PRUNE, UPDATE AND TRIM: ROBUST STRUCTURED PRUNING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have experienced significant growth and development in recent years. However, performing inference on LLMs remains costly, especially for long-context inference or in resource-constrained devices. This motivates the development of new post-training pruning (PTP) methods. These methods reduce LLMs' requirements by removing a substantial part of the model's parameters. The discarded weights are selected depending on their impact on the model's performance. Current PTP methods prune the models by removing the less informative hidden nodes from the FFN layers, and the least important attention layers. We propose **Putri**, a PTP method that Prunes the MLP layers, Updates them and TRIms the attention heads. It introduces three changes to the State-of-the-art. First, we update the unpruned weights of the FFN to compensate for the introduced pruning error. Second, the FFN layers are pruned sequentially, taking into account the updates done to the previous layers. Third, instead of removing full attention layers, we remove individual attention-heads. We extend this method such that it can also address Grouped-Query Attention. In summary, Putri is a structured pruning method which remains simple while showing SOTA performance. Pruning experiments on multiple models with a wide variety of sparsity ranges and on different datasets, validate the generality of Putri. Notably, we demonstrate that, unlike previous methods, Putri can prune LLMs on extreme sparsity ratios. The code is available at: https://anonymous.4open.science/r/Putri-C3A0/README.md.

1 INTRODUCTION

Large Language Models (LLMs) (Radford et al., 2018; 2019; Brown et al., 2020) show consistent improvements when increasing model scale, training data and compute. This enhanced performance has allowed LLMs to tackle a variety of tasks such as agentic workflow (Durante et al., 2024; Li et al., 2024; Wu et al., 2024) and robotics (Brohan et al., 2022; Driess et al., 2023; Brohan et al., 2024). Nevertheless, these tasks require the model to either run on a large number of tokens, which drastically increases the required memory, or run on a smaller device with less memory and computing capacity than data centers. These requirements have motivated the development of multiple post-training techniques that reduce the memory and computing requirements of LLMs by compressing model size.

For example, *Knowledge distillation* (Hinton et al., 2015) aims to train a smaller model (*student*) that mimics the prediction of the already trained bigger model (*teacher*). Even though *Knowledge distillation* achieves a highly optimized smaller model, it requires the computational effort to run inference on the *teacher* model and train the *student* model at the same time. Another approach is *Quantization* (Wu et al., 2020), which reduces the memory requirements by reducing the floating-point precision of the model's parameters. However, this technique is limited by the number of precision bits to store the weights. This sets a lower bound of how much it can compress the model. Another prominent method is *Pruning* (LeCun et al., 1989), which aims to remove the less informative parameters of a model while maintaining the original model performance. This allows Pruning to reduce the model size with a comparatively small computational effort (compared to Knowledge Distillation). At the same time it is (in theory) capable of reducing the model size as much as required for a certain task (in contrast to Quantization).

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.

The first Pruning methods aimed to remove the model's weights without considering the model's structure (*Unstructured pruning*, Zhu & Gupta, 2017; Frantar & Alistarh, 2023). These methods result in models with highly sparse weight matrices, i.e., weight matrices where most of the values are zero. Although these procedures allow for pruning at high sparsity levels without significantly reducing the model's performance, they do not always result in memory or speed gains proportional to the sparsity ratio (Xu et al., 2024; Zhang et al., 2024). This inability to turn the sparsity into meaningful memory and speed-up gains is a result from current software and hardware not being able to benefit from sparse matrices. In particular, current computation libraries such as Pytorch (Paszke et al., 2019) and Keras (Chollet et al., 2015) and accelerator hardware (GPUs, TPUs, etc.) are highly optimized for dense matrix operation and only partially benefit from sparse matrices. Even with specialized frameworks (NeuralMagic, 2022), these Unstructured Pruning methods do not lead to relevant speed-up or memory gains. Notably, these marginal gains are not proportional to the model's sparsity (Frantar & Alistarh, 2023).

Structured Pruning methods (Ma et al., 2023) take a different approach. Their sparsification methods take into account the underlying model structure. This guarantees both a reduction of the memory footprint and a substantial speed-up for inference. A principled way to do this is to remove rows or columns from the weight matrices (*Width pruning*, Ashkboos et al., 2024). This reduces the number of computations required while maintaining matrices dense for faster computation. Another Structured Pruning approach is *Depth pruning*, which is done by removing full transformer layers Men et al., 2025 or any sub-block such as the attention or FFN layer (Zhong et al., 2025; Sieberling et al., 2025).

State-of-the-art pruning methods such as *2SSP* (Sandri et al., 2025) achieve efficient post-training compression by applying a simple criteria. However, these methods leave several design choices unexplored. First, they do not update the remaining weights after a pruning step. This can lead to a significant reconstruction error at high sparsity. Second, when pruning attention layers, their approach works with a coarse granularity, i.e., full entire attention modules are removed instead of removing individual attention heads. Third, pruning multiple FFN layers independently ignores the distribution shift introduced by earlier pruning steps.

We address these limitations by introducing **Putri**, a robust structured pruning method with the following contributions:

- We show the importance of updating the unpruned weights. In particular, we demonstrate how some of the state-of-the-art (SOTA) pruning methods underperform for this reason.
- Instead of discarding entire attention layers, we introduce a fine-grained attention head removal. Previous attention head pruning methods did only consider standard Multi-Head Attention (MHA). To the best of our knowledge, our approach is the first method to tackle Grouped-Query Attention (GQA), which is a standard component in many modern LLMs.
- We sequentially prune the FFN layers, allowing each pruning decision to account for the perturbations introduced by the layers that were pruned earlier.
- Our method not only achieves SOTA performance, but it also demonstrates a more robust performance for higher sparsity ratio where the benefits for inference compute requirements are larger.

2 Related Work

Pruning is a model compression technique that aims to remove the unimportant weights. This can be done in an unstructured manner where you remove some of the weights' parameters depending on an importance score. For example, using the weights' magnitude (Zhu & Gupta, 2017), or given a calibration dataset, it can measure some statistics of the weights and use it to score and prune the weights' parameters. This includes the expected squared error (Sun et al., 2024), the standard deviation of the output (Mecke et al., 2025) or minimize the error reconstruction using the Hessian to approximate the loss (Frantar & Alistarh, 2023). Even though Unstructured Pruning methods can maintain good performance even for high sparsity, they require specialized software to benefit from the sparse matrices (NeuralMagic, 2022), they don't have reliable speed-ups (Ashkboos et al., 2024) and they do not reduce the memory requirements. This motivates the idea of pruning sub-structures of the model that can reliably lead to memory and speed-up improvements.

