Efficient Plug-and-Play Weight Refinement for Sparse Large Language Models

Jingcheng Xie, Yinda Chen, Xiaoyu Liu, Yinglong Li, Haoyuan Shi, Zhiwei Xiong*

University of Science and Technology of China
{icarus0131, cyd0806, liuxyu, yllee, haoyuan.shi}@mail.ustc.edu.cn, zwxiong@ustc.edu.cn

Abstract

One-shot pruning efficiently compresses Large Language Models but produces coarse sparse weights, causing significant performance degradation. Traditional fine-tuning approaches to refine these weights are prohibitively expensive for large models. This highlights the need for a training-free weight refinement method that works seamlessly with one-shot pruning and can efficiently recover the lost performance. To tackle this problem, we propose Efficient Iterative Weight Refinement (EIWR), a lightweight, plug-and-play, and training-free method that refines pruned weights through layer-wise iterative optimization. EIWR achieves efficient weight refinement via three key components: a Global Soft Constraint that eliminates costly row-wise Hessian inversions and expands the solution space; a Historical Momentum Strategy that leverages one-shot pruning priors to accelerate convergence and enhance final performance; and Neumann Series Extrapolation that significantly speeds up per-iteration computation. As a result, EIWR enables effective weight refinement with minimal time and memory overhead. Extensive experiments on LLaMA2/3 and Qwen under different pruning strategies and sparsity levels demonstrate that our method can efficiently refine sparse weights and mitigate performance degradation. For example, on LLaMA2-7B under 70% sparsity, EIWR reduces perplexity by 15% compared with SparseGPT on the WikiText2 benchmark, with only 1.81 additional minutes of computation and 1GB of additional memory. The code is in https://github.com/xiejingcheng/EIWR.

Introduction

Large language models (LLMs) have shown strong performance across a wide range of natural language processing tasks (Touvron et al. 2023; Bai et al. 2023). However, their ever-growing parameter counts introduce substantial computational and memory burdens, making efficient deployment increasingly challenging. This has sparked significant interest in model compression techniques that can retain performance while reducing inference costs.

Among these compression techniques, *one-shot pruning* has emerged as a particularly compelling solution (Frantar and Alistarh 2022). Unlike traditional pruning methods, which require computationally expensive fine-tuning (Mocanu et al. 2018; Evci et al. 2020), one-shot methods produce sparse models in a single forward pass, thereby avoiding backpropagation and minimizing data usage. Recent techniques such as SparseGPT (Frantar and Alistarh 2023) and Wanda (Sun et al. 2023) have demonstrated that well-crafted heuristics can yield competitive sparsity-accuracy trade-offs with minimal computational overhead.

Despite these advances, current one-shot pruning methods suffer from a fundamental limitation: the sparse weights are generated via coarse layer-wise approximations, whose accuracy degrades as sparsity increases (Sun et al. 2023; Meng et al. 2024), resulting in a large performance gap between the pruned and original models. For instance, SparseGPT exhibits noticeable degradation beyond 50% sparsity. This degradation stems from the fact that these approximations deviate further from the optimal values as sparsity increases, leading to rapid performance deterioration.

While some approaches attempt to recover performance through fine-tuning (Huang et al. 2025b; Lu et al. 2024b; An et al. 2024; Kusupati et al. 2020), such solutions reintroduce the very computational costs that one-shot pruning aims to eliminate. To avoid such costs, recent studies have explored training-free alternatives. Among them, DsnoT (Zhang et al. 2023) adopts a training-free perspective and effectively refines the pruning mask via a grow-and-prune mechanism without backpropagation, but leaves the sparse weights largely unoptimized. This limitation highlights a broader research trend that decomposes pruning into two decoupled subproblems: *mask selection* and *weight reconstruction* (Blumensath and Davies 2008; Frantar and Alistarh 2023). While prior efforts have mainly focused on the former, the latter remains underexplored. The unrefined sparse weights obtained through one-shot pruning weight reconstruction can lead to substantial performance degradation. To address this, it is desirable for a weight refinement method to be both efficient and plug-and-play, ensuring compatibility with existing one-shot pruning methods with minimal overhead and allowing seamless integration with other training-free techniques, particularly those focused on mask refinement.

Nevertheless, designing such a method is non-trivial for several reasons. Although the weight refinement problem can be formulated as a linear least squares system, directly

*Corresponding author.

solving it at scale remains impractical due to the *Different Row-Hessian Challenge* (Frantar and Alistarh 2022), where inverting a separate Hessian matrix for each row of the weight matrix becomes prohibitively expensive. Additionally, naive solvers that disregard the prior knowledge introduced by one-shot pruning often suffer from poor convergence or suboptimal results.

To tackle these challenges, we propose Efficient Iterative Weight Refinement (EIWR), a method to recover pruned weights without backpropagation or extra data. Meanwhile, EIWR is decoupled from the one-shot pruning pipeline, making it plug-and-play. EIWR addresses the unique difficulties of one-shot pruning in large models via three technical innovations: a Global Soft Constraint that bypasses per-row Hessian bottlenecks, a Historical Momentum Strategy that leverages pruning priors for stable convergence, and a Neumann Series Extrapolation that reduces iteration cost. Together, these components enable EIWR to efficiently refine sparse weights in large models. Moreover, EIWR offers strong practical utility, especially due to its efficiency and plug-and-play property. EIWR can be easily integrated into pruning strategies and combined with other training-free refinement methods for further gains.

