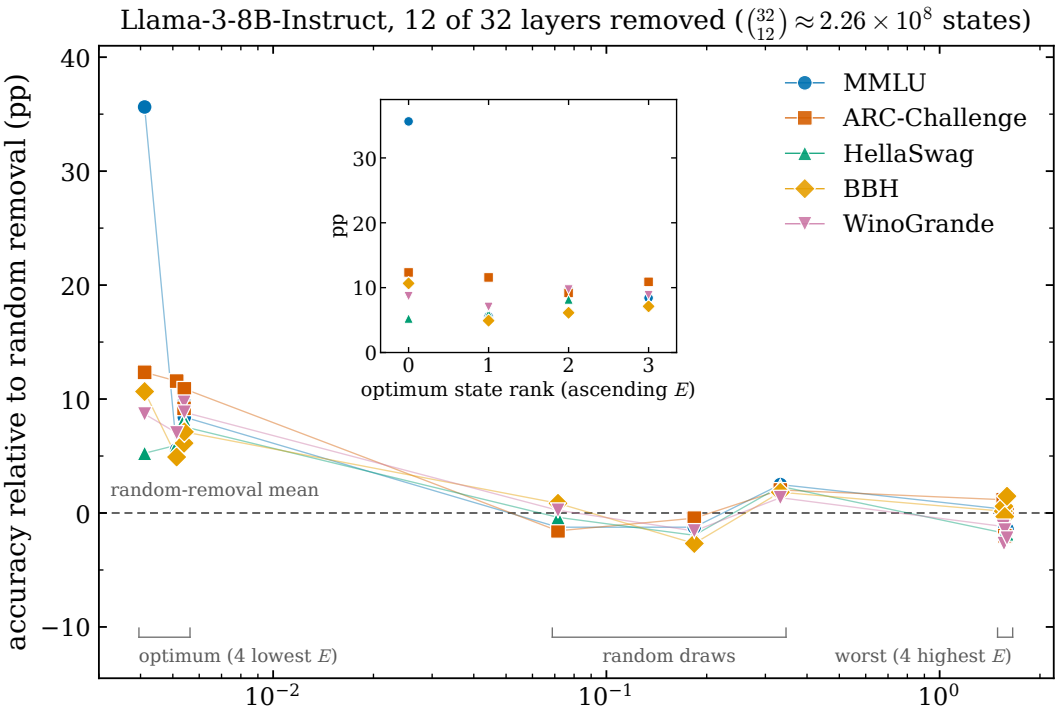


Figure 3: Percentage points (pp) on the different benchmarks as a function of energy after removing the random mean. We show the values for the same states as shown in Fig. 2. The inset shows the scores for the lowest 4 energy states.

4.2 Solving the CBO problem

Checking each CBO combination just involves the computation of the energy $E = \boldsymbol{x}^\top H^0 \boldsymbol{x}$, where H^0 is an $N \times N$ matrix. This is very cheap, but the number of combinations is given by the binomial coefficient $\binom{N}{M}$, which grows exponentially. Luckily, for the block removal problem, N is never exceedingly large (for the models here: $N = 28 \ldots 80$). Note that the problem is also tractable for large N, as long as M is small (i.e. for removing a few blocks from a very deep model).

In our code, we perform a batched matrix-matrix multiplication on an H200 GPU to accelerate the brute-force computation, and are able to fully check combinations up to the order of $O(10^{10})$. We note that the problem is also embarrassingly parallel and could be split over multiple GPUs for additional acceleration, but we did not attempt this.

The computation time is just a couple seconds for a few million combinations, and of the order of minutes below 1B combinations. For the case of 5.59B combinations, it took ~3.5 hours. The hardest still tractable problem would be to remove 8/80 blocks of Llama-3.3-70B-Instruct (29B combinations), which took about ~2 days. Removing more blocks from Llama-3.3-70B-Instruct becomes intractable.

