STADE: Standard Deviation as a Pruning Metric

Diego Coello de Portugal Mecke[1,2], **Haya Alyoussef**[1], **Maximilian Stubbemann**, **Ilia Koloiarov**[1,2], **Tom Hanika**[1], **Lars Schmidt-Thieme**[1,2]

[1] ISMLL, University of Hildesheim
[2] VWFS DARC

Abstract

Recently, Large Language Models (LLMs) have become very widespread and are used to solve a wide variety of tasks. To successfully handle many of these tasks, LLMs require longer training times and larger model sizes. This makes LLMs ideal candidates for pruning methods that reduce computational demands while maintaining performance. Previous methods require a retraining phase after pruning to maintain the original model's performance. However, state-of-the-art pruning methods, such as Wanda, prune the model without retraining, making the pruning process faster and more efficient. Building upon Wanda's work, this study provides a theoretical explanation of why the method is effective and leverages these insights to enhance the pruning process. Specifically, a theoretical analysis of the pruning problem reveals a common scenario in Machine Learning where Wanda is the optimal pruning method. Furthermore, this analysis reveals cases where Wanda is no longer optimal. To tackle those cases, we develop a new method, *STADE*, based on the standard deviation of the input. From a theoretical and empirical standpoint, *STADE* demonstrates better generality across different scenarios. Finally, extensive experiments on Qwen, Llama and Open Pre-trained Transformers (OPT) models validate these theoretical findings, showing that depending on the training conditions, Wanda's optimal performance varies as predicted by the theoretical framework. These insights contribute to a more robust understanding of pruning strategies and their practical implications.

Code — https://github.com/Coello-dev/STADE/tree/main

Introduction

Large Language Models (LLMs) (Radford et al. 2018, 2019; Brown et al. 2020) have revolutionized not only the field of Natural Language Processing (NLP) but also numerous real-world applications that affect everyday life. Their ability to generate coherent text, perform complex reasoning, and support a variety of conversational and decision-making tasks has led to widespread adoption in both research and industry. With the advent of increasingly autonomous systems (Durante et al. 2024; junyou li et al. 2024; Wu et al. 2024), these models now assist with tasks ranging from content creation and translation to automated customer support and strategic decision making.

Despite these impressive capabilities, LLMs are notorious for their substantial computational requirements (Kaplan et al. 2020a). The high memory footprint, extensive processing power, and significant energy consumption often limits their deployment on devices with limited resources, such as mobile phones or embedded edge devices. In addition, the large-scale training of these models contributes to increased operational costs and a non-negligible environmental impact. Consequently, the drive to reduce the computational and storage demands of LLMs has become a central focus in the field (Sevilla et al. 2022).

To mitigate these computational challenges, a variety of approaches have been explored. One prominent strategy involves reducing the storage requirements of model weights through *quantization* (Ma et al. 2024; Wu et al. 2020). Quantization techniques lower the numerical precision of weights and activations, resulting in reduced memory usage and accelerated inference speeds, often with minimal degradation in performance. Another effective approach is to remove unimportant weight parameters through *pruning* (LeCun, Denker, and Solla 1989). Pruning methods seek to eliminate redundancies in the network by removing weights that contribute little to overall model performance, thereby decreasing both the computational load and the inference latency.

Pruning techniques can be applied during training (Sanh, Wolf, and Rush 2020) or after the model has been fully trained, in what is known as *post-training pruning* (Ashkboos et al. 2024). The latter approach is particularly appealing when the goal is to adapt a pre-trained model for deployment on resource-constrained devices, as the main challenge is not the training process but rather fitting the model into a limited hardware environment. Although some post-training pruning strategies involve costly retraining steps (Agarwal et al. 2024; Xu et al. 2024), previous studies (Sun et al. 2024; Frantar and Alistarh 2023) have demonstrated that a model can maintain a large fraction of its original performance even when up to 50% of its weights are pruned without any retraining.

A notable pruning method is Wanda (Sun et al. 2024), which employs a simple yet effective strategy based on the L_2-*loss* to guide weight removal. Despite its empirical success, the fundamental reason for the superior performance of the L_2-*loss* over alternative norms (e.g., L_1 or L_∞) was neither formally analyzed or fully understood. As noted in the original paper: *"We find that l_2 norm tends to work better than other norm functions (e.g., l_1 and l_∞) in measuring* *activation magnitudes. This is possibly because l_2 norm is generally a smoother metric"* (Sun et al. 2024). Such observations have motivated deeper theoretical investigations into pruning criteria.

This work aims to provide a comprehensive analysis of the pruning problem. The contributions are as follows:

- A theoretical analysis of the pruning problem is presented, revealing a characterization of machine learning scenarios where *Wanda* emerges as the optimal pruning method.
- The analysis is extended to cases where *Wanda*'s approach is suboptimal, thereby motivating the development of a new method, *STADE.*
- Multiple experiments with different LLM model families validate empirically the theoretical analysis.
- Additionally, an ablation of layer-specific characteristics demonstrates that different pruning metrics can yield better performance when applied selectively across different layers of a model. To the best of our current knowledge, this is the first work to apply distinct pruning metrics to different layers, resulting in improved overall pruning effectiveness.

Extensive experiments have been performed across multiple models and configurations to validate the theoretical insights and assess the performance of the proposed *STADE* method. The experiments evaluate perplexity, on different pruning metrics and on different layers for various models, and reveal that the impact of pruning is highly dependent on the statistical properties of the input at each layer. These findings offer valuable guidance for future research in model compression and the efficient deployment of LLMs on consumer devices.

Related Work

The study of sparse subnetworks within large neural networks has been an area of intense investigation, particularly following the introduction of the *Lottery Ticket Hypothesis* (Frankle and Carbin 2019). This hypothesis proposes that within a randomly initialized neural network there exist subnetworks (or "winning tickets") that, when trained in isolation, can achieve performance on par with the full network. Subsequent investigations (Morcos et al. 2019; Frankle et al. 2020) have further elucidated the generalization capabilities and connectivity properties of these subnetworks, providing a theoretical basis for pruning methods.

Pruning strategies have evolved significantly over the past decade. Early methods relied on simple heuristics such as magnitude-based pruning (Zhu and Gupta 2017), which removes weights with the smallest absolute values under the assumption that these contribute least to network performance. This basic approach laid the groundwork for more sophisticated techniques that consider additional information about the network. For instance, the work in (Han et al. 2015) utilized the L_2 norm to evaluate the importance of weights, demonstrating that many redundant parameters could be pruned without significant loss in accuracy.

Advancements in pruning have also led to the development of methods that incorporate second-order information. The Optimal Brain Surgeon (OBS) algorithm (Dong, Chen, and Pan 2017), for example, leverages the Hessian matrix of the loss function to estimate the impact of removing individual weights. Although OBS provides more refined pruning decisions, its high computational complexity has restricted its practical application in large-scale models.

More recent approaches have shifted focus to dynamic pruning strategies that are integrated into the training process (Sanh, Wolf, and Rush 2020; Chen et al. 2021). These methods progressively reduce the number of active parameters during training, often resulting in models that are sparser and more computationally efficient. However, such strategies may conflict with the scaling laws observed for LLMs (Kaplan et al. 2020b), where performance improvements are closely tied to increases in model size, computational resources, and data availability. As a consequence, post-training pruning techniques have emerged as a pragmatic solution for adapting large pre-trained models to resource-limited environments.