We evaluate the effectiveness of our method on various LLMs, including LLaMA2/3 and Qwen2.5, under different pruning strategies and sparsity levels. It is also extremely lightweight, requiring only 1.81 additional minutes and 1GB of extra memory to process LLaMA2-7B under 70% sparsity, while reducing perplexity by 15% compared with SparseGPT on the WikiText2 benchmark. Moreover, by combining EIWR with other training-free techniques, we achieve even better performance, which validates its strong plug-and-play and compatibility capabilities.

Our contributions can be summarized as follows:

- We propose EIWR, a lightweight, training-free, and plug-and-play method for refining pruned weights in LLMs without backpropagation or additional data, effectively recovering the lost performance after one-shot pruning.
- EIWR addresses key challenges in sparse weight refinement through three novel components: a *Global Soft Constraint* to bypass the Different Row-Hessian Challenge, a *Historical Momentum Strategy* to incorporate structural prior information from one-shot pruning, and a *Neumann Series Extrapolation* to enable fast, scalable optimization.
- EIWR demonstrates broad compatibility with existing one-shot pruning strategies and complements other training-free techniques, enabling compound performance improvements across diverse model architectures and sparsity levels.

Related Work

Network Pruning. Prior to the rise of large language models, network pruning was a key technique for model compression, aiming to remove redundant weights and reduce computation (Molchanov et al. 2019; He, Zhang, and Sun 2017; Hu, Zhu, and Chen 2024; Hubara et al. 2021a; Lebedev and Lempitsky 2016; Liu et al. 2018). Unstructured pruning eliminates individual weights based on importance scores. Magnitude Pruning (Han et al. 2015) removes weights with small absolute values but struggles under high sparsity due to ignoring weight correlations. Optimal Brain Surgeon (Hassibi, Stork, and Wolff 1993) uses second-order information via Hessians for precise pruning, but its $O(d^4)$ complexity limits scalability. AdaPrune (Hubara et al. 2021b) iteratively optimizes masks and reconstructs weights, while Dynamic Sparse Training (Mocanu et al. 2018; Evci et al. 2020) trains sparse networks from scratch by pruning small-magnitude weights and regrowing important ones. Structured pruning (Dong et al. 2024; Li et al. 2024a,b) removes higher-level units (e.g., channels or neurons) to facilitate hardware acceleration. Filter Pruning (Liu et al. 2016) eliminates entire convolutional kernels for inference efficiency, often at the expense of accuracy. Group Fisher Pruning (Liu et al. 2021) leverages Fisher information to evaluate channel importance and achieves over 50

Pruning for Large Language Models. Large Language Models (Zhang et al. 2022; Kasneci et al. 2023; Naveed et al. 2023; Chang et al. 2024) contain billions of parameters, making fine-tuning-based pruning methods computationally infeasible. Recent work thus focuses on one-shot pruning tailored for LLMs (Huang et al. 2025a; Lee, Ajanthan, and Torr 2018). SparseGPT (Frantar and Alistarh 2023) uses a greedy approximation to address the Different Row-Hessian Challenge, but still incurs substantial reconstruction error. Wanda (Sun et al. 2023) introduces a lightweight criterion based on the product of weight magnitude and input activation norm. Several plug-and-play approaches enhance one-shot pruning performance. DSnoT (Zhang et al. 2023) introduces a training-free fine-tuning strategy by iteratively growing and pruning connections to optimize the sparse mask. While Alps (Meng et al. 2024) introduces improved conjugate gradient methods to optimize sparse weights, it is not plug-and-play, as it relies on a specially designed pruning framework (Boža 2024). OWL (Yin et al. 2023) and AlphaPruning (Lu et al. 2024a) propose dynamic sparsity allocation techniques that adapt pruning ratios based on layer-wise sensitivity analysis.

Method

Background and Motivation

One-shot pruning efficiently compresses large models by removing unimportant weights in a single pass without costly fine-tuning. Most methods formulate pruning as a layer-wise reconstruction problem: given calibration data, approximate the dense layer output using sparse weights by minimizing the ℓ_2 reconstruction error:

$$\min_{M_\ell, \hat{W}_\ell} \left\| W_\ell^{\text{dense}} X_\ell - \left((1 - M_\ell) \odot \hat{W}_\ell \right) X_\ell \right\|^2 \quad (1)$$

Here, $W_\ell^{\text{dense}} \in \mathbb{R}^{d_{row} \times d_{col}}$ is the dense weight matrix, $\hat{W}_\ell$ its pruned counterpart, $M_\ell \in \{0,1\}^{d_{row} \times d_{col}}$ the pruning mask, and $X_\ell \in \mathbb{R}^{NL \times d_{row}}$ the layer input with N calibration samples.

