Change Is the Only Constant: Dynamic LLM Slicing based on Layer Redundancy

Razvan-Gabriel Dumitru[1], Paul-Ioan Clotan[2], Vikas Yadav[3], Darius Peteleaza[4], Mihai Surdeanu[1]
[1]University of Arizona, [2]Università di Bologna, [3]ServiceNow AI,
[4]University Lucian Blaga of Sibiu
Correspondence: razvandumm@gmail.com

Abstract

This paper introduces a novel model compression approach through dynamic layer-specific pruning in Large Language Models (LLMs), enhancing the traditional methodology established by SliceGPT. By transitioning from constant to dynamic slicing, our method leverages the newly proposed Layer Redundancy (LR) score, which assesses how much change each layer changes its input by measuring the cosine similarity of the input to the output of the layer. We use this score to prune parts of individual layers based on redundancy in such a way that the average pruned percentage for all layers is a fixed value. We conducted extensive experiments using models like Llama3-8B and Mistral-7B on multiple datasets, evaluating different slicing bases and percentages to determine optimal configurations that balance efficiency and performance. Our findings show that our dynamic slicing approach not only maintains but, in many cases, enhances model performance compared to the baseline established by constant slicing methods. For instance, in several settings, we see performance improvements of up to 5% over the SliceGPT baseline. Additionally, a perplexity decrease by as much as 7% was observed across multiple benchmarks, validating the effectiveness of our method. The code, model weights, and datasets are open-sourced at https://github.com/RazvanDu/DynamicSlicing.

1 Introduction

Large Language Models (LLMs), characterized by their massive scale, often consist of billions to trillions of parameters, enabling them to perform a wide range of complex tasks with remarkable proficiency (Kevian et al., 2024; Touvron et al., 2023; Team et al., 2023; Jiang et al., 2023). However, the deployment of these models poses significant challenges, primarily due to the extensive computational resources requirements. As the scale of these models grows, so does the urgency to develop more efficient methods for their deployment. This has led to increased interest in model compression techniques that aim to reduce the computational burden without substantially sacrificing performance. Techniques such as knowledge distillation, quantization, or pruning variants have emerged as viable solutions (Wan et al., 2023), each offering a different approach to streamlining model architecture and operations (Wang et al., 2024).

In this paper, we improve the work on model pruning introduced by SliceGPT (Ashkboos et al., 2024), a pruning technique via a constant slicing percentage of each layer. While this approach reduces computational demands and maintains a level of performance, it does not account for the varying significance of different layers within the network. We propose a more nuanced, dynamic pruning method that adapts the degree of pruning based on the individual characteristics and contributions of each layer. Our method aims to optimize both the efficiency and the efficacy of the pruning process by preserving more functionality in critical areas of the model, leading to better performance and less degradation in tasks.

More specifically, we develop a new metric, namely Layer Redundancy (LR) score, to quantify the impact of each layer on the model's overall performance. This evaluation is essential, as it guides the order in which layers are pruned, ensuring that the most influential layers are preserved while less critical layers are removed. Our approach involves generating slicing functions tailored to the importance of each layer, allowing for a dynamic and informed pruning strategy. Our results from the extensive empirical studies across various datasets and base models show a substantial improvement in model accuracy across all datasets tested, accompanied by a notable reduction in perplexity. In order to thoroughly evaluate the effectiveness of our proposed dynamic slicing pattern, we also analyzed the median accuracy and perplexity across

Findings of the Association for Computational Linguistics: EACL 2024, pages 9912–9920

a range of models. The results consistently show the superiority of our method over conventional constant slicing techniques.

The main contributions of our paper are:

• We showed that dynamic, adaptive layer pruning can significantly improve computational efficiency without compromising model performance.

• The introduction of the Layer Redundancy score, a new metric to evaluate and guide the dynamic pruning of layers in LLMs.

• Extensive empirical validation shows that our method outperforms static pruning techniques in terms of both accuracy and perplexity across various settings.

2 Related Work

Recent advancements in model compression techniques (Hoefler et al., 2021; Zhu et al., 2023) have markedly improved the efficiency of deploying LLMs while striving to retain their performance. The field has seen a variety of approaches including knowledge distillation (Hinton et al., 2015; Hsieh et al., 2023), quantization (Ma et al., 2024), pruning (Ma et al., 2023; Yang et al., 2024), low-rank adaptation (Hu et al., 2021) or hybrid variants (Xu et al., 2023; Dettmers et al., 2024), each designed to address the growing computational and memory requirements of these models.