A wide range of post-training pruning techniques has been proposed in recent years. Some methods, such as LoRA-based pruning (Zhou et al. 2024), incorporate low-rank adaptations to guide the pruning process. However, retraining the pruned model often incurs significant computational overhead. Others, like SparseGPT (Frantar and Alistarh 2023), use Hessian-based metrics to carefully select which weights to remove, and adjusting the remaining parameters accordingly, thereby preserving critical network functionality. Additionally, strategies that minimize local reconstruction errors within individual blocks (Agarwal et al. 2024; Bai et al. 2024) or layers (Hubara et al. 2021b) of Transformer-based architectures have been investigated, underscoring the notion that different layers may require tailored pruning criteria. Some layer-wise pruning techniques employ structured sparsity, assigning a learned importance weight to each matrix, thereby determining its optimal sparsity level (Li et al. 2024). Others adopt a block-wise grouping strategy, optimizing sets of layers collectively (Xu et al. 2024) to balance sparsity and accuracy.

A central aspect of all pruning methodologies is the selection of an appropriate pruning metric that accurately distinguishes between essential and redundant weights. The metric adopted in Wanda (Sun et al. 2024)—which involves computing the L_2 norm of the input and multiplying it by the absolute value of the corresponding weight—has garnered considerable attention for its simplicity and effectiveness. This approach provides a smooth, continuous measure that captures the contribution of each weight to the overall activations. In contrast, more elaborate metrics, including those based on second-order derivatives or layer-specific statistical properties, may offer theoretical advantages but often come at the cost of increased computational overhead.

Overall, the evolution of pruning methods reflects a broader trend in machine learning towards achieving a balance between model efficiency and predictive performance. Early heuristic methods have given way to more principled approaches that take into account the underlying statistics and structure of the network. The continued development of these techniques is critical for the deployment of large-scale neural networks on platforms with limited computational resources. The insights provided by previous studies serve as a valuable foundation for the enhancements presented in this work, including the development of the *STADE* method, which refines pruning strategies by incorporating the statistical characteristics of layer inputs.

Methodology

Consider a data matrix $X \in \mathbb{R}^{N\times M}$ and a weight matrix $\mathbb{W} \in \mathbb{R}^{M\times H}$, where N is the number of instances in the dataset, M represents the number of features and H represents the number of output features. In Wanda (Sun et al. 2024), the pruning of each column $\mathbb{W}_{:,i} \in \mathbb{R}^M$ is performed according to the criterion:

$$\min_j \|X_{:,j}\|_2 |\mathbb{W}_{j,i}| \tag{1}$$

where $\|X_{:,j}\|_2$ is the L_2-$norm$ of feature j in the dataset. In the following section, the pruning problem is formalized and it is demonstrated that *Wanda* selection criterion is optimal for layers with a centered inputs, i.e., inputs whose expected value in each coordinate is 0. With this insight, a generalization to layers with uncentered inputs is derived, leading to the proposed method *STADE*.

Problem definition

Let $X \in \mathbb{R}^M$ be a random multivariate variable with $\mu_i = \mathbb{E}[X_i]$ and $\sigma_i = Var[X_i]$, and consider a linear layer with a weight matrix $\mathbb{W} \in \mathbb{R}^{M\times H}$ and a bias term $\mathbb{B} \in \mathbb{R}^H$. The pruning process for the m-th column will be focused on the weight matrix (denoted by $W = \mathbb{W}_{:,m} \in \mathbb{R}^M$) and the corresponding bias (denoted by $B = \mathbb{B}_m \in \mathbb{R}$). In this setting, the pruning problem aims to find the optimal $W^* \in \mathbb{R}^M$ and $B^* \in \mathbb{R}$ such that:

$$\begin{gathered}\min_{W^*,B^*} \mathbb{E}[\left((B + \sum_{i=1}^M X_i W_i) - (B^* + \sum_{i=1}^M X_i W_i^*)\right)^2] \\ \text{s.t. } \forall i \in \{1,...,M\}\backslash\{j\}, W_i^* = W_i, W_j^* = 0\end{gathered} \tag{2}$$

Note that the objective is to select the pruning weight W_j so that the output remains almost unchanged, while allowing the bias term to be updated.

STADE derivation

Starting from the formulation in Eq. 9, the objective function can be reformulated as follows:

$$\begin{aligned}&\mathbb{E}[\left((B + \sum_{i=1}^M X_i W_i) - (B^* + \sum_{i=1}^M X_i W_i^*)\right)^2] \\ =&\mathbb{E}[((B-B^*) + X_j W_j)^2] \\ =&\mathbb{E}[(B-B^*)^2 + 2(B-B^*)(X_jW_j) + (X_jW_j)^2 \\ =&(B-B^*)^2 + 2(B-B^*)\mathbb{E}[X_jW_j] + \mathbb{E}[(X_jW_j)^2] \\ =&(B-B^*)^2 + 2(B-B^*)\mu_jW_j + (\sigma_j^2+\mu_j^2)W_j^2\end{aligned} \tag{3}$$

To determine the optimal solution of the convex problem (with respect to B^*) in Eq. 3, the derivative is computed to obtain the stationary and minimum point:

$$\begin{aligned}&\frac{d}{dB^*}[\mathbb{E}[\left((B + \sum_{i=1}^M X_i W_i) - (B^* + \sum_{i=1}^M X_i W_i^*)\right)^2]] \\ &= \frac{d}{dB^*}[(B-B^*)^2 + 2(B-B^*)\mu_jW_j + (\sigma_j^2+\mu_j^2)W_j^2] \\ &= -2(B-B^*) - 2\mu_jW_j = 0 \Leftrightarrow B^* = \mu_jW_j + B\end{aligned} \tag{4}$$

Substituting the optimal bias in Eq. 3 yields the solution for W^*:

$$\begin{aligned}&\min_{W^*,B^*} \mathbb{E}[\left((B + \sum_{i=1}^M X_i W_i) - (B^* + \sum_{i=1}^M X_i W_i^*)\right)^2] \\ &= \min_{j,B^*}(B-B^*)^2 + 2(B-B^*)\mu_jW_j + (\sigma_j^2+\mu_j^2)W_j^2 \\ &= \min_j(B-(\mu_jW_j+B))^2 + 2(B-(\mu_jW_j+B))\mu_jW_j \\ &+ (\sigma_j^2+\mu_j^2)W_j^2 = \min_j(\mu_jW_j)^2 - 2(\mu_jW_j)\mu_jW_j \\ &+ (\sigma_j^2+\mu_j^2)W_j^2 = \min_j \sigma_j^2W_j^2 \approx \min_j \frac{||X_{:,j}-\mu_j||_2^2}{N-1}W_j^2 \\ &\approx \frac{1}{N-1}\left(\min_j ||X_{:,j} - \frac{1}{N}\sum_{n=1}^N X_{n,j}||_2|W_j|\right)^2\end{aligned} \tag{5}$$

Since the our goal is to find the optimal j that minimizes the loss ($arg\min_j$), the factor $\frac{1}{N-1}$ and the squaring operation can be omitted.

