Train Overcomplete, Deploy Compact: Scaling Recovery Capacity for Structured LLM Pruning

Seungmin Oh **Donggeon Lee** **Jongbin Ryu***
Ajou University, South Korea
{seungminoh, donggeon_lee, jongbinryu}@ajou.ac.kr

Abstract

Large language models achieve strong performance across diverse tasks, but deployment remains costly because of memory, latency, and energy demands. Structured pruning reduces these costs by removing architectural components, yet its recovery stage is often limited by a mismatch between the recovery module's representational capacity and the complexity of the removed knowledge. We call this bottleneck the capacity-knowledge asymmetry and propose **OverRep**, an **Over**complete **Rep**arameterization framework for structured LLM pruning. Following the principle of *"train overcomplete, deploy compact"*, OverRep temporarily overparameterizes the recovery module during training to absorb complex knowledge distilled from the original model. After recovery, the overcomplete re-parameterization is algebraically merged into a mathematically equivalent compact module, preserving the pruned model's inference-time architecture and computational cost. OverRep further introduces an annealed activation that enables nonlinear training dynamics while converging to a linear regime for exact algebraic merging. Across three backbone families, OverRep improves retained reasoning performance over strong recovery baselines by up to 5.5 and 8.4 points at 25% and 50% pruning, respectively, while keeping memory usage and TFLOPs comparable to existing recovery methods. Our code is available at https://github.com/mmai-laboratory/OverRep.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse tasks, but their large parameter counts impose substantial memory, latency, and energy costs during deployment. Improving model efficiency while preserving strong performance has therefore become a central challenge in contemporary LLM research.

*Corresponding author.

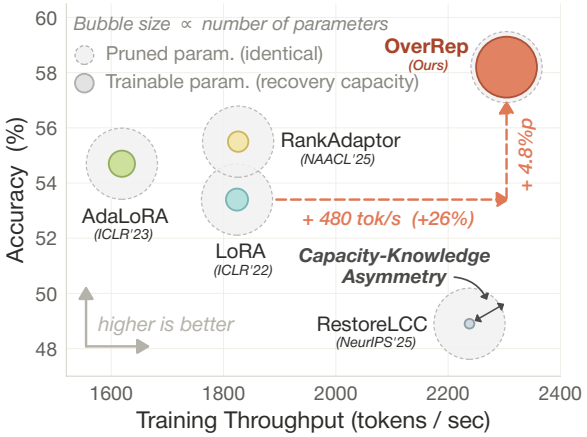


Figure 1: Recovery capacity, accuracy, and throughput during recovery of pruned LLaMA3-8B. Conventional recovery methods use limited recovery capacity relative to the pruned parameters, leading to capacity-knowledge asymmetry. In contrast, OverRep allocates larger recovery capacity and improves both accuracy and throughput, mitigating capacity-knowledge asymmetry.

Structured pruning offers a hardware-efficient way to reduce these costs by removing architectural components such as hidden dimensions, attention heads, or transformer blocks. A standard pruning pipeline consists of two stages: identifying redundant structures and then recovering the pruned model through fine-tuning. While the first stage has been extensively studied through effective pruning criteria (Ma et al., 2023; An et al., 2024; Chen et al., 2025; Wang et al., 2025b; Zhang et al., 2024; Men et al., 2025), the recovery stage remains comparatively underexplored. Existing recovery methods mainly rely on LoRA (Hu et al., 2022) or information-loss compensation such as RestoreLCC (Feng et al., 2025), but their limited representational capacity can make it difficult to reconstruct the complex transformations removed by pruning, especially at high pruning ratios.

We refer to this bottleneck as *Capacity-Knowledge Asymmetry*, as illustrated in Fig. 1. Structured pruning may remove many parameters

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 6336–6355
October 24-29, 2026 ©2026 Association for Computational Linguistics

that encode meaningful model knowledge, whereas only a small fraction of parameters is updated by conventional recovery methods. This creates a fundamental mismatch between the capacity available for recovery and the complexity of the knowledge that needs to be restored. The problem becomes particularly severe when multiple transformer blocks are pruned, as a compact recovery module must replicate the rich compositional mapping those blocks previously contained.

To address the capacity-knowledge asymmetry, we propose **OverRep**, a recovery framework that strategically enhances the representational capacity of the recovery module during fine-tuning and then folds the overcomplete module back into the target compact architecture at inference time through re-parameterization. Specifically, OverRep constructs a temporarily overcomplete recovery module (ORM) with expanded training-time capacity, allowing the pruned model to absorb complex knowledge distilled from the original model during recovery. We further introduce an annealed activation function that provides nonlinear training dynamics early on and smoothly converges to a linear regime, enabling exact algebraic merging for deployment. This design decouples training-time expressiveness from inference-time efficiency, allowing the pruned model to benefit from a richer recovery process without additional deployment cost. Our contributions are summarized as follows:

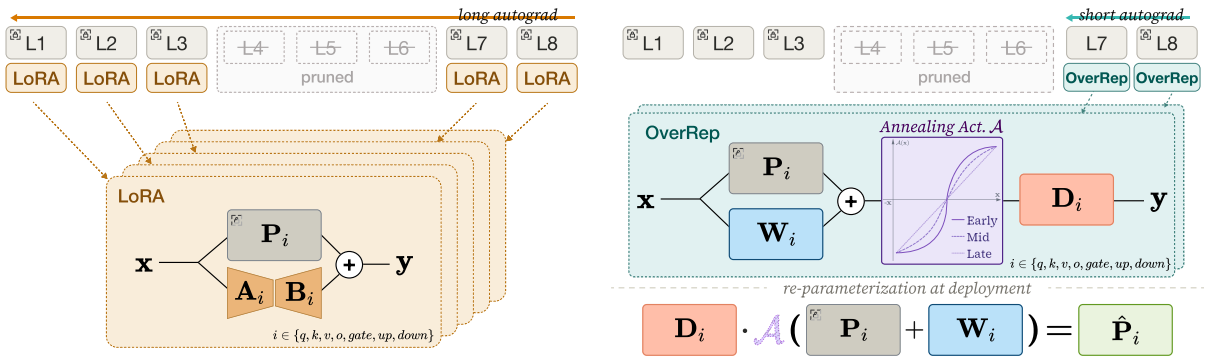
- We identify the *Capacity-Knowledge Asymmetry* as an underexplored bottleneck in structured LLM pruning recovery, where limited recovery capacity hinders reconstructing the knowledge removed by pruning.
- We propose **OverRep**, a recovery framework that temporarily constructs an overcomplete recovery module during training to absorb more pruned knowledge, then algebraically merges it with no inference-time overhead.
- We introduce an annealed activation that enables nonlinear training while preserving exact algebraic merging at deployment.
- We show that larger recovery capacity improves accuracy and throughput without proportionally increasing memory or TFLOPs.

