Efficient Post-Training Pruning of Large Language Models with Statistical Correction

Anonymous ACL submission

Abstract

Post-training pruning is an effective approach for reducing the size and inference cost of large language models (LLMs), but existing methods often face a trade-off between pruning quality and computational efficiency. Heuristic pruning methods are efficient but sensitive to activation outliers, while reconstruction-based approaches improve fidelity at the cost of heavy computation. In this work, we propose a lightweight post-training pruning framework based on first-order statistical properties of model weights and activations. During pruning, channel-wise statistics are used to calibrate magnitude-based importance scores, reducing bias from activation-dominated channels. After pruning, we apply an analytic energy compensation to correct distributional distortions caused by weight removal. Both steps operate without retraining, gradients, or second-order information. Experiments across multiple LLM families, sparsity patterns, and evaluation tasks show that the proposed approach improves pruning performance while maintaining computational cost comparable to heuristic methods. The results suggest that simple statistical corrections can be effective for post-training pruning of LLMs.

1 Introduction

Large Language Models (LLMs) (OpenAI, 2024, 2020; MetaAI, 2023a) have achieved strong performance across a broad range of language understanding and generation tasks (Wei et al., 2022; Bubeck et al., 2023; Hendrycks et al., 2021). However, multiple lines of work, such as pretraining language modeling (Frankle and Carbin, 2019), machine translation (Voita et al., 2023), and downstream fine-tuning (Sanh et al., 2020b), have consistently shown that these models are substantially overparameterized (Gale et al., 2019; Zhu and Gupta, 2017). In particular, large fractions of model parameters can be pruned or compressed with minimal impact on performance, even without additional retraining (Frantar et al., 2023a; Frantar and Alistarh, 2023; Lin et al., 2024). Exploiting this redundancy through post-training model reduction is therefore important for reducing computational, memory, and storage costs, and for enabling scalable deployment of LLMs across diverse hardware and application settings (Frantar and Alistarh, 2023; Mishra et al., 2021; DeepMind, 2022; Strubell et al., 2019; Kaplan et al., 2020).

LLM compression is commonly studied through quantization, knowledge distillation, and pruning(Zhu et al., 2024). Quantization reduces numerical precision(Frantar et al., 2023a; Xiao et al., 2024), and distillation transfers the behavior of a large model to a smaller student model(Hinton et al., 2015; Sanh et al., 2020a). Pruning directly removes redundant weights from the pretrained model, producing sparse parameter matrices that may reduce memory and computation when supported by sparse hardware kernels(Frantar and Alistarh, 2023; Sun et al., 2024). These directions are complementary. In this paper, we focus on *post-training pruning*, which performs pruning without retraining and can therefore serve as a lightweight component in a larger compression pipeline.

Post-training model reduction for LLMs involves a trade-off between pruning accuracy and computational cost. Post-training methods can be divided into reconstruction-based approaches that model weight interactions and heuristic approaches based on simple importance metrics (Cun et al., 1990; Hassibi et al., 1993). Reconstruction-based methods such as SparseGPT (Frantar and Alistarh, 2023) use second-order information to update the remaining weights after pruning to preserve reconstruction quality, at the cost of increased computational and memory overhead from Hessian estimation and matrix inversion. Heuristic approaches such as Wanda (Sun et al., 2024) avoid weight updates and instead perform greedy pruning based on weight–activation

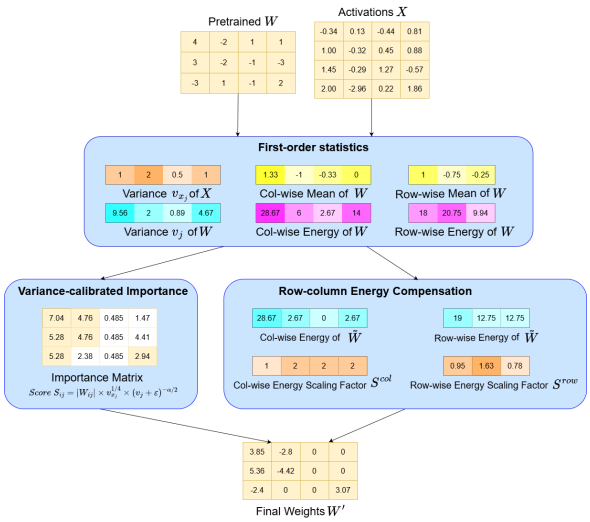

Figure 1: Overview of the proposed statistical correction framework. First-order statistics from pretrained weights W and activations X are used for variance-calibrated importance scoring and post-pruning energy compensation. The two components are applied independently and produce corrected weights while preserving the sparsity pattern.

magnitudes, which improves scalability. However, these methods analyze weights in isolation and rely on input signal strength to assess parameter importance, resulting in suboptimal pruning decisions (Han et al., 2015; Zhang et al., 2024; Dettmers et al., 2022; Frankle et al., 2021). In particular, channels dominated by extreme activation outliers may receive high importance scores despite their instability, as shown in Sec. 3.1.

Recent work has also explored alternative approaches for defining parameter importance. For example, DenoiseRotator (Gu et al., 2025) applies orthogonal transformations to redistribute importance scores before pruning, while RIA (Zhang et al., 2024) adopts a weight-pruning criterion based on relative importance among parameters. These methods rely on preprocessing or weight-only importance estimation and do not explicitly model activation reliability or post-pruning signal correction. In contrast, we focus on post-training pruning and weight adjustment directly on pretrained models.

We propose a post-training pruning framework based on first-order statistical properties of both model weights and activations. During pruning, channel-wise statistics of weights and activations are used to calibrate magnitude-based importance scores, reducing bias from activation-dominated channels (Sun et al., 2024). After pruning, signal distortion caused by weight removal is corrected through an analytic energy compensation that restores the scale of layer outputs. Both parameter selection and weight adjustment rely on first-order statistical correction and do not require gradient-based optimization or second-order information. As a result, the method has computational cost comparable to heuristic pruning approaches (Han et al., 2015; Sun et al., 2024) while avoiding reconstruction-based overhead. Moreover, the proposed weight adjustment is criterion-agnostic and can be applied after pruning regardless of the underlying importance metric, making it compatible with existing weight-pruning methods.

For pruning, we introduce a variance-calibrated importance metric that corrects systematic bias in magnitude-based heuristics. Existing methods such as Wanda (Sun et al., 2024) compute importance scores using the product of weight magnitude and input activation norm ($|W_{ij}| \cdot \|X_j\|_2$). This formulation can be overly sensitive to activation outliers (Dettmers et al., 2022; Kovaleva et al., 2021). Channels with large input norms tend to amplify the importance scores of all associated weights, including noisy or structurally insignificant parameters.