Wanda derivation

Many modern Transformers (Touvron et al. 2023a,b; Dubey et al. 2024) employ normalization layers. These design choices simplify the original problem by enforcing that the input X is normalized ($\mu_i = \mathbb{E}[X_i] = 0$). Incorporating these conditions to the previous derivations (Eqs. 4 and 5) leads to:

$$\begin{gathered}B^* = \mu_jW_j + B = 0 * W_j + B = B \\ \min_j ||X_{:,j} - \mu_j||_2|W_j| = \min_j ||X_{:,j}||_2|W_j|\end{gathered} \tag{6}$$

This derivation results in the *Wanda* criterion, where the bias term doesn't need to get updated. Please notice that *Wanda* is optimal under the previous assumptions, i.e., it is only optimal for layers with centered inputs.

STADE-W: Using different metrics for different layers

Based on the previous theoretical insight we introduce *STADE-W*, a pruning strategy that employs different pruning criterions depending on whether the input is normalized.

Method	Weight Update	Centered Input	Pruning criterion
Magnitude (Zhu and Gupta 2017)	✗	Any	$\|W_{i,j}\|$
Wanda (Sun et al. 2024)	✗	Any	$\|\|X_{:,j}\|\|_2\|W_{i,j}\|$
Sparsegpt (Frantar and Alistarh 2023)	✓	Any	$[\|W\|^2/diag[(X^TX+\lambda\mathbf{1})^{-1}]]_{i,j}$
STADE	✗	Any	$\|\|X_{:,j}-\frac{1}{N}\sum_{n=1}^{N}X_{n,j}\|\|_2\|W_{i,j}\|$
STADE-W	✗	Yes	$\|\|X_{:,j}\|\|_2\|W_{i,j}\|$
	✗	No	$\|\|X_{:,j}-\frac{1}{N}\sum_{n=1}^{N}X_{n,j}\|\|_2\|W_{i,j}\|$

Table 1: Comparison of pruning weight metrics across different methods. The column *Centered Input* indicates whether the pruning method distinguishes between inputs with zero mean (Yes), without zero mean (No), or treats them equivalently (Any).

The pruning metrics derived from the previous analysis are as follows:

$$\text{Wanda criterion:} \quad ||X_{:,j}||_2|W_{i,j}| \tag{7}$$

$$\text{STADE criterion:} \quad ||X_{:,j}-\frac{1}{N}\sum_{n=1}^{N}X_{n,j}||_2|W_{i,j}| \tag{8}$$

STADE-W applies the *STADE* metric for biased inputs (such as the second layer of an MLP or the output layer in multi-head attention) and the *Wanda* metric for unbiased inputs (such as the first layer of an MLP or the queries, keys, and values in multi-head attention). In theory, *STADE* should be able to identify that the mean is 0 and return the same output as *Wanda*. However, in practice the dataset used for calibration might lead to a slightly different mean estimation and therefore, *STADE* ends up underperforming.

Optimal pruning metric

In order to clarify which pruning metric to use in which linear layer we make the following distinctions:

- **Centered inputs**: When the input distribution is centered the optimal method is *Wanda*. This would be the case when the previous layer is a *Batchnorm* (Ioffe and Szegedy 2015), *Groupnorm* (Wu and He 2018) or *Layernorm* layer (Ba 2016), but not in the case of a *RMSnorm* layer (Zhang and Sennrich 2019).
- **Uncentered inputs**: In this case, the input mean is no longer 0 and therefore *Wanda* is no longer optimal. Therefore *STADE* should be use since it takes into account the non-zero mean.

This protocol will be used throughout the paper unless specified otherwise. Notice that within the same model, different layers could belong to different scenarios as mentioned before with *STADE-W*.

Experiments

Models and Evaluation. Most experiments are conducted using the Llama (Touvron et al. 2023a,b; Dubey et al. 2024) and Qwen (Bai et al. 2023; Yang et al. 2024; Qwen et al. 2025; Yang et al. 2025) models. In addition, the OPT family (Zhang et al. 2023) is also evaluated due to its architectural differences such as the usage of a bias term in the linear layers, the usage of *Layernorm* (Ba 2016) and the incorporation of positional embeddings instead of rotary position embeddings (Su et al. 2024).

Following previous research (Sun et al. 2024), C4 dataset (Raffel et al. 2019) is used for calibration, while raw-WikiText2 dataset (Merity et al. 2022) is employed to evaluate model perplexity. Moreover, the zero-shot capabilities of the pruning methods are assessed using nine tasks from the EleutherAI LM Harness Benchmark (Gao et al. 2024). These tasks include: *Boolq* (Clark et al. 2019), a yes/no question answering dataset containing 15,942 examples; the *Recognizing Textual Entailment (RTE)* suite, which combines RTE-1 (Dagan, Glickman, and Magnini 2006), RTE-2 (Dagan, Glickman, and Magnini 2005), RTE-3 (Delmonte et al. 2007), and RTE-5 (Bentivogli et al. 2009) challenges constructed from news and Wikipedia text; *HellaSwag* (Zellers et al. 2019), a challenging dataset for evaluating commonsense,; *WinoGrande* (Keisuke et al. 2019), a binary fill-in-the-blank task that requires commonsense reasoning; *Arc-Easy* and *Arc-Challenge* (Clark et al. 2018), which consist of multiple-choice science questions targeting grade-school level content and are split into easy and challenging subsets, *OpenBookQA* (Mihaylov et al. 2018), a dataset that involves questions requiring multi-step reasoning, additional commonsense knowledge, and comprehensive text comprehension; and *MMLU* (Hendrycks et al. 2021), a multitask test consisting of multiple-choice questions from various branches of knowledge.

Baselines. The main experiments employ pruning methods that do not involve weight updates to corroborate our theoretical analysis. These methods include *Magnitude* pruning (Zhu and Gupta 2017) and *Wanda* (Sun et al. 2024). Furthermore, we also do an ablation on methods with weight updates (*SparseGPT* (Frantar and Alistarh 2023)) for further insights.