When dealing with the intractable regime, the problem can be reformulated as a quadratic unconstrained binary optimization (QUBO) problem, where the constraint is now accounted for by a penalty term. In condensed matter physics, this is equivalent to an Ising glass model (all-to-all coupling) with conserved magnetization. This allows us to employ highly optimized classical or quantum solvers (Farhi et al., 2014; Quinton et al., 2025; Gurobi Optimization, LLC, 2024; Liang, 2026; Altelarrea-Ferré et al., 2026). Gurobi (Gurobi Optimization, LLC, 2024) is commonly regarded as the best solver on the market, and can be used without license for up to $N = 200$, which is more than enough for the block-removal problem. However, we find that the open-source tabu algorithm (Liang,

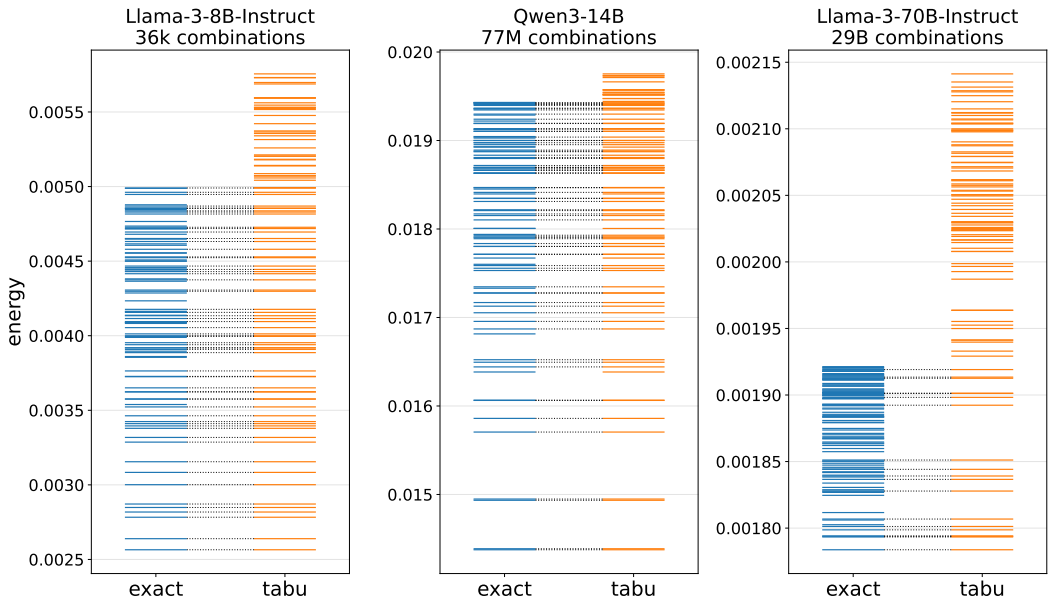


Figure 4: Comparison between the exact low-energy spectrum and the solutions obtained by the tabu algorithm for an easy, medium and hard CBO problem. Tabu was set to provide 1000 solutions, we show the lowest 100 in both cases for clarity. Matching solutions are joined by dotted lines.

2026) gives comparably good performance, reliably converging to the lowest-energy solutions in a matter of seconds for the hardest problems we can check. It can thus also be employed to further save resources even in the tractable case.

Figure 4 compares the exact low-energy states with those obtained by the tabu algorithm. We find that tabu reliably outputs the ground state for all cases shown, including the space of $\binom{80}{8} \approx 29$B combinations for Llama-3.3-70B-Instruct. It is also able to output the lowest excited states at a runtime of seconds, but fails to fully reproduce the spectrum further up, especially for very large combinatorial spaces. However, a faithful reconstruction of the full low-energy spectrum is not a requirement for our problem. We note that we can in principle run several solvers in parallel, or further iterate over tabu given the cheap running time, in order to generate even more solutions.

4.3 Results

Finally, we compare our method with several other methods for block removal. In Table 1, we show the full benchmark results, for our method (CBO), the ground state when only considering the diagonal elements in H^0 in Eq. (5), block influence (Men et al., 2025), Sliding-Window Merging (Ding et al., 2025), Norm ratio (Mistral.AI, 2026), and EvoPress (Sieberling et al., 2025) using the same calibration data (2048 samples randomly drawn from OpenHermes-2.5 (Teknium, 2023)), except for EvoPress, where we also used FineWeb-Edu (Lozhkov et al., 2024) recommended by the authors. We are interested in compression that is as deep as possible, but avoids rendering the models completely random (which would require re-training). Common experience is that smaller models are harder to compress, so we compress them by about $\sim$30% (neglecting the embeddings and other weights). In practice, this means that we remove 8/28 blocks from Falcon3-7B-Instruct Technology Innovation Institute (2024a), 12/32 blocks from Llama-3.1-8B-Instruct (Llama Team, 2024), and 12/40 blocks from Qwen3-14B (Qwen Team, 2025). For the larger Llama-3.3-70B-Instruct (Llama Team, 2024), we increase the compression rate to 50%, removing 40/80 blocks. Note that we do not show Sliding-Window Merging for Llama-3.3-70B-Instruct due to the costly parameter tuning needed to target the right compression rate.