2.1 STRUCTURED PRUNING

Structured Pruning prunes the model while taking into account the underlying structure. The different pruning methods can be grouped as follows:

Width pruning These pruning methods focus on reducing the number of columns and rows of the weights' matrices (Ashkboos et al., 2024; Sandri et al., 2025). Intuitively, it prunes some of the hidden neurons and/or channels from the model.

Depth pruning In this case, given a model with skip connections ($x_{l+1} = x_l + f_l(x_l)$) it aims to prune the model by removing the processing blocks $f_l(x_l)$ such that $x_{l+1} = x_l$. This can be applied to the full transformer layer (Men et al., 2025; Gromov et al., 2025) or to the individual sub-blocks, i.e., the FeedForward Layer and the Attention layer (Zhong et al., 2025; Sieberling et al., 2025; Sandri et al., 2025).

Head Pruning It prunes the attention layer by removing individual heads of the attention layer, which allows to remove multiple columns from the query, key and value matrices as well as multiple rows from the output matrix. To the best of our knowledge, existing methods can only be applied to standard Multi-Head Attention (MHA) (Guo et al., 2025) and not to Grouped-Query Attention (GQA). Therefore, they can not be used to prune modern LLMs with GQA such as Llama3 or Qwen3.

2.2 GROUPED-QUERY ATTENTION

When performing inference with a transformer model, the process is done in an auto-regressive manner. This process recomputes many calculations, in particular, the key and values of previous tokens in the attention layer is recomputed for each auto-regressive step. In order to avoid that, LLMs store the previous tokens' keys and values for faster inference (KV-cache). However, this speed-ups the process at the cost of additional memory requirements. To improve the memory efficiency, Multi-Query Attention (MQA, Shazeer, 2019) was proposed. MQA is a variation of MHA that uses only one key and value head that matches all the query-heads. This also reduces the KV-cache required, making the inference process even faster.

Nevertheless, MQA is not able to perform comparably to standard MHA. Grouped-Query Attention (GQA, Ainslie et al., 2023) is introduced as an option in between MHA and MQA, where each key-value head (kv-head) matches to multiple query heads. However, contrary to MQA there are multiple kv-heads instead of a single one. This allows GQA to benefit from the speedup by having less kv-heads than MHA and therefore less KV-cache, while still achieving the same performance as MHA.

3 METHODOLOGY

To prune LLMs, Putri first calculates the least important intermediate nodes of the MLP layer to prune. Then, the remaining unpruned weights of the MLP are updated. Both steps are done sequential over the MLP layers. Lastly, the attention layers are trimmed. However, instead of removing full attention layers, individual heads from MHA or kv-heads from GQA are removed.

3.1 MLP PRUNING AND UPDATE

Given a target sparsity for the MLP layers we prune all the MLP layers uniformly. This way, the same sparsity level is achieved on all the MLP layers. For each MLP layer, the pruning process aims to remove all the connections related to the less important intermediate nodes/hidden representations. In order to prune the intermediate nodes, first a score is assigned to each individual node as follows:

$$\text{score}(z_i^{(l)}) = \|z_i^{(l)}\|_2^2 \tag{1}$$

where $z_i^{(l)}$ represents the output of the intermediate node of node i in layer l. Analogously, it can be interpreted as the input to the *mlp.down_proj* layer as in Qwen and Llama notation. After removing the

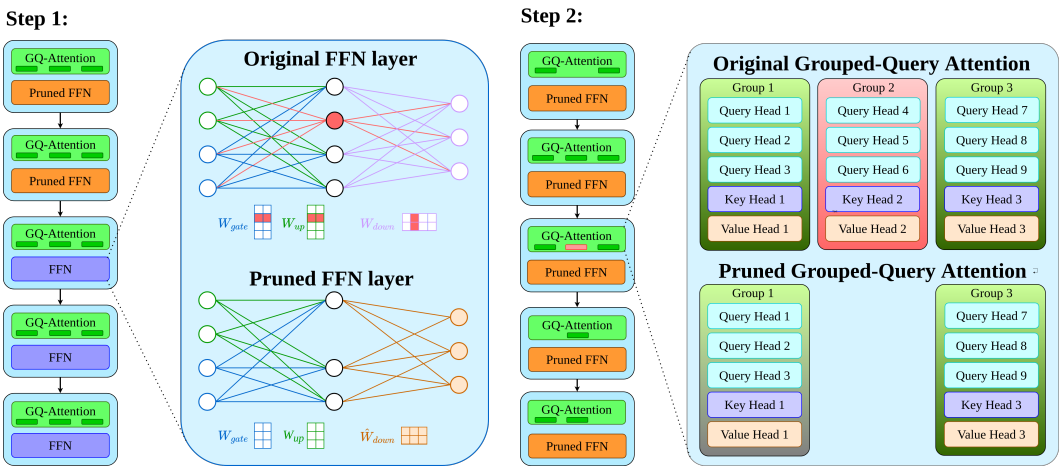


Figure 1: Diagram of the pruning process for Putri. First the FFN layers are pruned sequentially. Then the attention/kv-heads are pruned in an iterative manner.

nodes with the least score, each unpruned connection from the intermediate weights is reconstructed by minimizing the L_2-loss, i.e.:

$$arg\min_{\hat{W}_P} \|ZW - Z_P\hat{W}_P\|_2^2 = (Z_P^T Z_P)^{-1} Z_P^T ZW \tag{2}$$

where $Z \in \mathbb{R}^{N\times M_1}$ and $W \in \mathbb{R}^{M_1\times M_2}$ are the original intermediate nodes' values and weight parameters, while $Z_P \in \mathbb{R}^{N\times P}$ and $\hat{W}_P \in \mathbb{R}^{P\times M_2}$ are the nodes' values the P unpruned intermediate nodes and their corresponding updated weights. N refers to the number of instances in the dataset, M_1 the number of input features and M_2 the number of output features. Applying this update, allows the remaining weights to reduce some of the error obtained from pruning some intermediate representations.

In order to further improve performance, the MLP layers are not pruned in parallel but sequentially. This means that Z and Z_P are calculated with the previous layers pruned and updated weights. This allows each MLP layer to be pruned taking into account the output of the previously pruned layers, leading to a better pruning process (Figs. 2 and 3).

