Efficient LLM Pruning with Token-Dependency Awareness and Hardware-Adapted Inference

Anonymous ACL submission

Abstract

Structured pruning removes entire components, like attention heads or hidden dimensions to yield faster dense large language models. However, previous methods are time-consuming and inference speedup is bottlenecked by inefficient GPU parallel processing due to mismatch in pruned weight block dimensions with tensor cores. Moreover, pruning of heads in grouped query attentions is not widely attempted due to challenges with their interdependencies. To address these limitations, we propose (1) a structured pruning method for LLMs with grouped-query attentions(GQA) that learn appropriate key,value and shared query heads to retain according to its importance for accurate prediction. (2) a post-pruning weight update to better retain performance of pruned LLMs. (3) a post-pruning dimension adaptation step to enhance GPU utilization of pruned models and significantly speed up inference. Our method speeds up inference by up to 60% over previous approaches. Evaluated on several language benchmarks using variants of LLaMA models and Mistral, our method shows a reduction in pruning time by upto 90% with higher inference speed and performance over a range of sparsity ratios. Additionally, our findings suggest that pruning can reduce prediction confusion in models.

1 Introduction

Deploying Large Language Models (LLMs) on resource-constrained devices is challenging due to their high computational and memory demands (Le Scao et al., 2023). Pruning is an effective solution to reduce redundant model parameters and accelerate inference without sacrificing task performance. Structured pruning (An et al., 2024) involves removing layers, heads, intermediate dimensions which can lead to dense compressed models with faster inference. While effective in maintaining model accuracy, gradient-based methods (Ma et al., 2023) require substantial memory resources and forward-pass only method Dery et al. (2024) requires about 40 GPU hours for continuous evaluation of sub-models. This makes them impractical for scenarios with limited memory, power or time. On the other hand, unstructured pruning methods, which remove individual weights, offer faster pruning but necessitate specialized hardware to accelerate the pruned models (Frantar and Alistarh, 2023). Quantization techniques require specialized GPUs and libraries for acceleration (Dettmers et al., 2022; Zhang et al., 2024c).

Structured pruning methods often fail to prune token embedding representations due to the complex dependencies that span across layers of the model, thereby missing out on added acceleration. Pruning of attention heads in grouped query attentions (GQA) (Ainslie et al., 2023) introduces additional complexity since multiple query heads share a single key and value head. This interdependence implies that pruning a query head can disrupt the functionality of the entire group. Very few previous work undertake the structured pruning of GQA-based models like Mistral and LLaMA-3. The recent Bonsai (Dery et al., 2024) attempted pruning Mistral but takes over 40 hours to search for an optimal model limiting its use.

Prior work (An et al., 2024) shows 1.3x speedup on NVIDIA A100 for 50% pruning, not scaling linearly. A notable reason is that pruned weight matrices often cannot fully exploit the parallelism in GPU tensor cores (NVIDIA, 2024a) which often perform operations in certain fixed block sizes

speedups (Chen et al., 2024; Sheng et al., 2023; Liu et al., 2023c) involving complex algorithms.

To address these challenges, we propose an efficient structured pruning method for LLMs specifically for grouped query-based models - **T**oken dependency-aware **V**ariational **A**dapted pruning.

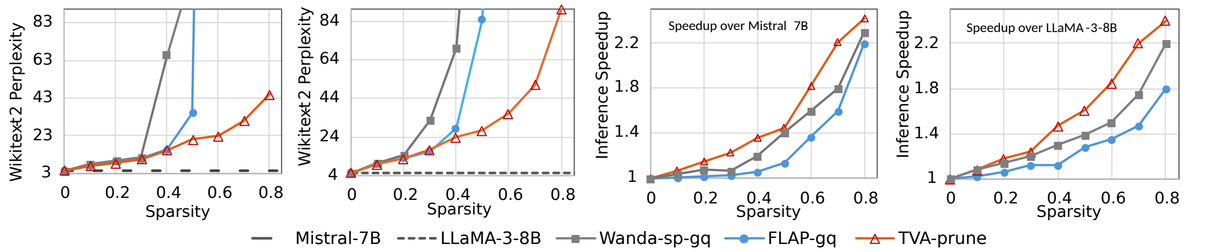

Figure 1: Comparison of sparsity, perplexity and inference speedup of GQA-based LLaMA-3-8B and Mistral-7B models pruned to different sparsity ratios with C4 train set and evaluated on Wikitext-2 validation set. Speedup is measured on NVIDIA A100(GB) for evaluation on the validation set.

We extend the formulation of the Variational Information Bottleneck (VIB) principle to include token dependency-awareness in pruning grouped-query-based models. Our method effectively removes redundant grouped-heads, intermediate and global token representation while preserving information flow on a single GPU, adhering to a user-defined sparsity criterion. Additionally, our post-pruning weight update and dimension adaptation ensures parallelism in the inference GPU and thus achieves higher inference speedup. Pre-trained LLMs including variants of LLaMA-7B (Touvron et al., 2023a,b), and Mistral-7B (Jiang et al., 2023a) are pruned, demonstrating superior performance compared to prior methods. Our major contributions are as follows:

- We propose an efficient structured approach to prune LLMs with grouped query-based attention (GQA) modules as in Mistral and LLaMA-3.
- Our framework includes an immediate post-pruning weight update that enhances pruned model performance, surpassing previous structured pruning methods, even on non-GQA-based models.
- We incorporate a post-pruning dimension adjustment that leverages GPU parallelism for faster inference not explored in previous work, with negligible changes in performance and model size.
- Evaluations on variants of LLaMA and Mistral models across language modeling and reasoning tasks demonstrate that our method outperforms previous state-of-the-art techniques.