To address this issue, we incorporate weight variance as a measure of channel stability. Feature channels with high weight variance tend to have heterogeneous and unstable distributions (Zhang et al., 2024; Voita et al., 2023; Kovaleva et al., 2021; Timkey and van Schijndel, 2021), making magnitude-based importance scores less reliable. In contrast, low-variance channels reflect more consistent parameter structure. We therefore down-weight activation-driven importance scores using an inverse variance factor. This effectively suppresses activation-dominated channels to reallocate sparsity toward distributionally stable features.

For weight adjustment, we observe that a major source of post-pruning degradation is mismatched signal energy rather than irrecoverable information loss. Pruning alters the aggregate signal strength of feature maps, leading to a distribution shift that accumulates across layers (Han et al., 2015; Li et al., 2016; Sun and Saenko, 2016). To address this effect, we introduce an energy-based compensation mechanism that restores the scale of latent representations after pruning.

Specifically, we apply a deterministic, closed-form rescaling that aligns the layer-wise output energy of the pruned model with that of the original model (Hassibi et al., 1993; Cun et al., 1990). This

update is non-iterative and requires no gradient computation or second-order information (Frantar and Alistarh, 2023; Sun et al., 2024). By restoring the dynamic range of intermediate activations, the proposed compensation alleviates pruning-induced distortion while retaining computational complexity comparable to heuristic pruning methods.

Our contributions are summarized as follows:

- We propose a post-training pruning framework that uses first-order statistical correction for both parameter selection and post-pruning weight adjustment.
- We introduce a variance-calibrated importance metric that corrects activation-induced bias in magnitude-based pruning without relying on gradient or second-order information.
- We develop an effective energy compensation method that restores activation scale after pruning. It can be applied independently of the pruning criterion, with computational cost comparable to heuristic methods in practice.
- We evaluate the proposed method on multiple LLMs and sparsity settings, achieving consistent improvements over existing post-training pruning methods (MetaAI, 2023b,a; Qwen, 2025).

2 Preliminaries

Post-training pruning is a widely used approach for compressing LLMs without the high cost of retraining (Frantar and Alistarh, 2023; Sun et al., 2024). We consider a pretrained LLM composed of stacked linear layers. Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ denote the pretrained weight matrix of a given linear layer. Post-training pruning methods construct a binary mask $M \in \{0,1\}^{d_{\text{out}} \times d_{\text{in}}}$, where $M_{ij} = 0$ indicates that the corresponding weight is pruned. By applying the mask the pruned weight matrix can be obtained as $\widetilde{W} = M \odot W$.

Heuristic-based weight pruning Early magnitude-based methods (Han et al., 2015; Cun et al., 1990) prune weights with small absolute values under the assumption that weight importance is independent of input data. More recent approaches incorporate activation statistics to better capture feature importance. For example, Wanda (Sun et al., 2024) computes importance scores using the product of weight magnitude and input activation norm, $|W_{ij}| \cdot \|X_j\|_2$. However, activation magnitudes in LLMs can be highly heterogeneous (Dettmers et al., 2022; Xiao et al., 2024; Kovaleva et al., 2021), making such heuristics sensitive to activation outliers. As a result, structurally insignificant weights may receive high importance scores due to amplification by a small number of dominant input features (Timkey and van Schijndel, 2021).

Reconstruction-based weight adjustment To address performance degradation after pruning, reconstruction-based methods further adjust the remaining weights in $\widetilde{W}$. Methods such as SparseGPT (Frantar and Alistarh, 2023) and its variants (Ma et al., 2023; van der Ouderaa et al., 2024; Frantar et al., 2023a,b; Hubara et al., 2021) formulate pruning as a reconstruction problem and update the retained weights to minimize the error between the original and pruned layer outputs. These methods typically rely on second-order information, using approximations of the Hessian to guide weight updates. While effective, the required matrix factorizations and iterative updates cause significant computational and memory overhead.

3 Method

We present a post-training pruning method based on first-order activation statistics that addresses two limitations of previous approaches (Frantar and Alistarh, 2023; Sun et al., 2024; Han et al., 2015): activation-induced bias in magnitude-based weight selection and signal energy mismatch after pruning. The method handles both issues directly using closed-form statistical quantities, without gradient-based optimization or second-order information.

3.1 Variance-Calibrated Weight Selection

Magnitude-based heuristic pruning methods such as Wanda (Sun et al., 2024) are sensitive to activation outliers (Dettmers et al., 2022; Kovaleva et al., 2021), which can inflate importance scores for unstable weight columns. We address this effect using Column-Variance Reweighting (CVR), which calibrates column-wise importance scores by weight variance and rectified channel activation.

Specifically, given the weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and its input activation X, feature channels with high weight variance tend to exhibit heterogeneous and unstable distributions, while activation outliers can be directly characterized through activation dispersion.

Therefore, based on weight W, we compute the variance of each input column as:

$$v_j = \mathrm{Var}_{i=1}^{d_{\text{out}}}(W_{ij}),$$

and define a weight-based calibration factor:

$$c_j = (v_j + \varepsilon)^{-\alpha/2},$$

where $\alpha \geq 0$ controls the reweighting strength and ε is a small constant for numerical stability.

At the same time, based on activation X, we estimate the channel-wise activation variance:

$$v_{x_j} = \mathbb{E}[x_j^2] - \mathbb{E}[x_j]^2,$$

and compute an activation-based calibration factor:

$$a_j^{(\text{var})} = v_{x_j}^{1/4}.$$

The proposed CVR rescales the column-wise importance scores using these calibration factors and the final weight importance is given by:

$$S_{ij} = |W_{ij}| \cdot a_j^{(\text{var})} \cdot c_j.$$

In contrast, Wanda defines a pruning metric based on per-channel activation statistics derived from the second moment, corresponding to an activation factor of the form $a_j^{(\text{var})} \propto \sqrt{\mathbb{E}[x_j^2]}$. This can amplify all weights in channels with large activation variance, including insignificant parameters. By explicitly accounting for activation variance and incorporating weight variance through c_j, CVR suppresses activation-dominated channels and reduces false amplification. Since c_j is computed solely from pretrained weights, CVR has no additional activation or gradient computation.

