LoRScore: Low-Rank Refinement of Activation-Aware Scores for Learnable Large Language Model Pruning

Anonymous authors
Paper under double-blind review

Abstract

Learnable LLM pruning adapts sparsity patterns to the language-modeling objective, helping preserve model quality under compression. However, existing approaches often attach trainable variables to individual weights or candidate mask patterns, introducing substantial optimization overhead. We propose LoRScore, a parameter-efficient framework that combines Wanda's activation-aware importance scores with a learnable low-rank residual while keeping pretrained weights frozen. The key idea is to constrain only the learned score correction, preserving fine-grained prior information while enabling coordinated, loss-driven adjustments to pruning decisions. By separating score learning from mask projection, LoRScore supports both unstructured and semi-structured sparsity within a shared parameterization and introduces no additional parameters at inference. Experiments across multiple LLM families and sparsity settings demonstrate competitive language-modeling and zero-shot performance with substantially fewer trainable parameters. Theoretical analysis connects the surrogate score gradients to first-order sensitivity and characterizes the induced low-rank updates, providing an interpretation of how the compact parameterization refines pruning decisions.

1 Introduction

Large language models (LLMs) achieve strong performance across language tasks (Touvron et al., 2023a;b), but their scale makes deployment costly in memory, computation, and bandwidth. Weight pruning seeks to remove redundant connections with minimal loss of model quality (Han et al., 2015). Unstructured pruning offers flexible weight selection, whereas semi-structured patterns support acceleration on compatible hardware (Mishra et al., 2021). Calibration-based methods guide pruning through local reconstruction or importance statistics. SparseGPT (Frantar & Alistarh, 2023) uses second-order information for approximate layer-wise reconstruction, while Wanda (Sun et al., 2024) combines weight magnitudes with input activation norms. These methods are efficient, but do not directly optimize the language-modeling loss of the complete sparse model.

To learn pruning decisions from the language-modeling objective, recent methods optimize masks under semi-structured $N{:}M$ constraints, retaining N weights in each group of M. MaskLLM (Fang et al., 2024) learns distributions over candidate patterns with pretrained weights frozen, requiring six logits per four-weight group for $2{:}4$ sparsity (Figure 1(a)). MaskPro (Sun et al., 2026) reduces this overhead through per-weight logits and sampling without replacement (Figure 1(b)). It updates these logits using a loss-residual policy-gradient estimator without backpropagating through the LLM. Both methods maintain trainable mask variables whose number scales with the number of weights being pruned. These parameter costs and optimization choices motivate a compact mask parameterization that retains access to backpropagated sensitivity signals and supports both unstructured and semi-structured sparsity.

To this end, we introduce **LoRScore**, a framework that learns pruning decisions through low-rank score residual. For each projection matrix, LoRScore represents a score correction as the product of two thin matrices, sharing trainable parameters across weights. Constraining the entire score matrix to be low rank, however, can restrict its ability to distinguish individual connections. We therefore apply the low-rank constraint only to a learned residual added to a fixed activation-aware

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

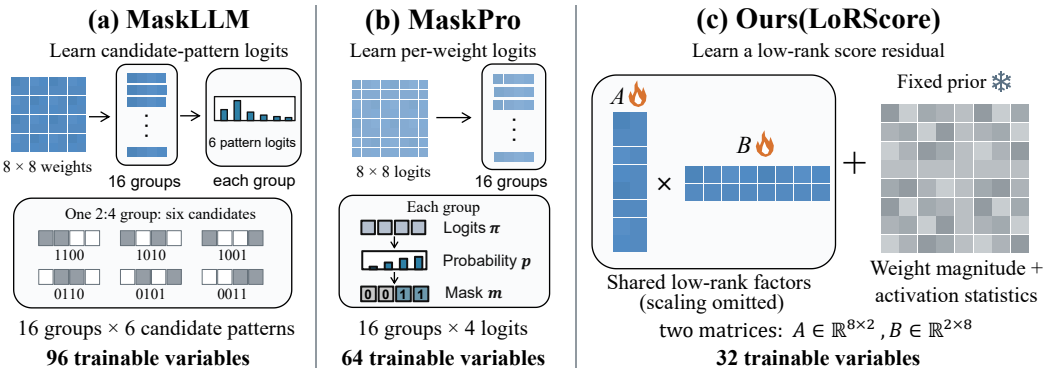


Figure 1: Comparison of trainable mask parameterizations for 2:4 pruning on an 8×8 weight matrix: (a) candidate-pattern logits in MaskLLM; (b) per-weight logits in MaskPro; (c) low-rank score residual over a fixed prior in LoRScore ($r = 2$).

prior (Figure 1(c)). The complete score matrix remains unrestricted in rank. For a $d_{\text{out}} \times d_{\text{in}}$ weight matrix, this representation requires only $r(d_{\text{out}}+d_{\text{in}})$ trainable pruning parameters. For a 4096×4096 projection under 2:4 sparsity, LoRScore ($r = 32$) uses approximately 0.26M parameters, compared with 25.2M for MaskLLM and 16.8M for MaskPro.

The fixed prior follows Wanda's activation-aware criterion (Sun et al., 2024), combining weight magnitudes with calibrated input activation norms. It supplies weight-specific importance information, allowing the low-rank factors to focus on loss-driven corrections. Prior mask information and parameter sharing have also been explored in earlier work (Fang et al., 2024; Sun et al., 2026; Sun & Sakuma, 2026). LoRScore combines a fixed activation-aware score prior with a low-rank residual, restricting the learned correction rather than the complete score representation.

LoRScore separates score learning from mask projection, allowing the same score parameterization to support both unstructured and N:M sparsity. Its central design is to retain a fixed, full-resolution activation-aware prior while constraining only the learned score correction to be low-rank. Unlike low-rank adaptation in weight space, these factors adjust pruning scores and the resulting masks while leaving pretrained weight values unchanged. Whereas MaskPro constructs policy-gradient updates from sampled-mask loss feedback, LoRScore uses a straight-through estimator (STE) (Bengio et al., 2013) to optimize its score factors under the language-modeling objective. Under this surrogate backward rule, we connect the score gradients to first-order Taylor sensitivity and characterize how factor-wise updates transform this signal into coordinated score corrections through shared low-rank factors. We alternate factor updates and progressively enforce the target sparsity constraint. After optimization, the final masks are exported and the score factors are discarded, leaving no additional learned modules at inference.

Experiments across multiple LLM families demonstrate competitive language-modeling and zero-shot performance. LoRScore shows a pronounced advantage over Wanda at high unstructured sparsity and modest average accuracy gains over MaskPro under semi-structured sparsity, while using substantially fewer trainable mask parameters. These results highlight a practical balance between pruning quality and optimization efficiency.

Our contributions are threefold:

(i) We propose LoRScore, which preserves a fixed activation-aware prior while learning a low-rank score residual. Constraining only the refinement retains fine-grained importance information with a compact trainable parameterization.

(ii) We support unstructured and semi-structured pruning through a shared score parameterization, combining STE-based optimization, alternating factor updates, and progressive sparsity scheduling. We analyze the surrogate gradients, update geometry, and expressiveness.

(iii) We demonstrate competitive pruning quality across LLM families with substantially fewer trainable parameters than per-weight and per-pattern mask learning. Ablations and resource measurements characterize the design choices and performance–efficiency trade-off.

2 Related Work

2.1 Pruning Pretrained Language Models

Pruning methods differ in the granularity of removed weights. Unstructured pruning removes individual connections, whereas structured pruning removes entire components, such as layers, attention heads, or hidden dimensions (Ma et al., 2023; Xia et al., 2024). Semi-structured pruning imposes local patterns, such as $N{:}M$ sparsity, which retains N of every M consecutive weights to support acceleration on compatible hardware. For weight-level pruning, early methods rank connections by magnitude (Han et al., 2015). SparseGPT (Frantar & Alistarh, 2023) improves pruning through layer-wise sparse reconstruction, using second-order information to compensate for removed weights. Wanda (Sun et al., 2024) ranks weights by the product of their magnitudes and input activation norms, without updating retained weight values. Both support unstructured and semi-structured sparsity, but neither directly optimizes the model-level language-modeling loss. Such activation-aware scoring provides a useful starting point for mask learning. Building on this prior, LoRScore learns score refinements through the language-modeling loss.

2.2 Learnable Pruning Masks

Learning-based approaches make pruning decisions trainable. Edge-Popup (Ramanujan et al., 2020) learns connection scores over fixed, randomly initialized weights through a straight-through estimator (STE) (Bengio et al., 2013), retaining the highest-scoring connections. Movement pruning (Sanh et al., 2020) extends score-based pruning to pretrained language models, jointly optimizing weights and scores during fine-tuning using first-order information. Extending mask learning to semi-structured sparsity requires accounting for local $N{:}M$ constraints. MaskLLM (Fang et al., 2024) addresses this by assigning trainable logits to the valid sparsity patterns within each group. MaskPro (Sun et al., 2026) reduces the parameter count by learning per-weight logits and sampling without replacement. Its policy-gradient estimator uses loss residuals and a moving-average tracker without backpropagating through the LLM. Both methods can incorporate prior masks into initialization. HyperPrune (Sun & Sakuma, 2026) shares a context-aware hypernetwork across layers to generate semi-structured masks using layer-wise reconstruction objectives. Following this line of mask learning, LoRScore combines STE-based optimization with a compact low-rank score parameterization under the language-modeling objective.

2.3 Low-Rank Adaptation and Pruning

Low-rank parameterization is widely used for efficient model adaptation. LoRA (Hu et al., 2022) freezes pretrained weights and learns low-rank updates to the effective weight matrices. LoRAPrune (Zhang et al., 2024) combines low-rank adaptation with structured pruning, using LoRA parameters and their gradients to estimate importance. Targeted low-rank refinement (Shen et al., 2025) compensates for discrepancies between dense and sparse weights through iterative sparse-plus-low-rank refinement. LoRScore applies low-rank adaptation to pruning scores, learning a residual over a fixed activation-aware prior to adjust weight rankings. Only the score residual is constrained to be low-rank; the prior retains fine-grained importance information. The learned factors guide mask selection and are discarded after pruning, leaving the retained pretrained weight values unchanged.

