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

Oshin Dutta [1] **Ritvik Gupta** [2] **Sumeet Agarwal** [1]

Abstract

Structured pruning removes entire components like attention heads or layers to yield faster dense models. However, previous methods require significant pruning time and overlook token embedding dimension, missing potential inference acceleration. Moreover, pruning of heads in grouped query attentions is not widely attempted due to challenges with their interdependencies. To address these limitations, we propose a structured pruning method for LLMs that incorporates the concept of Variational Information Bottleneck (VIB) to obtain compressed representations at each structural element while preserving essential information for accurate prediction. We enhance the formulation to account for all preceding layers' influence on the current compressed representation, ensuring a globally informed reduction in token embedding dimension and grouped query not explored in previous work. Additionally, we include a post-pruning step to adjust the pruned model dimensions, ensuring optimal use of Tensor Cores in GPUs which speeds up inference by up to 60%. Evaluated on several language benchmarks using variants of LLaMA models and Mistral, our method shows a reduction in pruning time by up to 90% with higher inference speed and competitive performance.

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. 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 & Alistarh, 2023). Quantization techniques require specialized GPUs and libraries for acceleration (Dettmers et al., 2022; Zhang et al., 2024).

Structured pruning methods often fail to prune token representations due to the complex dependencies that span across layers of the model, thereby missing out on added acceleration. Pruning of attention heads, especially 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. 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, 2024b) which often perform operations in certain fixed block sizes like-128x256. Certain work aim at inference speedups (Chen et al., 2024; Sheng et al., 2023; Liu et al., 2023) involving complex algorithms.

To address these challenges, we propose an efficient structured pruning method for LLMs- **T**oken dependency-aware **Va**riational Information Bottleneck-based pruning (**TVA-Prune**). We extend the formulation of the VIB principle to include global token dependency-awareness. Our method effectively removes redundant 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 adaptation step converts dimensions to match the block sizes of Tensor cores, ensuring parallelism in the inference GPU and thus achieving higher

[1]Department of Electrical Engineering, Indian Institute of Technology, Delhi, India [2]Machine Learning Department, Carnegie Mellon University, PA, USA. Correspondence to: Oshin Dutta <oshin.dutta@ee.iitd.ac.in>, Ritvik Gupta <ritvikgu@cs.cmu.edu>, Sumeet Agarwal <sumeet@iitd.ac.in>.

Accepted to ICML ES-FoMo Workshop, Vienna, Austria, 2024.

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

Input: Target model sparsity t, Frozen Pretrained model $f(\cdot;\theta_s)$
Initialize: VIB masks $\boldsymbol{z_i}$, $\boldsymbol{z_i}^h$ or $\boldsymbol{z_i}^{vh}$
for $e = 1, ..., Samples$ **do**
  Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$; apply VIB masks as in Eq 2
  Calculate sparsity loss $\mathcal{L}_s$ as in sec. 2.3
  Total loss $\mathcal{L}_{\text{total}} = \tilde{\mathcal{L}}_{\text{VIB}} + \mathcal{L}_s$
  Update $\boldsymbol{z}$ masks, sparsity coefficients λ_1, λ_2
end for
Convert masks to binary hard masks z_i^{mask}
Adapt dimensions by modifying masks to $\widehat{z}_i^{mask}$
Prune global token representations, attention heads, intermediate dimensions according to VIB-masks:
$\widehat{f}(\cdot;\theta_s) \leftarrow f(\cdot;\theta_s) \odot \widehat{z}_i^{mask}$
Optional: fine-tune remaining weights to recover performance with LoRA by minimizing: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{distil}} + \mathcal{L}_{\text{task}}$
Output: Compressed model $\widehat{f}(\cdot;\theta_s)$

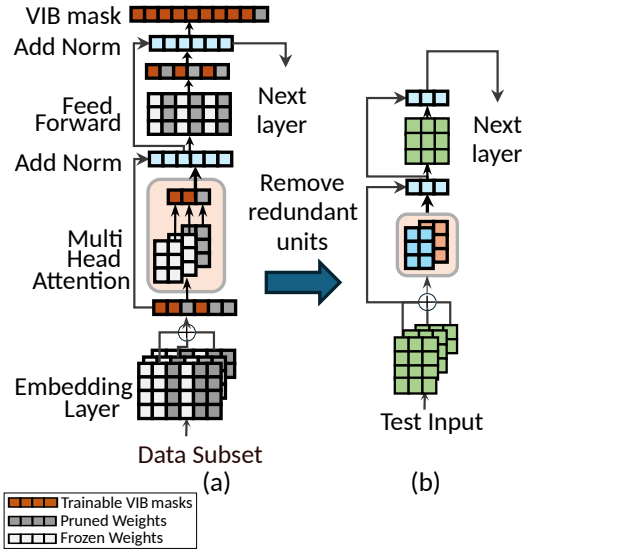

Figure 1: (a) Structured pruning of token representations and heads, considering global token dependencies using VIB masks. (b) Compressed dense model.

