ZEROTH-ORDER ADAPTIVE NEURON ALIGNMENT BASED PRUNING WITHOUT RE-TRAINING

Elia Cunegatti, Leonardo Lucio Custode & Giovanni Iacca
Department of Information Engineering and Computer Science
University of Trento, Italy
{elia.cunegatti, giovanni.iacca}@unitn.it

ABSTRACT

Network pruning focuses on computational techniques that aim to reduce a given model's computational cost by removing a subset of its parameters while having minimal impact on performance. Throughout the last decade, the most widely used pruning paradigm has been pruning and re-training, which nowadays is inconvenient due to the vast amount of pre-trained models, which are in any case too expensive to re-train. In this paper, we exploit functional information from dense pre-trained models, i.e., their activations, to obtain sparse models that maximize the activations' alignment w.r.t. their corresponding dense models. Hence, we propose NEURONAL, a *top-up* algorithm that can be used on top of any given pruning algorithm for LLMs, which modifies the block-wise and row-wise sparsity exploiting information from both the dense model and its sparse version to maximize the *neuron alignment* among activations. Differently from existing methods, our approach adaptively selects the best hyperparameters for the block-wise and row-wise sparsity ratios w.r.t. the model and the desired sparsity, and requires *no re-training*. We test our method over four LLM families, three sparsity ratios, and ten language tasks (three language modeling and seven zero-shot datasets), showing how it consistently outperforms the latest state-of-the-art methods in terms of performance-runtime trade-off. The code is available at https://github.com/eliacunegatti/NeuroAL.

1 INTRODUCTION

In recent times, Large Language Models (LLMs) have shown incredible performance over several language tasks Wei et al. (2022); Min et al. (2023); Chang et al. (2024). However, their performance usually improves with their sizes (i.e., the number of trainable parameters), which in turn is proportional to the computational cost of training and then using such models. One way to reduce this cost is through *network pruning*, i.e., applying algorithms that remove parameters while minimizing performance degradation. This approach has been extensively studied on Convolutional Neural Networks (CNNs) Frankle & Carbin (2019); Lee et al. (2019); Wang et al. (2020); Evci et al. (2020), but nowadays the focus has shifted towards pre-trained models Touvron et al. (2023a;b); Jiang et al. (2023).

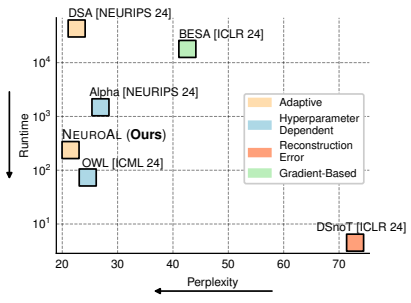

Figure 1: Perplexity vs. Runtime (seconds) trade-off among different *top-up* algorithms.

This shift has required a change of paradigm in pruning techniques: in fact, while in CNNs the main paradigm is iterative pruning (with re-training) Frankle & Carbin (2019), with pre-trained models (such as LLMs) in most cases it is not possible to fully re-train such models, because (1) training data are often not accessible, and (2) full re-training would be anyway too expensive. This calls for "exploiting" as much as possible the information contained in a pre-trained model to obtain a performant sparse version of it, using weight's information Jaiswal et al. (2024), activations Sun et al. (2023; 2024), or reconstruction error Frantar & Alistarh (2023), without the need for re-training.

More recently, a new category of pruning algorithms, which we may call *top-up* algorithms (i.e., methods that can be applied on top of a given pruning algorithm for LLMs), has emerged, aiming at further improving pruning performance. Such approaches can be divided into two categories: those that minimize the reconstruction error Guo et al. (2024); Xu et al. (2024); Zhang et al. (2024), and those that impose non-uniform sparsity distribution modifying the block-wise sparsity Yin et al. (2024); Lu et al. (2024); Li et al. (2024).