3 Method

LoRScore makes pruning decisions learnable by optimizing the scores that determine which connections to retain. Learning is performed through compact low-rank score factors under the language-modeling objective, allowing the resulting masks to adapt while pretrained weights remain frozen. As shown in Figure 2, the framework comprises three coupled components. *Low-rank score refinement* adds a trainable low-rank residual to a fixed activation-aware prior. *Mask projection* converts the refined scores into binary masks through row-wise Top-k or group-wise Top-N selection under the scheduled sparsity constraint. *Masked forward* computes the language-modeling loss using the masked weights, with an STE providing surrogate gradients for updating the score factors. Together, these components connect score refinement and mask selection to loss-driven optimization. We alternate factor updates (Section 3.1.3) and progressively enforce the target sparsity. After mask export, the score factors are discarded, leaving no additional learned modules at inference.

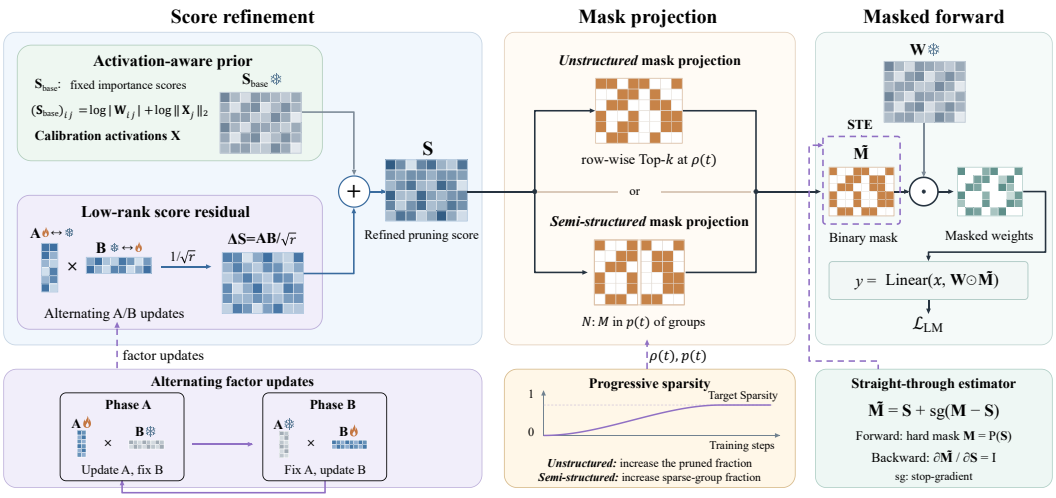


Figure 2: Overview of LoRScore. (1) **Score refinement** combines a fixed activation-aware prior with a low-rank correction learned through alternating updates of **A** and **B**. (2) **Mask projection** converts the refined scores into masks for unstructured or $N{:}M$ pruning under a progressive sparsity schedule. (3) **Masked forward** applies binary masks to frozen pretrained weights to compute the language-modeling loss, while an STE provides surrogate gradients for score optimization.

3.1 Score Refinement

3.1.1 Activation-Aware Prior

Consider a pretrained linear layer $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We collect its input activations $\mathbf{X} \in \mathbb{R}^{T \times d_{\text{in}}}$ from a small calibration set containing T tokens. For the j-th input channel, we compute the activation norm $a_j = \|\mathbf{X}_{:,j}\|_2$. Following Wanda (Sun et al., 2024), we assess weight importance jointly through weight magnitude and the activation norm of the corresponding input channel. We express this criterion in the log domain:

$$(\mathbf{S}_{\text{base}})_{ij} = \log\left(|W_{ij}| + \epsilon\right) + \log\left(a_j + \epsilon\right), \tag{1}$$

where $\epsilon > 0$ ensures numerical stability for zero or near-zero values. Since the logarithm is strictly increasing, this transformation preserves the ranking of the stabilized multiplicative criterion $(|W_{ij}| + \epsilon)(a_j + \epsilon)$. Consequently, it produces the same Top-k and $N{:}M$ masks as that criterion under identical selection and tie-breaking rules.

The log-domain representation converts multiplicative importance into an additive score, allowing a learnable residual correction to refine the activation-aware prior. This provides an informed starting point for score learning while retaining the importance information derived from pretrained weights and calibration activations. The prior is computed once from calibration data and remains fixed throughout training.

3.1.2 Low-Rank Score Residual

The prior remains fixed after calibration and does not directly optimize the language-modeling loss. We therefore introduce a learnable low-rank correction parameterized by $\mathbf{A} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_{\text{in}}}$, where $r \ll \min(d_{\text{out}}, d_{\text{in}})$. The refined importance score is defined as

$$\mathbf{S} = \mathbf{S}_{\text{base}} + \frac{\mathbf{AB}}{\sqrt{r}}. \tag{2}$$

We use $1/\sqrt{r}$ as a rank-dependent normalization factor (Kalajdzievski, 2023). Compared with learning a full score matrix, this parameterization reduces the number of trainable parameters per layer from $d_{\text{out}}d_{\text{in}}$ to $r(d_{\text{out}} + d_{\text{in}})$. Importantly, optimization takes place in score space rather than weight space: the pretrained weights **W** remain frozen, while the relative importance of individual weights is adapted through the learned correction.

We initialize $\mathbf{A} = \mathbf{0}$ and sample the entries of $\mathbf{B}$ independently from $\mathcal{N}(0, 1)$. This gives $\mathbf{S}^{(0)} = \mathbf{S}_{\text{base}}$, so the initial score ranking exactly matches that of the activation-aware prior. Under the same sparsity constraints and tie-breaking rules, projecting these scores also produces the same mask as projecting the prior. Score learning therefore begins from the calibrated importance estimates rather than a randomly initialized score matrix.

Equation 2 also admits a multiplicative interpretation in the original importance-score domain. Let $q_{ij}^{\text{base}} = (|W_{ij}| + \epsilon)(a_j + \epsilon)$ denote the stabilized activation-aware importance. Exponentiating the refined score yields $\widetilde{q}_{ij} = q_{ij}^{\text{base}} \exp\big((\mathbf{AB})_{ij}/\sqrt{r}\big)$. Thus, the additive correction in log space corresponds to a learned multiplicative reweighting of the analytic importance.

3.1.3 Alternating Factor Updates

We optimize the score factors in alternating phases: Phase A updates $\mathbf{A}$ while keeping $\mathbf{B}$ fixed, whereas Phase B updates $\mathbf{B}$ while keeping $\mathbf{A}$ fixed. We switch phases every epoch, starting with Phase A. This order follows the initialization $\mathbf{A} = \mathbf{0}$, under which the initial loss gradient with respect to $\mathbf{B}$ is zero. The pretrained weights and activation-aware prior remain frozen. Surrogate gradients are obtained through the STE described in Section 3.3.

3.2 Mask Projection

The refined scores are converted into binary masks through a sparsity-aware projection. At training step t, we write $\mathbf{M}_{\text{hard}}^{(t)} = \mathcal{P}_t(\mathbf{S}^{(t)})$, where $\mathcal{P}_t$ specifies the sparsity pattern and the sparsity level at that step. Unlike sampling-based mask learning in MaskLLM (Fang et al., 2024) and MaskPro (Sun et al., 2026), our projection is deterministic given the scores, sparsity constraints, and tie-breaking rules. It therefore avoids variance arising from mask sampling while supporting different sparsity patterns through the same score parameterization.

For unstructured pruning, Top-k projection retains the entries with the largest scores and sets the remaining mask entries to zero. At target sparsity $\rho^\star$, layer-wise selection retains $k = \text{round}((1 - \rho^\star)d_{\text{out}}d_{\text{in}})$ weights from the entire weight matrix. The same criterion also supports row-wise selection, which retains $k = \text{round}((1 - \rho^\star)d_{\text{in}})$ weights per output row. Ties are resolved consistently to preserve the prescribed number of retained entries.

For semi-structured $N{:}M$ pruning, each output row is partitioned into consecutive groups of M weights, and the N highest-scoring entries within each selected group are retained. At the target sparsity, every group $\mathcal{G}$ satisfies $\sum_{j\in\mathcal{G}} M_{ij} = N$, with $|\mathcal{G}| = M$. In particular, the $2{:}4$ pattern retains two weights in every group of four and can be used with compatible GPU sparse kernels (Mishra et al., 2021). Both projection rules operate directly on the refined scores without introducing additional trainable mask parameters.

3.2.1 Progressive Sparsity Scheduling

We gradually introduce the pruning constraint using a smoothstep schedule:

$$p(t) = 3u(t)^2 - 2u(t)^3, \qquad u(t) = \text{clip}\left(\frac{t - t_{\text{s}}}{t_{\text{e}} - t_{\text{s}}}, 0, 1\right), \tag{3}$$

where t_{s} and t_{e} delimit the transition. The schedule increases from zero to one with zero slope at both endpoints. For unstructured pruning, the sparsity is $\rho(t) = \rho^\star p(t)$. For $N{:}M$ pruning, a layer with G groups applies the exact constraint to $\text{round}(p(t)G)$ groups, while the remaining groups stay dense. At each step, we select groups with the largest boundary score gaps,

$$c_{\mathcal{G}} = S_{\mathcal{G},(N)} - S_{\mathcal{G},(N+1)},$$

where $S_{\mathcal{G},(k)}$ is the k-th largest score in group $\mathcal{G}$. This prioritizes groups with a clearer separation between retained and removed weights. Each group remains either dense or exactly $N{:}M$ sparse throughout the transition. After t_{e}, the target sparsity is maintained, with all groups following the prescribed $N{:}M$ pattern in the semi-structured setting. Mask selections continue to adapt to the learned scores.

3.3 Masked Forward

For each pruned linear layer, we compute $\mathbf{y} = (\mathbf{W} \odot \widetilde{\mathbf{M}})\mathbf{x} + \mathbf{b}$, keeping all pretrained parameters frozen. Since the hard projection $\mathbf{M} = \mathcal{P}(\mathbf{S})$ has zero gradients almost everywhere, we use a straight-through estimator (STE) (Bengio et al., 2013):

$$\widetilde{\mathbf{M}} = \mathbf{S} + \mathrm{sg}(\mathbf{M} - \mathbf{S}), \tag{4}$$

where sg denotes stop-gradient. Thus, $\widetilde{\mathbf{M}} = \mathbf{M}$ in the forward pass, while mask gradients pass directly to the scores during backpropagation. We optimize the next-token prediction loss of the resulting masked model:

$$\mathcal{L}_{\mathrm{LM}} = -\frac{1}{T-1}\sum_{t=2}^{T} \log p\Big(z_t \mid z_{<t}; \big\{\mathbf{W}_\ell \odot \widetilde{\mathbf{M}}_\ell\big\}_{\ell\in\mathcal{I}}\Big), \tag{5}$$

where $z_{1:T}$ is a training sequence and $\mathcal{I}$ indexes the pruned layers. Here, p denotes the full model with the indicated masked weights and all remaining parameters unchanged.