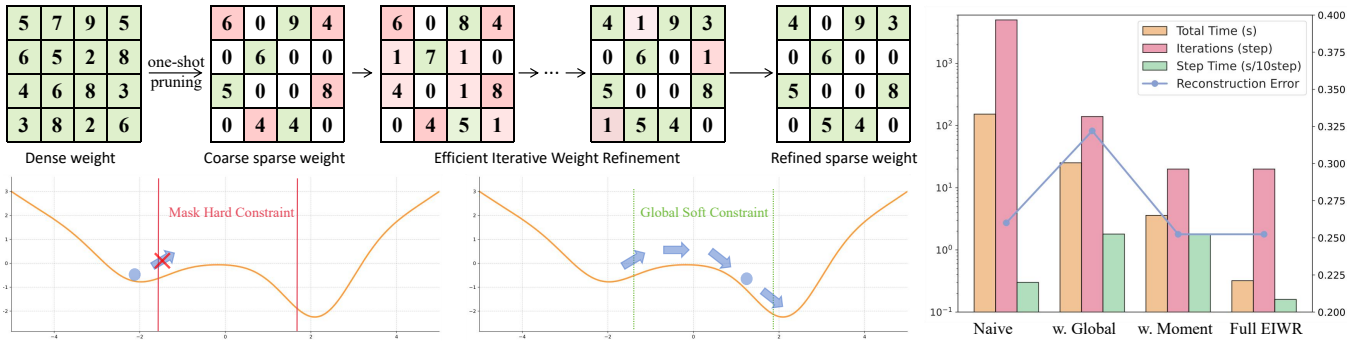

Figure 1: Overview of our proposed **EIWR** method. The goal is to iteratively refine the coarse sparse weights produced by one-shot pruning. During refinement, we replace the original *Mask Hard Constraint* with a *Global Soft Constraint* to alleviate the *Different Row-Hessian Challenge* and reduce computational overhead, while enabling exploration in a broader solution space. In addition, we adopt a *Historical Momentum Strategy* to leverage the structural prior from one-shot pruning for faster convergence, and apply *Neumann Series Extrapolation* to significantly reduce the per-iteration cost. The right panel shows the baseline naive backsolve method and the progressive assembly of the three components into the full EIWR framework.

In one-shot pruning, jointly optimizing $\hat{W}_\ell$ and M_ℓ is NP-hard. A common approach decomposes it into *Mask Selection* and *Weight Reconstruction*. While mask refinement has been extensively studied, weight reconstruction often relies on simple heuristics without further refinement, resulting in considerable performance loss.

We assume the pruning mask M_ℓ is fixed and focus on improving initial weights $\hat{W}_\ell^0$ by reducing the ℓ_2 reconstruction error. Under a fixed mask, Eq. (1) can turn into a linear least squares problem. A simple approach applies naive backsolve, computing optimal weights for each row independently:

$$w_{M_i}^i = (X_{M_i} X_{M_i}^T)^{-1} X_{M_i} \left(w_{\text{dense}}^i X_{M_i}^T \right) \tag{2}$$

where X_{M_i} denotes input features corresponding to non-pruned positions in row i. This avoids full fine-tuning overhead but suffers from two significant limitations.

First, it encounters the *Different Row-Hessian Challenge* (Frantar and Alistarh 2023). Since M_i varies across rows, the corresponding matrix $(X_{M_i} X_{M_i}^T)$ is also different for each row. As a result, every row requires an independent computation and inversion of $(X_{M_i} X_{M_i}^T)$, leading to a per-row complexity of $O(d_{\text{col}}^3)$ and a total complexity of $O(d_{\text{row}} \cdot d_{\text{col}}^3)$. Moreover, although both matrix multiplication and matrix inversion share the same asymptotic complexity of $O(d_{\text{col}}^3)$, on modern GPUs, inversion is typically an order of magnitude more expensive than multiplication. Second, this method ignores prior information from one-shot pruning, which encodes useful structural priors. These priors can guide the optimization trajectory, and disregarding them often leads to instability and inferior reconstruction quality.

Efficient Iterative Weight Refinement

To address the limitations of naive backsolve, including the high computational cost from the Different Row-Hessian Challenge and the disregard for prior information, we introduce Efficient Iterative Weight Refinement (EIWR). As shown in Figure 1 and detailed in Algorithm 1, EIWR is a lightweight, training-free method that refines pruned weights. It does not require backpropagation or labeled data and is designed to operate independently of the one-shot pruning pipeline, making it fully plug-and-play.

Algorithm 1: Efficient Iterative Weight Refinement

Require: W^0 (initial pruned weights), H (covariance matrix), M (pruning mask)
Ensure: $\hat{W}$ (refined weights)

$W \leftarrow W^0$
$\Lambda^0 \leftarrow 0$
$\alpha_0 \leftarrow \alpha_{\text{start}}$
$A_0 \leftarrow (1-\alpha_0)H + \alpha_0 I$
$X_0 \leftarrow A_0^{-1}$
$B \leftarrow HW_{\text{dense}}$
Compute X_1, X_2, X_3 via Neumann recurrence
for $k = 0$ to $T-1$ **do**
  $\alpha_k \leftarrow \alpha_{\max}(1 - e^{-\gamma k})$
  $\delta_k \leftarrow \alpha_k - \alpha_{k-1}$
  $B_k \leftarrow (1-\alpha_k-\delta_k)B + (\alpha_k+\delta_k)W - \Lambda^k$
  $\tilde{A}_k^{-1} \leftarrow X_0 + \delta_k X_1 + \delta_k^2 X_2 + \delta_k^3 X_3$
  $W \leftarrow B_k \tilde{A}_k^{-1}$
  $\Lambda^{k+1} \leftarrow \Lambda^k + \rho \cdot (M \odot W)$
end for
return $\hat{W} \leftarrow W - M \odot W$

Global Soft Constraint. A central challenge in post-pruning weight refining is the *Different Row-Hessian Challenge*, which arises from enforcing hard sparsity masks with row-wise heterogeneity. Specifically, each row of the weight matrix is pruned independently, resulting in distinct input subspaces for each least-squares subproblem. This leads to incompatible optimization conditions across rows and requires computing and inverting a separate covariance matrix $X_{M_i} X_{M_i}^T$ for every row i, which is prohibitively expensive in large models. Moreover, the lack of a shared solution space prevents assembling these subproblems into a unified

matrix formulation.