To verify that CVR reduces activation-dominated importance concentration, we measure the importance distribution on a representative linear layer of LLaMA-2-7B in Table 1. Compared with Wanda, CVR reduces the top-1% channel importance share from 66.8% to 49.8%, increases the prune rate within the top-5% highest-importance channels from 3.9% to 6.2%, and lowers the Gini coefficient from 0.88 to 0.75. This indicates that CVR spreads importance across more channels and avoids almost entirely preserving weights in a small set of activation-dominated channels.

Table 1: Effect of CVR on importance concentration in a representative linear layer of LLaMA-2-7B. CVR reduces channel concentration and increases pruning within the highest-importance channels.

Method	Top-1% share	Top-5% prune	Gini
Wanda	66.8%	3.9%	0.88
+CVR	49.8%	6.2%	0.75

3.2 Energy-Compensated Weight Adjustment

After pruning, the mask M is fixed and the remaining weights in $\tilde{W} = M \odot W$ are adjusted to obtain corrected weights W'. Pruning changes the aggregate signal strength of weight vectors, causing distributional distortions that can accumulate across layers. Two effects are particularly relevant (Frantar and Alistarh, 2023; Xiao et al., 2024).

Mean shift. Removing a subset of weights disrupts the balance between positive and negative contributions, making offset of weight vectors sensitive to sparsity patterns.

Energy collapse. Pruning reduces the centered ℓ_2 energy of weight vectors, attenuating the magnitude of signals propagated through the network even when structurally important weights are retained (Dettmers et al., 2022).

Existing reconstruction-based methods such as SparseGPT adjust pruned weights through reconstruction objectives, without explicitly correcting these distributional distortions. In contrast, we apply a lightweight post-pruning correction that matches the centered energy of the pruned weights to that of the original pretrained weights, using the original mean as a stable reference. The correction is applied independently along the input (column-wise) and output (row-wise) dimensions.

Intuition. Column-wise correction compensates for energy loss in pruned input connections, restoring the scale of signals entering each output unit. Row-wise correction compensates for energy loss in outgoing connections, restoring the scale of signals produced by each output unit. Applying both preserves the overall signal balance of the layer.

Column-wise correction. The column-wise mean of the original weights is

$$\mu_j^{\text{col}} = \frac{1}{d_{\text{out}}} \sum_{i=1}^{d_{\text{out}}} W_{ij}.$$

Algorithm 1 Post-Training Pruning with Variance-Calibrated Selection and Energy Matching

Input: Pretrained weights W, calibration data $\mathcal{D}$ (for mask construction), target sparsity (or $n:m$ pattern)
Output: Sparse corrected weights W'

1: **Procedure: Mask construction (Sec. 3.1)**
2: Collect activation statistics on $\mathcal{D}$ and compute activation calibration factors $a_j^{(\mathrm{var})}$.
3: Compute weight variance calibration factors $\{c_j\}$.
4: Compute importance scores $S_{ij} = |W_{ij}| \cdot a_j^{(\mathrm{var})} \cdot c_j$ and construct pruning mask M.
5: **Procedure: Post-pruning adjustment (Sec. 3.2)**
6: Apply mask: $\widetilde{W} \leftarrow M \odot W$.
7: Apply column-wise correction to $\widetilde{W}$.
8: Apply row-wise correction to $\widetilde{W}$ and obtain W'.

Using this mean for centering, the centered column-wise energies are defined as:

$$E_{\text{orig}}^{\text{col}}(j) = \sum\nolimits_{i=1}^{d_{\text{out}}} (W_{ij} - \mu_j^{\text{col}})^2,$$
$$E_{\text{pruned}}^{\text{col}}(j) = \sum\nolimits_{i=1}^{d_{\text{out}}} (\tilde{W}_{ij} - \mu_j^{\text{col}})^2.$$

We apply the transformation:

$$\tilde{W} \leftarrow (\tilde{W} - \mu^{\text{col}}) \odot s^{\text{col}} + \mu^{\text{col}},$$
$$s_j^{\text{col}} = \sqrt{E_{\text{orig}}^{\text{col}}(j) / \left(E_{\text{pruned}}^{\text{col}}(j) + \varepsilon\right)},$$

where ε is a small constant for numerical stability. s_j^{col} are clamped to a fixed range to avoid over-amplification in highly sparse columns. The pruning mask is re-applied to preserve exact sparsity.

Row-wise correction. A similar procedure is applied along the output dimension. Let

$$\mu_i^{\text{row}} = \frac{1}{d_{\text{in}}} \sum\nolimits_{j=1}^{d_{\text{in}}} W_{ij}$$
$$E_{\text{orig}}^{\text{row}}(i) = \sum\nolimits_{j=1}^{d_{\text{in}}} (W_{ij} - \mu_i^{\text{row}})^2,$$
$$E_{\text{pruned}}^{\text{row}}(i) = \sum\nolimits_{j=1}^{d_{\text{in}}} (\tilde{W}_{ij} - \mu_i^{\text{row}})^2.$$

denote the row-wise mean of the original weights and centered energies. The corrected weights are:

$$W' = (\tilde{W} - \mu^{\text{row}}) \odot s^{\text{row}} + \mu^{\text{row}},$$
$$s_i^{\text{row}} = \sqrt{E_{\text{orig}}^{\text{row}}(i) / \left(E_{\text{pruned}}^{\text{row}}(i) + \varepsilon\right)},$$

and the pruning mask is re-applied. In practice, column-wise correction is applied first, followed by row-wise correction.

Since the proposed compensation depends only on post-pruning weights and closed-form statistics, it is independent of pruning criterion and can be applied after mask construction, making it compatible with existing weight-pruning methods.

3.3 Algorithm Summary

Algorithm 1 summarizes the proposed post-training pruning pipeline. The method proceeds in two stages. It first collects first-order activation statistics on a pretrained model and applies variance-calibrated importance scoring (Sec. 3.1) to construct the pruning mask (Frantar and Alistarh, 2023; Sun et al., 2024). In the second stage, it performs mean-shifted energy matching (Sec. 3.2) to correct pruning-induced distortion in remaining weights using closed-form updates (Hassibi et al., 1993).

4 Experiment

4.1 Experimental Setup

Models. We evaluate our method on two representative LLM families: LLaMA-2 (MetaAI, 2023a) and Qwen2.5 (Qwen, 2025), covering a range of architectures and model scales. Specifically, we report results on LLaMA-2 (7B, 13B, 70B) and Qwen2.5 (7B, 14B, 32B, 72B).