Innovative approaches such as LLM-Pruner (Ma et al., 2023) and LaCo (Layer Collapse) (Yang et al., 2024) offer novel perspectives on model pruning. LLM-Pruner focuses on structured pruning by identifying and removing dependency groups within the model, aiming to minimize dependency on the original training corpus while preserving linguistic capabilities. Similarly, LaCo presents a layer-wise pruning strategy where subsequent layers collapse into preceding ones, achieving notable size reduction while maintaining good performance. A third approach (Gromov et al., 2024) explores the potential of simple layer-pruning strategies combined with parameter-efficient finetuning (PEFT), demonstrating minimal performance loss even when half of the model's layers are removed.

Among the innovative strategies in LLM optimization, SliceGPT (Ashkboos et al., 2024) emerges as a significant breakthrough in model compression. Developed to address the intensive computational and memory demands of deploying LLMs, SliceGPT employs a unique post-training sparsification technique. Although effective in practice, previous research has illustrated that the order in which layers are removed plays a critical role in model performance (Gromov et al., 2024; Men et al., 2024). This insight led us to explore variable slicing percentages across different layers, challenging the constant slice for all layers. Initial attempts by the creators of SliceGPT to implement this through spectral analysis of layers did not yield a reliable method for determining the optimal percentage to be removed from each layer, as spectral analysis only tells part of the story and doesn't correlate with how much can be sliced out of a layer.

3 Method

3.1 LR score

Building on concepts introduced in a recent, unpublished study (Men et al., 2024), we propose a novel metric for assessing layer usefulness. This metric quantifies the extent to which each layer modifies its input by measuring the cosine similarity between the input and output. Specifically, we define this as the Layer Redundancy (LR) score:

$$LR(\mathbf{L_i}) = \frac{\mathbf{L_i^I} \cdot \mathbf{L_i^O}}{\|\mathbf{L_i^I}\|\|\mathbf{L_i^O}\|} \quad (1)$$

In Eq 1 $\mathbf{L_i^I}$ refers to the input of $layer_i$, while $\mathbf{L_i^O}$ refers to the output of $layer_i$. Intuitively, the higher cosine similarity between inputs and outputs leads to higher LR score, implying that the layer is more redundant. To evaluate the score for all layers, we use the full validation set of the PG-19 data set (Rae et al., 2019), we pass the data as context through the LLMs and we sum up how much each layer processed its input according to the cosine similarity score. At the end we normalize the values linearly so that the $min(LR(\mathbf{L_i})) = 0$ and $max(LR(\mathbf{L_i})) = 1$, guaranteeing that they range from 0 to 1. Intuitively, a LR score closer to 0 corresponds to lesser redundancy while higher LR score (close to 1) would mean high redundancy. To the best of our knowledge, our work is the first one to use a per layer importance/redundancy score to prune variable parts of layers out (explained in the following section), instead of removing whole layers or removing a fixed constant portion of the layer (Men et al., 2024).

3.2 Defining a per-layer percentage

Our goal is to have a function that slices variable sized parts of each layers based on their LR score

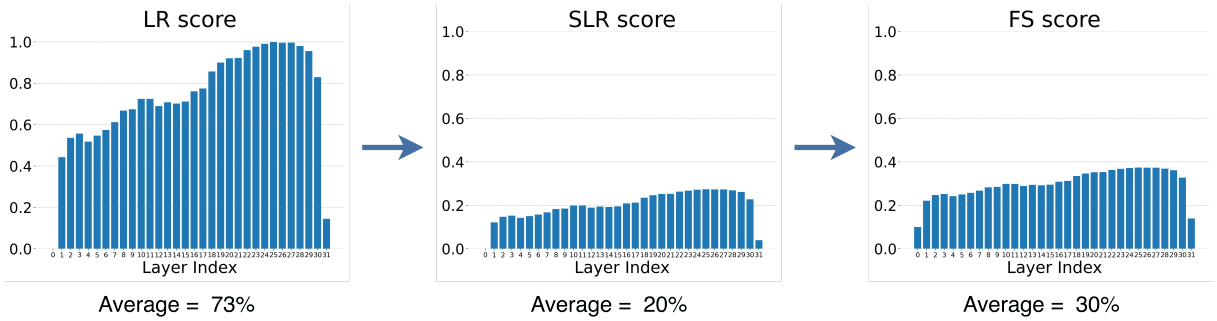

Figure 1: Example of the Layer Redundancy (LR) score as well as the transformations used to achieve a slice percentage of 30% ($S_P = 0.3$) with a base slice for all layers of 10% ($S_B = 0.1$). The example is shown for the 32 layers of llama3-8B.