For a given layer, let $\mathbf{G} = \partial\mathcal{L}_{\mathrm{LM}}/\partial\widetilde{\mathbf{M}}$. With $\mathbf{S} = \mathbf{S}_{\text{base}} + \mathbf{AB}/\sqrt{r}$, the STE yields the surrogate gradients

$$\nabla_{\mathbf{S}}\mathcal{L}_{\mathrm{LM}} = \mathbf{G}, \qquad \nabla_{\mathbf{A}}\mathcal{L}_{\mathrm{LM}} = \frac{\mathbf{G}\mathbf{B}^\top}{\sqrt{r}}, \qquad \nabla_{\mathbf{B}}\mathcal{L}_{\mathrm{LM}} = \frac{\mathbf{A}^\top\mathbf{G}}{\sqrt{r}}. \tag{6}$$

These gradients are biased surrogates for optimizing the discrete masks. Under the alternating schedule in Section 3.1.3, only the active factor is updated, while the other factor and $\mathbf{S}_{\text{base}}$ remain fixed.

3.4 Theoretical Analysis

We summarize the main properties of LoRScore; full statements and proofs are provided in Appendix B.

Score gradients encode first-order saliency. Let $\mathbf{G}$ denote the loss gradient with respect to the effective weights $\mathbf{W} \odot \widetilde{\mathbf{M}}$. Under the STE,

$$\nabla_{\mathbf{S}}\mathcal{L}_{\mathrm{LM}} = \mathbf{G} \odot \mathbf{W} \triangleq \widehat{\mathbf{G}}. \tag{7}$$

This signed, loss-dependent signal connects score learning to movement pruning (Sanh et al., 2020) and first-order Taylor saliency (Molchanov et al., 2019). It guides adjustments to weight rankings beyond the fixed activation-aware prior (Proposition 2).

Low-rank updates capture saliency through factor subspaces. For $\mathbf{S} = \mathbf{S}_{\text{base}} + \mathbf{AB}/\sqrt{r}$, plain gradient descent on one factor at a time induces

$$\Delta\mathbf{S}_A = -\frac{\eta}{r}\widehat{\mathbf{G}}\mathbf{B}^\top\mathbf{B}, \qquad \Delta\mathbf{S}_B = -\frac{\eta}{r}\mathbf{A}\mathbf{A}^\top\widehat{\mathbf{G}}. \tag{8}$$

At initialization, $\mathbf{A}^{(0)} = \mathbf{0}$ and $\mathbf{B}$ has i.i.d. standard Gaussian entries. The first Phase-A update satisfies $\mathbb{E}_{\mathbf{B}}[\Delta\mathbf{S}_A] = -\eta\widehat{\mathbf{G}}$ (Proposition 3), but individual updates may deviate substantially when $r \ll d_{\text{in}}$. Unbiasedness alone does not guarantee an accurate full-gradient approximation; subsequent alternating updates adapt the two factor subspaces during score optimization. These identities characterize plain gradient descent, rather than the exact updates of adaptive optimizers.

Consistency, expressiveness, and parameter efficiency. Zero initialization of $\mathbf{A}$ exactly recovers the fixed prior and its projected masks (Proposition 1). In the unstabilized log-score formulation, a rank-one residual can represent rankings induced by $|W_{ij}|a_j^\alpha$, encompassing magnitude pruning ($\alpha = 0$) and Wanda ($\alpha = 1$). Row-wise additive shifts leave within-row mask selection unchanged (Appendix B.4). LoRScore uses $r(d_{\text{out}}+d_{\text{in}})$ trainable parameters per layer, compared with $1.5d_{\text{out}}d_{\text{in}}$ categorical mask parameters for 2:4 pruning (Fang et al., 2024), and updates only one factor per phase (Appendix B.5).

4 Experiments

4.1 Experimental Setup

Models and Baselines. We evaluate unstructured pruning at 50%, 60%, and 80% sparsity on the 7B and 13B variants of LLaMA (Touvron et al., 2023a) and LLaMA-2 (Touvron et al., 2023b),

Table 1: WikiText validation perplexity of pruned LLaMA and LLaMA-2 models under unstructured sparsity of 50%, 60%, and 80%. Best results at each sparsity level are bolded.

Method	Weight Update	Sparsity	LLaMA-7B	LLaMA-2-7B	LLaMA-13B	LLaMA-2-13B
Dense	–	0%	5.68	5.12	5.09	4.57
Magnitude	✗	50%	17.29	14.89	20.21	6.37
SparseGPT	✓	50%	7.22	6.51	6.21	5.63
Wanda	✗	50%	7.26	6.42	6.15	5.56
LoRScore	✗	50%	**7.20**	**6.38**	**6.10**	**5.50**
Magnitude	✗	60%	6e2	4e3	2e2	11.23
SparseGPT	✓	60%	10.51	9.58	8.56	7.80
Wanda	✗	60%	10.66	9.71	8.56	7.75
LoRScore	✗	60%	**9.56**	**8.51**	**7.35**	**7.07**
Magnitude	✗	80%	1e5	nan	3e4	5e4
SparseGPT	✓	80%	2e2	1e2	**1e2**	**1e2**
Wanda	✗	80%	5e3	5e3	4e3	2e3
LoRScore	✗	80%	**1e2**	**73**	**1e2**	**1e2**

and 2:4 semi-structured pruning on Gemma-7B (Team et al., 2024), Vicuna-v1.3-7B (Chiang et al., 2023), LLaMA-2-7B, and DeepSeek-7B (Bi et al., 2024). Baselines include magnitude pruning (Han et al., 2015), SparseGPT (Frantar & Alistarh, 2023), and Wanda (Sun et al., 2024), with GBLM-Pruner (Das et al., 2023), Pruner-Zero (Dong et al., 2024), MaskLLM (Fang et al., 2024), and MaskPro (Sun et al., 2026) additionally evaluated for 2:4 sparsity. Baseline results are drawn from Wanda and MaskPro for unstructured and semi-structured pruning, respectively, with dense models as references. LoRScore uses the corresponding model checkpoints, sparsity settings, and evaluation protocols for comparability.

Datasets and Evaluation. We use C4 (Raffel et al., 2020) for calibration and mask learning, and evaluate language modeling using WikiText-2 perplexity (PPL) (Merity et al., 2016). Zero-shot accuracy is measured using the LM Evaluation Harness (Gao et al., 2021) on HellaSwag (Zellers et al., 2019), RACE (Lai et al., 2017), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), ARC-Easy/Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). Lower PPL and higher accuracy indicate better performance. We report token-level or word-level PPL according to the corresponding baseline protocol to facilitate consistent comparisons, so PPL values across protocols are not directly comparable.

Training and Pruning. The learning rate is selected from $\{1 \times 10^{-4}, 4 \times 10^{-4}, 1 \times 10^{-3}\}$ for each model and sparsity pattern by perplexity on held-out C4 validation data (sensitivity in Appendix D.1); all ablation and hyperparameter studies use 1×10^{-3}. The fixed Wanda prior uses 32 calibration sequences; both calibration and training use 1,024-token sequences. The low-rank refinement is initially zero, and updates alternate between **A** and **B** each epoch, starting with **A**. All attention and MLP projections are pruned, while embeddings and the LM head remain dense and pretrained weights stay frozen. Masks use row-wise top-k selection or top-2 selection within consecutive four-weight groups. Progressive pruning reaches the target after 60% of training and maintains it thereafter; for 2:4, it gradually increases the fraction of constrained groups to all groups. Detailed configurations and sensitivity results appear in Appendix D.

4.2 Main results

Unstructured Pruning Performance. Perplexity (PPL) measures language modeling quality on held-out text, with lower values indicating better performance. *Baseline WikiText PPL results are taken from Wanda (Sun et al., 2024).* LoRScore performs competitively at moderate sparsity, with a particularly pronounced advantage over Wanda at high sparsity. This suggests that loss-driven, low-rank score refinement helps preserve important connections when pruning becomes more aggressive, improving upon the fixed activation-aware rankings. Nevertheless, substantial degradation relative to dense models remains at extreme sparsity, indicating that score refinement alleviates but does not eliminate the effects of extensive weight removal.

Table 2: WikiText-2 perplexity and zero-shot accuracy under 2:4 sparsity. Model-name rows report dense results. Avg. is the mean accuracy across the seven tasks. Bold and underlined values indicate the best and second-best results below the separator within each model, respectively.

Model / Method	PPL ↓	Zero-shot Accuracy (%) ↑							
	WikiText-2	HellaS.	RACE	PIQA	WinoG.	ARC-E	ARC-C	OBQA	Avg.
Gemma-7B	112.39	60.54	40.19	79.71	73.09	81.65	49.91	32.80	59.70
MaskLLM	–	25.42	20.10	51.52	49.49	25.21	21.59	18.40	30.25
Magnitude	–	25.23	21.24	51.85	50.75	26.43	21.84	12.40	29.96
SparseGPT	–	26.07	22.39	55.11	50.36	30.64	18.43	14.80	31.11
Wanda	–	26.80	22.78	56.47	48.86	32.66	17.75	13.60	31.27
GBLM	–	26.81	22.49	54.52	51.00	32.38	17.66	14.00	31.27
Pruner-Zero	–	25.27	21.63	53.21	50.75	24.58	**22.70**	15.20	30.48
MaskPro	–	26.97	23.26	**57.88**	**52.82**	**32.92**	22.65	16.40	33.27
LoRScore	–	**31.54**	**30.24**	57.51	51.64	32.27	22.10	**18.02**	**34.76**
Vicuna-v1.3-7B	11.86	56.32	41.91	77.37	69.46	74.28	42.41	34.60	56.62
MaskLLM	14.91	49.07	39.13	75.24	65.35	65.57	33.57	25.60	50.50
Magnitude	389.92	40.19	28.61	67.03	57.62	54.59	28.75	19.40	42.31
SparseGPT	24.93	44.87	37.81	70.62	63.30	62.92	32.42	**25.00**	48.13
Wanda	25.24	44.28	37.89	70.57	61.56	61.70	32.17	23.00	47.31
GBLM	24.60	44.29	38.37	70.51	61.80	62.84	31.40	24.00	47.60
Pruner-Zero	24.02	44.77	37.42	71.22	62.75	62.33	32.76	24.00	47.89
MaskPro	21.10	46.81	**38.76**	71.60	**64.25**	64.23	33.19	24.80	49.09
LoRScore	**18.71**	**47.11**	37.69	**72.20**	63.27	**65.95**	**33.28**	**25.00**	**49.21**
LLaMA-2-7B	8.71	57.15	39.62	78.07	68.90	76.35	43.34	31.40	56.40
MaskLLM	12.55	51.17	38.56	74.70	65.04	69.57	35.67	26.80	51.64
Magnitude	307.39	45.43	31.48	70.08	60.93	61.87	30.20	21.80	45.97
SparseGPT	21.07	43.20	36.56	70.89	64.56	64.52	31.48	24.60	47.97
Wanda	23.44	41.32	35.89	70.46	62.12	62.79	30.20	24.20	46.71
GBLM	21.64	41.79	34.61	70.57	62.75	63.17	29.86	23.20	46.56
Pruner-Zero	22.09	41.17	34.64	70.18	62.35	61.32	27.05	22.80	45.64
MaskPro	17.17	46.18	37.13	73.07	**65.82**	66.12	32.85	26.20	49.62
LoRScore	**14.06**	**49.37**	**37.42**	**73.72**	64.96	**67.80**	**34.39**	**27.20**	**50.69**
DeepSeek-7B	9.70	56.94	39.62	79.27	70.40	75.25	43.60	32.60	56.81
MaskLLM	12.90	51.73	39.14	75.95	65.80	68.10	35.32	25.80	51.69
Magnitude	285.06	40.97	28.52	69.75	60.06	54.92	27.56	20.80	43.23
SparseGPT	19.12	45.58	37.80	73.94	65.43	66.37	32.94	24.80	49.55
Wanda	19.68	45.38	35.12	73.56	63.14	65.49	32.00	22.80	48.21
GBLM	19.55	45.34	36.17	73.99	62.98	65.82	32.85	23.60	48.68
Pruner-Zero	20.71	44.93	35.22	73.23	62.12	64.94	30.89	23.20	47.79
MaskPro	17.97	47.78	37.75	74.72	**65.59**	**66.74**	33.49	**28.60**	50.67
LoRScore	**16.35**	**49.13**	**38.18**	**74.86**	65.27	66.33	**33.53**	27.80	**50.72**