inference speedup. Pre-trained LLMs Including variants of LLaMA-7B (Touvron et al., 2023a;b), and Mistral-7B (Jiang et al., 2023) are pruned, demonstrating superior performance compared to prior methods.

2. Global Dependency-aware Structured Pruning

Background. For a transformer network $f(\cdot;\theta_s)$, we denote its layer embedding as $\{\boldsymbol{k}_i\}_{i=1}^{L}$, $\boldsymbol{k}_i \in \mathbb{R}^{n \times seq \times d}$. Our objective is to obtain condensed token representations at each layer, intermediate layers and compressed Multi-Head Attention (MHA) while preserving essential information for accurate prediction $\tilde{\boldsymbol{y}}$. This approach builds upon the VIB formulation proposed by (Dai et al., 2018) to prune CNNs and FFNs by retaining only the information relevant for prediction. But instead of obtaining compressed representation for just each input sample, transformers also require compressed representation for each of the sequences.

2.1. Global Token Dependency-Aware VIB loss

The VIB approach to compress representations for transformers oversimplifies the dependencies in a large language model (LLM) where longer-range interactions across layers can be crucial for performance, especially in tasks involving context, like language or sequence processing. Hence, we adapt the VIB formulation for LLMs by adding dependence of current layer token representation on all previous layers. The objective function given by the VIB principle to compress successive representations $\boldsymbol{k}_i$ (Dai et al., 2018) given the input and the output $\boldsymbol{Y}$ is,

$$\tilde{\mathcal{L}}_{\text{VIB}} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\epsilon,x,y}\left[-\log q(\boldsymbol{Y})\right] + \beta\,\mathrm{KL}\left[p(\boldsymbol{k}_i|\boldsymbol{k}_{i-1}), r(\boldsymbol{k}_i)\right] \quad (1)$$

Equation 1 minimizes the expectation of the negative log-likelihood of the variational output distribution, which ensures that the model is able to predict the output. The KL divergence acts as regularizer that minimizes the mutual information between successive layer representations.

The compressed representation is obtained by multiplying a random set of vectors $\boldsymbol{z}$ which are also the dimension masks with trainable parameters $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{1\times d}$, to the output of the LLM layer activations with d output dimension of the LLM layer.

$$\boldsymbol{k}_i = \boldsymbol{z}_i \odot f_i(\boldsymbol{k}_{i-1}, \boldsymbol{k}_{i-2}, \ldots); \quad \boldsymbol{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_i \quad (2)$$

With the above definition, we use encoder of the form $p\left(\boldsymbol{k}_i \mid \boldsymbol{k}_{i-1}\right) = \mathcal{N}(\boldsymbol{k}_i; f_i \odot \boldsymbol{\mu}_i\, \mathrm{diag}\left[f_i^2 \odot \boldsymbol{\sigma}_i^2\right])$ where f_i is a LLM layer that aggregates all previous layer token representations, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $r\left(\boldsymbol{k}_i\right) = \mathcal{N}\left(\boldsymbol{k}_i; \mathbf{0}, \mathrm{diag}\left[\boldsymbol{\xi}_i\right]\right)$ where $\boldsymbol{\xi}_i$ is learned from the data. During training with data samples, we backpropagate through the final layer which approximates the expectation in equation 1.

2.2. Pruning attention heads in grouped query.

Normal multi-head attention treats each head independently, making it simpler to prune individual heads without impacting others. In contrast, selectively pruning heads within