Pruning Methods. We compare the proposed method against widely used post-training pruning baselines, including Wanda (Sun et al., 2024), RIA (Zhang et al., 2024), and SparseGPT (Frantar and Alistarh, 2023). All baseline methods are evaluated using their original implementations and recommended settings; RIA is used with its full pipeline (relative importance + channel reallocation + LSA refinement).

Our method combines variance-calibrated weight selection (CVR) with energy-compensated weight adjustment and is denoted as Ours (CVR+EC). The EC weight adjustment can also be used as a plug-in correction on top of any baseline mask, denoted "+EC" in Section 4.4.

Calibration Data. For methods that require calibration data for mask construction (e.g., Wanda and SparseGPT), we follow the same protocol as Wanda (Sun et al., 2024). Specifically, we sample 128 sequences from the C4 dataset (Raffel et al., 2023) to collect activation statistics. Our method does not rely on additional calibration data beyond that used for mask construction.

Evaluation Metrics. We evaluate pruned models using perplexity on WikiText2 (Merity et al., 2016) for language modeling performance and average

Method	Weight Update	Sparsity	LLaMA-2			Qwen2.5			
			7B	13B	70B	7B	14B	32B	72B
Pretrained Dense	–	0	5.12	4.57	3.12	6.85	5.29	5.02	3.88
Wanda	No	50%	6.41	5.46	3.98	10.60	8.24	6.30	5.22
RIA	No	50%	6.48	5.61	3.92	7.78	6.88	5.89	4.88
SparseGPT	Yes	50%	6.51	5.63	3.98	7.83	**6.70**	**5.86**	4.89
Ours (CVR+EC)	Yes	50%	**6.19**	**5.32**	**3.81**	**7.67**	6.78	5.91	**4.79**
Wanda	No	4:8	7.51	6.14	4.47	19.49	19.79	7.07	5.93
RIA	No	4:8	8.13	6.62	4.37	9.56	8.26	6.51	5.54
SparseGPT	Yes	4:8	8.12	6.60	4.59	8.77	7.84	6.48	5.59
Ours (CVR+EC)	Yes	4:8	**7.05**	**5.86**	**4.24**	**8.56**	**7.60**	**6.42**	**5.33**
Wanda	No	2:4	9.51	7.31	5.16	61.50	53.41	8.08	6.69
RIA	No	2:4	11.22	8.52	4.91	13.38	11.57	7.65	6.23
SparseGPT	Yes	2:4	10.17	8.32	5.40	10.35	9.30	7.19	6.35
Ours (CVR+EC)	Yes	2:4	**8.17**	**6.50**	**4.63**	**9.67**	**8.43**	**7.02**	**5.86**

Table 2: Perplexity (↓) on WikiText-2 under different sparsity settings. **Bold**: best result in each column.

zero-shot accuracy on a suite of downstream tasks, including BoolQ (Clark et al., 2019), RTE (Wang et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC (Clark et al., 2018), and OBQA (Mihaylov et al., 2018). All evaluations are conducted without retraining or fine-tuning unless explicitly stated.

4.2 Language Modeling

We evaluate language modeling perplexity on WikiText2 (Merity et al., 2016). Table 2 reports results under 50% unstructured sparsity and structured 4:8/2:4 sparsity across LLaMA-2 and Qwen2.5 models from 7B to 72B. RIA is evaluated using its full published pipeline, including relative-importance scoring, heuristic channel reallocation, and LSA refinement for N:M sparsity.

Our method gives the lowest perplexity in most settings. Under 50% unstructured sparsity, it is best on all LLaMA-2 models and on Qwen2.5 7B/72B, while remaining within 0.1 PPL of SparseGPT on Qwen2.5 14B/32B. Under structured sparsity, it is best in all 4:8 and 2:4 settings except Qwen2.5 32B at 4:8, where RIA is ahead by 0.05 PPL. The gains are larger at 2:4 sparsity, where our method improves the best baseline by 0.36 PPL on LLaMA-2 13B, 0.49 PPL on LLaMA-2 7B, and 0.68 PPL on Qwen2.5 7B.

Overall, activation-based mask selection alone leaves a clear perplexity gap, especially under structured sparsity on Qwen2.5. The proposed closed-form post-pruning correction closes much of this gap while keeping computational cost comparable to Wanda (Section 4.5).

Table 3: Zero-shot average accuracy (%) on LLaMA-2 7B. Higher is better.

Method	50%	4:8	2:4
Dense		59.71	
Wanda	56.24	52.49	48.75
RIA	60.74	56.34	51.45
SparseGPT	60.70	56.68	54.51
Ours (CVR+EC)	**60.95**	**59.30**	**56.46**

4.3 Zero-shot Evaluation

We further evaluate the pruned LLaMA-2 7B models on seven zero-shot tasks following Wanda (Sun et al., 2024): BoolQ, RTE, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA. Table 3 reports the average accuracy under 50% unstructured, 4:8, and 2:4 sparsity. Full results across all model sizes are provided in Appendix C.

The zero-shot results on LLaMA-2 7B are consistent with the perplexity trends. Our method achieves the highest average accuracy under all three sparsity settings. At 2:4 sparsity, our method improves over SparseGPT by 1.95 points and over Wanda by 7.71 points.

4.4 EC as a Plug-in Post-Pruning Correction

Energy compensation (EC) uses only post-pruning weights and closed-form statistics of the original weights, making it independent of the pruning criterion. We verify this by applying EC to Wanda and RIA on LLaMA-2 13B and Qwen2.5 14B. As

Table 4: Effect of adding EC on top of different pruning baselines. WikiText-2 perplexity (↓). **Bold**: cell where adding EC reduces perplexity.

Baseline	Sp.	LLaMA-2 13B		Qwen2.5 14B	
		base	+EC	base	+EC
Wanda	50%	5.46	**5.46**	8.24	**7.30**
RIA	50%	5.61	**5.41**	6.88	**6.74**
Wanda	4:8	6.14	**6.03**	19.79	**7.85**
RIA	4:8	6.19	**5.79**	7.72	**7.29**
Wanda	2:4	7.31	**6.80**	53.41	**8.65**
RIA	2:4	8.52	**6.71**	11.57	**8.33**

Table 5: Runtime comparison for post-training pruning methods. Full-model pruning time is measured on LLaMA-2-7B under 2:4 sparsity on a single NVIDIA L40S GPU. Speedup is measured relative to SparseGPT.