while keeping the overall average of sliced out parts acorss all layers to be a fixed percentage of the LLM parameters specified by the user. For example, layers with higher LR values (or redundancy) can be sliced more compared to layers with lower LR score (or redundancy). To achieve this, we applied several transformations on the previously defined LR score such that we have control over how big of an impact we want the redundancy to have on the sliced percentage.

The first step is to control the average of the LR score so that we can then control the average pruned percentage of the LLM. Concretely, we denote the average desired slicing percentage as S_P (Slice Percentage). In order to investigate what happens as we go further away from a constant slice for each layer, we will also define S_B (Slice Base) to be a fixed constant value that will be guaranteed to be sliced from each layer such that $S_B <= S_P$. As $S_B = S_P$, the slicing becomes constant as presented by Ashkboos et al. (2024). We experimented with different values of S_B to see the effect of layer redundancy on the LLM's performance. The next step is to scale the LR function so that its average is $S_P - S_B$, thus once we add the base percentage for each layer (S_B) the total average will be $S_P - S_B + S_B = S_P$. This can be achieved by multiplying each LR_i value by the ratio of $S_P - S_B$ divided by the mean of the function. This is denoted as Slice per Layer Redundancy (SLR) and shown in Eq 2.

$$SLR(\mathbf{L_i}) = \text{LR}_i \cdot \frac{S_P - S_B}{\frac{1}{n}\sum_{i=1}^{n} \text{LR}_i} \tag{2}$$

At the end, the Final Slice (FS) for the layer $\mathbf{L_i}$, is the sum of SLR and S_B as shown in Eq 3.

$$FS(\mathbf{L_i}) = SLR(\mathbf{L_i}) + S_B \tag{3}$$

Average of FS for all n layers of a LLM is equal to S_P as shown in eq. (4).

$$\frac{\sum_{i=1}^{n} FS(\mathbf{L_i})}{n} = S_P \tag{4}$$

3.3 Slicing parts of layers

The methodology in *SliceGPT* utilizes a specialized version of Principal Component Analysis (PCA) (Abdi and Williams, 2010) for efficient data reduction. It projects the data matrix X onto a lower-dimensional subspace using the eigenvectors Q and a deletion matrix D. The reduced matrix Z is computed as XQD, where D selectively omits certain components from Q, resulting in a compressed representation Z. The approximate reconstruction $\tilde{X}$ is obtained by ZD^TQ^T, minimizing the reconstruction error $\|\mathbf{X} - \tilde{\mathbf{X}}\|^2$. Unlike SliceGPT, we control the dimension of the matrix D on a per-layer basis to achieve our dynamic slice.

4 Experiments

The first step of the process is to evaluate how redundant each one of the layers is using the procedure described above. We have done this for Llama3-8B and Mistral-7B (Jiang et al., 2023) using the full validation split of the pg-19 data-set (Rae et al., 2019). This will give us a LR score for each layer that we then need to process into a slicing pattern. Figure 1 shows an example of a slicing pattern evaluation process for Llama3-8B having a target S_P of 30% and a S_B of 10%.