To resolve this scalability bottleneck, we introduce a Global Soft Constraint that transforms the row-wise recovery (Equation 2) into a unified matrix-level optimization. Instead of operating in disjoint pruned subspaces, we project the dense reconstruction into a shared covariance space:

$$W = (W_{\text{dense}} X X^T)(X X^T)^{-1}, \tag{3}$$

where $H = XX^T \in \mathbb{R}^{d_{row} \times d_{row}}$ is the global input covariance. This formulation bypasses the Different Row-Hessian Challenge by aligning all rows to the same projection space, thereby eliminating the need for row-wise matrix inversion. Notably, many pruning methods already compute H, allowing us to reuse this intermediate at no extra cost.

We further augment this formulation with a progressive soft regularization that enforces sparsity in a relaxed yet principled manner. An auxiliary matrix Λ^k is introduced to accumulate penalties on weights that violate the target sparsity:

$$W^{k+1} = \left[W_{\text{dense}} H - \Lambda^k\right] H^{-1}, \quad \Lambda^{k+1} = \Lambda^k + (M \odot W^k), \tag{4}$$

where $M \in \{0,1\}^{d_{row} \times d_{col}}$ is the pruning mask. This mechanism does not immediately zero out undesired weights, but gradually encourages compliance with the sparsity pattern over iterations.

Unlike conventional hard-masked recovery, our approach offers two key advantages: it eliminates the substantial and redundant computation overhead introduced by the *Different Row-Hessian* Problem, and it allows temporary deviations from the sparsity mask to enhance optimization flexibility. As the soft regularization gradually accumulates, the solution naturally aligns with the desired sparse structure. This synergy between global structural consistency and progressive enforcement leads to improved convergence stability and reconstruction quality.

Historical Momentum Strategy. While the Global Soft Constraint alleviates the computational burden of row-wise Hessian inconsistencies, it overlooks the structural prior encoded in the initial weights from one-shot pruning. These weights, such as those produced by SparseGPT, are not arbitrary but derived from second-order loss approximations, offering a structurally aligned and task-relevant starting point. Discarding this informative prior forces the optimization to restart from scratch, leading to slower convergence and degraded reconstruction quality under high sparsity.

To address this, we introduce a historical momentum strategy that reuses and integrates previous solutions into the reconstruction process. Specifically, we interpolate between the current gradient descent direction and the previous estimate:

$$W^{k+1} = \left[(1-\alpha) W_{\text{dense}} H + \alpha W^k - \Lambda^k\right] \left((1-\alpha) H + \alpha I\right)^{-1} \tag{5}$$

Here, the momentum coefficient α modulates the trade-off between data-driven updates and structural memory, acting as a stabilizer that smooths the optimization trajectory across iterations.

To further adapt to the varying dynamics of sparsity-aware optimization, we employ a progressively scheduled momentum, where α increases over time:

$$\alpha_k = \alpha_{\max} - (\alpha_{\max} - \alpha_{\text{start}}) \exp(-\gamma k), \tag{6}$$

where α_{start} and $\alpha_{\max}$ are the initial and final momentum strengths, and γ controls the growth rate. This dynamic schedule encourages free exploration in early iterations while gradually leveraging the pruning-informed initialization to guide the solution toward structurally coherent optima. As a result, the optimization becomes both more stable and more responsive to pruning priors.

Neumann Series Extrapolation. Although the previous components substantially reduce the number of required iterations, each update still involves inverting a dynamic matrix of the form $((1-\alpha_k)H + \alpha_k I)^{-1}$, which incurs a computational cost of $\mathcal{O}(d^3)$. This operation remains a scalability bottleneck for high-dimensional models.

However, since the momentum coefficient α_k increases gradually over iterations, the matrix $A(\alpha_k) = (1-\alpha_k)H + \alpha_k I$ changes smoothly over time. This observation motivates a more efficient strategy, namely *Neumann Series Extrapolation*. It uses the Neumann series to extrapolate the inverse, providing sufficient accuracy while reducing the computational complexity of matrix inversion from cubic to quadratic, making it particularly suitable for iterative weight refinement in large-scale pruning tasks.

Specifically, let $X = A(\alpha_0)^{-1}$ and $B = I - H$. For a small change δ, the inverse can be expanded as

$$A(\alpha_0 + \delta)^{-1} = (A(\alpha_0) + \delta B)^{-1} = \sum_{k=0}^{\infty} (-\delta)^k X (BX)^k,$$

which converges when $\|\delta BX\| < 1$. Truncating the series to third order allows precomputing and reusing the base inverse X and its polynomial terms, so the inverse can be approximated at each step with only $\mathcal{O}(d^2)$ cost. Detailed derivation is provided in the supplementary material.

Theoretical Summary

Convergence Guarantee. We provide a theoretical analysis showing that EIWR converges globally under mild assumptions. First, we establish a one-step descent lemma that ensures consistent progress during iterations:

$$\Phi_{k+1} - \Phi_k \leq -\tfrac{\mu}{2} \|W_{k+1} - W_k\|^2 + \beta_k \epsilon_k^2, \tag{7}$$

where Φ_k denotes the augmented objective with a quadratic potential term, and ϵ_k is the iteration error.

Based on this, we prove that EIWR enjoys a *Q-linear convergence rate* toward the unique optimal solution $W_\star$:

$$\|W_{k+1} - W_\star\| \leq q \cdot \|W_k - W_\star\|, \quad \text{with } 0 < q < 1. \tag{8}$$

In addition, the original reconstruction error $E_k = X(W_k - W_{\text{dense}})$ also converges at the same linear rate, confirming the theoretical soundness and practical efficiency of our method. Detailed proofs are provided in the supplementary.