3.2 Attention Pruning

Previous works (Sieberling et al., 2025) showed that deciding which full attention layers to remove using the perplexity on 1 sequence achieves comparable performance to using multiple sequences. This reduces dramatically the time required to prune. However, we argue that a more fine-grained approach, such as pruning each head, could lead to a better performance. Notice that some of the modern LLMs pruned use Grouped-Query Attention (GQA) instead of Multi-Head Attention (MHA). We address this challenge by selecting the individual key-value heads to prune and all their associated q-heads. In the case of MHA, each key-value head has one unique query head associated, while for GQA there are multiple query heads associated to each key-value head. In particular, we select the key-value head to remove as follows:

$$arg\min_{a\in\theta_{Attn}} \text{perplexity}(f(X;\theta_{FFN},\theta_{Attn}\backslash\{a\}),Y) \tag{3}$$

where $X \in \mathbb{R}^{N\times M_1}$ are the inputs, θ_{FFN} are the weight parameters from the FFN layers, θ_{Attn} are the weight parameters from the Attention layers and a refers to a grouped head from θ_{Attn}, including multiple query heads, a key head and a value head. We notice that selecting one grouped head at a time and recalculating the perplexity each time, can improve the final performance but requires a

lot of compute. We address this limitation by removing the H less informative heads each iteration, where H is the number of kv-heads in the original attention layers.

3.3 Attention vs FFN Pruning ratio

Previous methods (Sandri et al., 2025) showed empirically that the ideal pruning ratio between the Attention and the FFN layers is obtained by pruning a number of attention layers equal to:

$$N_{Attn} = round(L \cdot s^{\frac{|\theta_{FFN}|}{\alpha|\theta_{Attn}|}}) \tag{4}$$

where L is the number of layers, s is the target sparsity, $|\theta_i|$ is the number of parameters that the i part of the model has (with $i \in \{FFN, Attn\}$) and α is the hyperparameter controlling the ratio ($\alpha = 1.5$ by default). In our case, to maintain this ratio while pruning kv-heads instead of attention layers, we modify Equation 4 slightly to calculate the number of key-value heads as follows:

$$N_{KV-Heads} = K \cdot round(L \cdot s^{\frac{|\theta_{FFN}|}{\alpha|\theta_{Attn}|}}) \tag{5}$$

where K is the number of kv-heads in each Grouped-Query Attention layer or attention heads for Multi-Head Attention layers.

4 Experiments

4.1 Set-Up

In order to do a fair comparison between all the pruning method, we use an extensive set-up with a wide variety of model families, model sizes, datasets with different tasks, pruning methods and sparsities ranges. The specific configurations used in the experiments consider the following options:

Models We evaluate the Llama-3 (Dubey et al., 2024) (including 3.0, 3.1 and 3.2 versions) and Qwen3 (Yang et al., 2025) family of models across the different sizes available. Both of them are SOTA open-sourced modern LLMs with GQA (Ainslie et al., 2023), RMSNorm (Zhang & Sennrich, 2019) and RoPE embeddings (Su et al., 2024).

Datasets For calibration during the pruning process we use the *C4* dataset (Raffel et al., 2019). The pruned models' perplexity is evaluated on the datasets *C4* (Raffel et al., 2019), *Wikitext-2* (Wiki, Merity et al., 2017) and *FineWeb-Edu* (FW, Penedo et al., 2024). Additionally, we also measure the zero-shot capabilities using LM Eval Harness (Gao et al., 2024). We evaluate on multiple benchmarks such as *Winogrande* (Sakaguchi et al., 2020), *ARC* (easy and challenge, "ARC-E" and "ARC-C" respectively) (Clark et al., 2018), *Hellaswag* (Zellers et al., 2019), *PiQA* (Bisk et al., 2020) and *MMLU* (Hendrycks et al., 2021).

Baselines We evaluate against multiple post-training Structured Pruning methods, which include width-wise pruning approaches (*2SSP*, Sandri et al., 2025) and depth-wise pruning on the transformer blocks (*ShortGPT*, Men et al., 2025; *UIDL*, Gromov et al., 2025) or the FFN and Attention sub-blocks (*Blockpruner*, Zhong et al., 2025; *Evopress*, Sieberling et al., 2025; *2SSP*, Sandri et al., 2025). Since most new open-sourced LLMs do not use MHA but GQA, we do not evaluate previous head pruning methods since they are not suitable for GQA.

Sparsity We experiment with a big range of high sparsities (25%, 37.5%, 50% and 75%).

4.2 Main experiment

In this section, we will focus on measuring the generalization capabilities of the different pruning methods. For that, we evaluate the perplexity over many datasets and multiple models in Table 1. The consistent improvements of Putri over the other methods, show the robustness of our method compared to the baselines. Additionally, notice that some of the other pruning methods are not able to

Table 1: Perplexity comparison of multiple pruning methods on Llama-3.1-8B, Qwen3-14B and Qwen3-32B models. We add some notations to specify the following cases: 'OOM' refers to an Out-Of-Memory error and '∞' is a successfully pruned model that returns infinite when the perplexity is evaluated.

Sparsity	Method	Llama-3.1-8B		Qwen3-14B		Qwen3-32B	
		Wiki	FW	Wiki	FW	Wiki	FW
0.0	-	6.406	8.375	8.641	9.43	7.594	8.32
0.25	UIDL	2976	3168	23.03	24.28	∞	∞
	ShortGPT	260	296	39.56	38.12	14.16	15.31
	BlockPruner	14.69	18	16.98	17.28	10.06	11.34
	EvoPress	13.56	15.62	14.95	15.19	OOM	OOM
	2SSP	12	14.44	11.51	12.91	10.02	11.31
	Putri (ours)	**11.81**	**13.38**	**10.62**	**12.28**	**9.125**	**10.48**
0.375	UIDL	111616	20736	30336	15376	∞	∞
	ShortGPT	8640	18304	151.4	131	28.44	29.8
	BlockPruner	27	29.25	46.78	39.47	14.22	15.58
	EvoPress	26.25	27	25.34	24.19	OOM	OOM
	2SSP	20.38	21.75	15.25	16.98	13.43	15.31
	Putri (ours)	**17.5**	**19.75**	**13.51**	**15.61**	**11.03**	**12.28**
0.5	UIDL	52736	38656	∞	∞	∞	∞
	ShortGPT	14208	10432	1318	622.5	∞	∞
	BlockPruner	163	163	221	131.5	39.25	35.03
	EvoPress	148	96	86.94	73.75	OOM	OOM
	2SSP	**34.75**	**41.25**	23.53	25	19.39	21.67
	Putri (ours)	39.25	41.75	**18.19**	**19.97**	**15.16**	**16.95**
0.75	UIDL	344064	196608	∞	∞	∞	∞
	ShortGPT	49664	49664	∞	∞	64736	∞
	BlockPruner	98816	28288	5524	4336	5884	3224
	EvoPress	17152	8640	7208	4840	OOM	OOM
	2SSP	278	260	212.6	160.5	122.1	99.62
	Putri (ours)	**163**	**185**	**124.5**	**68.75**	**44.66**	**41.78**

properly prune on higher sparsity degrees, leading to '∞' perplexity, i.e., the pruning process results in an unusable model.