We observe that when averaging over the benchmark set, CBO is scoring the highest in all cases except one (Falcon3-7B-Instruct where it loses to the diagonal matrix despite having the highest score in 3/5 benchmarks). For Qwen3-14B (Qwen Team, 2025) and Llama-3.3-70B-Instruct, the diagonal

Table 1: Benchmarks of block-removal methods: The #br column gives the number of removed blocks, #comb column gives the total number of all possible removal combinations $\binom{N}{M}$. The methods are: the CBO ground state, Diagonal (ground state when only considering the diagonal elements of H^0 in Eq. (5)), BI = block influence (Men et al., 2025), SWM = Sliding-Window Merging (Ding et al., 2025), Norm ratio (Mistral.AI, 2026), EvoPress (Sieberling et al., 2025) Note that for Llama-3.3-70B-Instruct, we used the tabu algorithm to solve the optimization problem as discussed in 4.2.

# br	#comb		MMLU	HellaSwag	Winogrande	ARC-C	BBH	Mean
Falcon3-7B-Instruct (28 blocks)								
0		Original	68.0	60.4	70.3	55.7	54.6	61.8
8	3.11M	CBO	**45.0**	37.8	**61.4**	36.6	**40.9**	44.3
		Diagonal	41.0	46.6	59.8	40.9	37.2	**45.1**
		BI	30.5	**49.0**	57.8	**41.8**	34.2	42.6
		SWM	39.9	39.2	59.7	33.3	38.3	42.1
		Norm ratio	24.5	42.0	56.3	30.7	30.9	36.9
		EvoPress OpenHermes	30.9	48.4	58.3	39.4	33.6	42.1
		EvoPress FineWebEdu	28.5	48.6	56.7	39.3	32.4	41.1
Llama-3.1-8B-Instruct (32 blocks)								
0		Original	68.3	59.8	73.7	53.6	51.1	61.3
12	225.8M	CBO	**59.9**	33.0	60.2	**34.1**	**39.1**	**45.3**
		Diagonal	32.2	32.4	**64.6**	30.1	36.7	39.2
		BI	38.4	33.2	56.7	28.3	35.6	38.4
		SWM	25.3	31.6	58.4	25.6	30.4	34.3
		Norm ratio	24.2	36.0	53.8	23.8	29.1	33.4
		EvoPress OpenHermes	25.2	**36.1**	54.8	23.6	30.4	34.0
		EvoPress FineWebEdu	24.4	35.9	54.7	23.8	29.8	33.7
Llama-3.3-70B-Instruct (80 blocks)								
0		Original	82.2	66.6	82.8	60.6	69.8	72.4
40	10^{23}	CBO	**76.9**	34.9	**69.0**	**37.5**	**48.8**	**53.4**
		Diagonal	23.3	30.0	55.5	25.1	28.9	32.5
		BI	54.0	28.1	61.3	28.8	34.7	41.4
		Norm ratio	27.4	37.7	61.1	26.5	33.9	37.3
		EvoPress OpenHermes	35.6	**43.7**	62.4	30.1	34.4	41.2
		EvoPress FineWebEdu	38.6	43.2	62.7	29.8	36.5	42.2
Qwen3-14B (40 blocks)								
0		Original	77.1	60.9	72.7	58.7	63.2	66.5
12	5.59B	CBO	**71.8**	34.7	59.3	**35.8**	**47.4**	**49.8**
		Diagonal	24.5	26.2	49.1	19.0	30.3	29.8
		BI	61.6	39.8	58.2	35.0	43.6	47.6
		SWM	53.1	35.1	**59.4**	32.1	36.6	43.2
		Norm ratio	25.0	31.5	50.2	21.3	29.2	31.4
		EvoPress OpenHermes	42.8	40.1	58.1	33.4	35.6	42.0
		EvoPress FineWebEdu	30.5	**43.0**	56.8	35.5	32.9	39.7