Pruning. The pruning strategy follows a layer-wise approach, which can be easily augmented with more complex procedures that assign different weights to each layer (Xu et al. 2024; Agarwal et al. 2024). The main focus is on unstructured pruning, where any weight in a matrix may be

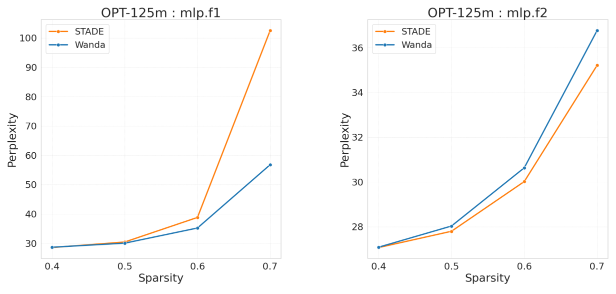
Methods	Sparsity	Llama-1	Llama-2		Llama-3		Qwen3				
		7B	7B	13B	3.0-8B	3.1-8B	1.7B	4B	8B	14B	32B
Dense	0%	5.68	5.47	4.88	6.14	6.24	16.67	13.64	9.72	8.64	7.6
Magnitude	2:4	42.53	37.76	8.89	2401.18	792.83	1808.24	1970.45	294.48	38.58	29.89
Wanda	2:4	11.52	12.12	8.98	24.31	22.87	61.63	**30.17**	16.41	13.14	10.33
STADE	2:4	**11.38**	**10.82**	**8.42**	**22.30**	**20.52**	**46.90**	133.17	**15.20**	**12.52**	**10.13**
Magnitude	4:8	16.83	15.91	7.32	181.47	212.46	614.71	150.43	115.48	21.18	36.49
Wanda	4:8	**8.57**	8.60	7.00	14.61	13.78	32.62	**22.25**	13.24	11.12	9.46
STADE	4:8	8.63	**8.29**	**6.86**	**13.69**	**12.93**	**27.76**	51.13	**12.62**	**10.69**	**9.29**
Magnitude	50%	17.29	16.03	6.83	205.45	134.28	174.10	111.22	54.56	15.22	49.09
Wanda	50%	**7.26**	**6.92**	5.97	9.83	9.65	20.63	**16.39**	11.35	10.00	**8.63**
STADE	50%	7.43	6.97	**5.95**	**9.63**	**9.47**	**18.67**	16.90	**11.19**	**9.60**	8.65

Table 2: Perplexity on Wikitext2 for different Llama and Qwen models. C4 dataset is used as the calibration dataset during the pruning process. 2:4 and 4:8 sparsity refers to a structure pruning approach where 2/4 weights are pruned out of every 4/8 weights (Mishra et al. 2021)

pruned. Additionally, the structured N:M pruning scenario will also be evaluated. In N:M structure pruning, out of every M weights N must be pruned (Hubara et al. 2021a). In particular, the 2:4 and 4:8 structured pruning schemes proposed by Nvidia (Mishra et al. 2021) for faster inference are adopted.

Large Language Modeling pruning

Table 2 reports the perplexity of Llama and Qwen models with various pruning methods. Notice that *STADE* outperforms the other methods consistently across the different pruning scenarios. These results follow our formal analysis, validating our theoretical understanding. Notice that the LLMs in Table 2 use RMSNorm (Zhang and Sennrich 2019) and therefore we do not use *STADE-W*. Since no layer receives a normalized input, it is no different from standard *STADE*.



(a) Centered input (mlp.f1 layer)

(b) Not centered input (mlp.f2 layer)

Figure 1: Perplexity comparison on OPT-125m when pruning only the specified layer type.

Pruning requirements of different layers

In order to see the effect of normalization layer, we investigate OPT models which use *Layernorm* (Ba 2016) instead of *RMSNorm* (Zhang and Sennrich 2019). Since the linear layers that receive the input after a *Layernorm* would be centered, a small ablation is done on the difference when pruning a layer with a centered input vs an uncentered input in Figure 1.

The experiment show that different layers benefit from different pruning methods. In particular, when a layer receives a centered input (first layer of the MLP block), *Wanda* performs better since it already assumes this scenario while *STADE* approximates the mean with the inputs. However, whenever the input is not centered *Wanda* is not able to keep up with *STADE* (second layer of the MLP block). This result is in line with our theoretical analysis and validates our characterization of the pruning problem. With these finding we propose *STADE-W*, a method that combines both *STADE* and *Wanda*. It uses *Wanda* when the input is centered and *STADE* otherwise. The results in Table 3 show that *STADE-W* improves model performance over *STADE* or *Wanda* when used individually on the OPT family.

Zero-shot comparison

While model perplexity serves as an important evaluation of pruning strategies, measuring prediction accuracy is equally crucial for large language models and their pruned variants. To test the impact of the different pruning methods on model accuracy, we evaluate on multiple zero-shot tasks across different datasets. The results are reported in Table 5.

The results on the zero-shot task align with those observed when evaluating perplexity (Table 2). *STADE* method demonstrates competitive performance across a range of models and tasks.

Method	Sparsity	opt-125m	opt-350m	opt-1.3b	opt-2.7b	opt-6.7b	opt-13b	opt-30b
Dense	0%	27.65	22.00	14.62	12.47	10.86	10.13	9.56
Magnitude	2:4	341.46	417.01	427.09	1152.92	264.04	484.64	1981.10
Wanda	2:4	80.24	113.54	28.23	21.20	15.89	**15.52**	13.44
STADE	2:4	109.68	100.16	**27.19**	24.08	16.44	17.61	15.35
STADE-W	2:4	**76.08**	**99.82**	27.55	**20.68**	**15.64**	15.57	**12.40**
Magnitude	4:8	169.09	160.73	240.13	166.93	196.15	450.06	564.03
Wanda	4:8	53.18	58.49	22.15	16.77	13.56	13.37	10.88
STADE	4:8	68.19	57.62	**21.34**	17.38	13.79	14.98	11.42
STADE-W	4:8	**52.64**	**56.69**	21.93	**16.66**	**13.41**	**13.34**	**10.85**
Magnitude	50%	193.35	97.78	1713.49	265.17	968.77	11609.08	168.09
Wanda	50%	**38.94**	36.21	18.42	14.22	11.98	**11.92**	**10.03**
STADE	50%	49.04	37.51	**17.75**	14.36	**11.87**	13.10	10.19
STADE-W	50%	39.22	**36.15**	18.38	**14.20**	11.97	11.96	10.05

Table 3: Perplexity on Wikitext2 with C4 as the calibration dataset.

Method	Weight Update	Sparsity	Qwen3 1.7B	Qwen3 8B
Dense	✗	0%	16.67	9.72
Magnitude	✗	2:4	1808.24	294.48
Wanda	✗	2:4	61.63	16.41
SparseGPT	✓	2:4	**31.74**	**14.48**
SparseGPT (w/o update)	✗	2:4	51.75	14.99
STADE	✗	2:4	46.90	15.20
Magnitude	✗	4:8	614.71	115.48
Wanda	✗	4:8	32.62	13.24
SparseGPT	✓	4:8	**25.38**	12.65
SparseGPT (w/o update)	✗	4:8	28.80	13.02
STADE	✗	4:8	27.76	**12.62**
Magnitude	✗	50%	174.10	54.56
Wanda	✗	50%	20.63	11.35
SparseGPT	✓	50%	23.73	11.49
SparseGPT (w/o update)	✗	50%	22.89	11.80
STADE	✗	50%	**18.67**	**11.19**

Table 4: Perplexity comparison with pruning methods that update weights (*SparseGPT*).

Weight update importance

SparseGPT is a pruning criterion that despite being comparable to *Wanda* and in some cases even outperforming it, it is known to be slower and more computationally demanding than other baselines. In this section, we compare the performance of *SparseGPT* against *STADE*, with a particular focus on the critical importance of its weight update mechanism.