To further investigate the collapse of the pruning methods, we evaluate them on a wider range of model sizes from the Qwen3 family of models. We particularly focus on the smaller models. Notice that successfully compressing small models is known to be more difficult than compressing big models Li et al. (2020). In particular, pruning small models is a harder task than pruning big models to the same sparsity ratio. The results in Table 2 show that not only Putri improvements are consistent across the model sizes, but also it is able to successfully prune to a much higher sparsity degree than previous methods.

Due to space constraints, we defer to the Appendix A.2 the following results. First, the perplexity results on the test partition of the C4 dataset. We consider Wikitext-2 and FineWeb-Edu more interesting datasets since the models are pruned with the train partition of C4. Therefore, this other datasets would better show the generalization to text not following the distribution of the calibration dataset C4. Second, the smaller models from the Llama3 family. Qwen3 has shown in general better capabilities than Llama3. Therefore, we will keep only the results on the bigger Llama3 models we were able to run. Lastly, we measure the performance of the pruning methods in extreme sparsities (over 90%) for exploration purposes.

The main goal of pruning is the reduction in memory requirements when running inference. To measure this improvement, we show the VRAM requirements of the pruned models. The results are available in the Appendix A.1.

Table 2: Perplexity comparison across different model sizes. We add some notations to specify the following cases: 'nan' refers to an invalid pruned model output or an error during the pruning process and '∞' is a successfully pruned model that returns infinite when the perplexity is evaluated.

Sparsity	Method	Qwen3-0.6B		Qwen3-1.7B		Qwen3-4B		Qwen3-8B	
		Wiki	FW	Wiki	FW	Wiki	FW	Wiki	FW
0.0	-	20.97	20.69	16.75	16.11	13.75	13.54	9.734	10.48
0.25	UIDL	23264	21344	∞	∞	1994	555.5	5072	739
	ShortGPT	189.1	146.6	104.8	103.2	48.56	43.28	38.56	33.91
	BlockPruner	41.28	38.41	30.86	28.94	20.41	21.02	22.67	18.66
	EvoPress	41.12	39.88	28.33	27.67	22.67	22.02	22.02	18.66
	2SSP	48	42.19	26.41	26.66	20.48	18.44	14.49	16.17
	Putri (ours)	**35.03**	**32.66**	**22.11**	**23.77**	**15.77**	**17.38**	**13.54**	**15.83**
0.375	UIDL	∞	∞	4544	2704	828	733.5	36320	∞
	ShortGPT	5568	2790	54944	23440	198.1	189.1	246.6	145
	BlockPruner	110.8	102	159.2	90.38	51.78	51	47.53	39.31
	EvoPress	125.4	103.2	106.1	74.31	44.31	42.03	42.59	38.25
	2SSP	112.4	94.69	50.59	46.69	35.19	28.72	21.12	24.42
	Putri (ours)	**52.19**	**51.28**	**32.78**	**32.28**	**23.86**	**26.14**	**18.25**	**20.2**
0.5	UIDL	∞	∞	∞	∞	3768	2946	∞	∞
	ShortGPT	∞	∞	58944	11072	2570	1499	1787	639.5
	BlockPruner	409.8	328	923.5	446.5	143.2	120.7	202.9	108.2
	EvoPress	356	272	7792	5976	117.9	100	211	120.7
	2SSP	371.8	255.4	166.2	109.9	63.84	56.56	41.94	40.09
	Putri (ours)	**108.2**	**97.69**	**65.62**	**55.47**	**40.97**	**36.44**	**28.44**	**27.3**
0.75	UIDL	∞	∞	∞	∞	∞	∞	∞	∞
	ShortGPT	∞	∞	∞	∞	∞	∞	∞	∞
	BlockPruner	20528	13256	∞	∞	14560	10088	13464	5884
	EvoPress	6716	4952	9544	6984	23264	14904	5112	3768
	2SSP	2866	2198	1080	768.5	1596	1063	536.5	395.8
	Putri (ours)	**1441**	**1038**	**884.5**	**647**	**309.2**	**207.6**	**164.2**	**140**

4.3 ZERO-SHOT

Perplexity measures how well a model predicts the next token on a held-out corpus, and is therefore a useful indicator of how much a pruning method distorts the model's learned distribution. However, it is only an indirect proxy for task-solving ability: it averages the prediction error uniformly over all tokens in the evaluation text, most of which are unrelated to any particular reasoning or knowledge-retrieval skill, so a moderate perplexity increase does not necessarily translate into a proportional loss of downstream performance, and vice versa. This gap is particularly relevant at the high sparsity ratios we study, where a pruned model can also collapse into degenerate behavior, e.g., repeating a narrow set of frequent tokens, which perplexity penalizes heavily even though such a model may still occasionally guess a multiple-choice benchmark answer correctly, while two methods with comparable perplexity can differ substantially in their actual ability to parse a question and select the correct answer. Since the ultimate goal of pruning an LLM is to preserve a model that can still correctly answer and solve different tasks, rather than only reproduce next-token statistics, we complement the perplexity evaluation with the zero-shot accuracy on a list of established benchmarks: Winogrande (WG), ARC-E, ARC-C, HellaSwag (HS), PiQA and MMLU. The results on Qwen3-8B are shown in Table 3.