2 Related Work

2.1 Structured Pruning

Pruning Criteria. A substantial body of prior work has focused on designing principled importance criteria for identifying redundant structures. LLM-Pruner (Ma et al., 2023) identifies groups of coupled structures, such as the projection matrices within multi-head self-attention, and prunes them jointly to preserve structural consistency using Taylor expansion-based importance estimation. Another line of research focuses on depth reduction to improve inference efficiency. LaCo (Yang et al., 2024) demonstrates that merging rear layers into a preceding layer with an output-preservation objective does not significantly degrade model performance. ShortGPT (Men et al., 2025) and Streamline (Chen et al., 2025) show that many transformer blocks are highly redundant and can be removed based on block influence scores or layer-wise cosine similarity of activations. FinerCut (Zhang et al., 2024) extends this direction by decoupling attention and feed-forward sub-layers, enabling finer-grained pruning decisions. SliceGPT (Ashkboos et al., 2024) applies principal component analysis to input activations to remove less informative embedding dimensions. Our empirical evaluation primarily considers block- or layer-level pruning.

Recovery Strategy. Most existing recovery methods utilize LoRA-based parameter-efficient fine-tuning (Hu et al., 2022; Zhang et al., 2023; Kopiczko et al., 2024). LoRA-based approaches are the most widely adopted, but they were originally designed for domain adaptation rather than for reconstructing the knowledge lost through pruning. Recognizing this mismatch, recent methods adapt LoRA to the reconstruction objective. RankAdaptor (Zhou et al., 2025) allocates layer-specific LoRA ranks via a performance model to address the uneven structural modification caused by pruning, and RestoreLCC (Feng et al., 2025) injects learnable component vectors into pruning-affected attention heads, identified through contrastive probing, to compensate for the lost directional information. Both refine LoRA-based recovery but remain limited by the low-rank parameterization. Another family of methods recovers the pruned model via layer-wise distillation, minimizing activation discrepancies with the original model at intermediate layers. This is adopted as the recovery procedure in works such as Streamline (Chen et al., 2025) and



(a) LoRA

(b) OverRep (Ours)

Figure 2: Comparison between standard recovery methods and OverRep. LoRA updates the frozen weight $\mathbf{P}_i$ in the pruned model via a low-rank component $\mathbf{A}_i \cdot \mathbf{B}_i$, which limits recovery capacity and requires a long backpropagation path. In contrast, OverRep is applied only after pruned layers, introducing training-time components $\mathbf{W}_i$ and $\mathbf{D}_i$ with an annealed activation $\mathcal{A}$. This design enables a shorter backpropagation path. At deployment, OverRep re-parameterizes all additional parameters into $\hat{\mathbf{P}}_i$, preserving performance while maintaining efficiency.

CoMe (Wang et al., 2025a).

While prior work has advanced pruning criteria and parameter-efficient recovery objectives, OverRep explores a complementary direction that increases the expressive capacity available during recovery. Even with the same pruning mask and distillation target, recovery effectiveness depends on whether the recovery module can approximate the transformation removed by pruning. OverRep therefore temporarily expands the recovery module during training and folds it back into the compact module at deployment, as shown in Fig. 2.

2.2 Re-parameterization

Re-parameterization builds a more expressive training-time topology, usually with multiple parallel branches, which are algebraically merged into a single equivalent structure at inference. This enhances representational capacity during optimization without inference-time overhead. Foundational work in convolutional networks (Ding et al., 2019, 2021a,b) established this paradigm by introducing multi-branch designs that capture richer spatial patterns. FastViT (Vasu et al., 2023) extended re-parameterization to vision transformers, while Neural Substitution (Oh and Ryu, 2024) generalized the framework by interpreting block-level skip connections as branch-level structural relationships.

OverRep adopts this principle of enhanced training-time topology and re-parameterization. Most previous methods expand capacity through width-wise branches leveraging convolutional kernel diversity. In contrast, OverRep uses a cascaded multiplicative-additive factorization designed for projection-dominated transformer blocks where kernel-shape diversity is absent. Classical re-parameterization requires merged branches to remain algebraically linear. OverRep enables nonlinear recovery while maintaining exact mergeability by using an annealed activation that starts and ends in the linear regime, with warm-up cosine annealing and final linear stabilization. A warm-up phase preserves the pretrained operating point, which is essential for recovering frozen pretrained models rather than training them from scratch. OverRep incorporates recovery-specific design choices, such as identity or zero initialization and selective placement within frozen recovery blocks.

3 Method

3.1 Preliminaries

We begin with the standard formulation of layer-wise distillation for recovering a pruned model. Consider a contiguous segment of n original transformer blocks that is compressed into a single recovery transformer block in the pruned model. For an input sample $\mathbf{z} \sim \mathcal{D}$, let $\mathbf{h}_\ell(\mathbf{z}) \in \mathbb{R}^{t \times d}$ denote the hidden state entering the ℓ-th transformer block, where t is the sequence length and d is the hidden dimension. For brevity, we write $\mathbf{h}_\ell$ for $\mathbf{h}_\ell(\mathbf{z})$ when the dependence on $\mathbf{z}$ is clear. Suppose blocks p through $p+n-2$ are pruned, while block $p+n-1$ is retained as the recovery module. Standard layer-wise distillation trains this retained block, denoted by $\mathcal{F}(\cdot;\theta)$ with pretrained parameters θ, to map the hidden state before the pruned segment directly to

the hidden state after the segment:

$$\mathcal{L}_{\text{std}}(\theta) = \mathbb{E}_{\mathbf{z}}\left[\|\mathcal{F}(\mathbf{h}_p;\theta) - \mathbf{h}_{p+n}\|_F^2\right]. \quad (1)$$

This objective is optimized using a single standard transformer block to approximate the composite transformation originally realized by n consecutive blocks. In practice, such compression often induces a substantial capacity-knowledge asymmetry bottleneck. The parameterization of θ may not adequately capture the complex nonlinear relationship between $\mathbf{h}_p$ and $\mathbf{h}_{p+n}$.

3.2 Overcomplete Re-parameterization for Scaling Recovery Capacity

OverRep addresses this bottleneck by replacing the compact recovery block during training with an Overcomplete Recovery Module (ORM). The central principle is *"train overcomplete, deploy compact"*: the ORM is temporarily overparameterized during training, but is algebraically folded back into the original compact architecture after training. Concretely, OverRep uses the pretrained parameters θ with auxiliary parameters ϕ, yielding an overcomplete re-parameterization $\mathcal{F}_{\text{ORM}}(\cdot;\theta,\phi)$. The recovery objective is defined as:

$$\mathcal{L}_{\text{ours}}(\theta,\phi) = \mathbb{E}_{\mathbf{z}}\left[\|\mathcal{F}_{\text{ORM}}(\mathbf{h}_p;\theta,\phi) - \mathbf{h}_{p+n}\|_F^2\right]. \quad (2)$$

To optimize $\mathcal{L}_{\text{ours}}$, OverRep either freezes θ and optimizes only ϕ, or jointly fine-tunes θ and ϕ to increase recovery flexibility. For notational simplicity, we present OverRep with a single recovery block, although $\mathcal{F}_{\text{ORM}}$ may denote a multi-block recovery module. In our experiments, we use a two-block instantiation, as detailed in Appendix A.