approximation is close to or at the random baseline, which means that the model is destroyed for all practical purposes. This signals the need to include the off-diagonal elements for these larger models. In all test cases, we see that CBO scores better on MMLU than the other methods, with > 20 percentage points higher scores for the Llama models. Interestingly enough, we find that CBO never gets the highest score for HellaSwag, but also that there is no single other algorithm that always gets the highest scores for this benchmark across the board. This demonstrates that there is probably no single algorithm that will always get the highest scores for all models across all benchmarks and that practitioners might have to explore different methods when applying depth pruning depending on their goals. Since we do not compensate for the loss of weights, and large chunks of the model are

removed at once, a combinatorial solution that is good for every single benchmark might not actually exist.

When looking at the total number of benchmarks and model configurations, we have a total of 5×4 possible tests, and we notice that CBO gets the highest score on 13 of them, while EvoPress get the highest scores on three (when adding the scores using the two different datasets together).

In App. B, we provide the specific block indices removed by each algorithm and in App. C we demonstrate the speedups gained from block removal.

5 Conclusion

In this work, we have introduced an algorithm based on CBO (constrained binary optimization) for selecting which combination of transformer blocks to remove from LLMs. Our approach introduces an auxiliary variable for each block (0: keep, 1: remove) and uses a second-order Taylor expansion of the loss to explicitly take into account interactions between blocks. The energies of the resulting CBO problem strongly correlate with high performance across multiple benchmarks.

First, we applied our method to Llama-3.1-8B-Instruct and demonstrated that the energy can serve as a good proxy for downstream performance and that the four lowest-lying states gave good results on the benchmarks (with the lowest-energy state significantly outperforming the others on MMLU for this model). Then, we showed that for strong compression of Llama-3.3-70B-Instruct, Qwen3-14B, and Llama-3.1-8B-Instruct solvers like tabu search can deliver correct solutions within seconds, thus alleviating the need to exactly solve the optimization problem. Lastly, we benchmarked CBO against other approaches for Llama-3.1-8B-Instruct, Qwen3-14B, Falcon3-7B-Instruct, and Llama-3.3-70B-Instruct, and found that for all but one tested case, CBO scores higher on average across the benchmark set. For Falcon3-7B-Instruct, a diagonal approximation got the average score, however, we saw that this method collapsed to random base lines for other models. On MMLU, it was also consistently better by a large margin, with around 20 more points than the next-best method for the Llama models. On the other hand, we find that other methods always outperform CBO on HellaSwag. We theorize that due to the nature of this problem, a block-removal solution that is good on each benchmark might not actually exist, and one needs to consider which downstream performance task one cares to optimize. We find CBO to be a strong general-purpose block-removal method across the evaluated benchmarks, with particularly strong performance on MMLU.

Efficiently compressing LLMs while preserving their knowledge remains an active and rapidly evolving research area, and several promising extensions of our work remain. For example, one could further study the influence of the calibration dataset or analyze the correlation between solver quality and downstream tasks. Finally, we note that our method is compatible with layer-merging approaches (Ding et al., 2025), quantization (Frantar et al., 2022; Lin et al., 2023), low-rank approximation (Yuan et al., 2023; Wang et al., 2025; Rausch et al., 2025), width pruning (Sun et al., 2023; Muralidharan et al., 2024; Guo et al., 2025; Rosenberg et al., 2024), and weight clustering (Aizpurua et al., 2026). Exploring how to effectively combine these techniques is left for future work.

AI use statement

We used generative AI tools in two limited capacities. First, for language editing: checking grammar and spelling and rephrasing for clarity in text that was written by the authors. Second, as a coding assistant, for writing and debugging parts of the implementation used to compute the Hessian, solve the CBO problem, adapt the reference implementations of the baseline methods we compare against, and generate the figures and tables. We did not use generative AI tools to generate research ideas, to design the method or experiments, to write the scientific content of the paper, or to produce or analyze experimental results. All AI-assisted text and code was reviewed and verified by the authors, and all reported numbers were produced by code the authors have checked and executed. We take full responsibility for the final content of this work, including its text, claims, and artifacts.

Ethics statement

This work advances compression algorithms for large language models and does not involve human subjects, newly released datasets, or personally identifiable information. All models and datasets used are publicly available and were used in accordance with their licenses.