In Table 3, it can be observed that Putri is overall the best method, maintaining the best average accuracy over most sparsity values. This advantage is consistent across most of the individual benchmarks rather than being driven by a single task, indicating that the gains Putri obtains in perplexity (Table 1) translate into an actual improvement in the model's ability to solve diverse downstream tasks, and not merely into a better fit of the language-modeling objective. Moreover,

Table 3: Zero-shot accuracy comparison with Qwen3-8B. 'Avg. Acc.' refers to the average accuracy over all the zero-shot tasks and 'nan' refers to an invalid pruned model output or an error during the pruning process

Sparsity	Method	WG	ARC-E	ARC-C	HS	PiQA	MMLU	Avg. Acc.
0	-	67.6	83.5	55.8	57.1	76.8	73.0	69.0
0.25	UIDL	51.1	62.0	30.1	37.3	68.6	25.6	45.8
	ShortGPT	52.9	57.4	27.9	34.3	65.3	26.6	44.1
	BlockPruner	60.9	**67.6**	**39.2**	43.6	70.8	43.3	54.2
	EvoPress	54.5	61.9	33.1	42.2	68.2	32.3	48.7
	2SSP	64.7	61.6	34.4	**49.1**	74.0	**52.0**	**56.0**
	Putri (ours)	**66.7**	58.7	33.1	47.0	**74.6**	51.7	55.3
0.375	UIDL	51.2	33.8	22.4	34.8	60.1	23.2	37.6
	ShortGPT	50.4	34.9	19.5	28.5	56.0	23.3	35.4
	BlockPruner	51.1	49.4	**27.0**	33.4	63.3	24.3	41.4
	EvoPress	53.0	**52.9**	25.4	34.8	64.1	26.0	42.7
	2SSP	61.2	50.6	25.3	41.5	69.3	40.0	48.0
	Putri (ours)	**62.9**	50.6	24.7	**41.8**	**69.4**	**43.0**	**48.7**
0.5	UIDL	52.2	26.9	**24.3**	26.8	57.3	25.8	35.5
	ShortGPT	49.7	30.2	21.4	26.4	56.2	23.0	34.5
	BlockPruner	50.4	41.4	**24.3**	30.4	61.1	23.8	38.6
	EvoPress	52.8	41.3	23.3	30.6	58.4	24.3	38.4
	2SSP	56.0	41.3	19.7	35.0	**63.2**	24.1	39.9
	Putri (ours)	**56.2**	**41.8**	19.8	**36.2**	62.6	**30.4**	**41.2**
0.75	UIDL	49.5	25.3	**23.4**	25.8	53.3	25.2	33.7
	ShortGPT	49.4	24.2	22.4	25.4	**54.1**	25.4	33.5
	BlockPruner	49.1	25.4	21.8	26.2	52.7	26.2	33.6
	EvoPress	48.3	25.4	20.5	25.7	53.3	**26.6**	33.3
	2SSP	49.1	27.8	21.3	27.0	53.0	23.8	33.7
	Putri (ours)	**50.5**	**28.0**	18.3	**27.4**	53.4	25.2	**33.8**

as sparsity increases, the accuracy of every method drifts towards the random-guess baseline of the corresponding multiple-choice benchmark; Putri is the method that best resists this collapse, retaining a higher average accuracy above chance level for a wider range of sparsities than the baselines.

4.4 Ablation study

Our method Putri incorporates 3 main components when compared to previous methods such as 2SSP: updating the weights of the FFN layers, pruning the FFN layers sequentially to take into account the previous layers pruning, and pruning attention heads instead of the full attention layer. To validate the usefulness of these additions, we perform the ablation experiments in Figures 2 and 3 where each individual component is removed. We denote 'Putri (no FFN upd.)' to not updating the unpruned FFN weights after removing the other weights, 'Putri (full Attn.)' to remove the full attention layer instead of the heads, and 'Putri (parallel upd.)' to not taking into account the result of the previously pruned FFN layers by using the output of the original model as an input, instead of the already pruned previous layers. Whenever the pruning method was not able to properly prune the model and resulted in a 'nan' error, it is not plotted in the graph. This happens for some methods at 95% and/or 99% sparsity.

We observe in Figures 2 and 3 that overall, removing any of these changes results in a decrease in performance. In particular, not updating the FFN weights significantly reduces the performance, especially for higher sparsities. We also observe that even though some of the other combinations can in particular scenarios compete and sometimes even beat the full method, the overall trend is that the complete method performs better than removing any of the individual parts.

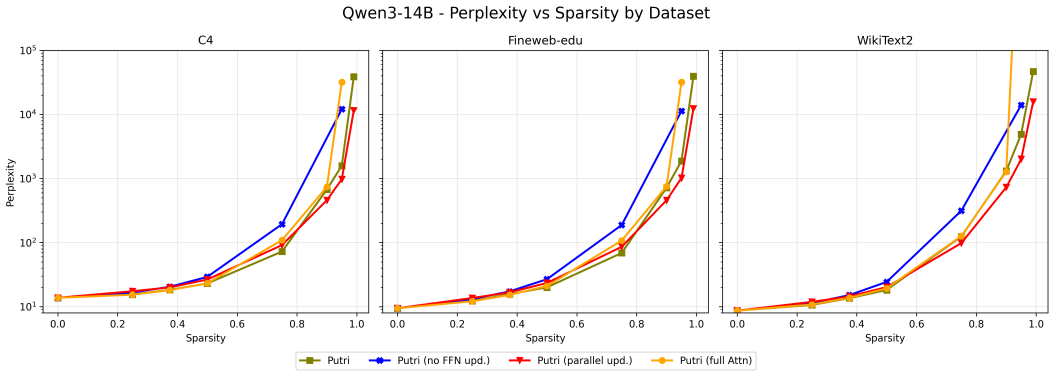


Figure 2: Ablation study on Qwen3-14B about the individual components of Putri.

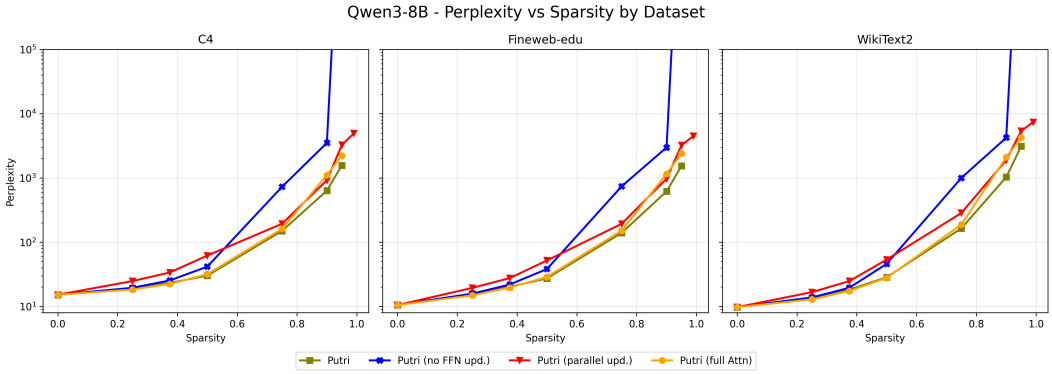


Figure 3: Ablation study on Qwen3-8B about the individual components of Putri.