Neumann Extrapolation Error Bound. To accelerate weight refinement, we use a one-shot Neumann extrapolation that computes the inverse matrix A_0^{-1} only once at start and reuses a fixed third-order approximation throughout iterations. The convergence radius is controlled by

$$\rho_k := \|X_0 \Delta_k\| \leq c\,|\delta_k|, \quad c = \frac{1 - \lambda_{\min}}{(1-\alpha_0)\lambda_{\min} + \alpha_0}, \quad (9)$$

which guarantees convergence whenever $c\,|\delta_k| < 1$. The residual approximation error is bounded by

$$\|A_k^{-1} - \tilde{A}_k^{-1}\| \leq \frac{(c\,\delta_k)^4}{1 - c\,\delta_k} \|A_0^{-1}\|. \quad (10)$$

In practice, layer-wise normalization ensures $\lambda_{\min}(H) \geq 0.2$, yielding $c \leq 1.2$, and with $|\delta_k| \leq 0.1$, the relative approximation error remains below 0.2%. This allows us to reduce per-step matrix inversion cost from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$ with negligible loss in precision. Detailed proofs are provided in the supplementary material.

Experiments

Models and Datasets. We evaluate our method on the LLaMA2, LLaMA3, and Qwen model families, covering model sizes ranging from 3 billion to 70 billion parameters. Model performance is assessed using perplexity and zero-shot evaluation benchmarks. Specifically, perplexity is evaluated on two test sets: raw WikiText2 and a subset of the C4 validation set. The zero-shot tasks are evaluated on six standard benchmarks: PIQA (Bisk et al. 2020), ARC-Easy, ARC-Challenge (Clark et al. 2018), BOOLQ (Clark et al. 2019), COPA (Roemmele, Bejan, and Gordon 2011), and MultiRC (Khashabi et al. 2018) (detailed in Appendix B.2).

Method. As our method is plug-and-play, we select SparseGPT (Frantar and Alistarh 2023) and Wanda (Sun et al. 2023) as base pruning strategies, as they are the two most widely used and representative approaches for one-shot pruning. We further compare against two strong training-free refinement methods that have demonstrated impressive performance: DsnoT (Zhang et al. 2023), which dynamically modifies mask selection, and PCG (Meng et al. 2024), which refines sparse weights via a conjugate gradient approach. The naive backsolve method is also included, but only evaluated on a single layer due to its high computational cost on large models.

Settings. Following SparseGPT, we use 128 segments of 2048 tokens sampled from the first shard of the C4 dataset as calibration data. Our method performs at most 20 refinement iterations. All evaluations are conducted on NVIDIA 3090 GPUs. Detailed experimental settings, the momentum schedule α_k and its ablation, and speedup ratios (which are identical for models at the same sparsity level) are provided in the supplementary material.

Reconstruction Error on a Single Layer

We conduct a detailed evaluation of our method's ability to refine pruned weights at the layer level. Specifically, we select a linear layer from LLaMA2-13B (8.self_attn.v_proj) with input and output dimensions of 5120. The layer is pruned to varying sparsity levels (50%-80%), and we compare reconstruction quality using the relative error $\|X\hat{W} - XW\|_F / \|XW\|_F$ between the pruned and dense layer outputs on 128 calibration samples. Initial refers to weights pruned by Wanda without refinement. Results of SparseGPT are provided in the appendix for reference.

Table 1: Comparison of weight refinement methods under different sparsity levels. Backsolve denotes the vanilla backsolve approach; Ours is our proposed EIWR method.

Sparsity	Initial	Backsolve		Ours (EIWR)	
	Error	Time(s)	Error	Time(s)	Error
50%	0.1995	267	0.1451	0.31	0.1555
60%	0.2722	213	0.1923	0.36	0.2055
70%	0.3750	153	0.2601	0.52	0.2524
80%	0.4904	109	0.2651	0.61	0.2600

As shown in Table 1, the initial solutions produced by one-shot pruning remain coarse and leave substantial room for improvement. Our method achieves much lower reconstruction error than the initial one-shot solution. Moreover, it closely approximates the result of the computationally expensive backsolve procedure, which is often regarded as a theoretical upper bound for single-layer refinement, while requiring less than one second across all sparsity levels. In contrast, backsolve takes hundreds of seconds per layer due to repeated inversions of heterogeneous row-wise Hessians.

Model Performance After Pruning

We comprehensively evaluate the effectiveness of EIWR across a wide range of models and downstream tasks. Table 2 reports the performance of LLaMA2, LLaMA3, and Qwen2.5 variants under 70% sparsity on eight representative datasets. The first two tasks (WikiText2 and C4) are evaluated using perplexity (↓), while the remaining six are zero-shot classification or reasoning tasks evaluated by accuracy (↑). Results for additional model architectures, sparsity levels, and semi-structured pruning are provided in the supplementary material.

We adopt SparseGPT and Wanda as base pruning methods, and compare several training-free refinement methods designed for one-shot pruning (DsnoT, PCG) alongside our proposed EIWR. DsnoT provides some improvements on zero-shot tasks, but performs poorly on perplexity tasks and has limited effectiveness on newer models such as LLaMA3 and Qwen2.5. The PCG method in ALPS brings some gains, but its improvements are limited due to reliance on specific mask choices. Across all settings, EIWR consistently enhances model performance and outperforms previous plug-and-play refinements.