2 Preliminaries

VIB-based Structured Pruning. Given a transformer model with pre-trained weights W, the objective is to remove rows and columns by eliminating redundant heads, intermediate layer and token embedding hidden dimensions to obtain compressed weights $\hat{W}$. We formulate this as a problem of searching for sparse masks m_{MLP}, m_{MHA} and m_{token} for MLP , Multi-Head Attention layers and token representation dimensions across all layers. Each of these masks have learnable parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$. VTrans (Dutta et al., 2024) estimates the importance of each token representation in each layer of LLMs using the Variational Information Bottleneck principle (Slonim and Tishby, 1999; Dai et al., 2018). A random set of vectors $z_i = \boldsymbol{\mu} + \boldsymbol{\eta} \odot \boldsymbol{\sigma}$ where $\boldsymbol{\eta}$ is sampled from $\mathcal{N}(0, I)$, is multiplied to the previous layer output and the mask parameters are trained using backpropagation with the following objective function as defined by (Dutta et al., 2024; Dai et al., 2018),

$$\arg\min_{\mu,\sigma} \mathcal{L} = \sum_{i}^{L} \beta_i \sum_{j=1}^{r_i} \log\left(1 + \left(\frac{\mu^{i,j}}{\sigma^{i,j}}\right)^2\right) - \mathbb{E}_{\boldsymbol{\eta}}\left[\log q\left(y_n \mid f(x_n, \eta)\right)\right] \quad (1)$$

Given a dataset of samples $\boldsymbol{x}, \boldsymbol{y}$, during backpropagation, the gradient is the unbiased estimate of the expectation. When for layer i and structure j, $\log \frac{\mu_{i,j}^2 + \epsilon}{\sigma_{i,j}^2} < 0$, the mask $m^{i,j}$ is 0, that is the corresponding weight parameters of structure j can be pruned.

Pruning grouped-query attention challenges. Grouped query Attention (GQA) (Ainslie et al., 2023) was introduced to reduce the amount of cache and speed up inference in large language models. It involves multiple query heads sharing a single key and value head. But structured pruning of heads with mask m_{MHA} in Multi-Head Attention modules (MHA) as in VTrans (Dutta et al., 2024) assumes that all query heads have a single key and value head. This does not hold

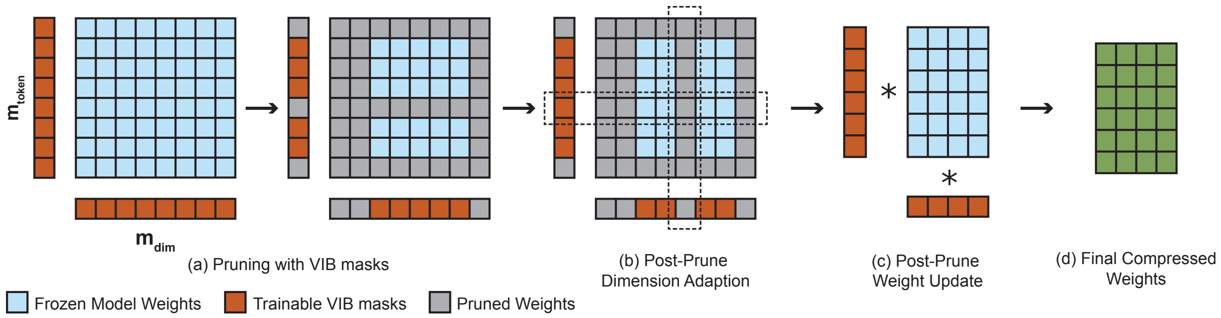

Figure 2: (a) Structured pruning of weight matrices considering global token masks and module-specific dimension mask. (b) Post-Prune Dimension Adaptation ensuring effective utilization of parallelism in GPUs during inference. Here, fifth row is unpruned and fifth column is pruned to ensure alignment with block sizes in GPUs. (c) Post-Prune Weight Update leveraging importance scores learned by VIB masks. (d) Final Model Weights

in grouped query-based attention modules, where pruning a query head can disrupt the group functionality and lead to inconsistencies in the attention module structure.

Inference speedup challenges. When evaluated on the test set of Wikitext-2 dataset, models pruned by 50% with prior methods (Dery et al., 2024; An et al., 2024) show about 1.3× speedup over the unpruned model on NVIDIA A100 (40GB). This speedup is relatively modest given the 50% reduction in parameters. Our investigation revealed that a key reason for this limited speedup is that pruned weight matrices often fail to fully leverage the parallelism of GPU tensor cores (NVIDIA, 2024a), which typically operate in fixed block sizes like 128x256. Additionally, some approaches targeting inference speedups (Sheng et al., 2023; Liu et al., 2023c) involve complex algorithms, potentially increasing computations and adding overhead before deploying pruned models.

3 Method

In this section, we describe the various aspects of our methodology to prune pre-trained LLMs in a structured manner: (1) Pruning attention heads in grouped-query attention (GQA)-based models (2) Post-pruning weight update (3) Dimension Adaptation of the weight matrices.

3.1 Pruning Attention Heads in Grouped Query

Multi-head attention allows for independent pruning of query, key, and value heads by masking individual heads. However, pruning within Grouped Query Attention (GQA) groups is more complex due to the need to maintain group functionality despite pruning.

For GQA we define separate masks for key-value heads m_v^i and query heads $m_q^i \in \mathbb{R}^d$ for i^{th} layer with d heads. Only key-value heads m_v^i are assigned trainable parameters $\boldsymbol{\mu_v^i}$, $\boldsymbol{\sigma_v^i}$. We define a random set of vectors for the key-value heads $\boldsymbol{z_v^i} \in \mathbb{R}^{dim' \times d}$ such that $dim' = batches \times seqlence\_length \times head\_dim \times group\_size$ and $\boldsymbol{z_v^i} = \boldsymbol{\mu_v^i} + \boldsymbol{\eta} \odot \boldsymbol{\sigma_v^i}$. $\boldsymbol{\eta} \in \mathbb{R}^{dim' \times d}$ is sampled from $\mathcal{N}(0, I)$. These random set of vectors are multiplied by the output of the accumulated heads. Sampling across groups ensures randomness within groups of query heads to weigh their importance for pruning. The learnable parameters are optimized as per Equation 1.