Reproducibility statement

The method is described in full in Sec. 3, and the experimental setup, including the calibration data, and the number of blocks removed per model are given in Sec. 4. The exact block indices removed by every method are listed in App. B. We also attached the code for reproducing all of our results.

References

Borja Aizpurua, Sukhbinder Singh, and Román Orús. Only relative ranks matter in weight-clustered large language models. https://arxiv.org/abs/2603.17917, 2026.

Erik Altelarrea-Ferré, Júlia Barberà-Rodríguez, David Jansen, and Antonio Acín. Beyond ground states: Physics-inspired optimization of excited states of classical hamiltonians. *Phys. Rev. Appl.*, 26:014093, Jul 2026. doi: 10.1103/ltys-tnj3. URL https://link.aps.org/doi/10.1103/ltys-tnj3.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. Streamlining redundant layers to compress large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IC5RJvRoMp.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL https://arxiv.org/abs/1803.05457. Dataset license: CC BY-SA 4.0.

Xuan Ding, Rui Sun, Yunjian Zhang, Xiu Yan, Yueqi Zhou, Kaihao Huang, Suzhong Fu, Angelica I Aviles-Rivero, Chuanlong Xie, and Yao Zhu. Sliding-window merging for compacting patch-redundant layers in LLMs. *arXiv:2502.19159*, 2025. URL https://arxiv.org/abs/2502.19159.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014. URL https://arxiv.org/abs/1411.4028.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. URL https://arxiv.org/abs/2210.17323.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. License: MIT License.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ngmEcEer8a.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. SlimLLM: Accurate structured pruning for large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2xjUkU7FDb.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com. Proprietary software; Licensed under the Gurobi End-User License Agreement.

Babak Hassibi, David Stork, and Gregory Wolff. Optimal brain surgeon: Extensions and performance comparisons. In J. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1993/file/b056eb1587586b71e2da9acfe4fbd19e-Paper.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv:2009.03300*, 2020. URL https://arxiv.org/abs/2009.03300. License: MIT.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened LLaMA: A simple depth pruning for large language models. *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models (ME-FoMo)*, 2024. URL https://openreview.net/forum?id=18VGxuOdpu.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 4163–4181, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.279. URL https://aclanthology.org/2022.emnlp-main.279/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Ricardo N. Liang. libtsqubo: A header-only c/c++ library for solving qubo problems through tabu search. https://github.com/rliang/libtsqubo, 2026. license: MIT.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023. URL https://arxiv.org/abs/2306.00978.

AI @ Meta Llama Team. The Llama 3 herd of models. *arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect, 2025. URL https://openreview.net/forum?id=JMNht3SmcG.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843. Dataset license: Creative Commons Attribution-ShareAlike.

Mistral.AI. Ministral 3. *arXiv:2601.08584*, 2026. URL https://arxiv.org/abs/2601.08584.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. *arXiv:2407.14679*, 2024. URL https://arxiv.org/abs/2407.14679.

OpenAI. GPT-4 technical report. *arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Finley Alexander Quinton, Per Arne Sevle Myhr, Mostafa Barani, Pedro Crespo del Granado, and Hongyu Zhang. Quantum annealing applications, challenges and limitations for optimisation problems compared to classical solvers. *Scientific Reports*, 15(1):12733, 2025. doi: 10.1038/s41598-025-96220-2. URL https://doi.org/10.1038/s41598-025-96220-2.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Roman Rausch, David Jansen, Sukhbinder Singh, and Román Orús. Globally optimized SVD compression of LLMs via fermi-function-based rank selection and gauge fixing. *arXiv:2512.03062*, 2025. URL https://arxiv.org/abs/2512.03062.

Gili Rosenberg, J. Kyle Brubaker, Martin J. A. Schuetz, Elton Yechao Zhu, Serdar Kadıoğlu, Sima E. Borujeni, and Helmut G. Katzgraber. Scalable iterative pruning of large language and vision models using block coordinate descent. *arXiv:2411.17796*, 2024. URL https://arxiv.org/abs/2411.17796.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL https://arxiv.org/abs/1907.10641. Dataset license: CC-BY; codebase license: Apache License 2.0.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. doi: 10.1162/08997660260028683.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. Evopress: Accurate dynamic model compression via evolutionary search. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=l7QzcZpjc5.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18098–18109. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d1ff1ec86b62cd5f3903ff19c3a326b2-Paper.pdf.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. SLEB: Streamlining LLMs through redundancy verification and elimination of transformer blocks. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023. URL https://arxiv.org/abs/2306.11695.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022. URL https://arxiv.org/abs/2210.09261. License: MIT License.