To further assess robustness under extreme sparsity, Table 3 presents perplexity on WikiText2 for LLaMA2-7B, LLaMA3-8B, and Qwen2.5-3B across sparsity levels from 60% to 90%. EIWR markedly reduces perplexity, especially under high sparsity. When sparsity increases, one-shot pruning methods typically suffer rapid performance degradation,

Table 2: Performance of LLaMA2, LLaMA3, and Qwen2.5 models under 70% sparsity across eight downstream tasks. WikiText2 and C4 are evaluated using perplexity (↓), while the remaining six tasks are evaluated using zero-shot accuracy or F1 score (↑). **MEAN** denotes the average performance over the six zero-shot tasks. Additional results on Vicuna, as well as other sizes of LLaMA and Qwen2.5, together with the performance of the dense model, are provided in the supplementary. Results for semi-structured sparsity, global sparsity, and combinations with quantization are provided in the supplementary.

Model	Algorithm	WikiText2↓	C4↓	PIQA↑	ARC-E↑	ARC-C↑	BOOLQ↑	COPA↑	MultiRC↑	MEAN
LLaMA2-7B	SparseGPT	23.5870	30.3359	0.6132	0.4428	0.2193	0.6245	0.7300	0.5419	0.5286
	w. Dsnot	67.5221	40.5157	0.6305	0.4831	0.2201	0.6214	0.7300	0.5437	0.5381
	w. PCG	21.9632	28.6312	0.6302	0.4851	0.2205	0.6314	**0.7400**	0.5431	0.5240
	w. EIWR	**20.2348**	**25.5027**	**0.6381**	**0.4847**	**0.2329**	**0.6593**	**0.7400**	**0.5769**	**0.5553**
	Wanda	73.0853	84.1034	0.5479	0.3085	0.1869	0.4881	0.6200	0.5311	0.4470
	w. Dsnot	72.9401	80.8218	0.5614	0.3068	0.1715	0.5275	0.6500	0.5703	0.4645
	w. PCG	32.4512	36.1264	0.5820	0.3569	0.1824	0.5823	0.6600	0.5695	0.4889
	w. EIWR	**22.9877**	**30.3113**	**0.6175**	**0.4322**	**0.2090**	**0.6226**	**0.6800**	**0.5719**	**0.5222**
LLaMA2-70B	SparseGPT	8.7688	12.0021	0.7535	0.7449	0.4104	0.7905	0.8900	0.5712	0.6934
	w. Dsnot	9.2142	14.1230	0.7612	0.7513	0.4272	0.8012	0.9000	0.5801	0.7035
	w. PCG	8.2141	11.8293	0.7692	0.7612	0.4376	0.8231	0.8900	0.5882	0.7116
	w. EIWR	**7.1234**	**10.3245**	**0.7982**	**0.7962**	**0.4532**	**0.8128**	**0.9100**	**0.5921**	**0.7271**
	Wanda	10.3869	13.5321	0.7378	0.7235	0.3899	0.7477	0.9000	0.5720	0.6785
	w. Dsnot	9.9231	12.3158	0.7423	0.7324	0.3982	0.7231	0.8900	0.5831	0.6782
	w. PCG	8.8731	12.8958	0.7512	0.7423	0.3723	0.7512	0.9000	0.5812	0.6900
	w. EIWR	**8.1241**	**11.7324**	**0.7821**	**0.7723**	**0.4272**	**0.7931**	**0.9200**	**0.5823**	**0.7058**
LLaMA3-8B	SparseGPT	39.1460	54.3701	0.5914	0.4128	0.2073	0.6709	0.6600	0.5715	0.5190
	w. Dsnot	124.7169	134.0848	0.6115	0.3939	0.1970	0.6290	0.6300	0.5614	0.5038
	w. PCG	37.3145	48.2415	**0.6115**	0.4149	0.2091	0.6421	0.6700	0.5814	0.5215
	w. EIWR	**32.8614**	**45.1304**	0.6077	**0.4696**	**0.2124**	**0.6602**	**0.6900**	**0.5709**	**0.5351**
	Wanda	106.3328	156.3624	0.5571	0.3127	0.1825	0.5217	0.5900	0.4581	0.4370
	w. Dsnot	119.9890	161.0210	0.5582	0.3110	0.1783	0.5262	0.5600	0.5736	0.4512
	w. PCG	35.9921	60.2310	0.5823	0.4241	0.2198	0.5193	0.6100	0.5631	0.4864
	w. EIWR	**34.0221**	**56.9829**	**0.6175**	**0.4322**	**0.2090**	**0.6226**	**0.6800**	**0.5719**	**0.5222**
Qwen2.5-3B	SparseGPT	28.1010	41.8826	0.6376	0.5113	0.2406	0.6342	0.6900	0.5719	0.5476
	w. Dsnot	33.4506	43.0498	0.6423	0.5142	0.2422	0.6431	**0.7000**	0.5699	0.5519
	w. PCG	25.3100	35.3241	0.6492	0.5253	0.2431	0.6342	0.6900	0.5811	0.5538
	w. EIWR	**24.3213**	**36.4024**	**0.6552**	**0.5365**	**0.2534**	**0.6545**	0.6900	**0.5879**	**0.5629**
	Wanda	152.2175	186.7293	0.5636	0.3358	0.1749	0.6183	0.5800	0.5715	0.4740
	w. Dsnot	89.0321	112.3214	0.5733	0.3720	0.1823	0.6211	0.5900	0.5715	0.4850
	w. PCG	40.1241	60.1264	0.5931	0.4231	0.1911	0.6031	0.6000	0.5731	0.4973
	w. EIWR	**37.0856**	**50.9128**	**0.6099**	**0.4494**	**0.1953**	**0.6235**	**0.6500**	**0.5819**	**0.5183**

Table 3: WikiText2 perplexity of sparse models pruned by the Wanda and SparseGPT metrics at 60%–90% sparsity. EIWR is applied as a post-pruning weight refinement method.