By introducing ϕ, the parameter space is substantially enlarged, facilitating more accurate reconstruction of the target hidden state $\mathbf{h}_{p+n}$. Importantly, ϕ is structured to be algebraically integrated with θ into a corresponding compact parameter set $\hat{\theta}$ through re-parameterization $\mathcal{R}$ after training as:

$$\hat{\theta} = \mathcal{R}(\theta,\phi), \quad \mathcal{F}_{\text{ORM}}(\mathbf{h};\theta,\phi) \equiv \mathcal{F}(\mathbf{h};\hat{\theta}),\ \forall\,\mathbf{h}. \quad (3)$$

Accordingly, OverRep increases training-time representational capacity while preserving the original architecture and incurring no additional parameters, memory, or inference-time latency overhead.

3.3 Overcomplete Recovery Module

The ORM enlarges the recovery module's optimization space during training without adding inference-time architectural overhead. Let $\mathcal{F}(\cdot;\theta)$ denote the standard recovery block, which contains a set of linear projection matrices as:

$$\begin{aligned}\theta &= \{\mathbf{P}_i : i \in \mathcal{I}\},\\ \mathcal{I} &= \{q,k,v,o,gate,up,down\},\end{aligned} \quad (4)$$

where q, k, v, and o denote the attention projections, while $gate$, up, and $down$ denote the feed-forward projections. Each projection $\mathbf{P}_i \in \mathbb{R}^{d_i^{\text{out}}\times d_i^{\text{in}}}$ may have its own input and output dimensions depending on the architecture. To construct the ORM $\mathcal{F}_{\text{ORM}}(\cdot;\theta,\phi)$, OverRep expands each base parameter $\mathbf{P}_i$ with an auxiliary parameter pair $(\mathbf{W}_i,\mathbf{D}_i)$. During training, the standard linear projection is replaced by an overcomplete mapping. For an input $\mathbf{x} \in \mathbb{R}^{d_i^{\text{in}}}$, the overcomplete projection is:

$$\mathcal{F}_{\text{ORM}}(\mathbf{x};\ \mathbf{P}_i,\mathbf{W}_i,\mathbf{D}_i) = \mathbf{D}_i(\mathbf{P}_i + \mathbf{W}_i)\mathbf{x}, \quad (5)$$

where $\mathbf{W}_i \in \mathbb{R}^{d_i^{\text{out}}\times d_i^{\text{in}}}$ serves as an additive parallel branch that broadens the optimization space, and $\mathbf{D}_i \in \mathbb{R}^{d_i^{\text{out}}\times d_i^{\text{out}}}$ functions as a cascaded multiplicative transformation applied in the output space. Together, these auxiliary parameters improve optimization flexibility during training. To ensure that the ORM reduces to the original module at initialization, we set $\mathbf{W}_i = \mathbf{0}$ and $\mathbf{D}_i = \mathbf{I}$. This identity initialization guarantees that recovery begins from the pretrained operating point.

Annealed Activation. Exact algebraic merging requires the ORM to be linear at deployment, but keeping it linear throughout recovery limits training-time flexibility. OverRep therefore introduces an *annealed activation* that provides nonlinear training dynamics while gradually returning to the identity map before re-parameterization. At training step s, OverRep replaces Eq. (5) with

$$\mathcal{F}_{\text{ORM}}^{(s)}(\mathbf{x};\ \mathbf{P}_i,\mathbf{W}_i,\mathbf{D}_i) = \mathbf{D}_i\mathcal{A}_s((\mathbf{P}_i + \mathbf{W}_i)\mathbf{x}), \quad (6)$$

where $\mathcal{A}_s$ interpolates between the recovery model's activation function σ and the identity map:

$$\mathcal{A}_s(\mathbf{x}) = \alpha_s \cdot \sigma(\mathbf{x}) + (1-\alpha_s)\mathbf{x}. \quad (7)$$

The coefficient $\alpha_s \in [0,1]$ follows a warm-up cosine schedule with a final linear-stabilization phase. Let S be the total number of recovery steps, and let ρ_{w} and ρ_{l} denote the warm-up ratio and the final linear-stabilization ratio. We define

$$\alpha_s = \begin{cases} s/S_{\text{w}}, & 0 \le s < S_{\text{w}},\\ \frac{1}{2}\left(1+\cos\left(\pi\frac{s-S_{\text{w}}}{S_{\text{d}}-S_{\text{w}}}\right)\right), & S_{\text{w}} \le s < S_{\text{d}},\\ 0, & S_{\text{d}} \le s \le S, \end{cases} \quad (8)$$

where $S_{\mathrm{w}} = \rho_{\mathrm{w}} S$ and $S_{\mathrm{d}} = (1-\rho_{\mathrm{l}})S$. Starting from $\alpha_0 = 0$ ensures that recovery begins from the pretrained operating point, while the warm-up phase rapidly introduces nonlinear training dynamics. The final ρ_{l} fraction of training keeps $\alpha_s = 0$, allowing the weights to stabilize in the exact linear regime before re-parameterization. Thus, at step S, $\mathcal{A}_S(\mathbf{x}) = \mathbf{x}$, and Eq. (6) reduces to

$$\mathcal{F}^{(S)}_{\mathrm{ORM}}(\mathbf{x}; \mathbf{P}_i, \mathbf{W}_i, \mathbf{D}_i) = \mathbf{D}_i(\mathbf{P}_i + \mathbf{W}_i)\mathbf{x}. \quad (9)$$

Re-parameterization. The re-parameterization operation $\mathcal{R}$ merges each triplet $\{\mathbf{P}_i, \mathbf{W}_i, \mathbf{D}_i\}$ into a single weight matrix $\hat{\mathbf{P}}_i$:

$$\hat{\mathbf{P}}_i = \mathbf{D}_i(\mathbf{P}_i + \mathbf{W}_i), \quad \forall i \in \mathcal{I}. \quad (10)$$

The pruned model is therefore trained using $\mathcal{F}^{(s)}_{\mathrm{ORM}}(\mathbf{h}; \theta, \phi)$ but deployed as the original compact module $\mathcal{F}(\mathbf{h}; \hat{\theta})$. In this way, OverRep benefits from enhanced recovery capacity while preserving the inference efficiency of the pruned model.

Optimization geometry of the merged projection. Although the merged model has the same architecture and function class as a directly fine-tuned one, OverRep changes the training parameterization and optimization geometry of the compact deployed weight. For a projection in the final linear phase, let $\mathbf{M}_i = \mathbf{P}_i + \mathbf{W}_i$ and $\hat{\mathbf{P}}_i = \mathbf{D}_i\mathbf{M}_i$, and let $\mathbf{G}$ denote the gradient of $\mathcal{L}$ with respect to the projection weight in use, i.e., $\partial\mathcal{L}_{\mathrm{ours}}/\partial\hat{\mathbf{P}}_i$ for OverRep and $\partial\mathcal{L}/\partial\mathbf{P}_i$ for direct fine-tuning. Under continuous-time gradient flow on $\mathbf{D}_i$ and $\mathbf{W}_i$,

$$\frac{d\hat{\mathbf{P}}_i}{dt} = -\left(\mathbf{D}_i\mathbf{D}_i^\top\right)\mathbf{G} \; - \; \mathbf{G}\left(\mathbf{M}_i^\top\mathbf{M}_i\right), \quad (11)$$