Furthermore, we evaluate Llama3-8B and Mistral-7B using different Slice Base (S_B) values in increments of 2%. Intuitively as we decrease the base percentage of the layers we have more extreme slicing patterns. In all our experiments we compare with the constant slice as a baseline. We experiment with Slice Percentages (S_P) of 30%, 35%,

Model	Technique	Pruned	Piqa Acc. (↑)	Hellaswag Acc. (↑)	Winogrande Acc. (↑)	Arc Easy Acc. (↑)	Wikitextv2 Perplexity (↓)	Average Acc. (↑)
Llama 3-8B	SliceGPT	30%	59.3%	37.2%	56.4%	**42.9%**	13.37	49.0%
		35%	57.7%	34.1%	54.3%	39.3%	16.58	46.4%
		40%	57.0%	32.4%	51.8%	35.9%	20.69	44.3%
	Dynamic Slicing	30%	**60.4%**	**38.4%**	**58.0%**	42.4%	**12.96**	**49.8%**
		35%	**58.4%**	**36.3%**	**57.2%**	**39.3%**	**15.64**	**47.8%**
		40%	**58.1%**	**34.0%**	**54.4%**	**36.8%**	**19.11**	**45.8%**
Mistral-7B	SliceGPT	30%	62.6%	38.0%	59.7%	51.1%	8.87	52.9%
		35%	58.5%	**35.9%**	**57.6%**	42.8%	10.80	48.7%
		40%	57.1%	**33.6%**	54.1%	38.2%	13.33	45.8%
	Dynamic Slicing	30%	**63.1%**	**38.6%**	**60.2%**	**51.7%**	**8.76**	**53.4%**
		35%	**58.5%**	34.9%	55.7%	**45.8%**	**10.38**	**48.8%**
		40%	**57.9%**	31.9%	**54.1%**	**40.1%**	**12.62**	**46.0%**

Table 1: Comparison of our technique in the smallest perplexity setting against the constant slicing proposed by SliceGPT, bold means higher value in comparison

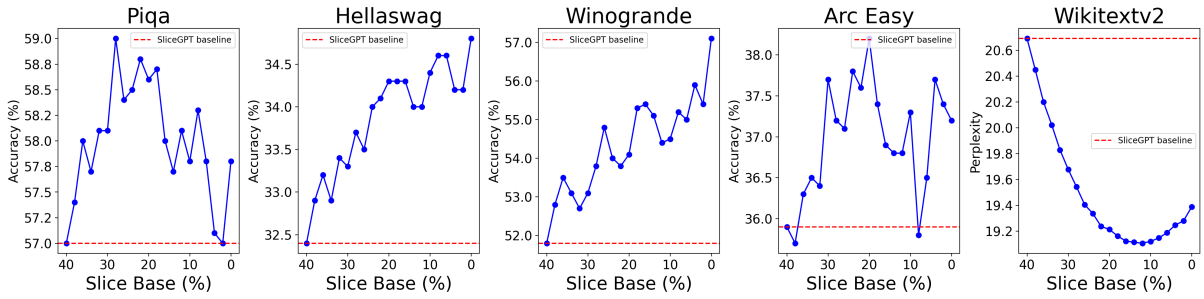

Figure 2: Llama3-8B with 40% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 40% slice.

40%, these representing the percentage of the LLM that we prune. For evaluations, we use: Piqa (Bisk et al., 2019), Hellaswag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), Arc Easy (Clark et al., 2018), and Wikitextv2 (Merity et al., 2016) within the library lm-evaluation-harness (Gao et al., 2023). For more experimental details, please refer to Section A.1 in the Appendix.

5 Results

We will first explore how the accuracy is affected by our dynamic slicing. Figure 2 (presented above), and Figures 3, 4, 5, 6, 7 (presented in Appendix) show the behaviors of Llama3-8b and Mistral-7b on 30%, 35%, 40% pruned percentage with respect to the accuracy on four data sets and perplexity on Wikitextv2. As observed in Figure 2, for the Llama3-8B model, the accuracy is improved across all of the 5 evaluated datasets, and the perplexity decreases by as much as 1.4 while pruning the exact same amount from the model. We also see huge accuracy improvements from 52% to 57% on Winogrande which has a baseline accuracy of 50%. An important finding in all cases is that the perplexity decreases and task accuracy (mostly) increases when we started to decrease the S_B (until a certain point) showcasing LLMs benefit more from dynamic slicing as proposed in our work.

To estimate a good S_B in our proposed dynamic slicing method, we evaluated the accuracy at the S_B point that achieves the minimum perplexity on Wikitextv2, thus using it as a calibration data set. The results shown in Table 1 indicate that our method outperforms SliceGPT on average for all pruning ratios and models explored. We also show even better results using the median accuracy over all S_B values (Table 3) and mean accuracy values (Table 4), leading us to believe that there are even better ways to choose an S_B value than the minimum perplexity.