Language Modeling and Zero-Shot Performance. Under 2:4 sparsity, LoRScore achieves substantial improvements over Wanda, highlighting the benefit of adapting the fixed activation-aware prior through loss-driven, low-rank score refinement. Using baseline results reported by MaskPro (Sun et al., 2026), LoRScore achieves lower WikiText-2 perplexity than MaskPro on Vicuna, LLaMA-2, and DeepSeek, while delivering higher HellaSwag accuracy across the evaluated models. Although these performance gains over MaskPro are modest, our efficiency evaluation also shows lower peak memory usage and shorter per-step training time. MaskLLM generally achieves stronger predictive performance, but LoRScore requires substantially fewer trainable parameters and less training memory. With pretrained weights frozen, it optimizes only $r(d_{\text{in}} + d_{\text{out}})$ score parameters per layer. Together, these results demonstrate a favorable balance between pruning quality and optimization efficiency under limited memory budgets.

Table 3: Component ablations of LoRScore under 50% unstructured and $2{:}4$ semi-structured sparsity. Lower PPL and higher accuracy (in percent) are better.

Method	Unstructured (50%)				Semi-structured (2:4)			
	LLaMA-7B		LLaMA-2-7B		LLaMA-7B		LLaMA-2-7B	
	PPL† ↓	Acc.↑	PPL† ↓	Acc.↑	PPL‡ ↓	Acc.↑	PPL‡ ↓	Acc.↑
w/o Wanda prior	8.02	51.73	7.05	52.36	18.82	48.38	18.38	48.50
w/o alternating opt.	8.96	51.71	7.54	52.73	22.88	47.30	19.16	48.85
w/o progressive sparsity	7.40	52.40	6.59	53.49	16.53	49.60	15.85	49.70
LoRScore (full)	**7.20**	**52.87**	**6.44**	**53.85**	**15.87**	**50.19**	**14.07**	**50.69**

4.3 Ablation Study

We ablate LoRScore on LLaMA-7B and LLaMA-2-7B under 50% unstructured and $2{:}4$ semi-structured sparsity. The variants remove the Wanda prior, update both factors jointly, or impose the target sparsity from the first step. PPL† and PPL‡ denote perplexity evaluated using the Wanda and MaskPro protocols, respectively. Acc. is the mean accuracy over the seven zero-shot tasks in Table 2. As shown in Table 3, full LoRScore achieves the lowest PPL and highest accuracy across all reported settings. Removing alternating updates causes the largest PPL increase, while removing the prior also consistently degrades performance. Progressive sparsification provides modest gains under unstructured pruning and clearer improvements under $2{:}4$ sparsity, allowing the score factors to adapt before the target constraint is fully enforced.

4.4 Computational and Memory Efficiency

We evaluate the training overhead of 2:4 mask learning for LLaMA-2-7B on a single NVIDIA RTX PRO 6000 GPU (96 GB). All methods use a unified framework with frozen pretrained weights, FP16 mixed precision, and identical Wanda initialization. Masks are optimized for attention and feed-forward projections across all 32 layers using C4, a sequence length of 1,024 tokens, and a batch size of one. We report median step time and peak GPU memory over 20 iterations after 10 warm-up iterations. As shown in Table 4, our method requires only 80M trainable parameters, approximately $1/81$ and $1/121$ of those required by MaskPro and MaskLLM, respectively. It uses 31.7 GiB of peak VRAM and takes 2.12 s per step, reducing these costs by 28% and 26% relative to MaskPro. MaskLLM runs out of memory even at batch size one. These results demonstrate the parameter and computational efficiency of low-rank mask optimization.

Table 4: Mask-learning costs for LLaMA-2-7B (2:4 sparsity, $r = 32$ for ours).

Metric	MaskLLM	MaskPro	**Ours**
Trainable params	9.71B	6.48B	**80M**
Mask state (GiB)	144.7†	–	**1.2**
Total memory (GiB)	$>$157†	44.2	**31.7**
Time / step (s)	OOM	2.87	**2.12**

Total memory: peak GPU memory during mask learning. Mask state: trainable parameters, gradients, and optimizer states.
†Estimated with FP32 states and Adam (16 bytes/param) plus FP16 frozen weights; exceeds the 96 GB GPU (OOM).

5 Conclusion and Discussion

We presented LoRScore, a parameter-efficient framework for learnable LLM pruning. By learning low-rank corrections to a fixed activation-aware prior, LoRScore adapts pruning decisions under the language-modeling objective while preserving fine-grained importance information and keeping pretrained weights frozen. Separating score learning from mask projection enables a shared parameterization for both unstructured and semi-structured sparsity. Our theoretical analysis connects the surrogate gradients to first-order sensitivity and characterizes the low-rank updates. Experiments demonstrate competitive language-modeling and zero-shot performance with substantially fewer trainable mask parameters. After mask export, the score factors are discarded, leaving no additional learned modules at inference. LoRScore still requires model backpropagation and biased STE gradients, limiting efficiency gains, while pruning quality degrades at extreme sparsity. Future work could explore adaptive ranks and more efficient score optimization.

AI Use Statement

We used generative AI tools to assist with manuscript editing and drafting, LaTeX formatting, feedback on experimental comparisons and result interpretation, and limited code development and debugging. The authors reviewed the AI-assisted text and interpretations against the experimental results and checked the assisted code for correctness. The authors take full responsibility for the final manuscript, methodology, results, and conclusions.

Reproducibility statement

We have made several efforts to facilitate the reproducibility of our work. All pretrained models and datasets used in our experiments are publicly available and identified in Section 4.1 and Appendix C. Section 3 describes the proposed method, and Algorithm 1 in Appendix A provides the training and mask export procedure. Training configurations, hyperparameters, and computational resources are described in Section 4.1, with additional configurations and hyperparameter analyses in Appendix D. The evaluation setup is described in Section 4.1, with further experiment-specific details in Appendix C. We will continue to refine the implementation and its documentation as this work develops and ultimately release the source code publicly to support reproduction and future research.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*, 2023.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-Zero: Evolving symbolic pruning metric from scratch for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 1135–1143, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017. doi: 10.18653/v1/D17-1082.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

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

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11256–11264. IEEE, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11890–11899. IEEE, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.

Li Shen, Anke Tang, Yong Luo, Tao Sun, Han Hu, and Xiaochun Cao. Targeted low-rank refinement: Enhancing sparse language models with precision. In *Forty-second International Conference on Machine Learning*, 2025.

Lu Sun and Jun Sakuma. Learning semi-structured sparsity for llms via shared and context-aware hypernetwork. In *The Fourteenth International Conference on Learning Representations*, 2026.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Yan Sun, Qixin Zhang, Zhiyuan Yu, Xikun Zhang, Li Shen, and Dacheng Tao. Maskpro: Linear-space probabilistic learning for strict (n:m)-sparsity on LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

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

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3013–3026, 2024.

This appendix provides the complete LoRScore algorithm (Appendix A), theoretical analysis (Appendix B), additional experimental results (Appendix C), and detailed configurations and hyperparameter sensitivity studies (Appendix D).

A Algorithm

We present the complete training and mask export procedure for LoRScore, including activation-aware score initialization, alternating factor updates, and progressive mask projection. The pruning schedule end ratio ρ_{end} determines the schedule endpoint $t_e \approx \rho_{\text{end}} T$, where T is the total number of training steps. At t_e, all groups satisfy the target $N{:}M$ constraint; the masks remain adaptive during subsequent score optimization.