Following the method described in (Dai et al., 2018), we observe that when $\alpha_v^{i,j} = \left(\frac{\mu_v^{i,j}+\epsilon}{\sigma_v^{i,j}}\right)^2$ with small ϵ approaches zero, it indicates that the key-value head j contains no significant information beyond what is captured in previous layers and pruning it results in minimal performance degradation. Since g queries share the same key-value head, token representations may share local dependencies within the groups. To retain these dependencies, we concatenate the key-head masks to form the query head mask m_q^i. Thus, the mask to prune key value heads and query heads with g groups can then be defined as,

$$m_v^{i,j} = \begin{cases} 1 & \text{if } \log \alpha_v^{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in d$$

$$m_q^i = \underbrace{[m_v^i, m_v^i, \ldots, m_v^i]}_{g \text{ times}} \tag{2}$$

For each key-value head pruned, our method prunes all connected query heads in the group, ensuring the pruning process does not disrupt group functionality.

Ensuring user-defined sparsity. Given a target sparsity t, during pruning, using binary masks m, we calculate the expected model sparsity s_e as the ratio of pruned parameters to the initial count. We use a Lagrangian term similar to Xia et al. (2022) by enforcing an equality constraint $s_e = t$ and introducing a violation penalty as,$\mathcal{L}_s = \lambda_1 \cdot (s_e - t) + \lambda_2 \cdot (s_e - t)^2$ where λ_1, λ_2 are jointly updated during the pruning.

3.2 Post-Pruning Dimension Adaptation

The dimensions in pre-trained unpruned models are optimized for efficient GPU execution using specific block sizes (NVIDIA, 2024b) like 128x256. However, pruned model dimensions may not align with these block sizes. To address this, we propose a post-pruning technique that initially identifies the indices where the masks m_{token} and m_{mlp} are non-zero and estimates the corresponding weight dimensions as $n = |I|; \quad I = \{i \mid m_{\text{token}}[i] \neq 0\}$. It adjusts the mask lengths such that each of the pruned weight dimension n' would be the nearest multiple of the specified tensor dimension T (say 128) as,

$$n' = \left(\left\lfloor \frac{n + T/2}{T} \right\rfloor \right) \times T \quad (3)$$

To account for the new dimension, it sorts $\log \alpha$ in descending order, gets the new threshold and recomputes the mask with d dimension based on the new threshold as,

$$\tau = (\log \alpha)_{\text{sorted}}[n']$$

$$\widehat{m}_{\text{token},j} = \begin{cases} 1 & \text{if } \log \alpha_j > \tau \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in d \quad (4)$$

Overall, there is a negligible change in the model size. In our experiments, we observe a significant inference speedup due to this step.

3.3 Post-Pruning Weight Update

Instead of applying binary masks as defined in Equation 2 that merely retain or discard weights, we leverage the continuous importance scores of representations (mean values $\boldsymbol{\mu}$) learned by the VIB. We modify all the binary masks to be weighted masks as,

$$\widehat{m}^{i,j} = \begin{cases} \mu^{i,j} & \text{if } \log \alpha_j > \tau \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in d \quad (5)$$

Model	Wikitext-2 PPL ↓	Inference Speedup	Tokens/s
Mistral-7B	4.77	1×	24.78
Wanda-sp-gq	116	1.1×	27.26
FLAP-gq	34.97	1.28×	31.73
Bonsai	47.50	1.66×‡	41.13
TVA-Prune	**18.37**	**1.67**×	**41.39**
Bonsai †	10.08	1.66×‡	41.13
TVA-Prune †	10.12	**1.67**×	**41.39**
LLaMA-3-8B	5.57	1 ×	25.13
Wanda-sp-gq	106	1.1×	27.64
FLAP-gq	34.90	1.2×	30.16
TVA-Prune	**27.50**	**1.61**×	**40.94**

Table 1: Performance comparison of Mistral-7B and LLaMA-3-8B models pruned by 50%. Our method outperforms others without any finetuning. †indicates finetuned with LoRA. ‡Result on Bonsai is taken from (Dery et al., 2024) where inference was performed on a different GPU.

Since each weight matrix W of a module has two dimensions, as for the MLP layer it is the intermediate dimension and the global token representation dimension, we update the unpruned weights using both the global token mask m_{token} and the intermediate mask m_{mlp}. The updated weights for layer i for the mlp module may be represented as,

$$W^i_{upd} = (\widehat{m}^i_{token} \otimes \widehat{m}^i_{mlp}) * W^i \quad (6)$$

By multiplying the pre-trained remaining weights with the mask weights, the model achieves a nuanced adjustment that emphasizes more relevant features. The compressed updated weights $\widehat{W}^i_{upd}$ can be obtained by removing the zeroed out rows and columns of W^i_{upd}.

Finetuning with LoRA. Although our method achieves better performance than previous approaches even without finetuning, post-pruning finetuning (Dery et al., 2024; Ma et al., 2023) using Low Rank Adapters (LoRA) (Hu et al., 2021) on a downstream task further improves performance. Additionally, we distil knowledge from the teacher logits H_t to the pruned student logits H_s (Xia et al., 2022) by minimizing the following: $\mathcal{L}_{dis} = \text{MSE}(\mathbf{H}_s, \mathbf{H}_t)$

4 Experiments

Datasets. We prune models using the training set of C4 (Raffel et al., 2020) and Wikitext-2 (Merity et al., 2016). We test our pruned models on the validation set of Wikitext-2 and on six zero-shot tasks designed to test for common sense reasoning using

Model	LLaMA-7B			LLaMA-2-7B		
	PPL ↓	Speedup	Tokens/s	PPL ↓	Speedup	Tokens/s
Unpruned	5.68	1×	26.21	5.11	1×	25.46
Wanda-sp	366.43	1.24×	32.50	132.0	1.29×	32.84
LLM-pruner	112.44	1.23×	32.24	95.26	1.29×	32.84
FLAP ‡	35.10	1.26×	33.02	25.40	1.32×	33.61
Bonsai	28.65	1.26×	33.02	22.32	1.28×	32.59
VTrans	25.87	1.32×	34.60	21.54	1.34×	34.12
TVA-Prune	**18.62**	**1.75×**	**45.87**	**18.44**	**1.82×**	**45.83**
TVA-Prune w/o DimAdapt	18.56	1.23×	32.23	18.49	1.25×	31.83
TVA-Prune w/o WUpdate	25.13	1.75×	45.87	21.32	1.82×	45.83
Wanda-sp †	67.24	1.24×	32.50	46.54	1.29×	32.84
LLM-pruner†	38.12	1.23×	32.24	29.56	1.29×	32.84
Bonsai †	11.02	1.26×	33.02	9.87	1.28×	32.59
VTrans†	10.79	1.32×	34.60	9.92	1.34×	34.12
TVA-Prune †	**10.58**	**1.75×**	**45.87**	**9.65**	**1.82×**	**45.83**