Method	W. Update	Time (s)↓	Speedup ↑
Wanda	✗	168	87×
RIA	✗	1,680	9×
SparseGPT	✓	14,834	1.00×
Ours	✓	170	87×

shown in Table 4, EC reduces perplexity in all 12 settings, with the largest gain from 53.41 to 8.65 for Wanda at 2:4 on Qwen2.5 14B. These results show that EC can be used as a plug-in correction after mask selection.

4.5 Efficiency and Runtime Analysis

Table 5 compares full-model pruning time on LLaMA-2-7B under 2:4 sparsity. Wanda is fastest since it uses mask selection only, while RIA adds channel reallocation and LSA refinement. SparseGPT is slower due to second-order statistics and reconstruction updates. Our method applies a closed-form correction and achieves Wanda-level runtime without additional calibration.

4.6 Unstructured Sparsity Robustness

Table 6 evaluates robustness to high unstructured sparsity on LLaMA-2-13B using perplexity. As sparsity increases, activation-based heuristics degrade rapidly, with Wanda and RIA reaching 45.77 and 50.85 PPL at 70% sparsity. SparseGPT is more stable because it performs reconstruction-based weight updates.

Our method gives the lowest perplexity across the evaluated sparsity range, including at high sparsity. This shows that closed-form correction of pruning-induced distribution shift improves robustness without second-order reconstruction.

Table 6: Perplexity (↓) of different pruning methods across unstructured sparsity levels on LLaMA-2-13B.

Method	10%	30%	50%	60%	70%
Wanda	4.60	4.79	5.59	7.02	45.77
RIA	4.58	**4.71**	5.46	7.60	50.85
SparseGPT	4.59	4.78	5.63	7.74	18.04
Ours	**4.57**	4.72	**5.32**	**6.33**	**10.52**

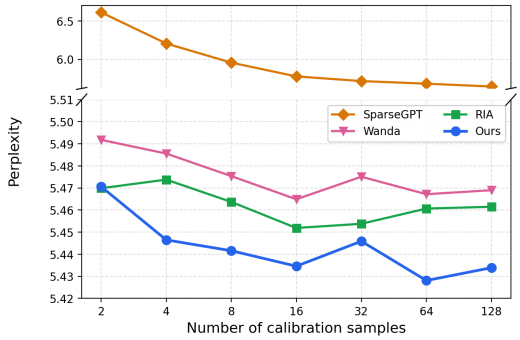

Figure 2: Perplexity (↓) across calibration set sizes on LLaMA-2-13B with 50% unstructured sparsity.

4.7 Effect of Calibration Set Size

Figure 2 evaluates calibration set size under 50% unstructured sparsity on LLaMA-2-13B. The proposed method gives lower perplexity than the baselines across all sizes and remains stable with limited calibration data.

We also test calibration-source robustness under 2:4 sparsity on LLaMA-2-7B. As shown in Table 7, our method has only 0.10 perplexity variation across C4, WikiText-2, and Pile, compared with 0.35 for Wanda, 0.32 for RIA, and 2.23 for SparseGPT, while zero-shot accuracy is unchanged. These results suggest that the correction depends mainly on intrinsic first-order weight statistics rather than a specific calibration distribution.

Table 7: Sensitivity to calibration dataset on LLaMA-2-7B (2:4 sparsity, WikiText-2 perplexity ↓). Max Diff is the range across the three sources.

Method	C4	Wiki	Pile	Max Diff
Ours (CVR+EC)	**8.16**	**8.06**	**8.16**	**0.10**
Wanda	9.51	9.16	9.49	0.35
RIA	10.56	10.24	10.54	0.32
SparseGPT	10.36	8.13	10.60	2.23

4.8 Ablation Study

Table 8 reports ablations on LLaMA-2 7B and 13B. Starting from Wanda, adding variance-calibrated reweighting (CVR) consistently reduces perplex-

Table 8: Effect of the variance reweighting strength α in CVR on WikiText-2 perplexity (50% unstructured sparsity). Lower is better. α=0 recovers the Wanda baseline; α=0.5 is used as default.

Configuration	7B	13B
Wanda (baseline)	6.42	5.46
+ CVR ($\alpha = 0$)	6.41	5.46
+ CVR ($\alpha = 0.25$)	6.35	5.44
+ CVR ($\alpha = 0.5$)	**6.31**	**5.43**
+ CVR ($\alpha = 0.75$)	6.32	5.44
+ CVR ($\alpha = 1.0$)	6.33	5.47
+ EC_{col}	6.21	5.33
+ EC_{row}	6.19	5.32

ity, showing that variance correction improves activation-based mask selection. Column-wise energy compensation (EC_{col}) further improves performance by correcting pruning-induced energy loss along input dimensions. Adding row-wise compensation (EC_{row}) gives the best results, reaching 6.19 / 5.32 PPL on 7B / 13B and improving over Wanda by 0.23 / 0.14 PPL.

We also vary the CVR strength α. Setting α=0 recovers the Wanda baseline, while $\alpha \in [0.25, 0.75]$ improves perplexity. The best result is obtained at α=0.5 on both models, and the variation across tested values is within 0.1 PPL. We therefore use α=0.5 in all remaining experiments.

4.9 Extended Experimental Results

Additional results are provided in the appendix. Appendix B reports per-layer matrix multiplication latency and peak GPU memory under 2:4 sparsity, confirming that the choice of pruning method does not affect post-deployment inference cost. Appendix D combines the proposed pruning with GPTQ quantization at INT8/INT4/INT3, and shows that the pruned model produced by our method retains the lowest WikiText-2 perplexity across all bit-widths. Appendix E provides long-form generation samples on DeepSeek-R1-Distill-Qwen-7B under 50% and 2:4 sparsity, comparing our method against Wanda, RIA, and SparseGPT.

5 Related Work

Post-training pruning reduces LLM inference cost without additional training. Existing methods fall into two groups. The first group focuses on mask selection, using weight magnitude (Han et al., 2015, 2016; Zhu and Gupta, 2017; Blalock et al., 2020; Molchanov et al., 2019), activation-aware scores (Sun et al., 2024; Molchanov et al., 2017; Lee et al., 2019), channel-wise normalization (Zhang et al., 2024), or pre-pruning rotations (Gu et al., 2025). The second group treats pruning as reconstruction and updates the remaining weights after pruning, using second-order information (Frantar and Alistarh, 2023; Ma et al., 2023; van der Ouderaa et al., 2024; Frantar et al., 2023b; Singh and Alistarh, 2020; Hubara et al., 2021).