Contributions In this paper, we first analyze the major limitation of current *top-up* algorithms. To do so, we carefully analyze the state-of-the-art top-up methods highlighting their limitations in terms of sensitivity to hyperparameters and large pruning runtime. Leveraging this knowledge, we introduce a new top-up method, called NEURONAL. The algorithm consists of a two-step approach that re-distributes the block-wise sparsity, i.e., the sparsity among Transformer blocks, and the row-wise sparsity, i.e., the sparsity for each row of a given layer's matrix, maximizing a metric which exploits information from both the dense and sparse model, namely the *neuron alignment* between dense and sparse activations. NEURONAL does not require the user to specify any hyperparameter-tuning, as it automatically selects the most-performing values from a suitable set, hence adapting to the underlying model and the target sparsity. Another advantage is that the *neuron alignment* only requires the computation of the activations of the dense and sparse models, which reduces the computation budget required, compared to other *top-up* approaches.

2 CURRENT LIMITATIONS OF TOP-UP ALGORITHMS

We analyze the sensitivity of the hyperparameters used by OWL, namely λ and M, and by AlphaPruning, namely ϵ. Concerning OWL, the first hyperparameter is used to set how much the sparsity can vary across blocks (i.e., $[s-\lambda, s+\lambda]$) while keeping the overall sparsity fixed as s. The second hyperparameter, M, defines the outliers' threshold: namely, for each block, the number of outliers is computed as the number of activations that are M times greater than the block's activations' mean. For AlphaPruning, instead, a hyperparameter called ϵ is used and manually tuned to set two tunable hyperparameters (s_1, s_2) that control the sparsity across blocks. We test the sensitivity of OWL and AlphaPruning to their hyperparameters, using three different sparsity ratios, two LLMs, and Wanda as the underlying pruning algorithm. Fig. 2 displays the perplexity on WikiText2 of the different hyperparameter settings obtained with OWL (first two rows) and AlphaPruning (last row); the gray square corresponds to the best *a-posteriori* hyperparameter selection. It can be seen that no single hyperparameter value achieves the best performance in all settings, which entails that careful tuning is required for these approaches to be effective.

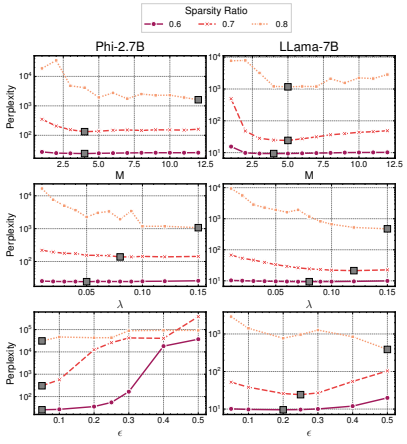

Figure 2: Perplexity for various hyperparameter settings of OWL (M,λ) and AlphaPruning (ϵ).

Another main limitation of some of the current approaches for non-uniform distribution is their computational runtime. This holds mainly BESA Xu et al. (2024) and DSA Li et al. (2024). On LLama-7B, the first approach, which relies on gradient information, requires $\sim$5 hours to find the best non-uniform distribution configuration. On the other hand, DSA requires $\sim$12 hours to find the distribution configuration. [1]

3 METHODOLOGY

Our proposed method is based on the idea of combining the concept of *neuron alignment*, which requires no *a priori* definition of outliers (hence no M parameter, as in OWL), with that of *adaptivity*, to remove the need for manually tuning λ and ϵ. The method takes as input both $\mathcal{D}$ and its sparse version $\mathcal{S}$ generated by $\mathcal{P}$ with sparsity ratio s, and uses a small calibration data C_λ to make a

[1] BESA and DSA are not included in these experiments due to their large runtime. Testing them on all combinations of sparsities and pruning algorithms is unfeasible with our GPU resources.

forward pass on both models, to retrieve the dense and sparse activations, respectively $\mathcal{A}_\mathcal{D}$ and $\mathcal{A}_\mathcal{S}$. The main idea behind NEURONAL is to maximize the neuron alignment by firstly modifying the vector of sparsity ratios for all blocks ($\mathbf{s}^\mathcal{B}$) and then for all rows ($\mathbf{s}^r$), where each row corresponds to the layer's weight matrix $W^{\ell_i^j}$ (for each layer ℓ_i^j in $\mathcal{B}_i$), where $W^{\ell_i^j} \in \mathbb{R}^{r \times m}$. The main strength of this approach is that it does not require any weight update nor gradient information, but just a block- and row-wise sparsity reallocation and mask update, using the same scoring criteria of $\mathcal{P}$.