5 Conclusion

In this paper we introduce Putri, a post-training structured pruning approach. In contrast to previous SOTA methods, Putri introduces three main changes. First, it updates the unpruned parameters in the last linear layer from the FFN to reduce errors induced by earlier steps. Second, the FFN pruning is done sequentially so that each layer takes into account the input from the previously pruned FFN layers, which also reduces the memory requirements since only one layer at a time needs to be in memory. Third, our method takes into account the fact that some attention layers depend highly on specific heads rather than on the combination of all of them. Therefore, Putri prunes at an attention head level and extends this approach to tackle Grouped-Query Attention.

Experiments on the Llama3 and Qwen3 model families, across a wide range of sparsities, datasets and tasks, validate our approach. Putri consistently achieves the best perplexity, and this advantage carries over to zero-shot accuracy, showing that the gains translate into actual downstream performance rather than only a better language-modeling fit. Our ablation study confirms that each of the three components contributes to this improvement, with updating the FFN weights having the largest impact. Putri is also able to prune models to substantially higher sparsity levels than existing methods before they collapse into an unusable model.

This robustness to pruning, especially at higher sparsity rates, is highly relevant for achieving meaningful memory and speed-up gains compared to the unpruned model. These improvements in speed and memory efficiency enable the deployment of these LLMs on edge devices and in time-sensitive use cases, such as on-device virtual assistants, real-time translation, and autonomous systems, where offloading inference to the cloud is impractical or too costly.

AI USE STATEMENT

In this work, we have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis or interpret results. We used generative AI tools to create or modify scientific figures or images, sourcing/searching for information and edit a research paper to improve readability. We have reviewed all AI-assisted work by making sure that all the created figures get the values from the corresponding log files, manually analyzing the information gathered and reviewing all the text changes. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv. org/abs/2307.15818*, 1:2, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Francois Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

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:1803.05457v1*, 2018.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zane Durante, Bidipta Sarkar, Ran Gong, Rohan Taori, Yusuke Noda, Paul Tang, Ehsan Adeli, Shrinidhi Kowshika Lakshmikanth, Kevin Schulman, Arnold Milstein, et al. An interactive agent foundation model. *arXiv preprint arXiv:2402.05929*, 2024.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

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. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers, 2025. URL https://arxiv.org/abs/2403.17887.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slimllm: Accurate structured pruning for large language models. 2025. URL https://arxiv.org/abs/2505.22689.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=bgzUSZ8aeg.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*, pp. 5958–5968, 2020.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Diego Coello de Portugal Mecke, Haya Alyoussef, Maximilian Stubbemann, Ilia Koloiarov, Tom Hanika, and Lars Schmidt-Thieme. Stade: Standard deviation as a pruning metric. *arXiv preprint arXiv:2503.22451*, 2025.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

NeuralMagic. Deepsparse. https://github.com/neuralmagic/deepsparse, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Fabrizio Sandri, Elia Cunegatti, and Giovanni Iacca. 2SSP: A two-stage framework for structured pruning of LLMs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Qd7LzJBg21.

Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL https://arxiv.org/abs/1911.02150.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. Evopress: Accurate dynamic model compression via evolutionary search. In *International Conference on Machine Learning*, pp. 55556–55590. PMLR, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation, 2020. URL https://arxiv.org/abs/2004.09602.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation, 2024. URL https://arxiv.org/abs/2402.16880.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5065–5080, 2025.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

A APPENDIX

A.1 INFERENCE PERFORMANCE

To validate the inference performance we measure the models memory requirement after the pruning process. In figure 4, we show the memory requirement (MB of VRAM) and perplexity of the Qwen3-8B model after being pruned with different methods. The memory requirements measures the peak VRAM requirement for a forward pass of an input with 2048 sequence length.

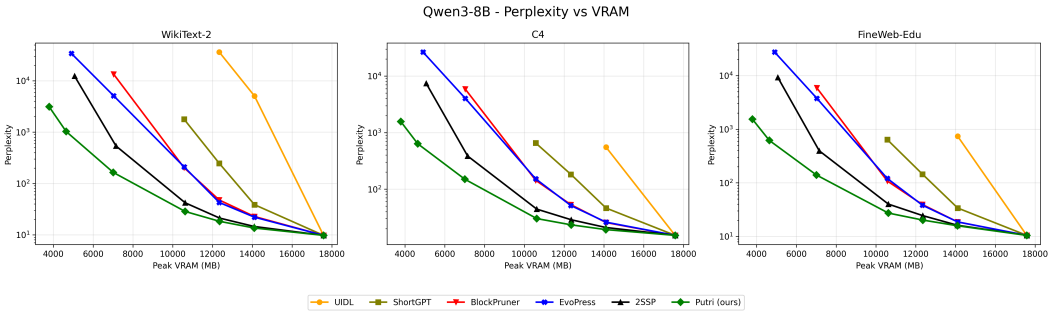


Figure 4: VRAM requirement and perplexity comparison for Qwen3-8B in different datasets and with different pruning methods.

Notice that given any VRAM requirement, Putri results in the lowest perplexity. Therefore, Putri defines a pareto frontier for the best perplexity and VRAM requirement combination, i.e, given any perplexity or VRAM requirement, Putri is able to minimize the other counterpart when compared to all the other methods. In regards to the other methods, some of them have less data points since for higher sparsities they result in ∞ perplexity or invalid models. For more information on those methods, please check Section A.2.

A.2 ADDITIONAL RESULTS

This section will provide an extensive list of all the experiment results that were not able to fit in the main paper due to space constraints. To not extend the tables over the margin constraints we abbreviate sparsity with S., EvoPress with E.V., ShortGPT with Short. and BlockPruner with B.P. Additionally, results over 10000 perplexity will be abbreviated to 10K notation.

In this section, we also extend the sparsity levels to extreme values (90%, 95% and 99%). In this regime, purely pruning the model results in really high sparsity values and therefore the pruned models can not be immediately used. However, Putri still has the lowest perplexity when compared to other methods. Therefore, if once wants to use pruning as a weight initialization method for a smaller model to train or distill, Putri would generate the best initialization. Nevertheless, this still would require additional research in the future to explore how model pruning could be effectively combined with model distillation and/or training smaller models.