Our work is complementary to both directions. Instead of changing only the pruning criterion or solving a reconstruction problem, we apply a closed-form post-pruning correction based on weight and activation statistics. This is motivated by prior observations that channel-wise heterogeneity and activation outliers strongly affect compression behavior (Bondarenko et al., 2021; Puccetti et al., 2022; Kovaleva et al., 2021; Timkey and van Schijndel, 2021; Wei et al., 2023; Dettmers et al., 2022; Xiao et al., 2024). A more detailed discussion of related work is provided in Appendix A.

6 Conclusion

We presented a post-training pruning method for LLMs based on lightweight statistical correction. The method combines variance-calibrated importance scoring with post-pruning energy compensation, using only first-order statistics and requiring no retraining, gradients, or second-order information. Across LLaMA-2 and Qwen2.5, it improves pruning performance under both unstructured and structured sparsity with runtime comparable to heuristic methods. Energy compensation also works as a plug-in correction for existing pruning baselines, reducing perplexity without changing the mask. Future work includes joint pruning–quantization and extension to encoder–decoder and multimodal models.

Limitations. The method relies on first-order weight and activation statistics, which may be less effective under severe distribution shift or highly non-stationary inputs. Performance also degrades at extreme sparsity, where model capacity becomes limited. On zero-shot tasks, SparseGPT remains stronger in some settings, suggesting that reconstruction objectives can preserve information not captured by first-order correction alone. Our evaluation is limited to decoder-only LLMs; other architectures are left for future work.

Ethical Considerations

This paper presents work whose goal is to advance the field of machine learning by studying efficiency aspects of large language model compression. The proposed approach focuses on improving post-training pruning through lightweight statistical correction and is intended as a methodological contribution to model compression.

The techniques studied in this work operate entirely on publicly available pretrained models and standard benchmark datasets, and do not involve the collection or processing of personal data. Model compression reduces the computational and energy cost of LLM inference, which may contribute to more efficient and accessible deployment of language technologies.

The proposed approach does not introduce new data sources, learning paradigms, or application domains. As such, we do not anticipate societal or ethical impacts beyond those already associated with the broader development and use of large language models.

References

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *Preprint*, arXiv:2003.03033.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. *Preprint*, arXiv:2109.12948.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *Preprint*, arXiv:2303.12712.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

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. *Preprint*, arXiv:1803.05457.

Yann Le Cun, John S. Denker, and Sara A. Solla. 1990. *Optimal brain damage*, page 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

DeepMind. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Preprint*, arXiv:2208.07339.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Preprint*, arXiv:1803.03635.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2021. Pruning neural networks at initialization: Why are we missing the mark? *Preprint*, arXiv:2009.08576.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *Preprint*, arXiv:2301.00774.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023a. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. 2023b. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Preprint*, arXiv:2208.11580.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *Preprint*, arXiv:1902.09574.

Tianteng Gu, Bei Liu, Bo Xiao, Ke Zeng, Jiacheng Liu, and Yanmin Qian. 2025. Denoiserotator: Enhance pruning robustness for llms via importance concentration. *Preprint*, arXiv:2505.23049.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Preprint*, arXiv:1510.00149.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. *Preprint*, arXiv:1506.02626.

B. Hassibi, D.G. Stork, and G.J. Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

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

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Seffi Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find n:m transposable masks. *Preprint*, arXiv:2102.08124.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Preprint*, arXiv:1502.03167.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. 2021. Bert busters: Outlier dimensions that disrupt transformers. *Preprint*, arXiv:2105.06990.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2019. Snip: Single-shot network pruning based on connection sensitivity. *Preprint*, arXiv:1810.02340.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. 2016. Revisiting batch normalization for practical domain adaptation. *Preprint*, arXiv:1603.04779.

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

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

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

MetaAI. 2023a. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

MetaAI. 2023b. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *Preprint*, arXiv:1809.02789.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *Preprint*, arXiv:2104.08378.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. *Preprint*, arXiv:1906.10771.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. *Preprint*, arXiv:1611.06440.

OpenAI. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. 2022. Outlier dimensions that disrupt transformers are driven by frequency. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, page 1286–1304. Association for Computational Linguistics.

Qwen. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. *Preprint*, arXiv:2003.02389.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *Preprint*, arXiv:1907.10641.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020a. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020b. Movement pruning: Adaptive sparsity by fine-tuning. *Preprint*, arXiv:2005.07683.

Sidak Pal Singh and Dan Alistarh. 2020. Woodfisher: Efficient second-order approximation for neural network compression. *Preprint*, arXiv:2004.14340.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *Preprint*, arXiv:1906.02243.

Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. *Preprint*, arXiv:1607.01719.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. *Preprint*, arXiv:2306.11695.

William Timkey and Marten van Schijndel. 2021. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. *Preprint*, arXiv:2109.04404.

Tycho F. A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M. Asano, and Tijmen Blankevoort. 2024. The llm surgeon. *Preprint*, arXiv:2312.17244.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2023. Neurons in large language models: Dead, n-gram, positional. *Preprint*, arXiv:2309.04827.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *Preprint*, arXiv:2304.09145.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2024. Smoothquant: Accurate and efficient post-training quantization for large language models. *Preprint*, arXiv:2211.10438.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *Preprint*, arXiv:1905.07830.

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

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *Preprint*, arXiv:1710.01878.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

A Extended Related Work

A.1 Post-Training Pruning for LLMs

Post-training pruning is commonly used to reduce the size and inference cost of LLMs without additional training. Early approaches rely on metrics derived from weight magnitude (Han et al., 2015, 2016; Zhu and Gupta, 2017; Blalock et al., 2020; Molchanov et al., 2019). Recent methods such as Wanda (Sun et al., 2024) use activation information (Molchanov et al., 2017; Lee et al., 2019) to better capture input-dependent behavior, where weight–activation statistics are adopted to guide and improve pruning.

Another line of work treats pruning as a reconstruction issue. Methods such as SparseGPT (Frantar and Alistarh, 2023) and related extensions (Ma et al., 2023; van der Ouderaa et al., 2024; Frantar et al., 2023a,b; Singh and Alistarh, 2020; Hubara et al., 2021) update the remaining weights after pruning to reduce reconstruction error. These methods typically rely on second-order information and have higher computational and memory costs.

Some other works take alternative approaches to modify the pruning process itself. DenoiseRotator (Gu et al., 2025) applies orthogonal transformations before pruning to redistribute parameter importance. RIA (Zhang et al., 2024) proposes a relative importance metric based on channel-wise normalization to mitigate scale imbalance during pruning. These methods mainly focus on how importance is defined or adjusted prior to pruning (Gale et al., 2019).