whereas directly optimizing $\mathbf{P}_i$ without $\mathbf{W}_i$ and $\mathbf{D}_i$ gives $d\mathbf{P}_i/dt = -\mathbf{G}$. The factorized parameterization therefore induces parameter-dependent, positive-semidefinite preconditioning on both sides of the gradient. At identity/zero initialization, where $\mathbf{D}_i = \mathbf{I}$ and $\mathbf{W}_i = \mathbf{0}$, Eq. (11) reduces to $-\mathbf{G} - \mathbf{G}\mathbf{P}_i^\top\mathbf{P}_i$. Thus, the initial update is shaped by the spectrum of the pretrained projection. Consequently, OverRep does not merely add trainable parameters. It optimizes the same compact weight under a different geometry that depends on the pretrained weight, analogous to the mechanism studied in overparameterized deep linear networks (Arora et al., 2018). This theoretical result is limited to a frozen-$\mathbf{P}_i$, continuous-gradient-flow model of the final linear phase and does not establish universal superiority or directly model AdamW or nonlinear activation annealing. The annealed activation provides a complementary benefit that the theoretical result does not capture. While $\alpha_s > 0$, OverRep optimizes over a nonlinear hypothesis class before converging to the linear regime required for exact merging, whose benefit is shown in § 4.4. Section 4.5 empirically separates parameterization from capacity using parameter-matched LoRA.

4 Experiments

We conduct comprehensive experiments on both reasoning and generation benchmarks to compare OverRep with state-of-the-art structured pruning methods. For pretrained backbones, we use LLaMA2 (Touvron et al., 2023), LLaMA3 (Grattafiori et al., 2024), and Qwen3 (Yang et al., 2025). We compare against representative recent pruning approaches, including LaCo (Yang et al., 2024), ShortGPT (Men et al., 2025), and Streamline (Chen et al., 2025). Additionally, to directly assess the effectiveness of OverRep in the recovery stage, we conduct controlled comparisons under an identical pruning criterion with recent recovery methods, including LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023), RankAdaptor (Zhou et al., 2025), and RestoreLCC (Feng et al., 2025). This isolates the recovery strategy from the pruning criterion, enabling a fair comparison and showing that OverRep outperforms existing recovery approaches.

We standardize both the datasets and the evaluation framework across all methods. Specifically, for recovery, we use a fixed subset of the `FineWeb-Edu` (Lozhkov et al., 2024) dataset consisting of 120,000 training samples and 4,000 test samples. For evaluation, we adopt the widely used `lm-eval-harness` (Gao et al., 2024) framework. This standardized setup enables fairer, more controlled comparisons across methods. Implementation details are provided in Appendix B.

4.1 Training Efficiency and Resource Analysis

We analyze the relationship among trainable capacity, training resources, throughput, and recovery accuracy in Fig. 3.[1] Although OverRep introduces substantially more trainable parameters, this increase does not translate into a proportional in-

[1] All analyses are conducted with a batch size of 1 and a sequence length of 1024 on two RTX 3090 GPUs (24GB).

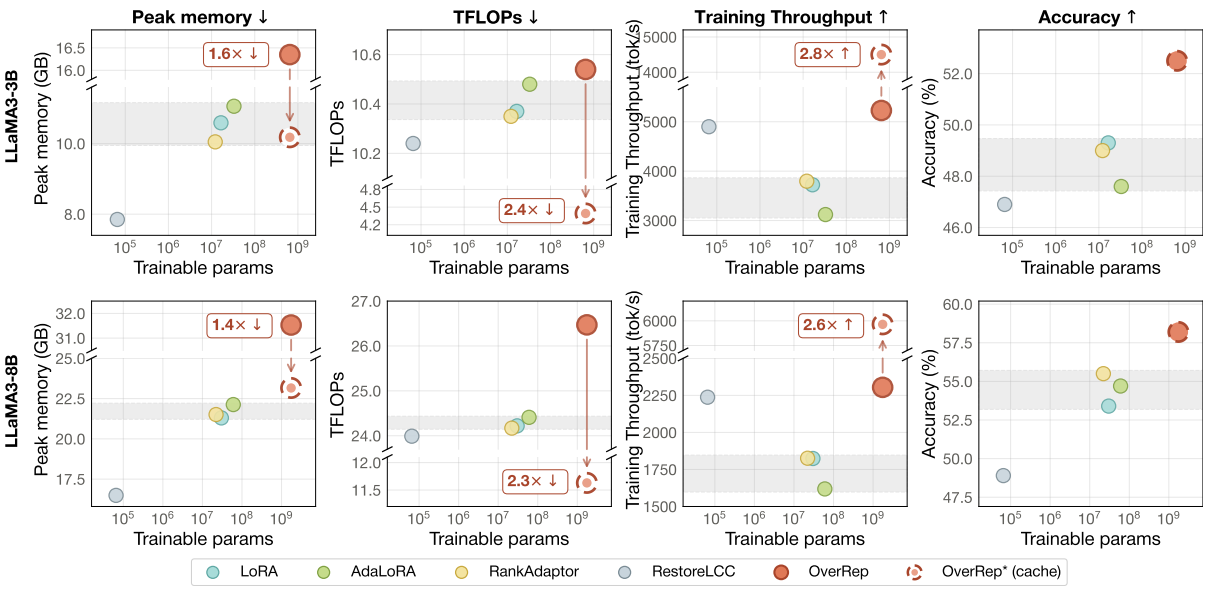


Figure 3: Training efficiency and recovery performance on LLaMA3-3B and -8B. Gray bands show the range of conventional baselines across resource and performance metrics. OverRep uses more trainable recovery parameters, yet maintains comparable memory and compute costs while improving throughput and accuracy. OverRep* further improves efficiency by caching frozen-prefix activations, without changing the final recovered model or its accuracy.

crease in training cost. On both backbones, OverRep remains comparable to, or only slightly higher than, existing recovery baselines in peak memory and TFLOPs, while achieving higher throughput and stronger reasoning accuracy. This is because OverRep localizes the recovery module after the pruned segment, providing a larger optimization space without requiring backpropagation through long autograd paths, as shown in Fig. 2. These results indicate that trainable parameter count alone is an incomplete proxy for recovery cost.

OverRep also supports a cached variant, OverRep*, which reuses precomputed activations from the frozen prefix before the recovery module. As shown in Fig. 3, this reduces peak GPU memory and TFLOPs by up to 1.6× and 2.4×, respectively, while improving throughput by up to 2.8× over vanilla OverRep. Since caching preserves the recovery objective and the final re-parameterized model, OverRep* achieves the same accuracy as OverRep while providing additional efficiency gains.

4.2 Reasoning Performance

We evaluate reasoning performance under two settings at two pruning ratios (PR). Table 1 reports task-level raw accuracy and average retained performance (RP)[2] for controlled recovery methods, showing how each recovery strategy behaves across benchmarks. Table 2 shows average RP for complete pruning pipelines, enabling comparison against methods that use their own pruning and recovery procedures. Detailed results of each task are provided in Appendix D.1.