Spars.	Method	Qwen3-0.6B			Qwen3-1.7B			Qwen3-4B		
		Wiki	FW	C4	Wiki	FW	C4	Wiki	FW	C4
0	-	20.97	20.69	29.45	16.75	16.11	22.28	13.75	13.54	20.05
0.25	UIDL	23K	21K	33K	∞	∞	∞	1994	555.5	771.5
	Short	189.1	146.6	187.6	104.8	103.2	134.1	48.56	43.28	62.34
	B.P.	41.28	38.41	52	30.86	28.94	39.03	20.41	21.02	29.56
	E.P.	41.12	39.88	53.75	28.33	27.67	37.91	22.67	22.02	30.5
	2SSP	48	42.19	56.56	26.41	26.66	34.38	20.48	18.44	24.75
	Putri	**35.03**	**32.66**	**42.59**	**22.11**	**23.77**	**28.83**	**15.77**	**17.38**	**22.36**
0.375	UIDL	∞	∞	∞	4544	2704	4688	828	733.5	831
	Short	5568	2790	3710	54944	23440	47744	198.1	189.1	236.2
	B.P.	110.8	102	131	159.2	90.38	125.9	51.78	51	70.94
	E.P.	125.4	103.2	136.2	106.1	74.31	101.6	44.31	42.03	58.34
	2SSP	112.4	94.69	118.3	50.59	46.69	54.19	35.19	28.72	35.75
	Putri	**52.19**	**51.28**	**62.59**	**32.78**	**32.28**	**36.22**	**23.86**	**26.14**	**30.62**
0.5	UIDL	∞	∞	∞	∞	∞	∞	3768	2946	3950
	Short	∞	∞	∞	58944	11072	12848	2570	1499	1780
	B.P.	409.8	328	398.8	923.5	446.5	566.5	143.2	120.7	163.6
	E.P.	356	272	358.8	7792	5976	6664	117.9	100	136.2
	2SSP	371.8	255.4	311.8	166.2	109.9	112.4	63.84	56.56	62.59
	Putri	**108.2**	**97.69**	**112**	**65.62**	**55.47**	**58.34**	**40.97**	**36.44**	**42.19**
0.75	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	21K	13K	14K	∞	∞	∞	15K	10K	15K
	E.P.	6716	4952	5792	9544	6984	7852	23K	15K	23K
	2SSP	2866	2198	2294	1080	768.5	790	1596	1063	1042
	Putri	**1441**	**1038**	**1114**	**884.5**	**647**	**649.5**	**309.2**	**207.6**	**224.5**
0.9	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	E.P.	∞	65K	64K	52K	24K	30K	58K	55K	61K
	2SSP	nan	nan	nan	30816	58K	42K	nan	nan	nan
	Putri	**4440**	**2414**	**2404**	**2610**	**1685**	**1705**	**916.5**	**705.5**	**711**
0.95	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	E.P.	∞	∞	∞	∞	∞	∞	∞	∞	∞
	2SSP	nan	nan	nan	nan	nan	nan	nan	nan	nan
	Putri	∞	∞	∞	**9400**	**6068**	**6116**	**3596**	**2258**	**2156**
0.99	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	E.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	2SSP	nan	nan	nan	nan	nan	nan	nan	nan	nan
	Putri	**64K**	**29K**	**27K**	∞	∞	∞	∞	∞	∞

Spars.	Meth.	Qwen3-8B			Qwen3-14B			Qwen3-32B		
		Wiki	FW	C4	Wiki	FW	C4	Wiki	FW	C4
0	-	9.734	10.48	15.19	8.641	9.43	13.73	7.594	8.32	12.26
0.25	UIDL	5072	739	551.5	23.03	24.28	34.5	∞	∞	∞
	Short.	38.56	33.91	46.16	39.56	38.12	52.19	14.16	15.31	23.2
	B.P.	22.67	18.66	25.59	16.98	17.28	24.23	10.06	11.34	16.14
	E.P.	22.02	18.66	26	14.95	15.19	21.3	OOM	OOM	OOM
	2SSP	14.49	16.17	20.92	11.51	12.91	16.88	10.02	11.31	15.34
	Putri	**13.54**	**15.83**	**19.2**	**10.62**	**12.28**	**15.46**	**9.13**	**10.48**	**13.91**
0.375	UIDL	36320	∞	∞	30336	15376	16368	∞	∞	∞
	Short.	246.6	145	182.5	151.4	131	161.8	28.44	29.8	42.09
	B.P.	47.53	39.31	53.03	46.78	39.47	54.81	14.22	15.58	22.06
	E.P.	42.59	38.25	51.19	25.34	24.19	34.03	OOM	OOM	OOM
	2SSP	21.12	24.42	28.66	15.25	16.98	21.05	13.43	15.31	19.47
	Putri	**18.25**	**20.2**	**23.3**	**13.51**	**15.61**	**18.28**	**11.03**	**12.28**	**15.46**
0.5	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short.	1787	639.5	655	1318	622.5	697	∞	∞	∞
	B.P.	202.9	108.2	143.2	221	131.5	181.9	39.25	35.03	48
	E.P.	211	120.7	151.9	86.94	73.75	101.6	OOM	OOM	OOM
	2SSP	41.94	40.09	44.22	23.53	25	29.75	19.39	21.67	26.56
	Putri	**28.44**	**27.3**	**30.09**	**18.19**	**19.97**	**22.98**	**15.16**	**16.95**	**20.2**
0.75	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞	65K	∞	∞
	B.P.	13K	5884	5884	5524	4336	4544	5884	3224	3568
	E.P.	5112	3768	4042	7208	4840	5316	OOM	OOM	OOM
	2SSP	536.5	395.8	385	212.6	160.5	151.9	122.1	99.62	94.31
	Putri	**164.2**	**140**	**149.6**	**124.5**	**68.75**	**72.31**	**44.66**	**41.78**	**44.56**
0.9	UIDL	∞	∞	∞	∞	∞	∞	nan	nan	nan
	Short.	∞	∞	∞	∞	∞	∞	nan	nan	nan
	B.P.	nan	nan	nan	nan	nan	nan	27K	28K	28K
	E.P.	34K	27K	27K	9928	7208	7976	OOM	OOM	OOM
	2SSP	12K	9256	7436	5316	2662	2432	2866	1476	1282
	Putri	**1030**	**617.5**	**634.5**	**1307**	**716.5**	**673**	**370.2**	**215.1**	**214.2**
0.95	UIDL	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	E.P.	∞	∞	∞	11K	8360	9328	OOM	OOM	OOM
	2SSP	∞	∞	∞	nan	nan	nan	30K	23K	25K
	Putri	**3124**	**1540**	**1571**	**4916**	**1880**	**1583**	**2714**	**831**	**871**
0.99	UIDL	∞	∞	∞	nan	nan	nan	nan	nan	nan
	Short.	∞	∞	∞	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	nan	nan	nan	nan	nan	nan
	E.P.	nan	nan	nan	nan	nan	nan	OOM	OOM	OOM
	2SSP	nan	nan	nan	nan	nan	nan	nan	nan	nan
	Putri	nan	nan	nan	**47K**	**40K**	**39K**	∞	∞	∞