Table 2: Performance comparison of pruning methods in a task-agnostic manner with C4 train set and zero-shot evaluation on Wikitext-2. Our method outperforms structured pruning (wanda-sp, Bonsai, LLM-pruner and FLAP) †indicates finetuned with LoRA. ‡adds extra bias parameters. w/o DimAdapt speed reduction is due to inefficient parallelism in GPU. Post-prune dimension adaption leads to negligible change in model size. Inference speedup is measured on the Wikitext-2 validation set while Tokens/s throughput is on one batch of data.

the EleutherAI LM Harness (Gao et al., 2023).

Baseline Models. We prune the LLaMA-7B, LLaMA-2-7B (Touvron et al., 2023a) and GQA-based models LLaMA-3-8B and Mistral-7B (Jiang et al., 2023a), to evaluate our method and compare against other structured pruning methods. We modify the pruning process in Wanda (Sun et al., 2023) to be structured (Wanda-sp) and account for grouped-query attention (Wanda-sp-gq). Similarly, we modify FLAP (An et al., 2024) to prune grouped-query and name it FLAP-gq. Additionally, we compare the pruned mistral model with Bonsai (Dery et al., 2024). Comparison of pruning of LLaMA 1 and 2 variants also includes other baseline methods: LLM-Pruner (Ma et al., 2023), LoRAPrune (Zhang et al., 2023b)and VTrans (Dutta et al., 2024).

Experimental Settings. To prune LLaMA-3, we use a sequence length of 400 to fit in a single GPU. Similarly, we reduce the maximum position embeddings of the Mistral model to 8192. For the task-specific experiments, we use the training set of Wikitext-2 dataset to prune models. We report the average of five runs with random seeds. The few hyperparameters used are listed in the Appendix. All experiments are conducted on a single NVIDIA A100 (40GB) GPU.

4.1 Language Modelling Tasks

Performance comparison on GQA models Table 1 shows TVA-Prune is highly effective for pruning Mistral-7B model with grouped query attention (GQA) with the least perplexity among all techniques without any finetuning. While Bonsai achieves a lower perplexity post-finetuning with LoRA, our method takes about 20 times lower compression time compared to Bonsai. TVA-Prune offers higher inference speed than all of the approaches. Similar, our method prunes LLaMA-3 and retains higher performance than other methods without any finetuning. The pruned LLaMA-3 model in our case is about 40% faster than FLAP and wanda.

Task-agnostic pruning Comparison. Table 2 compares our method with other structured pruning techniques to prune LLaMA-7B and LLaMA-2-7B. It highlights the superior performance of our method in terms of perplexity on Wikitext-2 without even finetuning. Upon finetuning, the performance is comparable to Bonsai and VTrans. We see that without the dimension adaptation component, the inference speedup of our method becomes similar to previous approaches. Similarly, without the weight update step, our method generalizes to the performance of VTrans, but with higher inference speedup of the pruned model. Our pruned

Method	BoolQ Acc	PIQA Acc	HellaSwag Acc	WinoGrande Acc	ARC-e Acc	ARC-c Acc	Average↑ Acc
LLaMA-3-8B	81.28	79.70	60.17	72.37	80.09	50.59	70.70
Wanda-sp-gq	41.60	54.57	26.90	52.24	29.62	18.55	37.24
FLAP-gq	43.73	56.74	27.71	50.98	33.20	20.64	38.83
TVA-Prune	**62.96**	**65.83**	**36.81**	**57.61**	**47.81**	**23.37**	**49.06**
Mistral-7B	83.66	80.57	61.22	73.87	80.89	50.42	71.77
Wanda-sp-gq	60.14	55.72	26.47	50.10	30.84	19.63	40.48
FLAP-gq	62.12	57.02	27.70	49.81	32.02	21.50	41.69
TVA-Prune	**62.20**	**67.03**	**37.64**	**57.14**	**46.29**	**22.26**	**48.76**
Wanda-sp-gq†	53.26	58.45	36.14	52.95	42.87	30.45	45.68
FLAP-gq†	54.25	68.24	40.75	57.89	49.95	**31.85**	50.49
TVA-Prune†	**64.52**	**70.50**	**44.02**	**58.95**	**56.15**	27.90	**53.67**

Table 3: Performance comparison of the 50% pruned LLaMA-8B and Mistral-7B models on six zero shot tasks. †denotes finetuned with LoRA. Finetuning enhances the performance all pruned models, yet our pruned model still generalizes better across the tasks. .

models observe a 40% faster inference over other models.

Task-specific pruning comparison. For task-specific pruning with Wikitext-2 training set it is observed that there is an improvement in perplexity on the validation set for all the methods. Our method outperforms Bonsai, FLAP and wanda by a wide margin. It yields model similar in performance to VTrans but with faster inference. Observations are deferred to the Appendix 10.

Various Sparsity Ratios. We see in Figure 1 that our method performs better than other methods from about 30% sparsity and maintains the stable performance as sparsity increases. This is in contrast to FLAP and Wanda where the performance deteriorates sharply after 50% and 40% sparsity ratio respectively. At sparsity ratios lower than 30%, our method performs similarly to FLAP. Below 30% sparsity, Wanda and FLAP retain similar performance to ours. Our models pruned from LLaMA-3 and Mistral have much faster inference at all sparsity levels than FLAP and Wanda.