	LLaMAV2-7B				LLaMAV3-8B				Qwen2.5-3B			
Sparsity	60%	70%	80%	90%	60%	70%	80%	90%	60%	70%	80%	90%
Wanda	10.05	73.83	4.74e3	1.41e4	22.84	106.33	2.06e3	6.32e3	19.87	152.21	7.91e3	1.52e6
w. EIWR	9.79	22.98	87.20	705.17	14.71	34.02	117.39	421.32	14.86	37.08	213.76	768.17
SparseGPT	9.61	23.66	107.22	1.32e3	13.83	39.14	176.42	2.32e3	13.55	28.10	123.03	2.59e3
w. EIWR	9.01	20.23	77.30	673.23	12.92	32.86	137.60	432.21	12.90	24.32	92.36	662.08

but our method effectively alleviates this issue.

Computational Efficiency

We further assess the computational efficiency of EIWR in terms of runtime and memory overhead during pruning. Table 4 reports the time and peak GPU memory required to prune LLaMA2 models of different sizes (7B, 13B, and 70B) using SparseGPT and Wanda, both with and without our method. As shown, integrating EIWR introduces only a moderate increase in overhead—typically adding less than

Table 4: Pruning time (in min) and peak GPU memory usage (in GB) with and without EIWR across LLaMA2 model scales. EIWR adds minimal overhead to both SparseGPT and Wanda.

Method	LLaMA2-7B		LLaMA2-13B		LLaMA2-70B	
	Time	Mem	Time	Mem	Time	Mem
SparseGPT	12.38	5.13	23.36	6.84	124.6	17.56
With Ours	14.19	6.17	26.56	9.15	139.4	22.51
Wanda	4.39	4.95	7.29	6.54	53.23	15.78
With Ours	7.67	5.21	10.23	8.92	61.23	21.79

Table 5: Compatibility of EIWR with Dsnot and AlphaPruning on LLaMA2-7B at 70% sparsity. Best results are **highlighted**.

EIWR	Dsnot	Alpha	PPL↓	Zero-shot↑
×	×	×	23.58	0.5286
✓	×	×	20.23	0.5553
✓	✓	×	19.73	0.5601
✓	×	✓	17.81	0.5691
✓	✓	✓	**17.22**	**0.5831**

5 min and 5 GB of memory in SparseGPT, even for the largest 70B model. Compared to the total cost of the forward pass during pruning, this additional overhead is negligible. For comparison, we also apply LoRA-based fine-tuning and full fine-tuning to refine the sparse weights of the 7B model to match the performance of EIWR. These methods require approximately 16 GPU-hours and 192 GPU-hours, respectively, which is more than 1,000 times the cost of EIWR. Detailed results are provided in the supplementary material. These results demonstrate that EIWR is both effective and lightweight, making it practical for real-world deployment.

EIWR Compatibility

We investigate the compatibility of EIWR with other orthogonal training-free pruning enhancements. In particular, we consider Dsnot (Zhang et al. 2023) and AlphaPruning (Lu et al. 2024a), which focus on adaptive sparsity allocation and are widely used to improve one-shot pruning quality. Experiments are conducted on LLaMA2-7B at 70% global sparsity, and results are shown in Table 5. EIWR alone already achieves strong performance, and combining it with either Dsnot or AlphaPruning leads to further improvements in both perplexity and zero-shot accuracy. These results demonstrate the *orthogonality* of our refinement strategy: EIWR introduces no interference and complements other techniques to yield compound benefits.

Ablation Study

Effect of Historical Momentum. We first investigate the role of prior information from one-shot pruning and the impact of the historical momentum strategy in guiding the refinement trajectory. To isolate this effect, we compare three variants: No-Moment, which disables momentum entirely and treats each iteration independently, thereby completely discarding prior information; Fixed-Moment, which uses a constant momentum factor α_k across iterations; and Ours, which employs a progressively increasing momentum schedule that gradually stabilizes updates.

Table 6: Ablation on the effect of historical momentum. We report the runtime and reconstruction error on a single layer, as well as the runtime and perplexity on the full model. Best results are **highlighted**.

Method	Single Layer		Full Model	
	Time(s)↓	Error↓	Time(min)↓	PPL↓
No-Moment	3.75	0.3217	31.25	20.61
Fixed-Moment	1.62	0.3192	16.87	18.95
Ours	**0.52**	**0.2651**	**10.23**	**16.47**

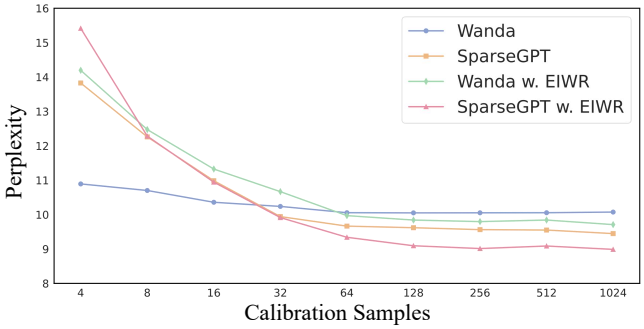

Figure 2: Performance of our method on LLaMA2-7B using SparseGPT and Wanda under 60% sparsity with varying calibration sample sizes.

As shown in Table 6, ignoring prior information leads to slower and less stable convergence. In contrast, the adaptive schedule in our method leverages this prior more effectively, significantly reducing reconstruction error while also accelerating convergence.