Controlled recovery. We isolate the effect of recovery by applying all methods to the same pruned model, using a shared pruning criterion that removes the last blocks except the final block. As shown in Tab. 1, OverRep generally achieves the strongest performance across backbones and pruning ratios, with clear gains at aggressive pruning ratios. In terms of RP, OverRep achieves the best result in all controlled settings, improving over the strongest recovery baseline by up to 5.5 points at 25% pruning and by up to 8.4 points at 50%.

Complete pruning pipeline. We compare OverRep with complete pruning pipelines that use their own pruning and recovery procedures. As shown in Tab. 2, OverRep achieves the strongest RP across all backbones and pruning ratios, suggesting that a fixed pruning mask combined with sufficient recovery capacity can outperform specialized pruning designs. Compatibility with other criteria is examined in Appendix C.

The improvements are especially pronounced under aggressive pruning, supporting our hypothesis that increased training-time recovery capacity mitigates the capacity-knowledge asymmetry.

[2] $\text{RP} = \frac{\text{Pruned model average accuracy}}{\text{Dense model average accuracy}} \times 100$

	PR	Method	ARC_C	ARC_E	BoolQ	Hella.	MathQA	MMLU	OBQA	PIQA	RACE	Wino.	Avg.	RP
LLaMA2-7B	25%	LoRA	37.6	65.4	77.6	66.2	25.3	24.6	37.8	72.5	39.0	**66.6**	51.3	88.7
		AdaLoRA	37.5	65.1	**77.8**	66.1	25.0	24.8	38.0	72.1	**39.1**	66.1	51.2	88.5
		RankAdaptor	39.2	64.4	68.9	66.9	23.4	28.1	37.6	69.6	35.9	62.8	49.7	86.0
		RestoreLCC	35.3	61.1	76.3	63.8	**25.3**	27.3	36.6	71.5	37.4	66.2	50.1	86.7
		OverRep	**41.0**	**70.5**	73.6	**67.7**	25.1	**38.5**	**40.2**	**74.3**	37.9	66.1	**53.5**	**92.6**
	50%	LoRA	27.8	44.3	62.4	45.1	22.0	**28.7**	30.4	61.0	32.6	59.1	41.3	71.5
		AdaLoRA	26.4	41.2	62.5	43.7	22.4	26.1	28.2	60.2	31.8	59.6	40.2	69.6
		RankAdaptor	27.7	44.8	62.4	45.2	22.2	28.6	30.0	61.0	32.4	58.6	41.3	71.4
		RestoreLCC	25.3	38.0	**62.6**	40.4	**23.1**	23.8	28.2	59.4	31.1	59.6	39.2	67.7
		OverRep	**31.5**	**60.0**	62.3	**51.1**	21.8	24.4	**33.8**	**66.9**	**32.7**	**59.9**	**44.4**	**76.9**
LLaMA2-13B	25%	LoRA	**46.8**	73.1	78.7	73.1	25.7	49.4	43.6	75.4	39.3	69.4	57.5	94.6
		AdaLoRA	43.9	69.4	**81.7**	70.8	26.1	**49.7**	42.0	74.5	38.6	70.3	56.7	93.3
		RankAdaptor	45.9	72.3	78.6	73.0	26.0	49.5	43.8	75.3	39.3	71.0	57.5	94.6
		RestoreLCC	38.8	65.3	77.7	67.4	25.5	44.8	40.0	72.3	36.9	69.5	53.8	88.6
		OverRep	46.0	**76.1**	76.7	**74.2**	**27.5**	47.4	**43.8**	**76.8**	**39.3**	**71.3**	**57.9**	**95.3**
	50%	LoRA	32.2	54.8	64.3	56.6	22.9	44.7	35.4	65.3	34.7	65.0	47.6	78.3
		AdaLoRA	30.0	48.8	62.4	51.3	22.4	33.9	31.4	64.7	33.8	63.8	44.3	72.8
		RankAdaptor	32.7	55.0	64.0	56.6	22.7	44.7	35.6	65.6	35.4	65.7	47.8	78.7
		RestoreLCC	29.4	43.2	62.2	45.4	22.6	29.7	31.8	59.2	33.0	63.4	42.0	69.1
		OverRep	**36.7**	**65.1**	**70.9**	**59.1**	**23.7**	**50.3**	**39.0**	**69.8**	**37.8**	**67.0**	**51.9**	**85.5**
LLaMA3-3B	25%	LoRA	35.7	54.8	68.2	58.3	**26.6**	46.3	34.2	69.2	36.2	63.5	49.3	84.7
		AdaLoRA	32.3	54.1	64.6	53.9	24.1	47.6	33.2	66.2	34.3	65.8	47.6	81.8
		RankAdaptor	35.9	56.5	66.9	58.1	26.2	45.7	32.8	68.4	35.3	63.9	49.0	84.1
		RestoreLCC	32.3	49.2	**70.4**	50.5	24.5	46.7	31.4	64.9	33.7	65.3	46.9	80.5
		OverRep	**39.2**	**66.0**	67.9	**59.5**	25.7	**54.9**	**36.6**	**70.5**	**38.0**	**66.8**	**52.5**	**90.2**
	50%	LoRA	26.1	41.0	49.3	36.9	23.0	23.1	29.2	60.9	29.7	52.1	37.1	63.8
		AdaLoRA	24.1	34.3	56.4	33.1	22.1	22.9	27.6	57.4	25.4	51.0	35.4	60.8
		RankAdaptor	25.2	42.8	51.6	36.5	23.4	23.2	29.4	60.8	27.6	51.8	37.2	63.9
		RestoreLCC	23.3	30.1	38.2	29.7	22.7	23.0	27.0	54.7	23.9	49.3	32.2	55.3
		OverRep	**29.1**	**55.4**	**61.4**	**42.6**	**23.7**	**24.8**	**32.6**	**64.5**	**30.4**	**56.5**	**42.1**	**72.3**
LLaMA3-8B	25%	LoRA	41.3	61.7	71.7	68.5	28.1	54.0	37.6	71.5	36.8	63.3	53.5	83.9
		AdaLoRA	41.1	64.9	67.6	63.7	29.7	56.5	36.8	71.0	37.9	69.5	53.9	84.5
		RankAdaptor	43.3	67.3	**75.6**	68.6	**29.7**	55.5	38.6	72.5	37.8	66.2	55.5	87.1
		RestoreLCC	38.1	60.0	62.4	57.6	28.1	36.0	33.4	69.2	36.2	68.3	48.9	76.8
		OverRep	**47.4**	**74.8**	75.4	**69.3**	29.2	**60.7**	**40.8**	**74.8**	**38.5**	**70.9**	**58.2**	**91.3**
	50%	LoRA	27.3	42.3	60.7	42.4	21.9	23.0	32.4	62.4	31.2	57.7	40.1	63.0
		AdaLoRA	25.0	37.8	62.0	38.3	21.7	22.9	28.4	59.7	27.9	56.1	38.0	59.6
		RankAdaptor	26.8	41.5	59.9	42.6	21.9	23.1	32.4	62.3	31.0	56.6	39.8	62.5
		RestoreLCC	24.5	33.4	58.1	31.8	21.4	22.9	26.4	53.7	23.0	51.5	34.7	54.4
		OverRep	**33.8**	**58.5**	**62.0**	**48.4**	**22.4**	**23.3**	**34.8**	**67.2**	**32.2**	**60.5**	**44.3**	**69.5**
Qwen3-4B	25%	LoRA	37.5	59.5	63.9	53.9	26.1	**36.9**	33.0	66.1	**36.0**	63.6	47.7	74.1
		AdaLoRA	32.3	53.8	**71.7**	49.8	26.5	32.0	29.8	63.9	34.3	**63.7**	45.8	71.2
		RankAdaptor	36.6	57.9	65.0	53.9	25.7	36.3	33.2	65.4	34.4	63.0	47.1	73.3
		RestoreLCC	31.2	40.8	70.1	40.5	23.5	23.2	31.6	59.4	26.8	60.4	40.8	63.4
		OverRep	**39.1**	**67.3**	64.3	**57.1**	**27.5**	24.3	**35.6**	**70.5**	35.7	62.9	**48.4**	**75.3**
	50%	LoRA	26.3	44.0	54.3	33.1	21.6	23.0	27.2	60.0	26.5	51.6	36.8	57.2
		AdaLoRA	24.6	37.8	45.5	31.1	20.6	22.9	28.4	57.5	24.9	51.8	34.5	53.7
		RankAdaptor	26.9	45.2	55.7	33.1	21.7	23.0	27.8	58.8	25.9	50.8	36.9	57.4
		RestoreLCC	27.0	28.3	60.4	27.6	19.6	22.9	28.0	52.2	21.5	49.3	33.7	52.4
		OverRep	**27.2**	**56.5**	**61.9**	**38.4**	**22.5**	**23.0**	**32.2**	**64.5**	**28.8**	**53.5**	**40.9**	**63.5**
Qwen3-8B	25%	LoRA	40.8	64.0	**69.8**	60.0	**31.0**	65.9	33.0	67.6	36.2	63.2	53.2	79.9
		AdaLoRA	35.3	57.7	62.3	56.2	29.0	52.3	32.2	67.2	35.9	**65.7**	49.4	74.2
		RankAdaptor	41.0	64.5	67.6	58.7	30.7	**67.6**	34.0	66.7	**39.2**	63.9	53.4	80.2
		RestoreLCC	32.9	44.4	62.2	45.2	23.8	63.9	34.2	62.1	30.0	63.2	46.2	69.4
		OverRep	**41.1**	**69.5**	62.7	**60.8**	29.1	64.4	**36.2**	**71.3**	36.4	65.0	**53.7**	**80.6**
	50%	LoRA	27.0	48.1	61.3	35.4	21.1	22.9	29.0	60.7	26.1	51.7	38.3	57.6
		AdaLoRA	25.3	39.5	**62.0**	32.5	21.0	22.9	30.8	59.6	25.6	49.4	36.9	55.4
		RankAdaptor	27.5	48.0	61.6	35.7	21.6	22.9	30.0	60.8	26.6	50.4	38.5	57.9
		RestoreLCC	27.5	27.1	38.0	28.9	18.7	22.8	30.0	54.9	22.6	51.5	32.2	48.4
		OverRep	**28.9**	**58.0**	59.5	**41.5**	**23.1**	**23.0**	**36.4**	**66.4**	**29.7**	**53.3**	**42.0**	**63.1**