Pruning Time comparison. As illustrated in Figure 3, our pruning method exhibits comparable pruning times to VTrans while achieving significantly better model performance. When compared to more rapid techniques such as LLM-pruner and FLAP, our method delivers more than double the performance improvement. Additionally, our approach prunes models six times faster than the faster variant of Bonsai($p = 0.2$) and twelve times faster than LoRA-prune. Moreover, our method allows further lowering of the pruning time with

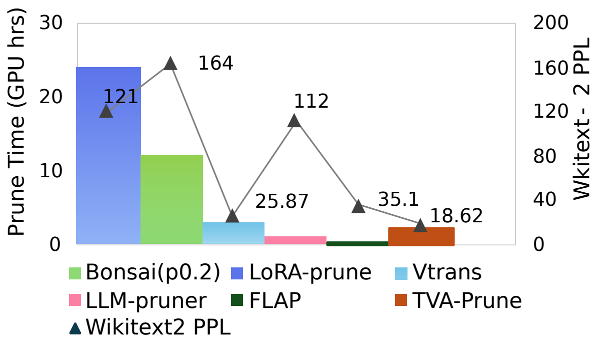

Figure 3: Comparison of Perplexity and time to prune LLaMA-7B by 50% with different structured pruning methods. Our method (TVA-prune) is more efficient yielding models with lower perplexity than methods taking similar or lower time to prune.

lower number of data samples as explored in Appendix A.

4.2 Performance on zero-shot tasks

In Table 3, we compare the performance of pruned models on six zero-shot reasoning tasks to assess the generalization efficiency of 50% pruned LLaMA-3 and Mistral models on unseen tasks. Our pruned LLaMA-3 models and pruned Mistral models outperform FLAP-gq and Wanda-sp-gq across all tasks. Since the TVA-prune (ours) model already generalises well to the tasks without any finetuning as per its capacity, finetuning it increases its performance by only about 3% on average. Despite a general decrease in performance for all the pruned models compared to their unpruned counterparts, our method most effectively preserves the generalization capabilities of the LLMs.

	LLaMA 2-7B	LLaMA 3-8B	Mistral 7B
TVA-prune	**18.44**	**27.50**	**18.37**
w/o weight update	+2.88	+5.32	+2.17
w/o dimension adapt	+0.05	+0.15	+0.42

Table 4: Performance of models on Wikitext-2 with and without post-prune weight update and dimension adaptation

	Dimension Multiple 0	8	64	128	256
50% pruned LLaMA-3-8B					
Speedup	0.9×	1.45×	**1.60**×	1.59×	1.59×
△PPL ↓	0	0	0	-0.1	-0.2
△Sparsity(%)	0	0	0.2	0.4	0.3
50% pruned Mistral-7B					
Speedup	1.1×	1.48×	1.52×	**1.67**×	1.40×
△PPL	-0.5	0	-0.5	-0.2	2.3
△Sparsity(%)	0	0	0.3	0.6	0.8
50% pruned LLaMA-2-7B					
Speedup	1.25×	1.52×	1.75×	**1.82**×	1.75×
△PPL	0	0	0	-0.05	-1.8
△Sparsity(%)	0	0	0.2	0.2	0.6

Table 5: Change in speedup, perplexity on Wikitext-2, and model sparsity on varying post-prune adapted dimension multiples. Across models it can be observed that adapting weight dimensions to be multiples of 64 or 128 yields the best speedup with least change in sparsity and often lower perplexity

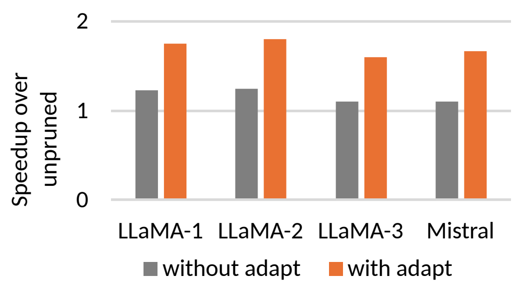

Figure 4: Inference speedup on NVIDIA A100(40GB) with and without our post-pruning dimension adaptation in 50% pruned models.

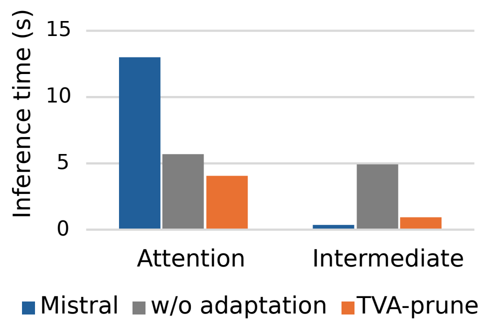

Figure 5: Time taken to infer on a single batch from Wikitext-2 by each module in Mistral-7B.

4.3 Ablation Study

Increase in speedup due to adaptation. Figure 4 illustrates the inference speedup on an NVIDIA A100 (40GB) GPU for 50% pruned models, comparing scenarios with and without post-pruning dimension adaptation. The y-axis represents the speedup factor over the unpruned models, with a value greater than 1 indicating faster performance. The results show that dimension adaptation significantly enhances speedup across all models.

Effect of adaptation in LLM modules. Figure 5 compares inference times for different modules in the Mistral-7B model: unpruned, TVA-prune with adaptation, and without adaptation. Adaptation significantly reduces attention module inference time over others. For the Intermediate module, the pruned model without adaptation increases inference time substantially, while adaptated model takes almost the same time as the unpruned module, likely due to parallel processing.

Which dimension multiple is the best? Adjusting weight matrix dimensions to be multiples of certain values, as shown in Table 5, optimizes GPU tensor core parallelism. When dimensions align with these multiples, computations parallelize more effectively, leading to significant speedups. As shown in the table, dimensions that are multiples of 64, 128, or 256 can maximize the utilization of tensor cores and increase throughput with minimal trade-offs as evidenced by the performance metrics of LLaMA and Mistral models.