Table 4 shows that even though *SparseGPT* is competitive, it is not always the best performing method. In particular, it is a competitive for scenarios where the weight selection is more restricted, i.e., it gets better with smaller models and with more constrained scenarios such as 2:4 pruning. Nevertheless, when pruning bigger models in more unstructured scenarios, *STADE* is able to show SOTA performance. Moreover, the efficacy of *SparseGPT* comes from the weight update and not from the weight selection criterion as shown with the *SparseGPT (w/o update)* ablation.

Notice that our theoretical analysis of the pruning problem presented focuses exclusively on cases without weight updates. The experimental findings corroborate that the performance of pruning methods can be improved by applying weight updates to the remaining parameters after pruning. These updates help to counterbalance the negative effects associated with the removal of weights, thereby better preserving the model's predictive capability. This suggests that beyond the initial selection of weights to prune, the adjustment of the unpruned weights is a crucial factor in achieving optimal performance.

Bias usage ablation

STADE method updates the bias term when pruning the models. In models like OPT which already have bias, this is reasonable. However, Llama and Qwen models do not have originally a bias term and therefore, this would results in adding a new bias term which could be considered as adding some extra weight variables. This could be considered an unfair advantage when compared to the other methods. To investigate this, a small ablation is done where we do not update the bias term.

The results shown in Tab. 6 demonstrate that there is little to no difference when adding the bias term and if one wants to remove this term the results are almost identical. Empirically we observe that for any layer, the sum of the absolute terms of the bias is always smaller than 10^{-2}, which explains why removing it has little to no impact.

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	45.73%	56.42%	63.52%	66.63%	69.48%
Magnitude	50%	32.91%	33.63%	35.40%	41.11%	60.65%
Wanda	50%	40.25%	49.85%	**57.66%**	62.48%	67.23%
STADE	50%	**40.36%**	**50.38%**	57.10%	**62.99%**	**67.64%**
Magnitude	2:4	30.21%	33.09%	33.11%	33.62%	48.54%
Wanda	2:4	33.28%	38.57%	**47.28%**	54.71%	59.81%
STADE	2:4	**33.85%**	**39.72%**	43.79%	**56.62%**	**59.94%**
Magnitude	4:8	31.17%	33.75%	35.43%	34.26%	54.53%
Wanda	4:8	**35.49%**	43.21%	**53.85%**	60.17%	63.33%
STADE	4:8	34.89%	**43.26%**	47.99%	**60.52%**	**63.69%**

Table 5: Zero shot accuracy averaged over 9 individual tasks (*Arc-Challenge*, *Arc-Easy*, *Boolq*, *HellaSwag*, *OpenBookQA*, *RTE-3*, *WinoGrande* and *MMLU*. The results for each individual tasks can be found in the Appendix.

Model	Sparsity	STADE	STADE (w/o bias)
Llama-7B	0.5	**11.38**	**11.38**
Llama2-7B	0.5	**10.82**	**10.82**
Llama2-13B	0.5	**8.42**	**8.42**
Llama3-8B	0.5	**22.30**	22.37
Llama3.1-8B	0.5	**20.52**	**20.52**
Qwen3-1.7B	0.5	**46.90**	**46.90**
Qwen3-4B	0.5	133.17	**131.50**
Qwen3-8B	0.5	**15.20**	**15.20**
Qwen3-14B	0.5	12.52	**12.51**
Qwen3-32B	0.5	**10.13**	**10.13**

Table 6: Ablation on the importance of the bias update in *STADE* algorithm.

Future Work

The exploration of various pruning criterions has revealed that no single method is universally optimal for every layer within a deep neural network. Future research should aim to deepen the understanding of how different layers and network depths interact with distinct pruning criteria, potentially leading to adaptive, layer-specific pruning strategies. In addition, investigating the benefits of pruning each layer with different sparsity ratios could further enhance model efficiency and performance, representing another promising direction for future work. Furthermore, methods such as *SparseGPT* demonstrate that incorporating weight updates for unpruned parameters can significantly enhance performance, suggesting that further investigation into efficient weight update mechanisms may yield substantial benefits.

In our work we focus on medium to small model sizes, as we are interested in their capabilities on many resource-constrained scenarios. Nevertheless, evaluating these pruning methods on larger models and other architectures (*mixture of experts* (Jiang et al. 2024)) would help to assess the scalability and effectiveness of the proposed techniques in on large-scale settings.

Conclusion

This work presents a comprehensive analysis of optimal weight pruning in neural networks and provides a theoretical framework that explains why *Wanda* is effective in many common deep learning scenarios. It demonstrates that while *Wanda* performs optimally in layers with centered inputs, its effectiveness diminishes in layers that receive uncentered inputs. In response to these observations, we propose a new pruning criterion (*STADE*) that handles this scenario. We demonstrate theoretically and empirically that *STADE* outperforms *Wanda* for uncentered inputs.

We also observe that different layers have different input statistics and therefore the optimal pruning criterion might change between layers. Building upon these insights, we introduce *STADE-W*, which dynamically combines *Wanda* and *STADE* based on the input statistics of each layer, making it, to the best of current knowledge, the first pruning method that employs different pruning criterions for different layers. We do extensive experiments on Qwen, Llama and Open Pre-trained Transformers models. We not only evaluate perplexity but also zero-shot performance. The results validate our theoretical analysis and reveal that pruning effectiveness varies according to the input characteristics of each layer. Notably, our method achieves state-of-the-art performance for the pruning problem. Moreover, our experiments demonstrate that incorporating weight update mechanisms (as exemplified by *SparseGPT*) can improve performance, further highlighting the benefits of updating the unpruned weights and a future research direction.

All together, these contributions not only advance the understanding of pruning strategies but also offer a new robust method for reducing the computational demands of large language models without significant performance loss. The insights provided herein pave the way for more efficient deployment of large-scale models in resource-constrained environments.

References

Agarwal, P.; Mathew, M.; Patel, K. R.; Tripathi, V.; and Swami, P. 2024. Prune Efficiently by Soft Pruning. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2210–2217.

Ashkboos, S.; Croci, M. L.; Nascimento, M. G. d.; Hoefler, T.; and Hensman, J. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Ba, J. L. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bai, G.; Li, Y.; Ling, C.; Kim, K.; and Zhao, L. 2024. SparseLLM: Towards global pruning of pre-trained language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Bentivogli, L.; Magnini, B.; Dagan, I.; Dang, H. T.; and Giampiccolo, D. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, T.; Cheng, Y.; Gan, Z.; Yuan, L.; Zhang, L.; and Wang, Z. 2021. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34: 19974–19988.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv:1803.05457v1*.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The PASCAL recognising textual entailment challenge. 177–190. ISBN 978-3-540-33427-9.

Dagan, I.; Glickman, O.; and Magnini, B. 2006. *The pascal recognising textual entailment challenge*, 177–190.

Delmonte, R.; Bristot, A.; Piccolino Boniforti, M. A.; and Tonelli, S. 2007. Entailment and Anaphora Resolution in RTE3. In Sekine, S.; Inui, K.; Dagan, I.; Dolan, B.; Giampiccolo, D.; and Magnini, B., eds., *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 48–53. Prague: Association for Computational Linguistics.