However, as tested in the previous observational study, finding the best block/row-wise sparsity requires defining a factor λ to control the block/row-wise sparsity difference between consecutive blocks/rows while ensuring the desired global sparsity. While OWL requires λ to be set *a priori*, NEURONAL automatically selects, from a suitable set of values, the best λ for each combination of $\mathcal{D}$, $\mathcal{P}$ and s, yielding an adaptive top-up method. The only constraint we set is that we use a linear sparsity schedule over λ for the block-wise step, demonstrated to be effective in our empirical study in Section 3.3.

3.1 BLOCK-WISE SPARSITY RATIO

The first step concerns the block-wise redistribution over the whole model. Our method takes as input the dense and sparse models ($\mathcal{D}$ and $\mathcal{S}$), the target sparsity (s), the calibration data C_λ, and a set of λ parameters (λ^{set}). Then, it computes a set of $|\lambda^{\text{set}}|$ vectors of block-wise sparsity values $\mathbf{s}^\mathcal{B}_{set} = \{\mathbf{s}^\mathcal{B}_{\lambda_1}, \mathbf{s}^\mathcal{B}_{\lambda_2}, \dots, \mathbf{s}^\mathcal{B}_{\lambda_{|\lambda^{\text{set}}|}}\}$, where each element $\mathbf{s}^\mathcal{B}_{\lambda_k}$ indicates a vector of block-wise sparsity values obtained with a linear schedule in $[s - \lambda_k, s + \lambda_k]$. For each $\mathbf{s}^\mathcal{B}_{\lambda_k}$, we then forward the calibration data C_λ through the model, and calculate the corresponding neuron alignment:

$$neur_{al} = \sum_{\mathcal{B}_i} \sum_{\ell_i^j} \frac{\left\| \tilde{A}^j_D - \tilde{A}^j_{\mathcal{S}_{\left(\mathbf{s}^\mathcal{B}_{\lambda_k}\right)}} \right\|_2}{|\tilde{A}^j_D|} \tag{1}$$

where $\tilde{A}$ means that the activations are normalized to sum up to one. Then, we select $\left(\mathbf{s}^\mathcal{B}_{set}\right)^*$, i.e., the λ parameters per block that minimize Eq. (1). Finally, we update the block-wise sparsity with the selected $\left(\mathbf{s}^\mathcal{B}_{set}\right)^*$, thus obtaining a sparsified model $\mathcal{S}_\mathcal{B}$.

3.2 ROW-WISE SPARSITY RATIO

The second step is complementary to the previous one, but in this case, the sparsity is modified w.r.t. the rows of each layer. In this case, for each layer ℓ_i^j (i.e., for each $W^{\ell_i^j} \in \mathbb{R}^{r \times m}$) we redistribute the sparsity across the r rows. Also in this case the λ parameters are critical for deciding how to control the sparsity difference between consecutive rows. We take our sparse model obtained with the block-wise redistribution ($\mathcal{S}_\mathcal{B}$) and, for each layer ℓ_i^j, we compute different row-wise sparsity values obtaining $\mathbf{s}^s_{set} = \{\mathbf{s}^r_{\lambda_1}, \mathbf{s}^r_{\lambda_2}, \dots, \mathbf{s}^r_{\lambda_{|\lambda^{\text{set}}|}}\}$, where each $\mathbf{s}^r_{\lambda_k}$ indicates a vector of row-wise sparsity in $[s - \lambda_k, s + \lambda_k]$, where each element is inversely proportional to the alignment of the corresponding row. In this case, we select in $\mathbf{s}^s_{set}$ the row-wise vector $\left(\mathbf{s}^r_{set}\right)^*$ that minimizes Eq. (1).