Model	LLaMA-7B			LLaMA-2-7B		
	PPL	Speedup	Inference Time (mins)	PPL	Speedup	Inference Time (mins)
Unpruned	5.68	1×	0.60	5.11	1×	0.62
Wanda-sp	366.43	1.24×	0.48 (-20.00%)	97.70	1.29×	0.48 (-22.58%)
LLM-pruner	112.44	1.23×	0.49 (-18.33%)	95.26	1.29×	0.48 (-22.58%)
FLAP ‡	35.10	1.26×	0.48 (-20.00%)	25.40	1.32×	0.47 (-24.19%)
Bonsai	22.62	1.26×	0.48 (-20.00%)	19.24	1.28×	0.48 (-22.58%)
TVA-Prune	**18.5**	**1.75×**	**0.34 (-43.43%)**	**14.15**	**1.8×**	**0.34 (-45.16%)**
TVA-Prune w/o PostAdapt	**18.5**	1.1×	0.54(-10.00%)	**14.15**	1.1×	0.56(-09.67%)
Wanda-sp †	67.24	1.24×	0.48 (-20.00%)	46.54	1.29×	0.48 (-22.58%)
LLM-pruner†	38.12	1.23×	0.49 (-18.33%)	29.56	1.29×	0.48 (-22.58%)
Bonsai †	10.92	1.26×	0.48 (-20.00%)	**9.15**	1.28×	0.48 (-22.58%)
TVA-Prune †	**10.58**	**1.75×**	**0.34 (-43.43%)**	9.58	**1.8×**	**0.34 (-45.16%)**

Table 1: Pruned models are evaluated on Wikitext-2. Our method outperforms structured pruning (wanda-sp, Bonsai, LLM-pruner and FLAP) †indicates finetuned with LoRA. ‡adds extra bias parameters. All inferences were performed on NVIDIA A100 (40GB) GPU. w/o post-adapt speed reduction is due to pruned dimensions not aligning with GPU tensor cores, which worsens for our method since it prunes token/hidden state dimensions across all layers unlike others. Post-adaption leads to negligible change in model size.

GQA (Grouped Query Attention) groups is more complex, as it requires ensuring that the group's overall functionality remains effective despite the pruning. In GQA, each head shares a common key-value pair, making the pruning process more intricate. In our method, for each key-value head pruned, all the heads in the query that share it are also pruned. This ensures that the pruning process does not disrupt the group's overall functionality. Let k_i^{vh} represent the compressed token representation of the i-th layer for the key-value head vh,

$$\boldsymbol{k_i}^{vh} = \boldsymbol{z_i}^{vh} \odot f_i^h(\boldsymbol{k_i}^{vh}, \boldsymbol{k_{i-2}}^{vh}, \ldots) \tag{3}$$

where the function f_i^h depends on the token representation of all previous layers for the key-value head vh. We train the masks for compression of heads, intermediate layers and token representation dimensions in an end-to-end manner.

2.3. Implementation of Pruning

In the inference phase, the vectors $\boldsymbol{z_i}$ are converted to hard binary masks z_i^{mask} using a thresholding operation on $\alpha_{i,j} = \mu_{i,j}^2/\sigma_{i,j}^2$ for pruning neurons.

Sparsity Control. Using z_i^{mask} hard masks, we calculate model sparsity as the ratio of pruned parameters to the initial count. We use a Lagrangian term 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 s_e is the expected model sparsity, t the target sparsity and $\lambda_1, \lambda_2 \in \mathbb{R}$ are jointly updated during the pruning.

Finetuning with LoRA. Like previous approaches (Dery et al., 2024; Ma et al., 2023), we finetuning on a downstream task using LoRA (Hu et al., 2021) to recover performance. Additionally, we distil knowledge from the teacher logits to the pruned student logits (Xia et al., 2022). The steps of our method is shown in Algorithm 1.

2.4. Post-Pruning Dimension Adaptation

The dimensions in pre-trained unpruned models are optimized for efficient GPU execution using specific block sizes (NVIDIA, 2024a) 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 z_i^{mask} are non-zero and estimates the corresponding weight dimensions as $n = |I|; \quad I = \{j \mid z_{i,j}^{mask} \neq 0\}$. 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,

$$n' = \left(\left\lfloor \frac{n + T/2}{T} \right\rfloor \right) \times T \tag{4}$$

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,

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

$$\widehat{z}_{i,j}^{mask} = \begin{cases} 1 & \text{if } \log \alpha_{i,j} > \tau \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in d \tag{5}$$

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

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
TVA-Prune	**18.37**	**1.67**×	**41.39**
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 2: Performance comparison of Mistral-7B and LLaMA-3-8B models pruned by 50%. Our method outperforms others without any finetuning. †indicates finetuned with LoRA.