Technology Innovation Institute. Falcon3-7b-instruct. https://huggingface.co/tiiuae/Falcon3-7B-Instruct, 2024a. Hugging Face model card.

Technology Innovation Institute. Welcome to the Falcon 3 family of open models! https://huggingface.co/blog/falcon3, 2024b. Hugging Face blog, 17 December 2024.

Teknium. OpenHermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xin Yu, Thiago Serra, Srikumar Ramalingam, and Shandian Zhe. The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 25668–25683. PMLR, 2022.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv:2312.05821*, 2023. URL https://arxiv.org/abs/2312.05821.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. License: MIT License.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023. URL https://arxiv.org/abs/2309.11998. Dataset license: LMSYS-Chat-1M Dataset License Agreement.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. *arXiv preprint arXiv:2406.10594*, 2024.

A Influence of the dataset

Here, we analyze the influence of different calibration datasets on the Qwen3-14B model. We report results for OpenHermes-2.5 Teknium (2023) (used in the main text) and compare them with those obtained using wikitext2 Merity et al. (2016) and LMSYS-Chat-1M Zheng et al. (2023), both for the ground state (0) and 1st excited state (1), using 2048 samples from each dataset. Overall, we observe that calibration datasets can be broadly categorized into low- and high-quality groups. While low-quality datasets should be avoided, OpenHermes-2.5 and LMSYS-Chat-1M clearly fall into the high-quality category.

B Block combinations

In Table 3, we show which blocks were removed by the different methods. For SWM, we show which layers were merged together.

Table 2: Benchmark results using different calibration datasets for Qwen3-14B.

# br	Dataset	Method	MMLU	HellaSwag	Winogrande	ARC	BBH	Avg.
			Qwen3-14B (40 blocks)					
12	OpenHermes-2.5	CBO:0	**71.8**	34.7	59.3	35.8	47.4	**49.8**
	lmsys-chat-1m	CBO:0	63.0	34.1	**61.2**	34.7	**48.5**	48.3
	lmsys-chat-1m	CBO:1	60.7	34.8	58.2	**36.7**	46.9	47.5
	WikiText	CBO:0	27.1	32.4	54.5	28.7	30.1	34.6
	WikiText	CBO:1	31.9	36.3	54.8	30.6	33.4	37.4

Table 3: Blocks removed per method across model families.

# br	Method	Blocks removed
		Falcon3-7B-Instruct (28 blocks)
8	CBO:0	[13, 15, 20, 21, 23–25, 27]
8	BI	[11–17, 24]
8	SWM	[[4,5], [14,15], [16,17], [21–26]]
8	Norm ratio	[6, 7, 9, 11–13, 15, 16]
8	EvoPress OpenHermes	[10–15, 17, 25]
8	EvoPress FineWebEdu	[10–17]
		Llama-3.1-8B-Instruct (32 blocks)
12	CBO:0	[20–31]
12	BI	[18–29]
12	SWM	[[6–8], [20–30]]
12	Norm ratio	[8–14, 19, 20, 22, 24, 25]
12	EvoPress OpenHermes	[7–12, 19, 22, 23, 25, 26, 28]
12	EvoPress FineWebEdu	[7–12, 14, 22, 24–26, 28]
		Llama-3.3-70B-Instruct (80 blocks)
40	CBO:0	[29, 40, 41, 43–79]
40	BI	[37–76]
40	Norm ratio	[19, 22, 24–29, 31–35, 38, 42, 43, 46, 47, 49–51, 53–71]
40	EvoPress OpenHermes	[2, 5, 8–10, 21–28, 30, 32, 42, 44, 46–50, 53, 54, 57–72]
40	EvoPress FineWebEdu	[2, 5, 8–10, 15, 22–30, 40, 45–58, 60–65, 67, 69, 72, 73]
		Qwen3-14B (40 blocks)
12	CBO:0	[2, 26, 30–39]
12	BI	[2, 3, 22–24, 31–37]
12	SWM	[[14–16], [21,22], [26,27], [24,25], [31–38]]
12	Norm ratio	[10–18, 20, 23, 39]
12	EvoPress OpenHermes	[2, 3, 7, 10, 11, 22–25, 32, 33, 36]
12	EvoPress FineWebEdu	[3, 7, 10, 18–24, 34, 36]