5.1 Analyses

To highlight strengths of pruning with our dynamic slicing, we also show comparison with ShortGPT (Men et al., 2024) where entire set of layers are pruned or removed. Please note that techniques such as ShortGPT that have shown to be effective,

Model	Technique	Pruned	Piqa Acc. (↑)	Hellaswag Acc. (↑)	Winogrande Acc. (↑)	Arc Easy Acc. (↑)	Wikitextv2 Perplexity (↓)	Average Acc. (↑)
Llama 3-8B	ShortGPT	28.1%	62.0%	32.5%	57.5%	39.4%	2.2×10^4	47.9%
		34.3%	61.2%	34.3%	55.8%	38.1%	4.9×10^4	47.4%
		37.5%	60.0%	34.1%	54.8%	37.5%	1.1×10^5	**46.6%**
	Dynamic Slicing	30%	60.4%	38.4%	58.0%	42.4%	1.3×10^1	**49.8%**
		35%	58.4%	36.3%	57.2%	39.3%	1.5×10^1	**47.8%**
		40%	58.1%	34.0%	54.4%	36.8%	1.9×10^1	45.8%
Mistral-7B	ShortGPT	28.1%	65.0%	39.8%	63.3%	46.6%	1.6×10^2	**53.9%**
		34.3%	57.2%	28.6%	56.1%	30.3%	1.1×10^4	43.1%
		37.5%	55.6%	29.3%	56.7%	30.0%	2.4×10^4	42.9%
	Dynamic Slicing	30%	63.1%	38.6%	60.2%	51.7%	8.7×10^0	53.4%
		35%	58.5%	34.9%	55.7%	45.8%	1.0×10^1	**48.8%**
		40%	57.9%	31.9%	54.1%	40.1%	1.2×10^1	**46.0%**

Table 2: Comparison of our technique with ShortGPT. For each of the pruning rations of our technique, the closest pruning ratio of ShortGPT is reported. Please note that removing 9, 11, and 12 layers completely as in ShortGPT results in 28.1%, 34.3%, and 37.5% pruned ratio respectively. Bold means higher value in comparison.

often provide lesser flexibility in pruning ratio as entire set of layers are removed. For example, removing 9 (least important) layers out of 32 layers results in a pruning ratio of 28.1% as shown in Table 2.

As shown in Table 2, our dynamic slicing with SliceGPT outperforms ShortGPT in majority of the cases even with higher pruning ration i.e., 28.1% vs. 30%, 34.3% vs. 35%, and 37.5% vs. 40%. Especially for higher pruning ratio (i.e., 35% or 40%), our dynamic slicing based SliceGPT approach outperforms ShortGPT with a larger margin across the four classification datasets shown in Table 2. Importantly, removing set of layers completely as in ShortGPT lead to very high perplexity values suggesting high degradation in text generation quality. On the other hand, our proposed dynamic slicing technique with SliceGPT results in exponentially better perplexity score highlighting benefits of pruning only parts of LLM layers instead of removing entire layers.

Additionally, our variable slicing scaled from layer importance can be easily extended and merged with techniques like ShortGPT by simply removing the least important layers completely and slicing moderately important layers using our proposed approach. This hybrid method leveraging the strengths of both techniques, could potentially enhance model efficiency and performance. We leave this for exploration in future work.

6 Conclusions

In conclusion, we propose a novel dynamic slicing strategy that shows considerable improvements in accuracy when compared against a fixed slicing method that prunes the same amount of parameters on average. We also show that layer redundancy is a powerful metric when removing percentages of layers, and also that there is room for improvement, leading to possible future work in the field.

Limitations

While our study introduces significant advancements in the dynamic pruning of Large Language Models, there are several limitations that are worth discussing:

- The effectiveness of our method has been demonstrated predominantly on the Llama3-8B and Mistral-7B models. However, its performance may vary with other architectures, especially those with different layer configurations or learning dynamics.

- We only experiment with one method to choose the S_B value (that gives lowest perplixity) and there can be other methods for estimating S_B which we leave it to more focused future works.

- Limited computational resources have constrained our ability to test on larger models. We believe that exploring the efficacy of our dynamic pruning method on more extensive architectures could provide valuable insights into its scalability and performance.