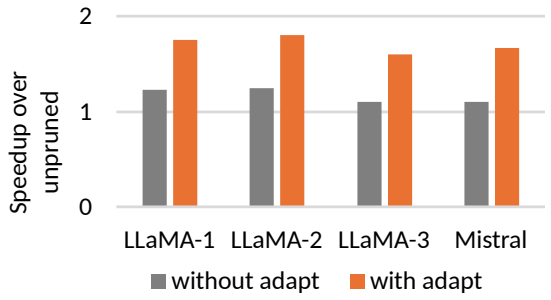

Figure 2: Inference speedup on NVIDIA A100(40GB) with and without our post-pruning dimension adaptation in 50% pruned models.

3. Experiments

All experiments are conducted on a single NVIDIA A100 (40GB) GPU. We prune the LLaMA-7B, LLaMA-2-7B and LLaMA 3 (Touvron et al., 2023a), and Mistral-7B (Jiang et al., 2023) models, to evaluate and compare the structured pruning methods on the validation set of Wikitext-2 (Merity et al., 2016). 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. We also evaluate on task-agnostic context with zero-shot tasks designed for common sense reasoning (Gao et al.). We use 5000 samples of C4 (Raffel et al., 2020) with a batch size of 1 and sequence length of 512 to prune the models. The baseline methods taken for comparison are structured Wanda (Sun et al., 2023), FLAP (An et al., 2024), LLM-Pruner (Ma et al., 2023), LoRAPrune (Zhang et al., 2023) and Bonsai (Dery et al., 2024).

	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 3: 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

3.1. Results

Performance Comparison of MHA-based models. Table 1 compares pruning techniques for LLaMA-7B and LLaMA-2-7B, highlighting TVA-Prune's superior performance in terms of perplexity on Wikitext-2 test set and inference speed-up. Only LLaMA-2 finetuned by Bonsai (Dery et al., 2024) achieves a better performance, at the cost of low inference speed-up and 20 times higher compression time.

Inference speedup. Figure 2 shows how our post-pruning dimension adaptation method leads to up to 60% gain in speedup over unpruned models.

Pruning Grouped Query Attention. Table 2 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. 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. We see in Figure 3 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.

Performance on zero-shot tasks In Table 4, we compare the performance of compression methods on six zero-shot reasoning tasks to assess the generalization efficiency of the 50% pruned LLaMA-7B models.

Method	WikiText2↓ PPL	BoolQ Acc	PIQA Acc	HellaSwag Acc	WinoGrande Acc	ARC-e Acc	ARC-c Acc	Average↑ Acc	Compression Time(hrs)
Unpruned model	5.68	76.5	79.8	76.1	70.1	72.8	47.6	70.48	-
Wanda-sp	132	50.58	55.01	29.56	51.78	31.27	23.04	40.21	0.16
LLM-Pruner†	16.41	60.28	69.31	47.06	53.43	45.96	29.18	45.95	1
FLAP ‡	31.8	60.21	67.52	40.0	57.54	49.66	28.49	50.57	0.3
LoRAPrune†	11.60	61.88	**71.53**	**47.86**	55.01	45.13	**31.62**	52.17	> 24
Bonsai†	10.92	**67.22**	60.85	43.09	**61.64**	54.92	26.28	52.33	40
TVA-Prune†	**10.58**	63.27	68.56	42.0	57.38	**56.97**	26.46	**52.44**	2

Table 4: Zero shot performance of the compressed LLaMA-7B. †Finetuned with LoRA ‡adds extra bias parameters. Inference done on NVIDIA A100 (40GB). Our method takes 10 to 20 times lower time to prune than other methods with similar performance. Method with lower compression time have much lower average performance.

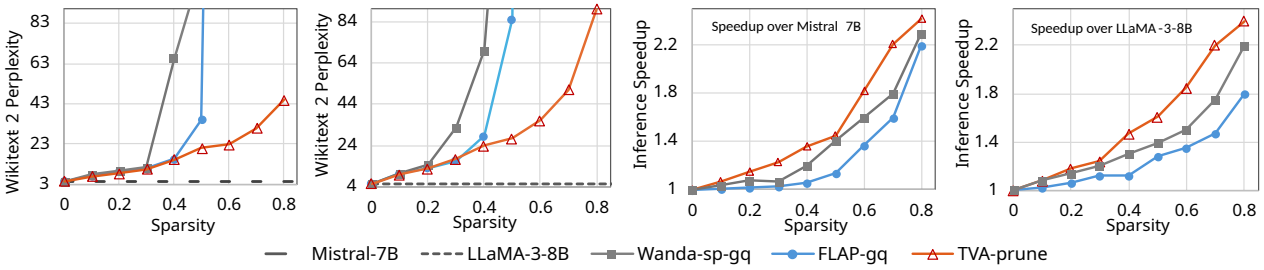

Figure 3: Comparison of sparsity, perplexity and inference speedup of GQA-based LLaMA-3-8B and Mistral-7B models pruned to different sparsity ratios with C4 train set and evaluated on Wikitext-2 validation set. Speedup is measured on NVIDIA A100(GB) for evaluation on the validation set.