3.3 NON-UNIFORM BLOCK-WISE SPARSITY DISTRIBUTION

The sparsity values for each block in $\mathbf{s}^\mathcal{B}_{\lambda_k}$ have been set such as given the sparsity window $[s - \lambda, s + \lambda]$, these schedules work by redistributing the sparsity across blocks in a monotonically *linear* way (i.e., the sparsity of block i is always larger than the sparsity of layer $i - 1, \forall i > 1$). We select this sparsity schedule for two main reasons: (1) as shown below such a straightforward sparsity schedule is already able to achieve similar results w.r.t. state-of-the art approaches, and (2) to align with the latest discovering in the literature of structured pruning where is consistently demonstrate how deeper blocks are redundant and can be removed with marginal performance degradation Gromov et al. (2024); Men et al. (2024); Kim et al. (2024). We motivated the choice of a linear schedule by testing three straightforward non-uniform sparsity schedules (namely *linear*, *exponential*, and *logarithmic*), which do not require any block scoring for

Table 1: Performance improvement w.r.t. uniform distribution averaged across three different datasets (WikiText2, C4, and PTB) using Wanda.

Sparsity	Model	Schedule			
		OWL	Exp	Log	Linear
60%	Phi-2.7B	+3.4%	+2.4%	**+7.8%**	+7.7%
	LLama-1 7B	+16.5%	+3.4%	+15.7%	**+18.1%**
70%	Phi-2.7B	+45.8%	+47.7%	+44.9%	**+52.5%**
	LLama-1 7B	**+66.8%**	+28.2%	+53.9%	+63.5%
80%	Phi-2.7B	+87.8%	**+89.3%**	+55.7%	+82.8%
	LLama-1 7B	**+81.5%**	+63.6%	-4.4%	+68.1%
Mean		**+50.3%**	+39.1%	+28.9%	+48.8%

sparsity allocation. Table 1 highlights how non-uniform sparsity schedules, without any block-based scoring, lead to a performance improvement close to OWL's. Overall, the *linear* schedule turns out to be the most reliable one since it does not show oscillations in performance across the different sparsity ratios.

4 EXPERIMENTS

Concerning the Language Modeling datasets, the numerical results in terms of perplexity computed over the three Language Modeling datasets at 70% sparsity are shown in Table 3. It can be seen how NEURONAL is able in almost all cases to outperform all the other baselines by a large margin. In no case NEURONAL performs worse w.r.t. the uniform distribution. The only model on which NEURONAL is not the best top-up algorithm for all pruning algorithms is OPT. In all other cases, NEURONAL outperforms OWL and AlphaPruning for all models and pruning algorithms.

Table 2: Accuracy on the seven Zero-Shot Tasks using Wanda as pruning algorithm.

Sparsity	Top-Up	Model				
		Phi-2.7B	LLama1 7B	LLama2 7B	Mistral-7B	OPT-6.7B
60%	Uniform	**52.36**	50.39	49.97	51.03	**46.24**
	DSnoT	49.36	49.12	48.83	50.51	45.75
	OWL	51.48	50.93	51.68	**52.49**	46.05
	AlphaPruning	43.14	51.08	51.21	49.87	44.50
	NEURONAL	52.04	**51.41**	**51.99**	52.09	46.10
70%	Uniform	39.89	36.90	34.37	36.85	36.31
	DSnoT	38.76	36.26	34.09	36.53	36.35
	OWL	41.20	43.31	40.57	38.77	38.77
	AlphaPruning	35.02	44.08	42.03	39.05	**39.53**
	NEURONAL	**41.87**	**44.57**	**43.10**	**41.56**	38.86
80%	Uniform	36.21	31.66	31.81	32.48	33.34
	DSnoT	32.73	31.78	32.27	32.14	**35.12**
	OWL	36.26	31.43	32.48	32.02	32.10
	AlphaPruning	30.74	35.34	32.09	32.29	32.16
	NEURONAL	**37.36**	**36.31**	**32.74**	**33.08**	33.05

As for the Zero-Shot tasks, the numerical results are shown in Table 2. We display only the mean across the seven Zero-Shot tasks. Again, NEURONAL turns out to outperform in the majority of cases all the baselines. In 10 cases out of 15 (w.r.t. the mean accuracy across all tasks), NEURONAL is the one that reaches the best performance and in 4 cases the second best.