Ethical Considerations

Ethical considerations are central to our development of a dynamic pruning method for large language models. Our research strives to reduce the computational costs and environmental impact of deploying large-scale models, aligning with the ethical responsibility to promote environmental sustainability and minimize negative consequences such as excessive energy consumption. This not only makes LLMs more accessible but also supports broader societal needs by enabling more efficient processing solutions that respect both individual rights and community values. By enhancing the efficiency of these models, we may enable populations with limited computational resources to harness the power of advanced NLP tools.

Furthermore, our methodology emphasizes the importance of using data sets that are free from harmful content. By using data sets that do not contain harmful data, we aim to ensure that the resulting models avoid biased outputs or content that could reduce their utility in practical applications. Ensuring the integrity and appropriateness of pruned models is essential, as these models often play significant roles in decision-making processes across various sectors. In this context, our approach is designed to be transparent and responsible, providing clear documentation and rigorous evaluation to maintain the reliability and fairness of the models.

Also, we have selected a color scheme that prioritizes accessibility, ensuring the visuals are clear and discernible to individuals with color vision deficiencies. This inclusive approach reflects our commitment to making our research accessible to a wider audience, including those with varying visual abilities.

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

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.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *Preprint*, arXiv:2403.17887.

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

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124.

Cheng-Yu Hsieh, Chun-Liang Li, CHIH-KUAN YEH, Hootan Nakhost, Yasuhisa Fujii, Alex Jason Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Darioush Kevian, Usman Syed, Xingang Guo, Aaron Havens, Geir Dullerud, Peter Seiler, Lianhui Qin, and Bin Hu. 2024. Capabilities of large language models in control engineering: A benchmark study on gpt-4, claude 3 opus, and gemini 1.0 ultra. *arXiv preprint arXiv:2404.03647*.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *Preprint*, arXiv:2403.03853.

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

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

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

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhensu Chen, Xiaopeng Zhang, and Qi Tian. 2023. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

A Appendix

A.1 Experimental details

For all our experiments we used 4 NVIDIA A100 GPUs, with 80GB of VRAM each, and running all of the data sets on one slicing pattern using one NVIDIA A100 takes around 30 minutes, leading to a total time of 10 hours for a plot that has 20 possible S_B values. We use a total of 1000 samples for each data set in all our configurations. Also, calculating the LR score can take upwards of 30-40 minutes per model on 1 NVIDIA A100 GPU, but that only has to be computed once.

A.2 Tables with mean/median results

For a detailed evaluation, Table 3 compares our technique in the median setting against the constant slicing proposed by SliceGPT, and Table 4 provides a comparison in the average setting.

A.3 Llama3-8B and Mistral-7B in various scenarios

Visual representations of our experiments are shown across several figures: Figure 3 and Figure 4 illustrate the performance of Llama3-8B with 30% and 35% of the network sliced, respectively, comparing it to the baseline accuracy achieved by SliceGPT. Similarly, Figures 5, 6, and 7 display the results for Mistral-7B with 30%, 35%, and 40% of the network sliced, again benchmarked against SliceGPT's constant slice accuracies. In each figure, the red line denotes the baseline accuracy set by SliceGPT for the respective slicing percentages.

Model	Technique	Pruned	Piqa Acc. (↑)	Hellaswag Acc. (↑)	Winogrande Acc. (↑)	Arc Easy Acc. (↑)	Wikitextv2 Perplexity (↓)	Average Acc. (↑)
LLaMa-3-8B	SliceGPT	30%	59.3%	37.2%	56.4%	42.9%	13.37	49.0%
		35%	57.7%	34.1%	54.3%	39.3%	16.58	46.4%
		40%	57.0%	32.4%	51.8%	35.9%	20.69	44.3%
	Dynamic Slicing	30%	**60.5%**	**38.1%**	**57.4%**	**43.0%**	**13.07**	**49.7%**
		35%	**58.7%**	**35.8%**	**56.2%**	**39.6%**	**15.75**	**47.6%**
		40%	**58.1%**	**34.2%**	**55.0%**	**37.2%**	**19.21**	**46.1%**
Mistral-7B	SliceGPT	30%	62.6%	38.0%	**59.7%**	51.1%	8.87	52.9%
		35%	58.5%	35.9%	**57.6%**	42.8%	10.80	48.7%
		40%	57.1%	33.6%	54.1%	38.2%	13.33	45.8%
	Dynamic Slicing	30%	**62.6%**	**38.6%**	59.5%	**52.4%**	**8.80**	**53.4%**
		35%	**59.2%**	**36.0%**	56.9%	**45.1%**	**10.42**	**49.3%**
		40%	**57.9%**	**34.3%**	**54.1%**	**40.0%**	**12.68**	**46.5%**