Dong, X.; Chen, S.; and Pan, S. J. 2017. Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon. arXiv:1705.07565.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Durante, Z.; Sarkar, B.; Gong, R.; Taori, R.; Noda, Y.; Tang, P.; Adeli, E.; Lakshmikanth, S. K.; Schulman, K.; Milstein, A.; et al. 2024. An interactive agent foundation model. *arXiv preprint arXiv:2402.05929*.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020. Linear Mode Connectivity and the Lottery Ticket Hypothesis. arXiv:1912.05671.

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

Gao, L.; Tow, J.; Abbasi, B.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; Le Noac'h, A.; Li, H.; McDonell, K.; Muennighoff, N.; Ociepa, C.; Phang, J.; Reynolds, L.; Schoelkopf, H.; Skowron, A.; Sutawika, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2024. A framework for few-shot language model evaluation.

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both Weights and Connections for Efficient Neural Networks. arXiv:1506.02626.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hubara, I.; Chmiel, B.; Island, M.; Banner, R.; Naor, J.; and Soudry, D. 2021a. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 34: 21099–21111.

Hubara, I.; Chmiel, B.; Island, M.; Banner, R.; Naor, J.; and Soudry, D. 2021b. Accelerated Sparse Neural Training: A Provable and Efficient Method to Find N:M Transposable Masks. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 21099–21111. Curran Associates, Inc.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. pmlr.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Hanna, E. B.; Bressand, F.; Lengyel, G.; Bour, G.; Lample, G.; Lavaud, L. R.; Saulnier, L.; Lachaux, M.-A.; Stock, P.; Subramanian, S.; Yang, S.; Antoniak, S.; Scao, T. L.; Gervet, T.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2024. Mixtral of Experts. arXiv:2401.04088.

junyou li; Zhang, Q.; Yu, Y.; FU, Q.; and Ye, D. 2024. More Agents Is All You Need. *Transactions on Machine Learning Research*.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020a. Scaling Laws for Neural Language Models. arXiv:2001.08361.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020b. Scaling Laws for Neural Language Models. arXiv:2001.08361.

Keisuke, S.; Ronan, L. B.; Chandra, B.; and Yejin, C. 2019. WinoGrande: An Adversarial Winograd Schema Challenge at Scale.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Li, L.; Dong, P.; Tang, Z.; Liu, X.; Wang, Q.; Luo, W.; Xue, W.; Liu, Q.; Chu, X.; and Guo, Y. 2024. Discovering sparsity allocation for layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ma, S.; Wang, H.; Ma, L.; Wang, L.; Wang, W.; Huang, S.; Dong, L.; Wang, R.; Xue, J.; and Wei, F. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2022. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*.

Mihaylov, T.; Clark, P.; Khot, T.; and Sabharwal, A. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *EMNLP*.

Mishra, A.; Latorre, J. A.; Pool, J.; Stosic, D.; Stosic, D.; Venkatesh, G.; Yu, C.; and Micikevicius, P. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*.

Morcos, A. S.; Yu, H.; Paganini, M.; and Tian, Y. 2019. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. arXiv:1906.02773.

Qwen; :; Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Tang, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2025. Qwen2.5 Technical Report. arXiv:2412.15115.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv e-prints*.

Sanh, V.; Wolf, T.; and Rush, A. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33: 20378–20389.

Sevilla, J.; Heim, L.; Ho, A.; Besiroglu, T.; Hobbhahn, M.; and Villalobos, P. 2022. Compute Trends Across Three Eras of Machine Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Su, J.; Ahmed, M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2024. A Simple and Effective Pruning Approach for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

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

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; and Micikevicius, P. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. arXiv:2004.09602.

Wu, Y.; and He, K. 2018. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19.

Wu, Z.; Han, C.; Ding, Z.; Weng, Z.; Liu, Z.; Yao, S.; Yu, T.; and Kong, L. 2024. OS-Copilot: Towards Generalist Computer Agents with Self-Improvement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Xu, P.; Shao, W.; Chen, M.; Tang, S.; Zhang, K.; Gao, P.; An, F.; Qiao, Y.; and Luo, P. 2024. BESA: Pruning Large Language Models with Blockwise Parameter-Efficient Sparsity Allocation. In *The Twelfth International Conference on Learning Representations*.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; and Fan, Z. 2024. Qwen2 Technical Report. arXiv:2407.10671.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Zhang, B.; and Sennrich, R. 2019. Root Mean Square Layer Normalization. arXiv:1910.07467.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2023. Opt: Open pre-trained transformer language models, 2022. *URL https://arxiv. org/abs/2205.01068*, 3: 19–0.

Zhou, H.; Lu, X.; Xu, W.; Zhu, C.; Zhao, T.; and Yang, M. 2024. Lora-drop: Efficient lora parameter pruning based on output evaluation. *arXiv preprint arXiv:2402.07721*.

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

Appendix

Pruning time

Different pruning methods take different time to calculate their corresponding pruning scores. We report in Table 7 the pruning time for the different methods in different pruning scenarios.

Sparsity	Wanda	STADE	STADE-W	SparseGPT
50%	72,89	72,02	74,55	222,31
2:4	77,87	74,80	73,04	204,52
4:8	70,10	73,39	71,51	215,36

Table 7: Pruning time comparison in seconds for different methods on Llama-3.2-1B.

Implementation details

When estimating the mean and the standard deviation, loading the full data results in an Out-Of-Memory error with our computational resources. Therefore, the mean and standard deviation is calculated using a in a moving fashion.

Calculating the sum and square of sums to later approximate the final results, leads to too high values at which point the new instances do not add any values to the value stored. In order to avoid that, the mean and standard deviation is calculated in each iteration as follows:

Algorithm 1: Mean and variance calculation

```
Input: dataloader D
Output: mean $\mu$, var $\mathbb{V}$
1: Let $N = 0$, $\mu = 0$ and $\mathbb{V} = 0$.
2: for $x_{batch}$ in D do
3:     $N_{new} = N + len(x_{batch})$
4:     $\mu_{new} = \frac{\mu * N}{N_{new}} + \frac{sum(x\_batch)}{N_{new}}$
5:     $\mathbb{V} = \frac{(N-1)*\mathbb{V}}{N_{new}-1} + \frac{N*\mu^2}{(N_{new}-1)} - \frac{N_{new}*\mu_{new}^2}{(N_{new}-1)} + \frac{sum(x_{batch}^2)}{N_{new}-1}$
6:     $N = N_{new}$
7:     $\mu = \mu_{new}$
8: end for
9: return $\mu$, $\mathbb{V}$
```

Training details

While the pruning methods shown don't have hyperparameter to tuned, there are some training details that we would like to mention:

- **C4 calibration dataset**: Following *Wanda*, when using C4 dataset only the file *'en/c4-train.00000-of-01024.json.gz'* is used during pruning to speed up the process. the full dataset can be fined in *'https://huggingface.co/datasets/allenai/c4/tree/main/en'*.
- **Sequence length**: Some LLMs allow over 10k context window. In order to run the models under our personal hardware constrains, we reduce the sequence length to 2048 in order to test more models. This is both applied during pruning and evaluation.