A.2 Weight and Activation Statistics

The statistical properties of weights and activations in LLMs have been studied in the context of model robustness and model compression (Bondarenko et al., 2021; Puccetti et al., 2022). In quantization, prior work shows that activation outliers are common in transformer-based models (Kovaleva et al., 2021; Timkey and van Schijndel, 2021; Wei et al., 2023) and can strongly affect compression performance (Dettmers et al., 2022; Xiao et al., 2024). These studies highlight the role of channel-wise heterogeneity and extreme activation values in determining sensitivity under compression.

A.3 Distribution Alignment after Pruning

Preserving intermediate feature distributions is important for maintaining model performance after compression (Ioffe and Szegedy, 2015). Reconstruction-based pruning methods address this through explicit optimization objectives (Frantar and Alistarh, 2023; Renda et al., 2020). Related work in quantization and domain adaptation shows that matching first- and second-order feature statistics can also be effective (Li et al., 2016; Sun and Saenko, 2016). These methods use closed-form corrections rather than iterative optimization.

For LLM post-training pruning, most prior work focuses on mask selection or reconstruction-based weight updates. Lightweight post-pruning corrections based on feature distribution alignment remain less explored.

B Inference Latency & Memory Analysis

Tables 9 and 10 report per-layer matrix multiplication latency and peak GPU memory usage, respec-

tively, for different pruning methods on LLaMA-2-7B under 2:4 sparsity, measured on a single NVIDIA L40S GPU. Under a fixed sparsity pattern, inference latency and memory are determined primarily by the sparsity structure and the underlying sparse kernel implementation (e.g., CUTLASS), rather than by the specific pruning criterion used to construct the mask. Consequently, all methods with the same 2:4 sparsity pattern exhibit near-identical deployment characteristics.

Table 9: Per-layer matrix multiplication latency (ms) on LLaMA-2-7B. Speedup is Ours relative to the dense baseline; Wanda's speedup is within $\pm 0.01\times$ of Ours and omitted.

Layer	Dense	Wanda	Ours	Speedup
q/k/v/o_proj	0.0453	0.0487	0.0480	0.94×
up/gate_proj	0.1635	0.1030	0.1038	1.58×
down_proj	0.1502	0.0916	0.0918	1.64×

Table 10: Peak GPU memory usage (GB) during inference on LLaMA-2-7B.

Method	Peak Memory (GB)
Wanda	7.56
Ours (CVR+EC)	7.56
SparseGPT	7.66

These results confirm that the choice of pruning method does not affect post-deployment inference efficiency under a fixed sparsity pattern. Latency gains are realized for computationally heavier layers (up/gate/down projections, $\approx 1.6\times$ speedup), while attention projection layers show marginal overhead due to their smaller size relative to the kernel launch cost. Peak memory is near-identical across all pruning methods and slightly lower than SparseGPT, which retains additional buffers during its reconstruction-based weight update step.

C Zero-shot Evaluation

We evaluate the pruned models on seven zero-shot tasks following Wanda (Sun et al., 2024): BoolQ, RTE, HellaSwag, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA. Table 11 reports the average accuracy under 50% unstructured, 4:8, and 2:4 sparsity.

Zero-shot accuracy shows smaller separation than perplexity. Our method obtains the best average accuracy on several LLaMA-2 settings, including all three sparsity patterns for 7B, 4:8 for 70B, and 2:4 for 13B. In the remaining cases, SparseGPT or RIA is usually strongest, with SparseGPT leading our method by 0.4–3.0 points in some columns.

Under structured sparsity, the gap between heuristic masking methods and correction-based methods becomes clearer. For example, at 2:4 on Qwen2.5 7B, Wanda drops to 60.98%, while our method retains 61.96%. This suggests that structured pruning requires not only mask selection but also post-pruning correction to preserve task-relevant representations.

D Combination with Quantization

The proposed pruning and EC correction operate on the dense weight matrix and do not depend on the numerical precision of the weights. We test whether the pruned model produced by our method remains amenable to post-training quantization by passing the pruned LLaMA-2-13B (50% unstructured sparsity) through standard GPTQ (Frantar et al., 2023a) at INT8/INT4/INT3, plus a Round-To-Nearest (RTN) INT4 reference, and comparing against the same quantization pipeline applied to the pruned models produced by Wanda, RIA, and SparseGPT. Calibration uses 128 C4 samples and group size 128; INT4/INT3 GPTQ runs use `-true-sequential -act-order`.

Figure 3 plots WikiText-2 perplexity across the five settings (FP16, RTN INT4, GPTQ INT8, GPTQ INT4, GPTQ INT3). Across all five settings, the pruned model produced by our method gives the lowest perplexity. The gap between methods widens as bit-width decreases: at GPTQ INT3 our method is 0.20 PPL lower than RIA, 0.27 PPL lower than SparseGPT, and 0.47 PPL lower than Wanda. The relative ordering between the four pruning methods is preserved under quantization, indicating that the post-pruning weight distribution induced by EC remains numerically robust to low-bit conversion. This suggests EC and GPTQ are compatible and can be stacked without retraining or special quantization-aware adjustments.

E Long-form Generation Examples

To verify that the proposed method does not cause output degeneration (e.g., repetition or collapse) under aggressive pruning, we provide generation examples on DeepSeek-R1-Distill-Qwen-7B under three sparsity settings using the prompt: *"write a poem about ocean in 100 words."* For each setting we compare Wanda, RIA, SparseGPT, and our

Table 11: Zero-shot accuracy (%) under different sparsity settings. Higher is better. **Bold**: best result in each column.