Sparsity	Method	Meta-Llama-3-8B			Llama-3.1-8B		
		Wiki	FW	C4	Wiki	FW	C4
0	-	6.312	8.5	8.125	6.406	8.375	8.25
0.25	UIDL	148	96	153	2976	3168	2976
	Short.	223	174	191	260	296	278
	B.P.	14.44	17.5	20.38	14.69	18	21
	E.P.	12.94	14.44	18.88	13.56	15.62	20.12
	2SSP	**11.81**	**14**	**16.62**	12	14.44	16.62
	Putri	12.38	19.75	21.75	**11.81**	**13.38**	**14.94**
0.375	UIDL	392	286	368	111616	20736	15168
	Short.	1408	856	968	8640	18304	14208
	B.P.	32	32	43.25	27	29.25	37.5
	E.P.	31.12	29.25	39.25	26.25	27	34.25
	2SSP	21.38	22	24.25	20.38	21.75	23.5
	Putri	**16.88**	**19.5**	**20.75**	**17.5**	**19.75**	**21.75**
0.5	UIDL	1000	688	732	52736	38656	34048
	Short.	56320	49664	46592	14208	10432	8640
	B.P.	131	99	108.5	163	163	185
	E.P.	96	99	108.5	148	96	105
	2SSP	34.75	**40.5**	46.75	**34.75**	**41.25**	47.5
	Putri	**32**	45.25	**45.25**	39.25	41.75	**46**
0.75	UIDL	17152	15168	17152	344064	196608	391168
	Short.	1204224	1359872	2113536	49664	49664	56320
	B.P.	7136	8640	6720	98816	28288	26624
	E.P.	20736	8640	7616	17152	8640	8640
	2SSP	346	268	268	278	260	252
	Putri	**216**	**237**	**223**	**163**	**185**	**203**
0.9	UIDL	880640	8912896	8912896	1859584	21364736	10092544
	Short.	323584	152576	162816	1859584	21364736	10092544
	B.P.	119296	184320	126976	501760	684032	643072
	E.P.	46592	41216	38656	28288	14208	20736
	2SSP	1592	1032	940	1544	908	908
	Putri	**568**	**458**	**472**	**644**	**416**	**472**
0.95	UIDL	569344	501760	442368	501760	501760	415744
	Short.	251904	143360	184320	532480	391168	471040
	B.P.	827392	778240	827392	1204224	995328	995328
	E.P.	92672	68096	72192	135168	98816	98816
	2SSP	4928	2976	2976	8096	2976	2976
	Putri	**2048**	**828**	**880**	**1696**	**1096**	**968**
0.99	UIDL	∞	∞	∞	–	–	–
	Short.	569344	569344	569344	532480	501760	442368
	B.P.	569344	569344	569344	532480	501760	442368
	E.P.	nan	nan	nan	nan	nan	nan
	2SSP	**22016**	14208	17152	34048	17152	17152
	Putri	24960	**9152**	**8096**	**11776**	**5216**	**3824**

Sparsity	Method	Llama-3.2-1B			Llama-3.2-3B		
		Wiki	FW	C4	Wiki	FW	C4
0	-	10.25	12	12.94	8.375	10.12	10.75
0.25	UIDL	7616	2976	4608	1496	908	1280
	Short.	245	237	245	158	168	203
	B.P.	42.5	38.75	54.5	23.5	22.75	29.25
	E.P.	51.25	48.25	54.5	22.75	21.38	26.62
	2SSP	18.62	20.12	25.38	**16.12**	**19.5**	**20.75**
	Putri	**16.38**	**19.5**	**21.75**	16.88	23.12	27.5
0.375	UIDL	52736	34048	34048	7616	5568	5568
	Short.	260	191	209	458	416	486
	B.P.	203	174	191	53.75	51.25	54.5
	E.P.	163	119.5	148	50.5	46.75	54.5
	2SSP	31.62	35.75	50.5	**26.62**	**29.25**	**32.5**
	Putri	**24.62**	**26.62**	**30.12**	31.62	45.25	54.5
0.5	UIDL	222208	162816	173056	46592	34048	36352
	Short.	3168	1592	1592	11072	9792	11776
	B.P.	1592	968	1032	139	127	153
	E.P.	536	392	404	163	135	153
	2SSP	56.25	79.5	77	49.75	49	54.5
	Putri	**35.75**	**38.75**	**53.75**	**40**	**40.5**	**48.25**
0.75	UIDL	778240	684032	880640	152576	126976	126976
	Short.	442368	1359872	6094848	38656	14208	19456
	B.P.	14208	8096	7136	4928	2976	2976
	E.P.	28288	18304	28288	7616	5216	5568
	2SSP	472	368	404	520	378	378
	Putri	**158**	**209**	**216**	**191**	**209**	**216**
0.9	UIDL	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞
	B.P.	∞	∞	∞	152576	173056	173056
	E.P.	49664	56320	59904	12544	8640	8096
	2SSP	2976	1408	1456	2464	1368	1320
	Putri	**1096**	**856**	**752**	**752**	**604**	**664**
0.95	UIDL	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	∞	∞	∞
	E.P.	nan	nan	nan	501760	143360	126976
	2SSP	17152	19456	14208	5568	2896	2976
	Putri	**2624**	**1320**	**1240**	**1984**	**1240**	**1000**
0.99	UIDL	∞	∞	∞	∞	∞	∞
	Short.	∞	∞	∞	∞	∞	∞
	B.P.	nan	nan	nan	∞	∞	∞
	E.P.	nan	nan	nan	nan	nan	nan
	2SSP	nan	nan	nan	13376	11776	11776
	Putri	nan	nan	nan	**10432**	**5920**	**5920**