Table 3: Perplexity on the three Language Modeling datasets computed over five different LLMs for four different top-up algorithms (Uniform, DSnoT, OWL, and NEURONAL) on three pruning algorithms (Magnitude, MULTIFLOW, and Wanda) at 70% sparsity.

Algorithm	Top-Up	Phi-2.7B			LLama-1 7B			LLama-2 7B			Mistral 7B			OPT 6.7B		
		WikiText2	C4	PTB	WikiText2	C4	PTB	WikiText2	C4	PTB	WikiText2	C4	PTB	WikiText2	C4	PTB
Magnitude	Uniform	764.6	384.4	983.9	2.53e4	2.25e4	3.26e4	1.42e5	1.02e4	2.02e6	221.9	232.9	748.7	1.00e4	5.39e3	6.54e3
	DSnoT	539.0	258.0	656.2	1.02e7	2.77e6	4.99e7	1.31e8	2.90e7	2.25e8	192.7	189.9	566.2	**6.16e3**	**3.93e3**	**4.36e3**
	OWL	419.6	242.7	358.5	1.20e4	6.58e3	5.39e4	3.39e5	1.24e4	3.28e6	111.7	124.2	545.5	1.57e4	8.48e3	9.67e3
	AlphaPruning	2.52e4	1.60e4	2.34e4	424.9	391.5	5.08e4	3.37e3	3.60e3	1.73e5	91.3	106.5	717.1	1.22e4	7.22e3	7.51e3
	NEURONAL	**281.7**	**180.9**	**321.1**	**231.8**	**219.9**	**4.46e3**	**155.8**	**264.8**	**2.61e3**	**46.5**	**43.1**	**612.8**	2.11e4	1.07e4	1.09e4
MULTIFLOW	Uniform	388.4	298.8	610.8	80.9	71.9	172.4	60.0	58.8	1.26e3	9.37e2	6.56e2	2.06e3	9.44e2	1.25e3	843.1
	DSnoT	325.5	261.9	328.8	67.6	65.0	114.7	66.6	75.8	6.89e2	**57.4**	**63.3**	**2.65e2**	241.8	153.3	263.9
	OWL	197.9	141.3	293.9	25.1	25.8	78.9	29.2	31.0	5.47e2	329.0	7.64e2	1.72e3	240.9	495.6	337.8
	AlphaPruning	1.22e5	8.99e4	9.52e4	32.2	35.2	103.8	31.3	34.0	287.3	230.8	292.8	1.72e3	**133.8**	**63.7**	**153.9**
	NEURONAL	**105.4**	**87.1**	**179.5**	**20.7**	**21.2**	**46.2**	**22.1**	**23.9**	**265.5**	202.5	334.7	1.41e3	209.7	83.7	202.1
Wanda	Uniform	227.6	182.7	346.2	85.1	86.2	157.0	78.0	81.0	599.3	60.7	73.6	298.3	157.5	260.1	209.2
	DSnoT	221.9	172.6	257.6	72.9	76.0	121.0	76.1	85.7	491.8	81.3	79.9	304.8	191.4	173.3	182.6
	OWL	132.7	116.2	183.7	24.6	27.3	61.2	30.5	36.6	333.7	41.0	51.8	253.5	**54.4**	**69.7**	**100.7**
	AlphaPruning	4.22e4	3.05e4	2.23e4	26.9	31.1	77.4	32.0	37.7	273.8	39.4	49.8	286.8	93.8	53.7	120.9
	NEURONAL	**88.3**	**77.7**	**129.5**	**21.5**	**23.2**	**44.2**	**24.0**	**27.4**	**207.0**	**28.8**	**33.7**	**232.0**	172.6	84.0	182.7