Table 3: Comparison of our technique in the median setting against the constant slicing proposed by SliceGPT, bold means higher value in comparison

Model	Technique	Pruned	Piqa Acc. (↑)	Hellaswag Acc. (↑)	Winogrande Acc. (↑)	Arc Easy Acc. (↑)	Wikitextv2 Perplexity (↓)	Average Acc. (↑)
LLaMa-3-8B	SliceGPT	30%	59.3%	37.2%	56.4%	42.9%	13.37	49.0%
		35%	57.7%	34.1%	54.3%	39.3%	16.58	46.4%
		40%	57.0%	32.4%	51.8%	35.9%	20.69	44.3%
	Dynamic Slicing	30%	**60.6%**	**38.0%**	**57.4%**	**43.0%**	**13.11**	**49.8%**
		35%	**58.6%**	**35.6%**	**55.8%**	**39.7%**	**15.9**	**47.4%**
		40%	**58.1%**	**34.0%**	**54.6%**	**37.1%**	**19.34**	**46.0%**
Mistral-7B	SliceGPT	30%	62.6%	38.0%	**59.7%**	51.1%	8.87	52.9%
		35%	58.5%	**35.9%**	**57.6%**	42.8%	10.80	48.7%
		40%	57.1%	33.6%	54.1%	38.2%	13.33	45.8%
	Dynamic Slicing	30%	**62.6%**	**38.1%**	59.5%	**52.3%**	**8.81**	**53.1%**
		35%	**59.0%**	35.7%	57.0%	**44.8%**	**10.50**	**49.1%**
		40%	**57.8%**	**33.6%**	**54.2%**	**39.6%**	**12.76**	**46.3%**

Table 4: Comparison of our technique in the average setting against the constant slicing proposed by SliceGPT, bold means higher value in comparison

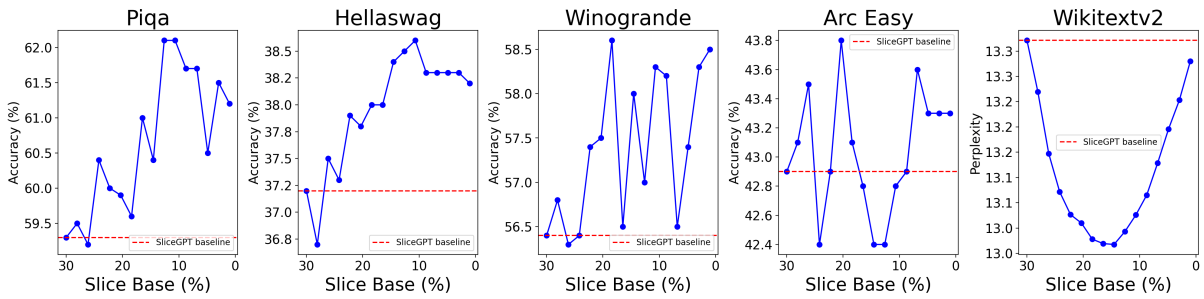

Figure 3: Llama3 8B with 30% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 30% slice.

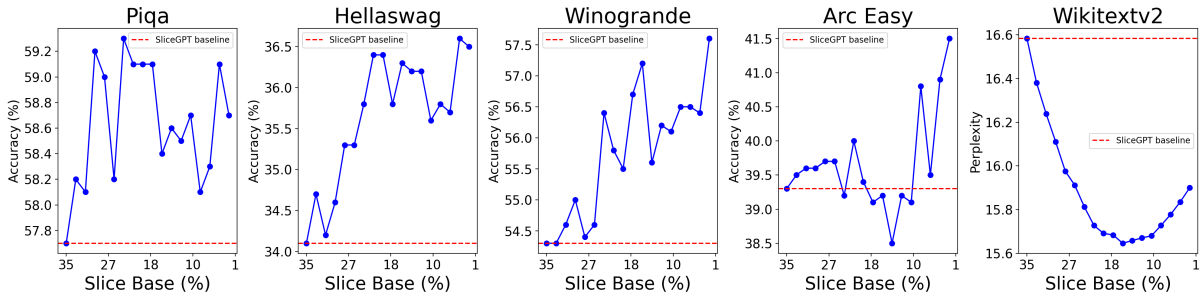

Figure 4: Llama3 8B with 35% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 35% slice.