Effect of post-prune weight update. Table 4 presents the performance of LLaMA and Mistral models with and without post-prune weight update and dimension adaptation. Omitting the weight update results in performance drops of 2.88 for LLaMA-2-7B, 5.32 for LLaMA-3-8B, and 2.17 for Mistral-7B, highlighting the crucial role of weight updates in maintaining high performance. Without dimension adaptation, the performance decreases slightly by 0.05 for LLaMA-2-7B, 0.15 for LLaMA-3-8B, and 0.42 for Mistral-7B. These results suggest that while dimension adaptation provides inference speedup benefits, the post-pruning weight update is significantly more critical for preserving the performance of the models. Overall, the combination of both techniques ensures the best performance retention in pruned models.

	ARC-e	HellaSwag
Pruned	-17.30	-57.26
Unpruned	-8.53	-35.95

Table 6: Average log likelihood of correct predictions by the 50% pruned and unpruned Mistral models shows that the pruned model is more uncertain about its correct predictions

4.4 Qualitative Analysis

As seen in Table 3, the pruned Mistral model shows a significant drop in factual knowledge, particularly in its ARC-e and ARC-c performance. However, further analysis reveals that the pruned model correctly classifies 62 ARC-e samples (2% of the total) that the unpruned model does not. As illustrated in Table 13, the pruned model often selects the annotated correct answer in cases where choices may seem ambiguous. For instance, on the question "Which is the best way to help prevent the flu from becoming a pandemic?", the unpruned model closely weighs "getting a vaccination" and "washing hands often," ultimately choosing the latter, while the pruned model selects "getting a vaccination." In most cases of incorrect prediction, the unpruned model shows confusion in its logits by weighing two choices in a multiple-choice question almost equally and choosing an answer different from the annotated one, while the pruned model displays less confusion and selects the correct answer. This raises the question of whether more 'knowledge' in models leads to more confusion and if pruning alleviates this. Additionally, in some instances (seen in Table 13), the pruned model is more factually accurate than the unpruned model, suggesting that overfitting during pretraining might cause the unpruned model's incorrect answers. Table 6 shows that the pruned model is overall more uncertain about its choices than the unpruned model.

5 Related Work

Efficient Transformers. As LLMs continue to grow in size, several methods have been developed to reduce their memory and computational constraints (Yun et al., 2024; Xiao et al., 2023). These methods broadly fall into two categories: quantization (Frantar et al., 2023; Dettmers et al., 2022, 2023; Zhang et al., 2024c; Lin et al., 2024; Shao et al., 2023) that reduce full precision weights to fewer bits and pruning (Xia et al., 2022; Sanh et al., 2020; Jiang et al., 2023b; Lagunas et al., 2021; Li et al., 2023) that removes weights and tokens. While being orthogonal approaches to compression, they have been combined to enhance compression further (Namburi et al., 2023; Saha et al., 2024). Several methods use knowledge distillation for performance recovery during pruning (Ko et al., 2023; Liu et al., 2023a; Lee et al., 2023; Liu et al., 2023b).

Pruning LLMs. Pruning methods are broadly categorized as unstructured pruning (Frantar and Alistarh, 2023; Xu et al., 2024; Zhang et al., 2024b) that targets individual weights within the LLM but yield models with faster inference only with specialized accelerators. Structured pruning (An et al., 2024; Zhang et al., 2024a) aims to eliminate redundant structures for faster inference but previous gradient-based methods (Ma et al., 2023; Zhang et al., 2023a) are limited by their substantial memory requirements, where as forward-pass only method Bonsai (Dery et al., 2024) takes nearly 40 hours for pruning during its search for optimal submodels. VTrans (Dutta et al., 2024), although a faster method, fails to prune heads in grouped query attentions and provides limited inference speedups.

6 Conclusion

We propose TVA-prune, a structured pruning method that can effectively compress grouped-query attention (GQA) based LLMs. It involves sharing key-head Variational Information Bottleneck-masks within groups of query heads to prune key, value and query heads and still maintain the structural integrity. The post-pruning dimension adaptation technique enhances parallelism, resulting in higher acceleration for pruned models without compromising performance. Morever, the post-pruning weight updates leverages importance scores of token representations and significantly contributes to maintaining the performance of the pruned models. We demonstrate the effectiveness of TVA-prune in retaining performance even in higher sparsity ratios. Further, it is more effective even in pruning MHA-based models. By training only the masks to prune LLM modules, TVA-prune has considerably lower pruning time compared to other gradient-based and some forward-pass-only approaches. Overall, TVA-prune shows improved resource utilization, achieving significant enhancements in both speed and performance suggesting its potential effectiveness in optimizing large language models for practical deployment.

7 Limitations

Although we have tackled recently published GQA-based and other diverse set of models, sparsity targets and datasets, there is still a vast list of benchmarks and models that could potentially reveal distinct behavior compared to the findings in this work. Hence, our work does not aim to provide an exhaustive set of results to universally characterize all models. In the experiments, we test with upto 8B models on a single GPU of 40GB. Extending it to larger models would require larger memory or more GPUs which we have not tested for in this work. However, our methodology would theoretically require low amount of memory, computations and provide greater inference speedups even for larger models. We plan to explore layer-wise pruning using our formulation to prune models larger than 13B on a single GPU in future work.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.

Bin Dai, Chen Zhu, Baining Guo, and David Wipf. 2018. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, pages 1135–1144. PMLR.

Lucio Dery, Steven Kolawole, Jean-Francois Kagey, Virginia Smith, Graham Neubig, and Ameet Talwalkar. 2024. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *Preprint*, arXiv:2306.03078.

Oshin Dutta, Ritvik Gupta, and Sumeet Agarwal. 2024. Vtrans: Accelerating transformer compression with variational information bottleneck based pruning. *https://arxiv.org/abs/2406.05276v2*.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv: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. 2023. A framework for few-shot language model evaluation.

E Hu, Y Shen, P Wallis, Z Allen-Zhu, Y Li, S Wang, L Wang, and W Chen. 2021. Low-rank adaptation of large language models. *arXiv*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ting Jiang, Deqing Wang, Fuzhen Zhuang, Ruobing Xie, and Feng Xia. 2023b. Pruning pre-trained language models without fine-tuning. *Preprint*, arXiv:2210.06210.