Intuitive explanation of STADE

In this section we formulate an easy and intuitive explanation for *STADE*. We will assume that the input multivariate distribution $X \in \mathbb{R}^2$ is normally distributed, i.e., $X_i \sim \mathcal{N}(\mu_i, \sigma_i)$. **Notice that STADE does not require the input to be normally distributed,** this is just a simplification for the sake of the explanation. In the same way as in the Methodology section, we will consider the pruning problem for one column. In this case the corresponding output of the linear layer ($\hat{y}$) can be calculated as:

$$\begin{aligned} \hat{y} &= B + x_1 W_1 + x_2 W_2 = \\ &= B + (\mu_1 + \epsilon_1 \sigma_1) W_1 + (\mu_2 + \epsilon_2 \sigma_2) W_2 \\ &= (B + \mu_1 + \mu_2) + \sigma_1 W_1 \epsilon_1 + \sigma_2 W_2 \epsilon_2 \end{aligned} \tag{9}$$

Notice that $\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 1)$ and therefore will affect the same way when pruning the weight. However, when deciding whether to prune W_1 or W_2, it's not only about the value of the weight but also about the standard deviation of the corresponding input. This is due to the fact that the mean of the input can be added to the bias and therefore omitted when pruning the weights.

STADE* variation derivation

Not all models use a bias term in their linear layers. Therefore, we consider a variation of *STADE* without the possibility of updating the bias (*STADE**), i.e., the linear layer to prune has no bias term and the pruning method is not allowed to add a bias term in order to keep the model structure. To do so we expand on the previous derivations from the main paper as follows:

$$\begin{aligned} &\min_{W^*,0} \mathbb{E}\left[\left((B + \sum_{i=1}^{M} X_i W_i) - (B^* + \sum_{i=1}^{M} X_i W_i^*) \right)^2 \right] \\ &= \min_{j,0} (B - B^*)^2 + 2(B - B^*)\mu_j W_j + (\sigma_j^2 + \mu_j^2) W_j^2 \\ &= \min_j (\sigma_j^2 + \mu_j^2) W_j^2 \\ &\approx \left[||X_{:,j} - \frac{1}{N}\sum_{n=1}^{N} X_{n,j}||_2^2 + (\frac{1}{N}\sum_{n=1}^{N} X_{n,j})^2 \right] |W_{i,j}|^2 \end{aligned} \tag{10}$$

We originally wanted to include this in the main paper. Nevertheless, standard *STADE* had better performance than *STADE** even when not updating the bias, i.e., the pruning criterion even though locally optimal (optimal for the linear layer pruning) is not optimal globally (for the model pruning). This can be observed in the following tables. The only cases, where it outperforms *STADE* is when *STADE* has a

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B	Qwen3-32B
Dense	0%	20.95	16.67	13.64	9.72	8.64	7.60
Magnitude	2:4	85481.66	1808.24	1970.45	294.48	38.58	29.89
Wanda	2:4	<u>190.03</u>	61.63	<u>30.17</u>	16.41	13.14	<u>10.33</u>
STADE	2:4	13785.28	<u>46.90</u>	133.17	<u>15.20</u>	12.52	**10.13**
STADE (w/o bias)	2:4	13785.28	<u>46.90</u>	131.50	<u>15.20</u>	12.51	**10.13**
STADE*	2:4	193.29	60.66	30.25	16.41	13.10	-
STADE-W	2:4	171.60	47.24	32.37	15.54	12.57	-
SparseGPT	2:4	**91.05**	**31.74**	**21.32**	**14.48**	<u>12.47</u>	-
SparseGPT (no update)	2:4	6278.69	51.75	86.57	14.99	**12.18**	-
Magnitude	4:8	99815.32	614.71	150.43	115.48	21.18	36.49
Wanda	4:8	<u>73.71</u>	32.62	<u>22.25</u>	13.24	11.12	<u>9.46</u>
STADE	4:8	304.30	<u>27.76</u>	51.13	**12.62**	10.69	**9.29**
STADE (w/o bias)	4:8	304.30	27.79	52.95	<u>12.63</u>	<u>10.68</u>	**9.29**
STADE*	4:8	74.73	32.46	22.50	13.24	11.12	-
STADE-W	4:8	77.89	28.67	22.57	12.71	**10.67**	-
SparseGPT	4:8	**60.55**	**25.38**	**18.64**	12.65	11.16	-
SparseGPT (no update)	4:8	513.65	28.80	51.60	13.02	10.71	-
Magnitude	50%	1455.57	174.10	111.22	54.56	15.22	49.09
Wanda	50%	34.20	20.63	16.39	11.35	10.00	**8.63**
STADE	50%	<u>34.01</u>	<u>18.67</u>	16.90	<u>11.19</u>	<u>9.60</u>	<u>8.65</u>
STADE (w/o bias)	50%	34.04	**18.66**	16.90	<u>11.19</u>	<u>9.60</u>	<u>8.65</u>
STADE*	50%	34.06	20.57	<u>16.39</u>	11.35	10.01	-
STADE-W	50%	**33.96**	18.98	**16.04**	**11.10**	**9.54**	-
SparseGPT	50%	34.14	23.73	17.39	11.49	10.08	-
SparseGPT (no update)	50%	89.12	22.89	20.03	11.80	9.95	-

Table 8: Wikitext perplexity for the Qwen family. Notice that *STADE-W* here is applied after the *RMSnorm* layers. As explained in the Methodology, Qwen3 uses *RMSnorm* which does not normalize the inputs and therefore it is not applicable as it was with the OPT family. We observe huge spikes for Qwen3-0.6B and Qwen3-4B. To the best of our knowledge the only difference with the other models is the usage of tie-embeddings. Nevertheless, Qwen3-1.7B also uses them and doesn't exhibit those spikes. We were not able to identify the source behind it. We will investigate it in our future work.

big spike/jump on the perplexity. It seems that it is performing worse in general, but it is always to have consistent results avoiding the huge spikes. We will investigate this phenomena more in the future work.

Additional experiments

Tables 8 to 16 show experiments on more models and additional pruning metrics measuring both perplexity and zero-shot performance.

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	31.40%	39.76%	50.77%	55.80%	58.62%
Magnitude	50%	20.90%	19.11%	22.70%	28.07%	48.81%
Wanda	50%	23.38%	30.46%	39.76%	50.85%	**55.20%**
STD (w/o bias)	50%	**23.98%**	**33.53%**	**41.72%**	**51.88%**	55.03%
Magnitude	2:4	20.65%	20.56%	22.44%	18.69%	37.12%
Wanda	2:4	19.71%	19.71%	**32.17%**	38.65%	42.24%
STD (w/o bias)	2:4	**21.67%**	**20.90%**	29.86%	**42.32%**	**44.54%**
Magnitude	4:8	20.56%	21.50%	23.46%	19.45%	44.37%
Wanda	4:8	19.71%	24.91%	**38.05%**	46.16%	50.60%
STD (w/o bias)	4:8	**21.25%**	**26.37%**	33.11%	**47.95%**	**51.96%**