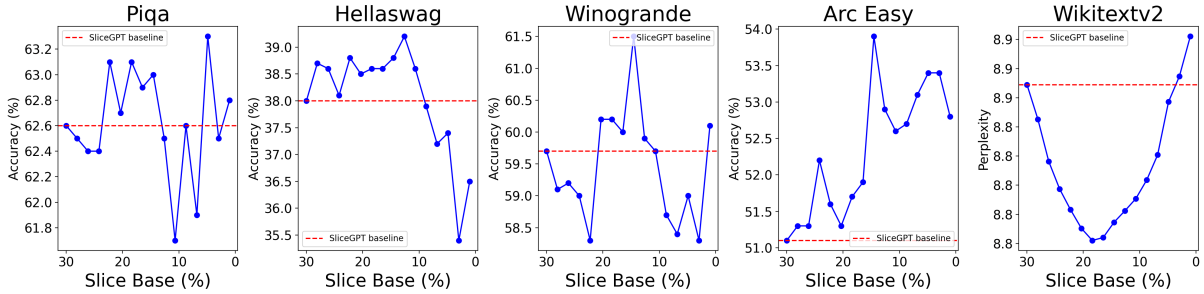

Figure 5: Mistral 7B with 30% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 30% slice.

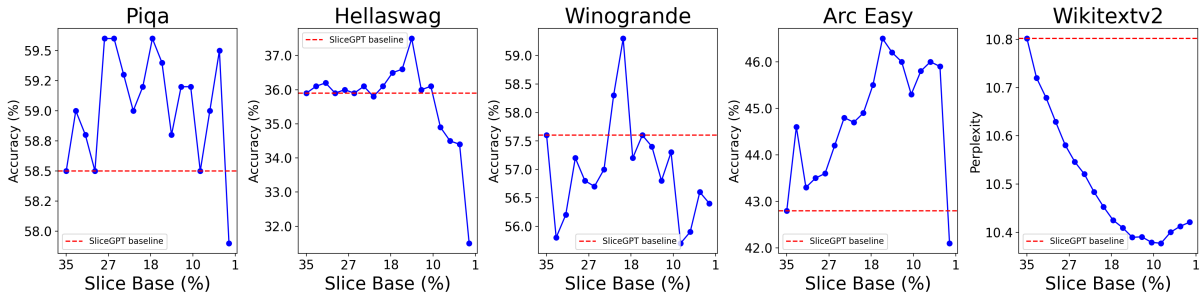

Figure 6: Mistral 7B with 35% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 35% slice.

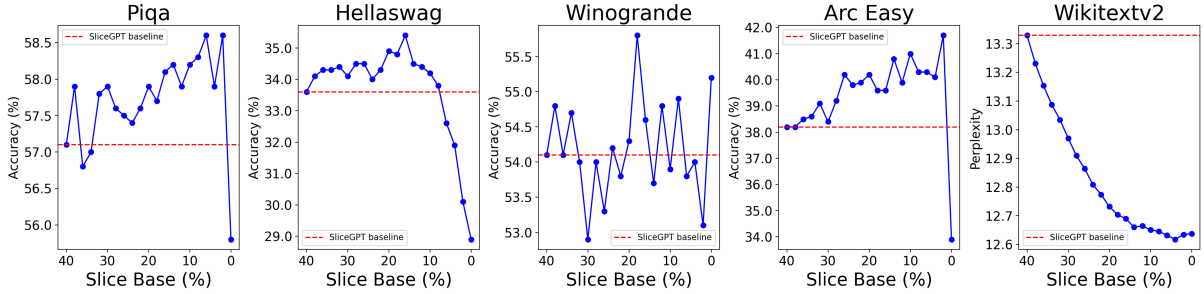

Figure 7: Mistral 7B with 40% of the network sliced on average, the red line is the baseline accuracy achieved by SliceGPT with a constant 40% slice.