Algorithm 1 LoRScore for $N{:}M$ Pruning

Require: Pretrained model $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$, target layers $\mathcal{I}$, calibration set $\mathcal{C}$, training set $\mathcal{D}$
Require: Rank r, sparsity pattern $N{:}M$, stabilization constant $\epsilon > 0$
Require: Phase lengths K_A, K_B, schedule $t_{\text{s}} < t_{\text{e}} \le T$, training steps T
Ensure: Hard masks $\{\mathbf{M}^{*(l)}\}_{l\in\mathcal{I}}$ and sparse weights $\{\widehat{\mathbf{W}}^{(l)}\}_{l\in\mathcal{I}}$

Freeze all pretrained model parameters
Collect activation statistics $\{a_j^{(l)}\}$ on $\mathcal{C}$
for each target layer $l \in \mathcal{I}$ **do**
    $S^{(l)}_{\text{base},ij} \leftarrow \log(|W^{(l)}_{ij}| + \epsilon) + \log(a^{(l)}_j + \epsilon)$
    $\mathbf{A}^{(l)} \leftarrow \mathbf{0}$; initialize $B^{(l)}_{kj} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$
    Partition each row into groups of M weights; let G_l be the total number of groups
end for
for $t = 1, \ldots, T$ **do**
    $q \leftarrow (t-1) \bmod (K_A + K_B)$
    Set $\mathbf{A}$ active if $q < K_A$, otherwise $\mathbf{B}$; freeze the other factor in every target layer
    Clear factor gradients and sample a minibatch $\mathcal{B}_t$ from $\mathcal{D}$
    $u \leftarrow \text{clip}\Big(\frac{t-t_{\text{s}}}{t_{\text{e}}-t_{\text{s}}}, 0, 1\Big)$; $p(t) \leftarrow 3u^2 - 2u^3$
    for each target layer $l \in \mathcal{I}$ **do**
        $\mathbf{S}^{(l)} \leftarrow \mathbf{S}^{(l)}_{\text{base}} + \mathbf{A}^{(l)}\mathbf{B}^{(l)}/\sqrt{r}$
        Compute $c_{\mathcal{G}} \leftarrow S^{(l)}_{\mathcal{G},(N)} - S^{(l)}_{\mathcal{G},(N+1)}$ for every group $\mathcal{G}$
        $k_l(t) \leftarrow \text{round}(p(t) G_l)$
        Select the $k_l(t)$ groups with the largest $c_{\mathcal{G}}$
        Form $\mathbf{M}^{(l)}$ by retaining the top-N scores in each selected group; keep all other groups dense
        $\widetilde{\mathbf{M}}^{(l)} \leftarrow \mathbf{S}^{(l)} + \text{sg}\big(\mathbf{M}^{(l)} - \mathbf{S}^{(l)}\big)$
    end for
    Run the model on $\mathcal{B}_t$ using $\mathbf{W}^{(l)} \odot \widetilde{\mathbf{M}}^{(l)}$ in target layers and compute $\mathcal{L}_{\text{LM}}$
    Backpropagate $\mathcal{L}_{\text{LM}}$ through the STE
    Apply an optimizer step only to the active factors
end for
for each target layer $l \in \mathcal{I}$ **do**
    $\mathbf{S}^{(l)} \leftarrow \mathbf{S}^{(l)}_{\text{base}} + \mathbf{A}^{(l)}\mathbf{B}^{(l)}/\sqrt{r}$
    Form $\mathbf{M}^{*(l)}$ by retaining the top-N scores in every group of M weights
    $\widehat{\mathbf{W}}^{(l)} \leftarrow \mathbf{W}^{(l)} \odot \mathbf{M}^{*(l)}$
end for
return $\{\mathbf{M}^{*(l)}, \widehat{\mathbf{W}}^{(l)}\}_{l\in\mathcal{I}}$

B Additional Theoretical Analysis

This appendix provides the statements and proofs summarized in Section 3.4. We analyze initialization consistency, the interpretation of the straight-through score gradient, the geometry of low-rank factor updates, the expressiveness of the score correction, and its parameter complexity.

We consider a frozen weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ with scores

$$\mathbf{S} = \mathbf{S}_{\text{base}} + \frac{1}{\sqrt{r}}\mathbf{AB}, \qquad \mathbf{A} \in \mathbb{R}^{d_{\text{out}} \times r}, \quad \mathbf{B} \in \mathbb{R}^{r \times d_{\text{in}}},$$

where $r \geq 1$ and

$$S_{\text{base},ij} = \log(|W_{ij}| + \epsilon) + \log(a_j + \epsilon), \qquad a_j \geq 0, \quad \epsilon > 0.$$

The activation statistics and the prior are fixed during score optimization.

Unless otherwise stated, comparisons between masks use the same sparsity level, the same constrained-group set, and fixed deterministic tie-breaking. The identities below apply to each layer at the current network state. Gradients are computed through the full masked model, so the optimization problems of different layers remain coupled through the loss.

B.1 Consistency with the Prior

Proposition 1 (Initialization consistency)**.** *If $\mathbf{A}^{(0)} = \mathbf{0}$, then for any $\mathbf{B}^{(0)}$,*

$$\mathcal{P}(\mathbf{S}^{(0)}) = \mathcal{P}(\mathbf{S}_{\text{base}}).$$

Moreover, the initial scores induce the same within-row and within-group rankings as the stabilized Wanda criterion

$$Q_{ij}^{\text{stab}} = (|W_{ij}| + \epsilon)(a_j + \epsilon).$$

Consequently, row-wise Top-k and group-wise Top-N selection recover the corresponding stabilized Wanda masks under the same sparsity constraints and tie-breaking rule.

Proof. Since $\mathbf{A}^{(0)} = \mathbf{0}$,

$$\mathbf{S}^{(0)} = \mathbf{S}_{\text{base}} + \frac{1}{\sqrt{r}}\mathbf{A}^{(0)}\mathbf{B}^{(0)} = \mathbf{S}_{\text{base}}.$$

Applying the same deterministic mask operator gives the first equality. Furthermore,

$$S_{\text{base},ij} = \log Q_{ij}^{\text{stab}}.$$

The logarithm is strictly increasing on positive arguments, so the two criteria have identical rankings and ties. Their Top-k and Top-N selections therefore coincide. □

This consistency result concerns the stabilized criterion used by the parameterization. For $\epsilon > 0$, its rankings can differ from those of the unstabilized criterion $|W_{ij}|a_j$.

If the training schedule initially constrains only a subset of groups, the result applies to the pruning decisions within those groups. Equality with a fully constrained baseline additionally requires the same constrained-group set.

A separate model-selection guarantee is available if a baseline mask constructed from the stabilized prior at the target sparsity is explicitly included among the feasible candidates. Selecting the candidate with the lowest value of a fixed calibration objective then yields a mask whose value on that objective is no higher than the baseline's. This property follows from candidate selection; it does not imply monotonic improvement during training or improved performance on unseen data.

B.2 Score Gradients and First-Order Sensitivity

Let

$$\mathbf{W}_{\text{eff}} = \mathbf{W} \odot \widetilde{\mathbf{M}}$$

denote the effective weights, where $\widetilde{\mathbf{M}}$ has the hard mask as its forward value. Consistent with the main text, define

$$\mathbf{G} = \frac{\partial \mathcal{L}_{\text{LM}}}{\partial \mathbf{W}_{\text{eff}}} = \sum_t \boldsymbol{\delta}_t \mathbf{x}_t^\top,$$

where $\boldsymbol{\delta}_t = \partial \mathcal{L}_{\text{LM}} / \partial \mathbf{y}_t$ is the error signal at the layer output. Define the continuous-gate gradient as

$$\widehat{\mathbf{G}} := \frac{\partial \mathcal{L}_{\text{LM}}}{\partial \widetilde{\mathbf{M}}} = \mathbf{G} \odot \mathbf{W}.$$

Proposition 2 (Surrogate score gradient)**.** *Under an identity straight-through estimator from scores to masks, the surrogate score gradient is*

$$\nabla_{\mathbf{S}}^{\text{STE}} \mathcal{L}_{\text{LM}} = \mathbf{G} \odot \mathbf{W} = \widehat{\mathbf{G}}. \tag{9}$$

Proof. Since $W_{\text{eff},ij} = W_{ij} \widetilde{M}_{ij}$ and $\mathbf{W}$ is frozen, differentiation with respect to the continuous gate gives

$$\frac{\partial \mathcal{L}_{\text{LM}}}{\partial \widetilde{M}_{ij}} = G_{ij} W_{ij}.$$

The identity STE passes this signal directly from the mask to the score, yielding Eq. 9. □

Equation equation 9 describes a surrogate backward rule. The exact hard-selection objective is locally constant with respect to scores whenever its pruning decisions remain unchanged, so its ordinary score derivative differs from the STE signal.

The identity-Jacobian assumption applies to the score-to-mask map actually used in the backward pass. If an implementation additionally gates this path to selected groups, the same gate also multiplies the resulting score gradient.

The signal $\widehat{\mathbf{G}}$ can be nonzero at pruned positions because it contains no explicit multiplicative factor M_{ij}. Its value nevertheless depends on the current mask through the input activations and output errors.

To interpret this signal, consider the loss as a function of continuous mask entries, evaluated at the current hard mask. Holding all other entries fixed, a first-order expansion gives

$$\Delta \mathcal{L}_{\text{remove},ij} \approx -\widehat{G}_{ij}$$

when a retained gate changes from 1 to 0, and

$$\Delta \mathcal{L}_{\text{restore},ij} \approx +\widehat{G}_{ij}$$

when a pruned gate changes from 0 to 1. Thus, a negative $\widehat{G}_{ij}$ locally favors the presence of the connection, whereas a positive value locally favors its removal. These are linear sensitivity estimates for finite gate changes, rather than guarantees of the resulting loss difference.

Under a fixed $N{:}M$ constraint, a feasible change typically replaces one retained connection with one pruned connection within the same group. If connection p is removed and connection q is restored, the corresponding first-order prediction is

$$\Delta \mathcal{L}_{p \to q} \approx -\widehat{G}_p + \widehat{G}_q.$$

The relevant decision therefore depends on relative sensitivities within the group.

This interpretation connects score learning to Taylor-based importance estimation and the signed score signal used in movement pruning (Molchanov et al., 2019; Sanh et al., 2020). Direct gradient descent on an unconstrained score matrix would accumulate $-\widehat{\mathbf{G}}$. LoRScore instead uses this signal through the low-rank factor gradients derived below. These updates couple different score entries, so the sign of an individual score increment need not equal the sign of $-\widehat{G}_{ij}$.

Compared with the fixed activation-aware prior, the gradient also incorporates output errors and therefore supplies feedback tied to the language-modeling objective.

B.3 Low-Rank Updates as Sketched Saliency

Applying the chain rule to the STE signal gives

$$\nabla_{\mathbf{A}}^{\mathrm{STE}}\mathcal{L} = \frac{1}{\sqrt{r}}\widehat{\mathbf{G}}\mathbf{B}^\top, \qquad \nabla_{\mathbf{B}}^{\mathrm{STE}}\mathcal{L} = \frac{1}{\sqrt{r}}\mathbf{A}^\top\widehat{\mathbf{G}}. \tag{10}$$

Here, $\mathcal{L}$ denotes the language-modeling loss. At initialization, $\nabla_{\mathbf{B}}^{\mathrm{STE}}\mathcal{L} = \mathbf{0}$ because $\mathbf{A}^{(0)} = \mathbf{0}$. Thus, $\mathbf{A}$ is the only factor with a potentially nonzero initial loss gradient.