Table 1: Experimental results on reasoning tasks reported as accuracy and average RP.

PR	Method	L2-7B	L2-13B	L3-3B	L3-8B	Q3-4B	Q3-8B
25%	LaCo	80.8	88.9	80.7	81.7	73.1	73.4
	ShortGPT	88.2	92.1	82.5	88.2	72.6	72.6
	Streamline	89.4	94.4	86.1	88.9	74.2	74.4
	OverRep	**92.6**	**95.3**	**90.2**	**91.3**	**75.3**	**80.6**
50%	LaCo	70.9	75.2	63.4	62.2	56.7	55.9
	ShortGPT	70.7	77.5	61.5	65.0	56.6	55.8
	Streamline	73.0	82.1	68.4	64.9	61.1	61.7
	OverRep	**76.9**	**85.5**	**72.3**	**69.5**	**63.5**	**63.1**

Table 2: Average reasoning performance of complete pruning pipelines, reported as average RP.

4.3 Generation Performance

We further evaluate OverRep on generation benchmarks to examine whether increased recovery capacity also benefits reasoning-intensive generation. Table 3 reports the average RP over three generation benchmarks (Reddy et al., 2019; Cobbe et al., 2021; Joshi et al., 2017), with detailed raw performance provided in Appendix D.2.

Controlled recovery. OverRep achieves stronger generation performance than controlled recovery baselines across all backbones and pruning ratios. The gains are especially clear under 50% pruning,

PR	Method	L2-7B	L2-13B	L3-3B	L3-8B	Q3-4B	Q3-8B
25%	*controlled recovery methods*						
	LoRA	51.2	60.6	35.0	39.9	19.8	23.9
	AdaLoRA	51.4	55.6	37.4	47.8	22.2	26.7
	RankAdaptor	45.9	60.6	36.7	42.8	17.9	28.2
	RestoreLCC	55.2	57.1	37.2	36.4	13.1	18.9
	complete pruning pipelines						
	LaCo	38.2	62.0	32.0	47.6	18.1	21.6
	ShortGPT	47.8	48.1	26.4	32.1	21.2	24.8
	Streamline	60.3	64.3	45.7	46.8	28.6	31.0
	OverRep	**61.9**	**67.2**	**47.1**	**50.0**	**32.7**	**31.4**
50%	*controlled recovery methods*						
	LoRA	19.1	36.9	9.4	12.1	5.8	7.4
	AdaLoRA	16.9	28.0	4.3	7.3	2.8	3.2
	RankAdaptor	21.2	35.8	8.9	12.3	6.1	7.3
	RestoreLCC	20.5	23.4	2.2	2.0	2.2	2.6
	complete pruning pipelines						
	LaCo	21.5	24.6	6.2	**16.4**	2.5	2.0
	ShortGPT	25.2	34.7	4.1	12.1	1.6	1.0
	Streamline	13.3	30.1	6.5	5.7	6.9	7.4
	OverRep	**29.4**	**42.1**	**10.7**	14.7	**10.2**	**8.7**

Table 3: Average generation performance of controlled recovery methods and complete pruning pipelines, reported as average RP.

where the recovery module must restore more removed knowledge. This suggests that OverRep is not limited to multiple-choice reasoning tasks, but also improves recovery on generation benchmarks.

Complete pruning pipeline. In the generation benchmarks, OverRep remains highly competitive against complete pruning pipelines. At 25% pruning, OverRep achieves the best RP across all backbones, and at 50% pruning, it achieves the best RP on five of six backbones.

Overall, the generation results provide complementary evidence to the reasoning benchmarks. OverRep consistently outperforms recovery baselines and remains competitive with complete pruning pipelines, indicating that increased training-time recovery capacity benefits both reasoning and generative evaluation settings.

4.4 Ablation Study

We conduct ablation studies on LLaMA3-3B with 25% pruning to analyze the contribution of each OverRep component. Table 4 compares different ORM configurations by varying the application of $\mathbf{W}_i$, $\mathbf{D}_i$, and the annealed activation $\mathcal{A}_s$. We measure training time without feature caching. Full application applies each component to the corresponding projections in both recovery blocks, whereas partial application applies each component only to selected projections in the second recovery block, whose pretrained parameters are frozen.