Jongwoo Ko, Seungjoon Park, Yujin Kim, Sumyeong Ahn, Du-Seong Chang, Euijai Ahn, and Se-Young Yun. 2023. Nash: A simple unified framework of structured pruning for accelerating encoder-decoder language models. *arXiv preprint arXiv:2310.10054*.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Hayeon Lee, Rui Hou, Jongpil Kim, Davis Liang, Hongbo Zhang, Sung Ju Hwang, and Alexander Min. 2023. Co-training and co-distillation for quality improvement and compression of language models. *arXiv preprint arXiv:2311.02849*.

Jianwei Li, Qi Lei, Wei Cheng, and Dongkuan Xu. 2023. Towards robust pruning: An adaptive knowledge-retention pruning strategy for language models. *arXiv preprint arXiv:2310.13191*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Chang Liu, Chongyang Tao, Jianxin Liang, Jiazhan Feng, Tao Shen, Quzhe Huang, and Dongyan Zhao. 2023a. Length-adaptive distillation: Customizing small language model for dynamic token pruning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Jiduan Liu, Jiahao Liu, Qifan Wang, Jingang Wang, Xunliang Cai, Dongyan Zhao, Ran Lucien Wang, and Rui Yan. 2023b. Retrieval-based knowledge transfer: An effective approach for extreme large language model compression. *arXiv preprint arXiv:2310.15594*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023c. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Satya Sai Srinath Namburi, Makesh Sreedhar, Srinath Srinivasan, and Frederic Sala. 2023. The cost of compression: Investigating the impact of compression on parametric knowledge in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5255–5273.

NVIDIA. 2024a. Inference optimization. `https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization`.

NVIDIA. 2024b. Linear/fully-connected layers user's guide. `https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J Goldsmith, and Mert Pilanci. 2024. Compressing large language models using low rank and low precision decomposition. *arXiv preprint arXiv:2405.18886*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Noam Slonim and Naftali Tishby. 1999. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Association for Computational Linguistics (ACL)*.

Chaojun Xiao, Yuqi Luo, Wenbin Zhang, Pengle Zhang, Xu Han, Yankai Lin, Zhengyan Zhang, Ruobing Xie, Zhiyuan Liu, Maosong Sun, et al. 2023. Variator: Accelerating pre-trained models with plug-and-play compression modules. *arXiv preprint arXiv:2310.15724*.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. 2024. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. *Preprint*, arXiv:2402.16880.

Jungmin Yun, Mihyeon Kim, and Youngbin Kim. 2024. Focus on the core: Efficient attention via pruned token compression for document classification. *arXiv preprint arXiv:2406.01283*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023a. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. *Preprint*, arXiv:2305.18403.

Mingyang Zhang, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, Bohan Zhuang, et al. 2023b. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. 2024a. Finercut: Finer-grained interpretable layer pruning for large language models. *arXiv preprint arXiv:2405.18218*.

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

Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. 2024c. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. *Proceedings of Machine Learning and Systems*, 6:381–394.

A Sample size vs pruning time.

	2k	4k	6k
Prune Time (hrs)	1	2	3
LLaMA-3 PPL	30.12	27.50	27.73
Mistral PPL	21.63	18.37	18.29

Table 7: Varying sample size impacts the performance of pruned models and pruning time. Reducing sample size favors pruning time but increases perplexity, while increasing sample size does not significantly improve performance.

Table 7 illustrates the impact of varying sample sizes (2k, 4k, 6k) on the performance and pruning time of pruned models, specifically LLaMA-3 and Mistral. As the sample size increases, pruning time also increases, while its reduction results in higher perplexity (PPL). Since increasing the sample size does not lead to significant improvements in perplexity, we choose to prune with 4000 samples for all types of models.

B Proportion of sub-layer parameters pruned

Figure 6 shows the proportion of the remaining parameters in the attention, the intermediate layers and the embedding layer after pruning each of the pre-trained LLaMA models to 50% sparsity. Lower number of attention parameters can be related to a slightly higher inference speedup in case of LLaMA-2 pruned model with respect to LLaMA-1 pruned model.

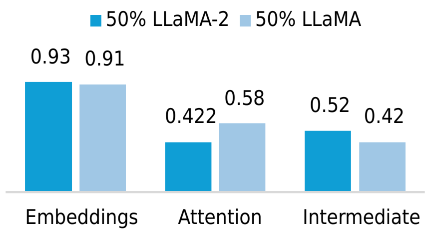

Figure 6: Proportion of remaining parameters in each of the LLM modules after pruning 50% of the total model parameters.

C Hyper-parameters for pruning and finetune

The hyper-parameters used for pruning LLaMA and Mistral models on one NVIDIA A100 (40GB) is given in Table 8 and for finetuning is given in

Table 9. We take a data blockor sequence_length of 400 while pruning LLaMA-3-7B to fit the training of masks in a single GPU.

VIB LR	Dataset size	block_size
$5\times10^{-2}, 1\times10^{-1}$	4000	512

Table 8: Hyper-parameters for pruning LLaMA and Mistral with TVA-Prune

weights LR	LoRA-rank	LoRA-α	η (distill Weight)	block_size
$1\text{x}10^{-4}$	128	4×rank	0.01	512

Table 9: Hyper-parameters for fine-tuning LLaMA and Mistral compressed models

D Algorithm

Algorithm 1 Pruning LLM with VIB masks, followed by post-prune adaptation

Input: Target model sparsity t, Pretrained model weights W
Initialize: VIB masks $\boldsymbol{m_i}, \boldsymbol{m_i}^h$or $\boldsymbol{m_i}^{vh}$
for $e = 1, ..., Samples$ **do**
  Sample $\eta \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$,apply random vectors $\boldsymbol{z}$ as in sec 3.1
  Calculate VIB loss as in Eq 1 and sparsity loss $\mathcal{L}_s$ as in sec. 3.1
  Backprop through total loss $\mathcal{L}_{\text{total}} = \mathcal{L} + \mathcal{L}_s$
  Update $\boldsymbol{\mu}, \boldsymbol{\sigma}$ of $\boldsymbol{m}$ masks, sparsity coefficients λ_1, λ_2
end for
Adapt dimensions as in section 3.2
Update weights as in section 3.3
Get sparser weights as in Eq 6
Remove zeroed out columns and rows in weights

Optionally fine-tune remaining weights with LoRA by minimizing: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{dis}} + \mathcal{L}_{\text{task}}$
Output: Compressed model weights $\widehat{W}$

E Task-specific pruning

Table 10 shows task specific pruning with Wikitext-2 dataset.