Effect of Calibration Sample Size. The number of calibration samples plays an important role in our method. A larger calibration set better captures the input distribution of the model, but also increases the computational cost. Figure 2 shows the impact of the number of calibration samples on pruning performance. We observe that the performance steadily improves with more samples, but saturates around 128. This indicates that our method achieves optimal results with only 128 calibration samples, which is the standard setting in one-shot pruning, without requiring additional data.

Conclusion

We propose Efficient Iterative Weight Refinement (EIWR), a lightweight, training-free, and plug-and-play method for recovering pruned weights in large language models. Experiments across various models and pruning strategies demonstrate that EIWR effectively improves performance with minimal overhead. Its plug-and-play nature ensures broad compatibility, making it a practical solution for efficient deployment of sparse LLMs.

Acknowledgments

The AI-driven experiments, simulations and model training were performed on the robotic AI-Scientist platform of Chinese Academy of Sciences.

References

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

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*.

Bisk, Y.; Zellers, R.; Gao, J.; Choi, Y.; et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 7432–7439.

Blumensath, T.; and Davies, M. E. 2008. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5): 629–654.

Boža, V. 2024. Fast and effective weight update for pruned large language models. *arXiv preprint arXiv:2401.02938*.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3): 1–45.

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. *arXiv preprint arXiv:1905.10044*.

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 preprint arXiv:1803.05457*.

Dong, P.; Li, L.; Tang, Z.; Liu, X.; Pan, X.; Wang, Q.; and Chu, X. 2024. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. *arXiv preprint arXiv:2406.02924*.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, 2943–2952. PMLR.

Frantar, E.; and Alistarh, D. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35: 4475–4488.

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.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Hassibi, B.; Stork, D. G.; and Wolff, G. J. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, 293–299. IEEE.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, 1389–1397.

Hu, Y.; Zhu, J.; and Chen, J. 2024. S-ste: Continuous pruning function for efficient 2: 4 sparse pre-training. *Advances in Neural Information Processing Systems*, 37: 33756–33778.

Huang, W.; Hu, Y.; Jian, G.; Zhu, J.; and Chen, J. 2025a. Pruning large language models with semi-structural adaptive sparse training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24167–24175.

Huang, W.; Zhang, Y.; Zheng, X.; Liu, Y.; Lin, J.; Yao, Y.; and Ji, R. 2025b. Dynamic Low-Rank Sparse Adaptation for Large Language Models. *arXiv preprint arXiv:2502.14816*.

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. *Advances in neural information processing systems*, 34: 21099–21111.

Kasneci, E.; Seßler, K.; Küchemann, S.; Bannert, M.; Dementieva, D.; Fischer, F.; Gasser, U.; Groh, G.; Günnemann, S.; Hüllermeier, E.; et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*, 103: 102274.

Khashabi, D.; Chaturvedi, S.; Roth, M.; Upadhyay, S.; and Roth, D. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 252–262.

Kusupati, A.; Ramanujan, V.; Somani, R.; Wortsman, M.; Jain, P.; Kakade, S.; and Farhadi, A. 2020. Soft threshold weight reparameterization for learnable sparsity. In *International conference on machine learning*, 5544–5555. PMLR.

Lebedev, V.; and Lempitsky, V. 2016. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2554–2564.

Lee, N.; Ajanthan, T.; and Torr, P. H. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Li, L.; Dong, P.; Tang, Z.; Liu, X.; Wang, Q.; Luo, W.; Xue, W.; Liu, Q.; Chu, X.; and Guo, Y. 2024a. Discovering sparsity allocation for layer-wise pruning of large language models. *Advances in Neural Information Processing Systems*, 37: 141292–141317.

Li, W.; Li, L.; Lee, M.; and Sun, S. 2024b. Adaptive layer sparsity for large language models via activation correlation assessment. *Advances in Neural Information Processing Systems*, 37: 109350–109380.

Liu, L.; Zhang, S.; Kuang, Z.; Zhou, A.; Xue, J.-H.; Wang, X.; Chen, Y.; Yang, W.; Liao, Q.; and Zhang, W. 2016. Pruning Filters for Efficient ConvNets.

Liu, L.; Zhang, S.; Kuang, Z.; Zhou, A.; Xue, J.-H.; Wang, X.; Chen, Y.; Yang, W.; Liao, Q.; and Zhang, W. 2021. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, 7021–7032. PMLR.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Lu, H.; Zhou, Y.; Liu, S.; Wang, Z.; Mahoney, M. W.; and Yang, Y. 2024a. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in neural information processing systems*, 37: 9117–9152.

Lu, X.; Zhou, A.; Xu, Y.; Zhang, R.; Gao, P.; and Li, H. 2024b. Spp: Sparsity-preserved parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2405.16057*.

Meng, X.; Behdin, K.; Wang, H.; and Mazumder, R. 2024. Alps: Improved optimization for highly sparse one-shot pruning for large language models. *Advances in Neural Information Processing Systems*, 37: 37594–37625.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1): 2383.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11264–11272.

Naveed, H.; Khan, A. U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; and Mian, A. 2023. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Roemmele, M.; Bejan, C. A.; and Gordon, A. S. 2011. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, 90–95.

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

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

Yin, L.; Wu, Y.; Zhang, Z.; Hsieh, C.-Y.; Wang, Y.; Jia, Y.; Li, G.; Jaiswal, A.; Pechenizkiy, M.; Liang, Y.; et al. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, Y.; Zhao, L.; Lin, M.; Sun, Y.; Yao, Y.; Han, X.; Tanner, J.; Liu, S.; and Ji, R. 2023. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*.