Configuration	Attn. $\mathbf{W}_i$	Attn. $\mathbf{D}_i$	MLP $\mathbf{W}_i$	MLP $\mathbf{D}_i$	Rea.	Gen.	Train time
Plain	–	–	–	–	86.9	35.8	1.8h
Single-component							
Attn $\mathbf{W}_i$	✓	–	–	–	86.7	37.4	2.0h
Attn $\mathbf{D}_i$	–	✓	–	–	86.9	38.0	1.9h
MLP $\mathbf{W}_i$	–	–	✓	–	88.6	43.5	2.7h
MLP $\mathbf{D}_i$	–	–	–	✓	88.7	41.9	4.2h
Double-component							
$\mathbf{W}_i$ only	✓	–	✓	–	88.5	44.5	3.1h
$\mathbf{D}_i$ only	–	✓	–	✓	88.8	45.0	4.4h
Full-component							
Uniform	✓	✓	✓	✓	88.9	45.5	5.8h
Hybrid	✓	P	P	✓	89.0	46.7	5.4h
Hybrid+$\mathcal{A}_s$	✓	P	P	✓	90.2	47.1	5.5h

Table 4: Ablation study of the components of OverRep using LLaMA3-3B at 25% pruning. ✓/P/– indicate full application, frozen-block partial application, and no application, respectively. Rea. and Gen. denote the average RP on reasoning and generation benchmarks.

PR	Method	Params. (M)	Rea.	Gen.	Time (h)
25%	LoRA $r = 16$	16.5	84.7	35.0	8.9
	LoRA $r = 630$	650	79.4	34.3	13.2
	OverRep	650	**90.2**	**47.1**	**4.6**
50%	LoRA $r = 16$	10.4	63.8	9.4	6.2
	LoRA $r = 1000$	651	64.8	**11.4**	9.2
	OverRep	650	**72.3**	10.7	**4.6**

Table 5: Iso-parameter comparison on LLaMA3-3B. The LoRA rank is scaled to match the ∼650M trainable parameters of OverRep under the same protocol, reported as average RP.

The results show that MLP-side overcomplete components are more effective than attention-only components, suggesting that feed-forward transformations play a central role in recovering the mappings removed by layer pruning. We also find that $\mathbf{W}_i$ and $\mathbf{D}_i$ are complementary, as using both components yields stronger performance than using either one alone. Finally, the hybrid configuration with frozen-block partial application and annealed activation achieves the best overall performance, indicating that selective capacity expansion in the frozen block, combined with nonlinear training dynamics, provides an effective trade-off between recovery flexibility and training cost.

4.5 Analysis

Iso-parameter LoRA. To empirically separate parameterization from capacity, we additionally con-

PR	Method	Rea.	Gen.	Time (h)
25%	LoRA	84.7	35.0	8.9
	Full FT	85.9	48.2	12.1
	Logit-KD	87.1	58.1	36.4
	OverRep	90.2	47.1	4.6
	OverRep-KD	**91.8**	**61.0**	7.0
50%	LoRA	63.8	9.4	6.2
	Full FT	67.4	19.2	7.9
	Logit-KD	67.5	22.9	32.5
	OverRep	72.3	10.7	4.6
	OverRep-KD	**73.4**	**26.7**	7.0

Table 6: Comparison with full fine-tuning ('Full FT') of the retained blocks and full-model logit-level distillation ('Logit-KD') on LLaMA3-3B, reported as average RP.

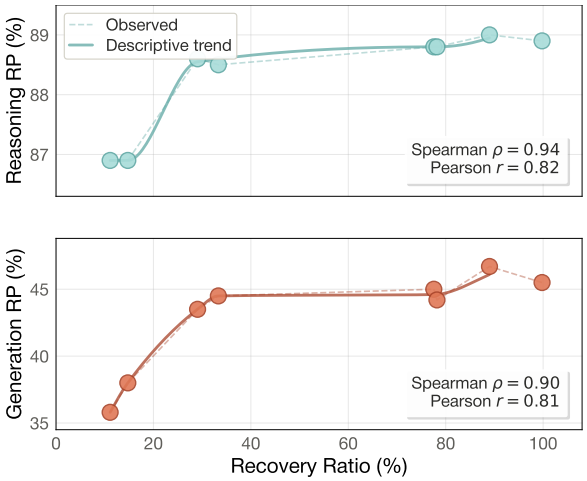


Figure 4: Association between recovery ratio (RR) and retained performance (RP) after pruning. The plots show a positive association between RR and RP on reasoning and generation benchmarks. RR is used as a diagnostic indicator of training-time recovery capacity.

duct a LoRA rank-scaling study, keeping the protocol fixed and preserving the standard scaling $\alpha = 2r$ at every rank. As shown in Tab. 5, even under an iso-parameter budget, high-rank LoRA does not close the performance gap with OverRep and requires up to 2.9× more GPU-hours.

Full capacity controls. We evaluate full fine-tuning (FT) of the retained blocks and full-model logit-level distillation from the dense teacher. We also combine the same KD objective with OverRep, denoted OverRep-KD, which replaces the layer-wise MSE objective with logit-level distillation while keeping the OverRep parameterization unchanged. As shown in Tab. 6, full fine-tuning still trails OverRep in reasoning at both pruning ratios despite its higher cost, indicating that the recovery parameterization provides benefits beyond exposing more trainable parameters. OverRep-KD also outperforms the same logit-KD objective without OverRep while using less than one-quarter of the GPU-hours, demonstrating that OverRep and logit-level distillation are complementary.

Capacity-knowledge asymmetry in practice. We further examine how trainable recovery capacity relates to post-pruning performance using the recovery ratio (RR)[3]. A small RR indicates limited trainable capacity to compensate for the transformations removed by pruning.

As shown in Fig. 4, larger RR values are generally associated with higher retained performance on reasoning and generation benchmarks, with positive rank-based and linear correlations. This trend is consistent with the capacity-knowledge asymmetry hypothesis, although it should be interpreted as descriptive evidence, as other design factors may also affect performance. In OverRep, RR increases only during recovery through overcomplete parameterization, and we merge the additional parameters before deployment. Thus, higher training-time recovery capacity adds no inference-time parameters, memory, or latency overhead.

5 Conclusion

We presented Overcomplete Reparameterization (OverRep), a post-pruning recovery framework for structured LLM pruning. Motivated by capacity-knowledge asymmetry, we revisit the recovery stage as a capacity-limited reconstruction problem, where a compact trainable module must recover the transformations pruned away. OverRep addresses this bottleneck under the principle of *"train overcomplete, deploy compact"*: temporarily increasing training-time recovery capacity through an overcomplete recovery module, while preserving the compact inference-time architecture via algebraic re-parameterization. Its auxiliary components enlarge the optimization space, and the annealed activation enables nonlinear training dynamics before converging to a linear regime that permits merging.

Across multiple backbones, OverRep improves performance without increasing inference-time overhead. Our analysis shows that increased recovery capacity does not necessarily require a proportional increase in training cost. These results suggest training-time overcomplete parameterization as an effective and deployment-friendly strategy for structured LLM pruning.