C Inference throughput and latency

We measured inference speed with the vLLM serving engine Kwon et al. (2023) (version 0.18.0) and its benchmarking client `vllm bench serve`, driven by our internal evaluation harness. For each model, the original and the compressed checkpoint were served with identical settings: bfloat16 weights, a maximum context length of 16,384 tokens, at most 16,384 batched tokens and 256 sequences per scheduler step, and 90% of GPU memory available for weights and KV cache. Llama-3.1-8B-Instruct Grattafiori et al. (2024), Falcon3-7B-Instruct Technology Innovation Institute

(2024b) and Qwen3-14B Yang et al. (2025) were served on one NVIDIA H200 GPU. Llama-3.3-70B-Instruct Grattafiori et al. (2024) was served on two H200 GPUs with a tensor-parallel degree of two. The Qwen3-14B checkpoints are stored in float32 and were cast to bfloat16 when loaded, like all other models.

Every request had 1,000 random input tokens and generated exactly 1,000 output tokens, with the end-of-sequence token ignored, greedy decoding and streamed output. We varied the number of concurrent requests c in powers of two, from 1 to 256 for the smaller models and from 1 to 128 for Llama-3.3-70B. At each level the client sent $\max(16, 4c)$ requests after 10 warm-up requests, so that the server ran at the target concurrency for several consecutive batches rather than a single one. All requests completed without errors. Each configuration was measured once, so the tables report single measurements without confidence intervals.

Throughput is the number of output tokens generated per second, summed over all concurrent requests. Time per output token (TPOT) is the median interval between consecutive output tokens of a request. Time to first token (TTFT) is the median delay until the first output token arrives. The reply time in Table 5 is derived rather than measured directly, as TTFT $+ 999\times$TPOT, which is the time needed to stream a complete 1,000-token answer.

Table 4 reports throughput at 64 concurrent requests. The speed-up of each compressed model is within 5% of the ratio L/L' between its original and retained number of layers. The speed-up is also stable across load: between 1 and 64 concurrent requests it changes by at most 0.06 for any model. Table 5 shows that the time per output token decreases by about the same factor, both for a single user and at 64 concurrent requests, and so does the time to stream a full reply. The median TTFT at 64 concurrent requests is the exception. For Llama-3.1-8B-Instruct, Qwen3-14B and Falcon3-7B-Instruct it changes by less than 10% (280 to 296 ms, 345 to 336 ms and 292 to 318 ms), while for a single user it decreases by 13 to 48% across all four models.

Table 4: Output-token throughput of the original (baseline) and compressed models at 64 concurrent requests. L/L' is the ratio of the original to the retained number of layers. Each value is a single measurement over 256 requests.

Model	Layers removed	Throughput (tok/s)		Speed-up	L/L'
		Baseline	Compressed		
Llama-3.3-70B-Instruct	40 / 80	2,002	3,946	1.97×	2.00
Llama-3.1-8B-Instruct	12 / 32	6,940	10,639	1.53×	1.60
Qwen3-14B	12 / 40	4,463	6,192	1.39×	1.43
Falcon3-7B-Instruct	8 / 28	7,528	10,361	1.38×	1.40

Table 5: Latency of the original → compressed models for a single user and at 64 concurrent requests. Reply time is the time to stream a 1,000-token answer (TTFT $+ 999\times$TPOT). TPOT is the median time per output token. Medians over 16 requests (1 user) and 256 requests (64 users), single measurement.

	1 user		64 users	
Model	Reply time (s)	TPOT (ms)	Reply time (s)	TPOT (ms)
Llama-3.3-70B-Instruct	22.2 → 11.3	22.1 → 11.3	31.8 → 15.9	31.5 → 15.6
Llama-3.1-8B-Instruct	5.2 → 3.2	5.1 → 3.2	9.3 → 5.9	9.0 → 5.6
Qwen3-14B	8.6 → 6.2	8.6 → 6.2	14.2 → 10.0	13.8 → 9.7
Falcon3-7B-Instruct	4.6 → 3.4	4.6 → 3.3	8.5 → 6.0	8.2 → 5.7