The update identities in this subsection concern plain gradient descent on this loss, without momentum, adaptive preconditioning, or weight decay. They characterize the geometry induced by the parameterization; they are not exact descriptions of AdamW iterates.

Proposition 3 (First-step moments under Gaussian initialization)**.** *Let* $\mathbf{A}^{(0)} = \mathbf{0}$ *and*

$$B_{kj}^{(0)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).$$

Condition on a fixed initial forward state and minibatch, and assume that the resulting saliency matrix $\widehat{\mathbf{G}}^{(0)}$ *is independent of* $\mathbf{B}^{(0)}$*. Holding* $\mathbf{B}^{(0)}$ *fixed during the update, let*

$$\Delta\mathbf{A} = -\frac{\eta}{\sqrt{r}}\widehat{\mathbf{G}}^{(0)}(\mathbf{B}^{(0)})^\top, \qquad \Delta\mathbf{S} = \frac{1}{\sqrt{r}}\Delta\mathbf{A}\mathbf{B}^{(0)}.$$

Then, taking moments over the Gaussian initialization,

$$\mathbb{E}_{\mathbf{B}^{(0)}}[\Delta\mathbf{S}] = -\eta\widehat{\mathbf{G}}^{(0)},$$

and

$$\mathrm{Var}_{\mathbf{B}^{(0)}}[\Delta S_{ij}] = \frac{\eta^2}{r}\left(\|\widehat{\mathbf{G}}_{i,:}^{(0)}\|_2^2 + \left(\widehat{G}_{ij}^{(0)}\right)^2\right).$$

Proof. For brevity, write $\widehat{\mathbf{G}} = \widehat{\mathbf{G}}^{(0)}$ and $\mathbf{B} = \mathbf{B}^{(0)}$. The induced score update is

$$\Delta\mathbf{S} = -\frac{\eta}{r}\widehat{\mathbf{G}}\mathbf{B}^\top\mathbf{B}.$$

Let the ℓth row of $\mathbf{B}$ be $\mathbf{z}_\ell^\top$, where $\mathbf{z}_\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{d_{\text{in}}})$ independently. Then

$$\Delta S_{ij} = -\frac{\eta}{r}\sum_{\ell=1}^{r}\left(\widehat{\mathbf{G}}_{i,:}\mathbf{z}_\ell\right)z_{\ell j}.$$

For a standard Gaussian vector $\mathbf{z}$,

$$\mathbb{E}\left[\left(\widehat{\mathbf{G}}_{i,:}\mathbf{z}\right)z_j\right] = \widehat{G}_{ij},$$

and the Gaussian fourth-moment identity gives

$$\mathbb{E}\left[\left(\widehat{\mathbf{G}}_{i,:}\mathbf{z}\right)^2 z_j^2\right] = \|\widehat{\mathbf{G}}_{i,:}\|_2^2 + 2\widehat{G}_{ij}^2.$$

Therefore,

$$\mathrm{Var}\left[\left(\widehat{\mathbf{G}}_{i,:}\mathbf{z}\right)z_j\right] = \|\widehat{\mathbf{G}}_{i,:}\|_2^2 + \widehat{G}_{ij}^2.$$

Summing the means and variances of the r independent terms proves both statements. □

The independence assumption is satisfied at initialization when the prior and other sources of forward-pass randomness are independent of $\mathbf{B}^{(0)}$: because $\mathbf{A}^{(0)} = \mathbf{0}$, the initial scores and forward computation do not depend on $\mathbf{B}^{(0)}$. The proposition therefore establishes an unbiased first-step sketch over random initializations.

This conclusion does not generally extend to later updates with the same fixed matrix. After optimization begins, the scores, masks, and saliency can depend on $\mathbf{B}^{(0)}$, so the independence used in the proof need not hold. Reusing the same matrix also does not supply independent sketch errors that automatically average out across steps.

Summing the entrywise variances yields the exact first-step mean-squared error

$$\mathbb{E}_{\mathbf{B}^{(0)}}\left[\left\|\Delta\mathbf{S}+\eta\widehat{\mathbf{G}}^{(0)}\right\|_F^2\right]=\frac{\eta^2(d_{\text{in}}+1)}{r}\left\|\widehat{\mathbf{G}}^{(0)}\right\|_F^2.$$

Although the variance decreases as $1/r$, unbiasedness alone does not imply a small relative error for an individual initialization or preservation of the full-score update's pruning decisions.

For arbitrary current factors, the induced plain gradient-descent updates have the deterministic forms

$$\Delta\mathbf{S}_A=-\frac{\eta}{r}\widehat{\mathbf{G}}\mathbf{B}^\top\mathbf{B}$$

when $\mathbf{B}$ is fixed, and

$$\Delta\mathbf{S}_B=-\frac{\eta}{r}\mathbf{A}\mathbf{A}^\top\widehat{\mathbf{G}}$$

when $\mathbf{A}$ is fixed. The first update has its row space contained in the row space of $\mathbf{B}$; the second has its column space contained in the column space of $\mathbf{A}$.

The operator $\mathbf{A}\mathbf{A}^\top/r$ is positive semidefinite but is generally not an orthogonal projector. If $\mathbf{A}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is a compact singular value decomposition, then

$$\frac{1}{r}\mathbf{A}\mathbf{A}^\top=\frac{1}{r}\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top.$$

Thus, Phase B produces an update within $\text{col}(\mathbf{A})$, with direction-dependent scaling determined by the singular values of $\mathbf{A}$.

Both updates have a nonpositive inner product with the current surrogate score gradient:

$$\left\langle\widehat{\mathbf{G}},\Delta\mathbf{S}_A\right\rangle_F=-\frac{\eta}{r}\left\|\widehat{\mathbf{G}}\mathbf{B}^\top\right\|_F^2\le 0,$$

and

$$\left\langle\widehat{\mathbf{G}},\Delta\mathbf{S}_B\right\rangle_F=-\frac{\eta}{r}\left\|\mathbf{A}^\top\widehat{\mathbf{G}}\right\|_F^2\le 0.$$

These identities follow directly from the Frobenius inner product and hold without a Gaussian or independence assumption. They show that each update does not increase the linear model defined by the current STE signal. Because the actual forward map uses discrete mask selection, they do not guarantee a decrease in the hard-mask loss.

Alternating the phases allows the row and column subspaces available to the correction to change in turn. The products $\widehat{\mathbf{G}}\mathbf{B}^\top$ and $\mathbf{A}^\top\widehat{\mathbf{G}}$ resemble the matrix products used in subspace iteration, providing an interpretation of the staged schedule in Section 3.1.3. Actual optimization uses additive updates and evolving saliency matrices, however, so this analogy does not establish convergence to the dominant singular subspace of an accumulated saliency matrix.

B.4 Expressiveness and Invariance

For a fixed sparsity level and constrained-group set, row-wise Top-k and group-wise Top-N selection depend only on the relevant score rankings. With fixed tie-breaking, adding a row-wise shift $\mathbf{u}\mathbf{1}^\top$ therefore leaves the mask unchanged:

$$\mathcal{P}\left(\mathbf{S}+\mathbf{u}\mathbf{1}^\top\right)=\mathcal{P}(\mathbf{S}).$$

Every constrained group lies within one row, so all scores in that group receive the same additive constant. This invariance also holds if constrained groups are selected solely through within-group boundary score gaps with fixed tie-breaking, since row-wise shifts preserve those gaps.

A column-wise correction $\mathbf{1}\mathbf{v}^\top$ can change the relative importance of input channels. Adding this correction gives

$$S_{ij}=\log(|W_{ij}|+\epsilon)+\log(a_j+\epsilon)+v_j,$$

which induces the same rankings as

$$(|W_{ij}|+\epsilon)(a_j+\epsilon)e^{v_j}.$$

Thus, the correction implements a channel-wise reweighting of the activation statistic.

In particular, for any $\alpha \in \mathbb{R}$, choosing

$$v_j = (\alpha - 1)\log(a_j + \epsilon)$$

gives the criterion

$$Q_{ij}^{(\alpha)} = (|W_{ij}| + \epsilon)(a_j + \epsilon)^{\alpha}.$$

At $\alpha = 0$, this recovers magnitude-pruning rankings; at $\alpha = 1$, it recovers the stabilized Wanda criterion. The required correction has rank at most one.

This construction is representable for any $r \geq 1$. For example, set

$$A_{i1} = \sqrt{r}, \qquad B_{1j} = v_j,$$

and set all remaining factor entries to zero. Then $\mathbf{AB}/\sqrt{r} = \mathbf{1}\mathbf{v}^{\top}$. This is an expressiveness statement about the full factor parameterization. It does not guarantee that a phase with one factor fixed can realize every such correction, or that optimization will find it.

For general rank r,

$$\exp(S_{ij}) = (|W_{ij}| + \epsilon)(a_j + \epsilon)\exp\left(\frac{1}{\sqrt{r}}\sum_{\ell=1}^{r} A_{i\ell}B_{\ell j}\right).$$

The rows of $\mathbf{B}$ provide up to r shared channel patterns, and $\mathbf{A}$ assigns row-dependent coefficients. This permits coordinated refinements that vary across both input and output channels.

The rank constraint applies only to the additive residual $\mathbf{S} - \mathbf{S}_{\text{base}}$. Neither the complete score matrix nor the resulting binary mask is required to have rank at most r.

Exact score representation and mask equivalence must also be distinguished. With both factors unrestricted, a target score matrix $\mathbf{T}$ is exactly representable if and only if

$$\text{rank}\,(\mathbf{T} - \mathbf{S}_{\text{base}}) \leq r.$$

Necessity follows from $\text{rank}(\mathbf{AB}) \leq r$; sufficiency follows from a rank factorization of the residual, with zero padding when its rank is smaller than r and a rescaling to absorb $1/\sqrt{r}$.

Matching pruning decisions requires only matching the rankings relevant to the specified selection rule. For example, for any $\beta > 0$, the criterion

$$(|W_{ij}| + \epsilon)^{\beta}(a_j + \epsilon)^{\alpha}$$

has the same rankings as

$$(|W_{ij}| + \epsilon)(a_j + \epsilon)^{\alpha/\beta},$$

because raising positive values to the power $1/\beta$ is strictly increasing. The latter criterion is attainable through the rank-one construction above. Consequently, a high-rank difference between two score matrices does not by itself establish that their pruning decisions cannot be reproduced by low-rank score residual.

B.5 Parameter Complexity

For $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the learned factors contain

$$P_{\text{LoRScore}} = r(d_{\text{out}} + d_{\text{in}}) \tag{11}$$

scalar parameters in total. The activation-aware prior and pretrained weights are fixed. This parameter count is independent of whether the scores are used for unstructured or semi-structured mask selection.