Method	No. of GPUs	Pruning Time (GPU hrs)	Wikitext2 PPL
LLM-pruner	4	1	29.56
FLAP	1	0.7	25.40
Bonsai	1	40	9.15
TVA-Prune	1	2	9.58

Table 5: Comparison of pruning methods for LLaMA-2-7B model to 50% sparsity using structured pruning, including only methods with finetuned perplexity (PPL) below 50.

Compression Speedup. As shown in Table 5, our method requires fewer GPUs compared to LLM-pruner. It achieves a pruning time reduction of more than 90% relative to Bonsai, while maintaining a similar perplexity. These results indicate that our method offers superior efficiency in resource utilisation, significantly faster pruning times, and effective model performance.

Which dimension multiple is the best? Adjusting weight matrix dimensions to be multiples of certain values, as shown in Table 3, 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.

4. Conclusion

Our experiments demonstrate the efficacy of our structured pruning framework that prunes token dimensions while considering previous layer representations in the VIB framework. It involves only training the masks to prune LLM modules while freezing the model parameters, thus significantly reducing pruning time over other approaches. Our post-pruning adaptation step aims to fully utilize the GPU tensor cores for a higher acceleration of the pruned LLaMA and Mistral models. Our method uses a single GPU and prunes LLMs in a few hours, making it suitable for environments with limited computational resources. Additionally, the evaluation of our models on varied datasets including zero-shot classification underscores the robustness and versatility of our pruning strategy.

Acknowledgement

We express our gratitude to the anonymous reviewers, Professors Surendra Prasad and Brejesh Lall from IIT Delhi, as well as our colleagues at Cadence India for their valuable feedback and insights. This research is funded by Cadence India, with additional support for the first author from a Ministry of Education fellowship.

References

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

An, Y., Zhao, X., Yu, T., Tang, M., and Wang, J. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.

Chen, Z., May, A., Svirschevski, R., Huang, Y., Ryabinin, M., Jia, Z., and Chen, B. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

Dai, B., Zhu, C., Guo, B., and Wipf, D. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, pp. 1135–1144. PMLR, 2018.

Dery, L., Kolawole, S., Kagey, J.-F., Smith, V., Neubig, G., and Talwalkar, A. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*, 2024.

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

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

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., et al. A framework for few-shot language model evaluation, 12 2023. *URL https://zenodo. org/records/10256836*, 7.

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

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

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

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

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

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

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X.,

Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023b.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. In *Association for Computational Linguistics (ACL)*, 2022.

Zhang, M., Shen, C., Yang, Z., Ou, L., Yu, X., Zhuang, B., et al. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*, 2023.

Zhang, Z., Liu, S., Chen, R., Kailkhura, B., Chen, B., and Wang, A. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. *Proceedings of Machine Learning and Systems*, 6:381–394, 2024.

A. Proportion of sub-layer parameters pruned

Figure 4 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.

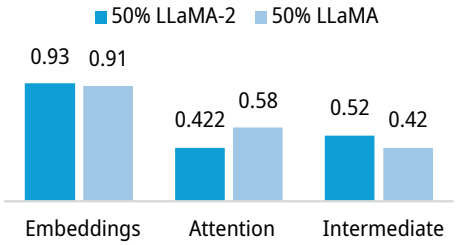

Figure 4: Proportion of remaining parameters in each of the LLM modules after pruning 50% of the total model parameters.

B. Hyper-parameters for pruning

The hyper-parameters used for pruning LLaMA and Mistral models on one NVIDIA A100 (40GB) is given in Table 6 and for finetuning is given in Table 7.

VIB LR	Dataset size	block_size
$5 \times 10^{-2}, 1 \times 10^{-1}$	5000	512

Table 6: Hyper-parameters for pruning LLaMA and Mistral with TVA-Prune

C. 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, 2024a). 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 of executing multiple threads in parallel. Efficient distribution of workload across all SMs is crucial for maximizing GPU performance.

weights LR	LoRA-rank	LoRA-α	η (distill Weight)	block_size
1x10^{-4}	128	4×rank	0.01	512

Table 7: Hyper-parameters for fine-tuning LLaMA and Mistral compressed models