Table 9: Zero-shot performance on Arc Challenge (Clark et al. 2018).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	60.90%	72.22%	80.51%	83.46%	84.22%
Magnitude	50%	28.75%	34.01%	47.94%	58.12%	76.47%
Wanda	50%	48.65%	62.12%	**72.90%**	80.09%	81.31%
STD (w/o bias)	50%	**48.70%**	**64.31%**	72.39%	**80.43%**	**81.94%**
Magnitude	2:4	26.52%	28.91%	33.46%	36.57%	63.09%
Wanda	2:4	**32.79%**	47.35%	**59.26%**	72.69%	**72.47%**
STD (w/o bias)	2:4	27.86%	**49.71%**	50.17%	**74.03%**	71.42%
Magnitude	4:8	27.48%	30.51%	42.63%	40.32%	72.35%
Wanda	4:8	**40.74%**	56.65%	**67.89%**	77.23%	78.41%
STD (w/o bias)	4:8	31.65%	**58.25%**	57.58%	**77.78%**	**78.87%**

Table 10: Zero-shot performance on Arc Easy (Clark et al. 2018).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	64.53%	77.46%	85.11%	86.64%	89.33%
Magnitude	50%	46.36%	46.94%	38.32%	52.32%	79.60%
Wanda	50%	**62.20%**	**73.61%**	**82.51%**	**84.86%**	88.07%
STD (w/o bias)	50%	62.08%	73.43%	80.52%	84.68%	**88.20%**
Magnitude	2:4	38.65%	50.28%	46.94%	43.33%	65.78%
Wanda	2:4	46.76%	63.85%	**73.39%**	**82.17%**	85.81%
STD (w/o bias)	2:4	**52.75%**	**67.22%**	70.18%	81.68%	**86.64%**
Magnitude	4:8	42.51%	51.10%	42.08%	41.28%	68.59%
Wanda	4:8	48.44%	**70.83%**	**78.41%**	**85.11%**	**87.43%**
STD (w/o bias)	4:8	**57.68%**	66.33%	74.50%	84.92%	**87.43%**

Table 11: Zero-shot performance on Boolq (Clark et al. 2019).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	37.55%	46.12%	52.27%	57.14%	60.97%
Magnitude	50%	26.07%	28.16%	29.79%	34.51%	49.76%
Wanda	50%	32.24%	38.56%	45.03%	50.08%	55.11%
STD (w/o bias)	50%	**32.86%**	**40.26%**	**45.92%**	**51.65%**	**56.66%**
Magnitude	2:4	25.70%	26.56%	27.03%	26.98%	42.16%
Wanda	2:4	**27.17%**	29.90%	**37.46%**	42.35%	48.00%
STD (w/o bias)	2:4	26.51%	**31.08%**	36.22%	**43.76%**	**49.53%**
Magnitude	4:8	26.21%	26.89%	30.43%	28.04%	45.68%
Wanda	4:8	**29.09%**	33.91%	**41.22%**	46.26%	51.69%
STD (w/o bias)	4:8	28.27%	**35.16%**	40.10%	**47.91%**	**53.08%**

Table 12: Zero-shot performance on HellaSwag (Zellers et al. 2019).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	21.00%	28.40%	29.20%	31.00%	35.00%
Magnitude	50%	15.00%	13.40%	13.80%	18.60%	30.40%
Wanda	50%	16.60%	21.20%	26.20%	28.60%	33.00%
STD (w/o bias)	50%	**18.00%**	**22.80%**	**26.80%**	**30.60%**	**33.60%**
Magnitude	2:4	14.00%	13.80%	13.00%	14.80%	26.20%
Wanda	2:4	13.00%	13.60%	22.20%	23.00%	28.60%
STD (w/o bias)	2:4	**15.40%**	**15.40%**	**22.80%**	**23.20%**	**30.20%**
Magnitude	4:8	13.20%	14.40%	14.20%	15.40%	28.20%
Wanda	4:8	15.60%	17.40%	**24.80%**	26.40%	31.40%
STD (w/o bias)	4:8	**15.80%**	**17.80%**	22.80%	**28.40%**	**31.60%**

Table 13: Zero-shot performance on OpenbookQA (Mihaylov et al. 2018).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	54.15%	70.76%	75.81%	78.34%	77.62%
Magnitude	50%	51.26%	52.71%	51.62%	52.71%	69.31%
Wanda	50%	**54.15%**	**70.04%**	**72.92%**	70.04%	81.23%
STD (w/o bias)	50%	51.26%	67.15%	67.51%	**70.76%**	**82.31%**
Magnitude	2:4	42.96%	49.46%	47.29%	52.71%	48.74%
Wanda	2:4	51.62%	52.35%	**55.96%**	61.73%	**70.40%**
STD (w/o bias)	2:4	**53.07%**	**52.71%**	54.87%	**69.31%**	65.70%
Magnitude	4:8	46.21%	51.26%	52.71%	52.35%	54.87%
Wanda	4:8	**53.07%**	52.35%	**71.12%**	**72.92%**	**70.04%**
STD (w/o bias)	4:8	47.65%	**53.07%**	55.23%	70.76%	69.68%

Table 14: Zero-shot performance on RTE (Bentivogli et al. 2009).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	56.20%	60.93%	66.06%	67.72%	72.85%
Magnitude	50%	49.88%	51.62%	51.62%	55.09%	64.80%
Wanda	50%	54.06%	**57.38%**	62.12%	**69.61%**	**72.77%**
STD (w/o bias)	50%	**55.41%**	56.91%	**62.19%**	68.35%	71.82%
Magnitude	2:4	48.78%	52.09%	51.30%	51.78%	61.01%
Wanda	2:4	**52.25%**	**53.67%**	**56.98%**	62.43%	68.27%
STD (w/o bias)	2:4	50.12%	52.33%	53.75%	**63.85%**	**68.75%**
Magnitude	4:8	49.80%	50.20%	51.38%	51.93%	64.33%
Wanda	4:8	**52.80%**	**53.99%**	**61.01%**	**66.61%**	**69.85%**
STD (w/o bias)	4:8	52.09%	53.75%	56.98%	65.43%	69.22%

Table 15: Zero-shot performance on Winogrande (Keisuke et al. 2019).

Method	Sparsity	Qwen3-0.6B	Qwen3-1.7B	Qwen3-4B	Qwen3-8B	Qwen3-14B
Dense	0%	56.20%	60.93%	66.06%	67.72%	72.85%
Magnitude	50%	49.88%	51.62%	51.62%	55.09%	64.80%
Wanda	50%	54.06%	**57.38%**	62.12%	**69.61%**	**72.77%**
STD (w/o bias)	50%	**55.41%**	56.91%	**62.19%**	68.35%	71.82%
Magnitude	2:4	48.78%	52.09%	51.30%	51.78%	61.01%
Wanda	2:4	**52.25%**	**53.67%**	**56.98%**	62.43%	68.27%
STD (w/o bias)	2:4	50.12%	52.33%	53.75%	**63.85%**	**68.75%**
Magnitude	4:8	49.80%	50.20%	51.38%	51.93%	64.33%
Wanda	4:8	**52.80%**	**53.99%**	**61.01%**	**66.61%**	**69.85%**
STD (w/o bias)	4:8	52.09%	53.75%	56.98%	65.43%	69.22%

Table 16: Zero-shot performance on MMLU (Hendrycks et al. 2021).