F Improved stability of our pruning method

In Table 11 we compare the standard deviation of performance measured over 5 random seeds for different pruning methods and observe that our method yields models with more consistent performance.

Model	LLaMA-2-7B	
	PPL ↓	Speedup
Unpruned	5.11	1×
Wanda-sp	97.70	1.29×
FLAP ‡	14.92	1.32×
Bonsai	19.24	1.28×
VTrans	11.88	1.33x
TVA-Prune	**11.86**	**1.82**×

Table 10: Performance comparison of task-specific pruning with Wikitext-2 train set and evaluation on the validation set. Our method outperforms other structured pruning methods (wanda-sp, Bonsai and FLAP) and is similar to VTrans with faster inference. †indicates finetuned with LoRA. ‡adds extra bias parameters.

Method	Sparsity Ratios					Average
	0.3	0.5	0.6	0.7	0.8	deviation
Wanda-sp	1.2	65	27	783	4340	1043
FLAP	0.18	3.4	24.09	110.9	3220	671
TVA-prune (ours)	0.16	1.65	5.42	7.72	14.18	6

Table 11: Comparison of standard deviation of performance (perplexity) measured over 5 random seeds on Wikitext-2. Our pruning method yields models more consistent in performance across sparsity ratios

G Zero shot results on LLaMA-1 and LLaMA-2

In Table 12 we show the zero-shot performance of LLaMA-1 and LLaMA-2 models.

H More explanation on optimizing GPU Performance with adjusted pruned weight dimensions

Having pruned weight dimensions in multiples of 256 enhances the performance of pruned models on NVIDIA V100 and A100 GPUs. Tiles are fixed-size blocks of matrix elements that GPUs process in parallel. Aligning matrix dimensions with preferred tile sizes like 256x128 ensures optimal use of Tensor Cores, minimizing computational waste due to tile quantization, where partially filled tiles perform unnecessary operations (NVIDIA, 2024b). Wave quantization occurs when the number of tiles doesn't match the number of streaming multiprocessors (SMs), leading to underutilized SMs and reduced performance. SMs are the primary computational units in NVIDIA GPUs, each capable

Method	BoolQ Acc	PIQA Acc	HellaSwag Acc	WinoGrande Acc	ARC-e Acc	ARC-c Acc	Average↑ Acc
LLaMA-7B	75.12	78.23	57.65	70.1	75.11	41.32	66.25
Wanda-sp†	50.58	55.01	37.57	54.65	41.72	31.89	45.23
LLM-Pruner†	60.28	69.31	47.06	53.43	45.96	29.18	45.95
FLAP †	60.21	67.52	40.07	57.54	49.66	28.49	50.57
LoRAPrune†	61.88	**71.53**	**47.86**	55.01	45.13	31.62	52.17
TVA-Prune†	**62.92**	68.25	41.95	**58.42**	**56.68**	**27.14**	**52.56**
LLaMA-2-7B	77.70	78.07	57.16	69.06	76.34	43.43	66.96
Wanda-sp†	51.43	55.46	37.24	53.98	42.26	28.68	45.51
FLAP †	60.54	**66.78**	40.76	57.32	50.18	**28.74**	50.72
TVA-Prune†	**63.24**	66.12	**42.37**	**58.84**	**56.82**	28.42	**52.63**

Table 12: Performance comparison of the 50% pruned LLaMA-1-7B and LLaMA-2-7B models on six zero shot tasks. †denotes finetuned with LoRA. Finetuning enhances the performance all pruned models, yet our pruned model still generalizes better across the tasks. .

of executing multiple threads in parallel. Efficient distribution of workload across all SMs is crucial for maximizing GPU performance.

I Qualitative comparison on examples from ARC-easy dataset

In Table 13 we show examples where the pruned model predicts more accurately than the unpruned Mistral model.

Instances where the choices might seem ambiguous; pruned model correctly chooses the annotated answer	
Question : Choices: Annotated: unpruned pruned	Which is the best way to help prevent the flu from becoming a pandemic? "getting a vaccination", "taking antibiotics", "eating fruits and vegetables", "washing hands often" "getting a vaccination" "washing hands often" "getting a vaccination"
Question : Choices: Annotated: unpruned pruned	Which of these traits is most influenced by environment? "weight", "hair color", "blood type", "handedness" "weight" "hair color" "weight"
Question : Choices : Annotated: unpruned: pruned:	What safety procedure should a student follow when a thermometer is broken during a lab experiment? "tell the teacher immediately", "stop the experiment immediately", "sweep the glass into a biohazard container", "use a paper towel to pick up the pieces" "tell the teacher immediately" "stop the experiment immediately" "tell the teacher immediately"
Instances where pruned model is factually correct while the unpruned is not	
Question: Choices : Annotated: unpruned: pruned:	"Which is a characteristic of both plants and animals?" "life cycles", "learned behaviors", "produce their own food", "reproduce using seeds" "life cycles" "produce their own food" "life cycles"
Question" Choices: Annotated: unpruned: pruned:	Which of these atomic structures has the least amount of mass? "an"an ion", "a proton", "a neutron", "an electron" "an electron" "a proton" "an electron"
Question: Choices: Annotated: unpruned: pruned:	"What should be added to soil to increase its water retention?" "pebbles", "sand", "rocks", "clay" "clay" "sand" "clay"
Instances where pruned model chooses incorrectly while unpruned chooses correctly	
Question: Choices: Annotated: unpruned: pruned:	"To express the distance between the Milky Way galaxy and other galaxies, the most appropriate unit of measurement is the" "meter." "kilometer.", "light-year.", "astronomical unit." "light-year." "light-year." "meter"

Table 13: Examples from the ARC-easy dataset where Mistral 50% pruned model's answers are compared to the unpruned model