For comparison, MaskLLM (Fang et al., 2024) assigns a trainable logit to each candidate pattern in an $N{:}M$ group, whereas MaskPro (Sun et al., 2026) uses one logit per weight. Assuming complete groups of M weights, the corresponding parameter counts are

$$P_{\text{MaskLLM}} = \frac{d_{\text{out}}d_{\text{in}}}{M}\binom{M}{N}, \qquad P_{\text{MaskPro}} = d_{\text{out}}d_{\text{in}}.$$

For 2:4 sparsity,

$$P_{\text{MaskLLM}} = 1.5 d_{\text{out}} d_{\text{in}}.$$

For a 4096×4096 projection with $r = 32$, LoRScore uses $262{,}144$ learned scalar parameters, compared with $25{,}165{,}824$ for MaskLLM and $16{,}777{,}216$ for MaskPro. The corresponding ratios are $1/96$ and $1/64$. For a square $d \times d$ matrix, the ratio to a full trainable score matrix is

$$\frac{P_{\text{LoRScore}}}{d^2} = \frac{2r}{d}.$$

At fixed rank, the factor parameter count therefore grows linearly with layer width.

Under alternating optimization, Phase A updates rd_{out} parameters and Phase B updates rd_{in} parameters. Both factors remain stored, and optimizer states for previously updated factors may also remain resident. Alternation reduces the number of parameters updated in each phase without changing the total factor parameter count.

These comparisons concern learned parameters. Total training memory additionally depends on frozen weights, the fixed prior, activations, optimizer-state management, and whether dense scores and masks are materialized. Parameter-count reductions therefore do not imply proportional reductions in peak memory or training time.

C Additional Experimental Results

We extend the main evaluation in three directions: additional zero-shot tasks, additional semi-structured sparsity patterns (4:8 and 8:16), and additional models, namely OPT-2.7B, LLaMA-13B, LLaMA-2-13B, LLaMA-3.1-8B, and LLaMA-30B. Any differences from the main experimental settings are specified in the corresponding subsections.

Table 5: Additional zero-shot evaluation of LLaMA-2-7B under 2:4 sparsity. Scores are reported in percent (↑). Bold and underlined values denote the best and second-best results among pruned models, respectively.

Benchmark	Dense	SparseGPT	Pruner-Zero	MaskPro	LoRScore
WMDP	39.29	26.61	26.52	<u>26.95</u>	**27.21**
Prost	23.60	24.26	24.03	<u>24.41</u>	**26.57**
AExams	21.04	<u>23.65</u>	<u>23.65</u>	<u>23.65</u>	**24.21**
AClue	27.47	25.33	25.31	<u>26.24</u>	**26.47**
ANLI-R1	36.40	33.20	<u>33.60</u>	**34.40**	33.10
ANLI-R2	37.20	<u>34.10</u>	33.90	<u>34.10</u>	**35.30**
ANLI-R3	37.58	33.08	33.00	**35.67**	<u>34.33</u>
SciQ	94.00	<u>91.10</u>	<u>91.10</u>	<u>91.10</u>	**91.80**
MathQA	28.24	23.72	23.55	<u>23.95</u>	**24.97**
Haerae	22.27	18.88	<u>18.91</u>	18.79	**20.44**
BoolQ	77.68	<u>71.10</u>	69.13	**71.12**	70.46
ComQA	32.92	20.80	20.08	<u>22.03</u>	**23.18**
LogiQA	25.65	21.66	21.78	<u>22.89</u>	**23.20**
COPA	87.00	<u>81.00</u>	79.00	79.00	**82.00**
WiC	49.84	47.81	47.22	<u>49.84</u>	**51.88**
WSC	36.54	<u>36.54</u>	<u>36.54</u>	<u>36.54</u>	**48.08**
CB	42.86	41.07	39.29	**57.14**	<u>42.86</u>
MultiRC	56.97	**57.20**	56.37	<u>56.93</u>	**57.20**
RTE	62.82	58.48	<u>59.12</u>	**61.37**	58.76

C.1 Additional Zero-Shot Tasks

We evaluate LoRScore on additional zero-shot benchmarks to examine its performance beyond the seven tasks reported in the main text. Table 5 reports additional zero-shot results for LLaMA-2-7B under 2:4 sparsity with rank $r = 16$. LoRScore achieves the best or tied-best performance

Table 6: Comparison of sparsity patterns across LLaMA models. Lower perplexity (PPL) and higher average accuracy (Avg.) are better.

	LLaMA-7B		LLaMA-2-7B		LLaMA-13B		LLaMA-2-13B	
Pattern	PPL↓	Avg.↑	PPL↓	Avg.↑	PPL↓	Avg.↑	PPL↓	Avg.↑
Dense	9.46	56.40	8.71	56.40	8.27	58.22	7.74	59.26
4:8	14.35	50.88	12.60	51.63	11.27	53.81	10.99	54.34
8:16	13.83	51.81	11.53	52.41	–	–	–	–

among pruned models on 14 of 19 tasks. Its unweighted mean score reaches 42.21%, compared with 41.90% for MaskPro, with notable gains on WSC and WiC of 11.54 and 2.04 percentage points, respectively. However, it remains below MaskPro on five tasks, indicating that the benefits vary across benchmarks. Overall, these results demonstrate competitive downstream performance with a compact rank-16 score parameterization while keeping pretrained weights frozen.

C.2 Additional N:M Sparsity Patterns

Beyond the $2{:}4$ setting, we evaluate LoRScore under two additional semi-structured sparsity patterns, $4{:}8$ and $8{:}16$, to examine its applicability to different group sizes. We evaluate both patterns on LLaMA-7B and LLaMA-2-7B and additionally report $4{:}8$ results for LLaMA-13B and LLaMA-2-13B. For an $N{:}M$ pattern, exactly N weights are retained within each group of M weights. Thus, $2{:}4$, $4{:}8$, and $8{:}16$ all yield 50% sparsity, while larger groups permit more flexible allocation of the retained weights within each group.

The experiments follow the main training setup, with model-size-specific learning rates. The low-rank score correction has rank $r = 32$, and the fixed activation-aware prior is initialized using the Wanda criterion with 32 calibration samples. We prune all linear projections in the attention and MLP blocks while keeping the pretrained weights frozen. The low-rank score factors are optimized on 5,000 C4 samples with a sequence length of 1,024 and a batch size of 4 for 10 epochs. We use a learning rate of 10^{-3} for both 7B models and 4×10^{-4} for all experiments on both 13B models. Optimization alternates between the two factors every epoch: one factor is updated while the other remains fixed. For each model, the score parameterization and training hyperparameters remain unchanged across sparsity patterns; only the grouping and selection constraint used for mask projection differs. At the target sparsity, mask projection retains the N highest-scoring weights within each group of M weights.

After training, we export and fix the final binary masks, apply them to the pretrained weights, and discard the low-rank score factors. We report WikiText-2 test perplexity following the MaskPro evaluation protocol, together with zero-shot raw accuracy on seven downstream tasks: HellaSwag, RACE, PIQA, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA. The Avg. metric is the arithmetic mean of the accuracies on these seven tasks. Lower perplexity and higher average accuracy indicate better performance.

Table 6 summarizes the results. On LLaMA-7B, changing the sparsity pattern from $4{:}8$ to $8{:}16$ reduces perplexity from 14.35 to 13.83 and increases average accuracy from 50.88% to 51.81%. LLaMA-2-7B exhibits the same trend: perplexity decreases from 12.60 to 11.53, while average accuracy increases from 51.63% to 52.41%. These correspond to average accuracy gains of 0.93 and 0.78 percentage points, respectively, at the same overall sparsity. The results are consistent with the greater selection flexibility afforded by larger groups. Under $4{:}8$ sparsity, LoRScore achieves perplexities of 6.99 and 6.27 and average accuracies of 53.81% and 54.34% on LLaMA-13B and LLaMA-2-13B, respectively. The $8{:}16$ results for the two 13B models are not yet available and are marked as – in the table.

C.3 Generalization Across Architectures and Model Scales

To examine whether score refinement transfers to a different model family, a larger parameter scale, and newer LLaMA generations, we evaluate LoRScore on OPT-2.7B, LLaMA-30B, LLaMA-1-13B, and LLaMA-3.1-8B. For OPT-2.7B, we prune all linear projections in the attention and MLP blocks (`q_proj`, `k_proj`, `v_proj`, `out_proj`, `fc1`, and `fc2`) at 50%, 60%, and 80% unstructured sparsity; for the three LLaMA models, we apply 2:4 semi-structured sparsity. All models use rank $r = 32$, Wanda-based initialization, alternating factor updates, and the progressive sparsity schedule; the learning rate is set to 10^{-3} for OPT-2.7B, 10^{-5} for LLaMA-30B, 4×10^{-3} for LLaMA-1-13B, and 4×10^{-4} for LLaMA-3.1-8B. Table 7 reports WikiText-2 perplexity under both the token-level and word-level protocols, together with zero-shot accuracy on seven tasks.

LoRScore transfers across all four settings. Under 2:4 sparsity, it retains approximately 91% of the dense model's average zero-shot accuracy on LLaMA-30B (55.95 vs. 61.45), 87% on LLaMA-1-13B (50.68 vs. 58.22), and 82% on LLaMA-3.1-8B (49.18 vs. 59.88). The corresponding token-level perplexity increases from 4.10 to 6.80, from 5.09 to 9.08, and from 5.84 to 10.85, respectively. Retention is thus highest for the largest model and lowest for LLaMA-3.1-8B, which may reflect that the more extensively trained LLaMA-3.1 has less redundancy to remove at a comparable scale. On OPT-2.7B, the average accuracy decreases by 5.5, 7.8, and 13.7 percentage points at 50%, 60%, and 80% sparsity, respectively, and token-level perplexity grows from 12.47 to 24.27 and 32.15 at 50% and 60% sparsity before rising sharply at 80%, showing that LoRScore degrades gracefully up to moderate sparsity on a non-LLaMA architecture.

Table 7: Results on OPT-2.7B (unstructured sparsity) and LLaMA-30B, LLaMA-1-13B, and LLaMA-3.1-8B (2:4 semi-structured sparsity). WikiText-2 perplexity for OPT-2.7B and LLaMA-30B is reported under the token-level (Tok.) and word-level (Word) protocols; zero-shot accuracy is in %. Non-dense rows report LoRScore results.