Runtime vs. Perplexity NEURONAL provides a good trade-off between performance and runtime. In Table 4, we show for all baselines, the runtime in seconds required to obtain the non-uniform sparsity distribution for the given model (in this case LLama-7B V1) as well as the performance computed as the perplexity over WikiText2. The results confirm how NEURONAL can achieve, in 3 out of 4 cases, the best results in terms of perplexity while maintaining a low computational budget. In terms of runtime, the only comparable methods are DSnoT and OWL, compared to which however NEURONAL achieves better performance. On the other hand, DSA is the closest in terms of perplexity to NEURONAL, while requiring four orders of magnitude more time to obtain the best sparsity distribution. Overall, the performance-runtime trade-off of NEURONAL improves when increasing the sparsity ratio.

Table 4: Runtime (seconds) vs. perplexity trade-off comparison among different *top-up* algorithms over LLama-7B pruned at different sparsity ratios using Wanda.

Metric	Top-up pruning algorithms						
	Uniform	DsNoT	OWL	BESA	DSA	AlphaPruning	NEURONAL
Runtime	-	4.5s	73.3s	$\sim 1.8 \times 10^4$ s	$\sim 4.3 \times 10^4$ s	1479.4s	237.1s
Perplexity @ 65%	20.9	19.1	13.1	18.5	**12.6**	14.0	12.8
Perplexity @ 70%	85.1	72.9	24.6	42.6	22.6	26.9	**20.7**
Perplexity @ 75%	927.4	646.7	152.5	257.9	103.3	110.2	**61.2**
Perplexity @ 80%	5.22e3	3.71e3	986.5	2.21e3	736.81	768.4	**302.8**

5 CONCLUSION

In this paper, we proposed NEURONAL, a new approach to prune LLMs based on the neuron alignment between sparse and dense activations. The main novelty of our approach is that it exploits

information from both the dense and the sparse models while also being adaptive since it is designed to automatically select the best hyperparameters for a given model, pruning algorithm, and target sparsity. Throughout extensive experiments, we showed how our approach outperforms, in most cases, the latest state-of-the-art methods both on Language Modeling datasets and Zero-Shot tasks, with different LLM families and sparsity ratios.

REFERENCES

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*, 2019.

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

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

Song Guo, Fan Wu, Lei Zhang, Xiawu Zheng, Shengchuan Zhang, Fei Chao, Yiyu Shi, and Rongrong Ji. EBFT: Effective and Block-Wise Fine-Tuning for Sparse LLMs. *arXiv preprint arXiv:2402.12419*, 2024.

Ajay Jaiswal, Shiwei Liu, Tianlong Chen, Zhangyang Wang, et al. The emergence of essential sparsity in large pre-trained models: The weights that matter. *Advances in Neural Information Processing Systems*, 36, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened LLaMA: A Simple Depth Pruning for Large Language Models. *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models (ME-FoMo)*, 2024.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-Shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations*, 2019.

Lujun Li, Peijie Dong, Zhenheng Tang, Xiang Liu, Qiang Wang, Wenhan Luo, Wei Xue, Qifeng Liu, Xiaowen Chu, and Yike Guo. Discovering Sparsity Allocation for Layer-wise Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*, 2024.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. AlphaPruning: Using Heavy-Tailed Self Regularization Theory for Improved Layer-wise Pruning of Large Language Models. In *Advances in Neural Information Processing Systems*, 2024.

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

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2023.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive Activations in Large Language Models. In *Conference on Language Modeling*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *International Conference on Learning Representations*, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. BESA: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *International Conference on Learning Representations*, 2024.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, AJAY KUMAR JAISWAL, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *International Conference on Machine Learning*. PMLR, 2024.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse LLMs. In *International Conference on Learning Representations*, 2024.