Method	Weight Update	Sparsity	LLaMA-2			Qwen2.5			
			7B	13B	70B	7B	14B	32B	72B
Dense	–	0	59.71	63.03	67.08	72.18	75.81	75.12	78.69
Wanda	No	50%	56.24	60.83	67.03	66.77	69.79	71.28	73.70
RIA	No	50%	60.74	64.39	**71.49**	66.42	70.39	**71.31**	74.15
SparseGPT	Yes	50%	60.70	**65.55**	67.28	**67.49**	**70.99**	70.63	**74.58**
Ours (CVR+EC)	Yes	50%	**60.95**	64.31	70.78	66.66	68.91	70.37	73.64
Wanda	No	4:8	52.49	58.75	66.06	64.80	66.69	**70.91**	72.93
RIA	No	4:8	56.34	62.84	70.16	65.32	68.86	69.84	73.25
SparseGPT	Yes	4:8	56.68	**62.96**	65.84	**66.85**	**69.25**	70.47	**73.75**
Ours (CVR+EC)	Yes	4:8	**59.30**	62.27	**70.50**	65.49	68.14	69.53	72.64
Wanda	No	2:4	48.75	55.03	64.14	60.98	62.55	**68.67**	71.75
RIA	No	2:4	51.45	57.27	68.57	60.47	62.69	67.99	71.81
SparseGPT	Yes	2:4	54.51	59.92	63.89	**64.06**	**65.80**	66.65	**72.16**
Ours (CVR+EC)	Yes	2:4	**56.46**	**61.66**	**68.72**	61.96	63.29	67.36	71.34

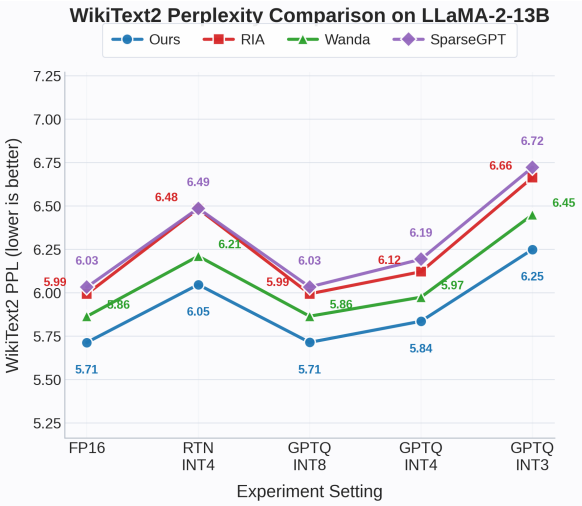

Figure 3: WikiText-2 perplexity on LLaMA-2-13B (50% unstructured sparsity) under different post-pruning quantization settings. Each curve corresponds to one pruning method (Ours, RIA, Wanda, SparseGPT); the x-axis sweeps quantization configurations (FP16, RTN INT4, GPTQ INT8/INT4/INT3). Lower is better.

method (CVR+EC), together with the unpruned dense model as a reference. Reasoning traces (`<think>...</think>`) are omitted; we show only the final poem.

Dense model (reference)

The sea's deep breath, a rhythmic sigh,
Through endless waves, where time does fly.
Salt-kissed air whispers tales of yore,
As ancient currents tell their lore.
Beneath the surface, secrets hide,
Mysteries lost to the tide's wide stride.
The endless blue, a boundless sky,
Where sailors brave the storms that vie.
With every wave, life finds its place,
In harmony, beyond our trace.
Oh, ocean vast, your beauty's might,
A canvas painted in the night.

50% unstructured sparsity

Wanda.

Deep blue unfolds its secrets slow,
Where sunken cities hold their lore,
Or ghostly currents softly glide below,
Reflecting stars beyond the sky.
Then tide rolls inland from afar,
A symphony across the shore,
While farthest depths conceal who knows what,
And lonely souls reflect no more.
Crashing waves repeat their endless song,
Like time itself, forever strong,
Though mirrored skies may seem along,
No harbor waits upon the wing.

RIA.

The ocean's heartbeat echoes deep,
Where ancient mysteries gently sleep.
Waves crash with rhythm, soft and bold,
A symphony only nature could behold.
Sunlight dances on the foam,
While salt-tinged air whispers home.
Mirrored stars in silver sheen,

A canvas where the dreams convene.
Listen close, the sea will tell
Of voyages held within its swell.

SparseGPT.

Deep blue stretches far and wide,
A canvas where the sky meets tide.
Whispers of waves on shores untold,
Stories ancient, never grown old.
Salt spray dances in the breeze,
Painting moments of fleeting ease.
Beneath the surface, secrets hide,
Worlds where mysteries gently glide.
From dawn's first light to twilight's hush,
The ocean breathes in endless rush.
Vast and free, it never sleeps,
Reflecting all our human deeps.

Ours (CVR+EC).

Deep blue stretches beyond the eye,
Where secrets hide beneath the sky.
A lonely shore against the sand,
Or distant ship upon the land.
Waves crash where the tempest rages,
And calm reflects across the ages.
Beneath the surface lies the story,
Of sailors lost and hope's discovery.
It whispers promises of shores unseen,
Reflects the sun, then darkens when
The tide turns inland, leaving only
The empty space where once there flowed
The endless water, vast and deep,
With currents whispering secrets keep.

2:4 structured sparsity

Wanda.

Beneath the sky vast and endless,
Salt whispers from the breeze blowing,
Waves crash shore-to-shore,
Building tension ever growing.
Sun's golden light paints evening skies,
A canvas richly painted with hues bright.
But softer, calmer, sand at my feet,
Coastal waters wait patient.
Humanity clings to skin like salt upon our own,
Yet we are drawn back to where the tide meets land.
Soaking into the rhythm of all things constant,
In the quiet depths, we find ourselves home.

RIA.

The ocean whispers, soft and low,
A symphony where currents flow.
Waves crash high, then sink below,
A rhythm steady, ebb and grow.
Salt-kissed wind carries tales of yore,
Of sailors brave on stormy shore.
Beneath the surface, secrets sleep,
In depths where ancient stories keep.
Reflecting skies of endless blue,
The sea holds dreams of me and you.

SparseGPT.

The ocean stretches wide and deep,
Where waves their endless secrets keep.
Salt-kissed winds across the shore,
Carry stories from days before.
Sunlight dances on the foam,
Painting paths from sea to home.
Sailors brave the rolling tide,
With courage as their faithful guide.
Mysteries dwell beneath the blue,
Where ancient creatures softly move.
The ocean whispers, vast and free,
A timeless song of you and me.

Ours (CVR+EC).

The ocean whispers secrets,
But the waves are crashing,
Leaving behind ripples,
Of blue water flowing.
It holds ancient stories,
In every tide,
From deep sea creatures,
To distant shores.
A lonely traveler,
With open arms,
Gazing at the horizon,
As night falls.
My heart beats with hope,
Like a beating pulse,
Reflecting on the shore,
And feeling free.

Across the three sparsity settings, the outputs from our method remain coherent, on-topic, and free of repetition or collapse, and are qualitatively comparable to the unpruned dense reference. Complete generation logs for all methods and sparsity patterns are available in the accompanying code release.