Model	Sparsity	Wiki2 PPL↓		Zero-shot accuracy (%)↑							
		Tok.	Word	HellaS.	RACE	PIQA	WinoG.	ARC-E	ARC-C	OBQA	Avg.
OPT-2.7B	Dense	12.47	20.27	45.84	36.27	73.78	61.09	60.82	26.79	25.00	47.08
	50%	24.27	44.85	38.77	33.01	69.26	57.77	51.22	22.10	19.20	41.62
	60%	32.15	62.70	36.37	29.67	67.41	55.80	47.01	21.42	17.40	39.30
	80%	146.76	383.42	28.84	25.93	60.07	50.83	34.76	18.26	14.80	33.36
LLaMA-30B	Dense	4.10	7.37	63.35	40.57	81.07	75.93	80.39	52.82	36.00	61.45
	2:4	6.80	15.08	54.88	38.85	77.69	72.06	73.95	42.24	32.00	55.95
LLaMA-1-13B	Dense	5.09	8.27	59.85	39.43	79.16	72.45	77.31	46.33	33.00	58.22
	2:4	9.08	15.30	50.76	37.22	73.78	66.46	64.31	33.45	28.80	50.68
LLaMA-3.1-8B	Dense	5.84	7.55	60.04	38.66	80.09	74.35	81.48	51.37	33.20	59.88
	2:4	10.85	14.34	47.68	34.74	73.23	65.35	64.35	32.94	26.00	49.18

D Experimental Setup and Hyperparameter Analysis

Unless otherwise specified, we follow the main experimental setup and keep all pretrained weights frozen. We prune all linear projections in the attention and MLP blocks. The pruning scores combine a fixed activation-aware prior, initialized using the Wanda criterion with 32 calibration samples, with a trainable low-rank refinement of rank $r = 32$. The low-rank factors are trained on 5,000 C4 sequences of length 1,024 for 10 epochs with a batch size of 4. **We use a learning rate of 10^{-3} for the 7B models** and 4×10^{-4} for the 13B models. The two low-rank factors are updated alternately every epoch, with one factor optimized while the other remains fixed. Because the 7B models in this section use a larger learning rate (10^{-3}) than in the main experiments (4×10^{-4}), their absolute results are slightly worse than those reported in the main table (e.g., for LLaMA-2-7B with 2:4 sparsity, the Wikitext-2 perplexity has increased). Since all variants within each study share the same learning rate, this difference does not affect the relative comparisons drawn from these experiments.

During training, the sparsity constraint is imposed progressively according to the schedule defined in the main text. The normalized endpoint t_e/T specifies the fraction of total training steps at which the target sparsity constraint is fully imposed. Score optimization continues after this endpoint, allowing

the masks to adapt while maintaining the target sparsity. After training, we export the final binary masks and discard the low-rank factors.

For hyperparameter comparisons, we vary one parameter at a time while keeping the model, sparsity pattern, and remaining settings fixed. We evaluate WikiText-2 test perplexity following the MaskPro protocol and report the arithmetic mean of zero-shot raw accuracies on seven downstream tasks: HellaSwag, RACE, PIQA, WinoGrande, ARC-Easy, ARC-Challenge, and OpenBookQA.

D.1 Effect of learning rate.

We examine the sensitivity of LoRScore to the learning rate of the low-rank factors on LLaMA-2-7B with 50% unstructured sparsity ($r = 32$), varying it from 10^{-5} to 10^{-2} while keeping all other hyperparameters fixed (Table 8). PPL† and PPL‡ denote perplexity evaluated using the Wanda and MaskPro protocols, respectively.

LoRScore is robust over a wide range of learning rates: from 10^{-5} to 10^{-3}, perplexity stays within 11.23–12.07 and the average zero-shot accuracy within 53.29–54.28, with the best results obtained at 10^{-4}. Even the smallest learning rate remains competitive, likely because the activation-aware prior already provides a strong starting point for unstructured pruning, so the learned correction mainly refines it. In contrast, an overly large learning rate (10^{-2}) destabilizes optimization: perplexity rises to 101.38 and the average accuracy drops to 44.16. These results indicate that LoRScore does not require careful tuning within a moderate range of learning rates, whereas excessively large updates should be avoided.

Table 8: Effect of the learning rate on LoRScore (LLaMA-2-7B, 50% unstructured sparsity, $r = 32$). All other hyperparameters are fixed. Perplexity is measured on WikiText-2; zero-shot results report accuracy (%).

Learning rate	PPL† ↓	PPL‡ ↓	HellaS.	RACE	PIQA	WinoG.	ARC-e	ARC-c	OBQA	Avg.↑
Dense	5.12	8.76	57.13	39.62	78.07	69.22	76.35	43.34	31.40	56.45
1×10^{-5}	6.46	11.37	52.76	40.86	76.44	66.54	72.14	37.80	31.00	53.93
1×10^{-4}	**6.38**	**11.23**	53.09	40.48	76.55	66.93	72.69	39.42	30.80	**54.28**
1×10^{-3}	6.65	12.07	52.86	38.09	76.39	67.64	69.65	37.63	30.80	53.29
1×10^{-2}	50.21	101.38	43.54	31.48	68.50	60.14	56.52	26.54	22.40	44.16

D.2 Effect of Score Rank

We investigate the effect of score rank by varying $r \in \{8, 16, 64\}$ on LLaMA-7B and LLaMA-2-7B under 2:4 semi-structured sparsity. All runs in this ablation use a learning rate of 10^{-3}, whereas the results reported in the main experiments use 4×10^{-4}, 10 training epochs, up to 5,000 C4 examples, a sequence length of 1,024, and a training batch size of 4, with all other settings held fixed within each model. PPL† and PPL‡ denote perplexity evaluated using the Wanda and MaskPro protocols, respectively.

Table 9: Effect of score rank under 2:4 sparsity. Best PPL and Avg. per model are bold.

Model	r	PPL† ↓	PPL‡ ↓	HellaS.	RACE	PIQA	WinoG.	ARC-E	ARC-C	OBQA	Avg. ↑
LLaMA-7B	8	10.17	17.31	47.68	35.79	73.29	64.09	65.40	33.11	25.20	49.22
LLaMA-7B	16	**9.40**	16.97	48.01	34.74	73.18	64.64	64.02	32.00	24.80	48.77
LLaMA-7B	32	9.86	**16.78**	48.12	35.60	73.05	65.64	64.56	33.53	25.20	49.39
LLaMA-7B	64	9.82	17.22	48.64	34.83	73.07	63.38	64.65	32.68	27.60	**49.26**
LLaMA-2-7B	8	**8.30**	15.68	46.98	36.46	72.91	63.22	65.28	32.85	26.20	49.13
LLaMA-2-7B	16	**8.30**	15.85	47.19	35.02	72.36	62.90	64.90	32.25	24.40	48.43
LLaMA-2-7B	32	8.32	**15.58**	47.82	35.89	72.52	64.25	65.19	32.00	24.40	48.87
LLaMA-2-7B	64	8.41	15.75	47.69	35.60	72.31	63.54	66.84	33.02	26.20	**49.31**

Table 10: Sensitivity to ρ_{end}. PPL_1 and PPL_2 denote the two perplexity metrics, and Avg. is the mean accuracy (%) across seven downstream tasks. Best results are highlighted in bold.

ρ	$\text{PPL}_1 \downarrow$	$\text{PPL}_2 \downarrow$	HellaS.	RACE	PIQA	WinoG.	ARC-E	ARC-C	OBQA	Avg.↑
0.3	**8.25**	**15.31**	47.93	35.02	**73.50**	**64.56**	**66.41**	33.02	25.00	**49.35**
0.5	8.31	15.50	**47.96**	34.64	72.74	64.33	66.25	**33.11**	24.80	49.12
0.6	8.32	15.58	47.82	**35.89**	72.52	64.25	65.19	32.00	24.40	48.87
0.8	8.45	16.19	47.14	35.69	72.63	63.46	65.95	31.74	**25.40**	48.86

As shown in Table 9, small score ranks already achieve competitive performance, while increasing the rank does not consistently improve both perplexity and downstream accuracy. Although $r = 64$ achieves the highest average zero-shot accuracy on both models, its improvements over $r = 8$ are only 0.04 and 0.18 percentage points, respectively. These results support using a compact low-rank parameterization: increasing the refinement capacity offers limited additional gains in average accuracy under the evaluated settings.

D.3 Sensitivity to pruning schedule end ratio.

We evaluate $\rho_{\text{end}} \in \{0.3, 0.5, 0.6, 0.8\}$ while keeping the remaining experimental settings fixed. Performance varies modestly over the tested range: the two perplexity metrics range from 8.25 to 8.32 and from 15.31 to 15.58, respectively, while average downstream accuracy ranges from 48.87% to 49.35%. Among the evaluated settings, $\rho_{\text{end}} = 0.3$ achieves the lowest perplexity on both metrics and the highest average accuracy, although individual tasks favor different values. Increasing ρ_{end} from 0.3 to 0.6 raises the two perplexities by 0.07 and 0.27 and reduces average accuracy by 0.48 percentage points. These results suggest limited sensitivity to ρ_{end} within the evaluated range, with a modest aggregate advantage for $\rho_{\text{end}} = 0.3$.

D.4 Effect of Training Epochs and Training Samples

We study how the training budget affects LoRScore on LLaMA-2-7B under 2:4 sparsity ($r = 32$), varying either the number of training epochs or the number of C4 training samples while keeping all other hyperparameters fixed (learning rate 10^{-3}). The default configuration uses 10 epochs and 5,000 samples. Table 11 reports WikiText-2 perplexity under the word-level protocol together with the average zero-shot accuracy over seven tasks.

Table 11: Effect of the training budget on LoRScore (LLaMA-2-7B, 2:4, $r = 32$). Left: number of training epochs (5,000 samples). Right: number of training samples (10 epochs).

Epochs	PPL↓	Avg.↑
5	14.99	49.59
10	15.58	48.87

Samples	PPL↓	Avg.↑
3,000	15.03	49.33
4,000	15.33	49.17
5,000	15.58	48.87

Reducing the training budget does not degrade LoRScore. Halving the number of epochs from 10 to 5 yields slightly lower perplexity (14.99 vs. 15.58) and a higher average accuracy (49.59 vs. 48.87), and training on 3,000 instead of 5,000 samples shows the same trend (15.03 and 49.33). These differences are small and comparable to the variation we observe between repeated runs of the same configuration, so we interpret them as evidence that LoRScore converges quickly rather than as a benefit of shorter training. Because only the low-rank factors are optimized, a modest training budget suffices to refine the activation-aware prior, which keeps the cost of LoRScore low.