[3] $\mathrm{RR} = \frac{\text{Trainable parameters}}{\text{Pruned parameters}} \times 100$

Limitations

OverRep focuses on improving the recovery stage of structured pruning, with main results based on layer pruning. This setting provides a controlled testbed for studying recovery capacity. However, the channel-wise pruning and hybrid-architecture results in Appendix C remain preliminary because they use smaller recovery budgets than the main protocol and cover only one or two backbones each. Attention-head and mixed structured pruning remain unexplored. A comprehensive evaluation across pruning granularity is left for future work.

OverRep also increases the number of trainable parameters during recovery. Although our efficiency analysis shows that this expansion does not proportionally increase the practical training cost and can be further accelerated through feature caching, the cached variant assumes a frozen, reusable prefix before the recovery module. Thus, its benefit may vary with pruning patterns, hardware, and training implementations. Future work could extend overcomplete recovery to more diverse pruning structures and hardware settings.

Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF), Electronics and Telecommunications Research Institute (ETRI), and Institute of Information & Communications Technology Planning & Evaluation (IITP), funded by the Ministry of Education (RS-2025-25423987), the Korean government [26CS1100, Development of Proprietary Physical AI-based Small-scale Computers and Integrated Soft Suits], and the Korean government (MSIT) (RS-2026-25518808, RS-2026-25617480, and IITP-2026-RS-2023-00255968).

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10865–10873. AAAI Press.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. 2018. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439. AAAI Press.

Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, Andrew Tao, Anna Shors, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Bobby Chen, Boris Ginsburg, Boxin Wang, and 180 others. 2025. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *Preprint*, arXiv:2504.03624.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models. In *International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, and Reiichiro Nakano. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. 2019. ACNet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1911–1920.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. 2021a. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. 2021b. RepVGG: Making VGG-style ConvNets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13728–13737.

Zijian Feng, Hanzhang Zhou, Zixiao Zhu, Tianjiao Li, Jia Jim Deryl Chua, Lee Onn Mak, Gee Wah Ng, and Kezhi Mao. 2025. Restoring pruned large language models via lost component compensation. In *Advances in Neural Information Processing Systems*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*.

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

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, Vienna, Austria. Association for Computational Linguistics.

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

Seungmin Oh and Jongbin Ryu. 2024. Neural substitution for branch-level network re-parameterization. In *Proceedings of the Asian Conference on Computer Vision*, pages 104–120. Springer.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. 2023. FastViT: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. 2025a. Layer as puzzle pieces: Compressing large language models through layer concatenation. In *Advances in Neural Information Processing Systems*.

Yuxin Wang, MingHua Ma, Zekun Wang, Jingchang Chen, Shan Liping, Qing Yang, Dongliang Xu, Ming Liu, and Bing Qin. 2025b. CFSP: An efficient structured pruning framework for LLMs with coarse-to-fine activation information. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9311–9328, Abu Dhabi, UAE. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. LaCo: Large language model pruning via layer collapse. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6401–6417.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*.

Yang Zhang, Yawei Li, Xinpeng Wang, Qianli Shen, Barbara Plank, Bernd Bischl, Mina Rezaei, and Kenji Kawaguchi. 2024. Finercut: Finer-grained interpretable layer pruning for large language models. In *Machine Learning and Compression Workshop at NeurIPS 2024*.

Changhai Zhou, Shijie Han, Lining Yang, Yuhua Zhou, Xu Cheng, Yibin Wang, and Hongguang Li. 2025. Rankadaptor: Hierarchical rank allocation for efficient fine-tuning pruned LLMs via performance model. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5796–5810, Albuquerque, New Mexico. Association for Computational Linguistics.

Appendix

Unless otherwise noted, 'Rea.' and 'Gen.' in the appendix denote the average raw score over the 10 reasoning and 3 generation tasks, not RP.

A Practical Implementation

For notational simplicity, the method sections describe OverRep using a single recovery block. In our implementation, we instantiate $\mathcal{F}_{\text{ORM}}$ with two consecutive transformer blocks following the pruned segment. This provides additional recovery flexibility while keeping the active training path short. Let $\mathcal{F}_{\text{ORM},1}$ and $\mathcal{F}_{\text{ORM},2}$ denote the first and second OverRep recovery blocks, respectively. The practical recovery objective is written as

$$\tilde{\mathbf{h}} = \mathcal{F}_{\text{ORM},1}(\mathbf{h}_p; \theta_1, \phi_1), \tag{12}$$

$$\mathcal{L}_{\text{ours}} = \mathbb{E}_{\mathbf{z}} \left[\left\| \mathcal{F}_{\text{ORM},2}(\tilde{\mathbf{h}}; \bar{\theta}_2, \phi_2) - \mathbf{h}_{p+n+1} \right\|_F^2 \right], \tag{13}$$

where θ_1 denotes the pretrained parameters of the first recovery block, $\bar{\theta}_2$ denotes the frozen pretrained parameters of the second recovery block, and ϕ_1, ϕ_2 are the corresponding OverRep auxiliary parameters. During recovery, we jointly optimize θ_1 and ϕ_1 in the first block, keep $\bar{\theta}_2$ frozen, and optimize only ϕ_2 in the second block. After training, each overcomplete projection in both blocks is independently merged using the same projection-wise re-parameterization rule in Eq. (10). Therefore, the deployed model contains only standard compact transformer blocks and incurs no additional inference-time overhead.

For the annealed activation schedule, we set the warm-up ratio ρ_{w} to 0.01 and the final linear-stabilization ratio ρ_{l} to 0.2.

B Implementation Details

A major challenge in comparing structured pruning methods is that prior work often differs in recovery datasets, data scale, and evaluation protocols. To ensure a controlled comparison, we standardize the recovery data and evaluation framework across all methods whenever possible, as described in Appendix B.1. For evaluation, we use `lm-eval-harness` (Gao et al., 2024).

For reasoning benchmarks, we evaluate on ARC-Challenge and ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MathQA (Amini et al., 2019), MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), RACE (Lai et al., 2017), and Winogrande (Sakaguchi et al., 2021). For generation benchmarks, we evaluate on CoQA (Reddy et al., 2019), GSM8K (Cobbe et al., 2021), and TriviaQA (Joshi et al., 2017). CoQA is measured using F1 score in the zero-shot setting, while GSM8K and TriviaQA are measured by exact match in the 8-shot and 5-shot settings, respectively, following Gao et al. (2024).

Backbone	HF repo.	dim.	Inter dim.	# attn heads	# kv heads	# L	25% # L	50% # L
LLaMA2-7B	meta-llama/Llama-2-7b-hf	4096	11008	32	32	32	24	16
LLaMA2-13B	meta-llama/Llama-2-13b-hf	5120	13824	40	40	40	30	20
LLaMA3-3B	meta-llama/Llama-3.2-3B	3072	8192	24	8	28	19	12
LLaMA3-8B	meta-llama/Llama-3.1-8B	4096	14336	32	8	32	23	14
Qwen3-4B	Qwen/Qwen3-4B-Base	2560	9728	32	8	36	25	15
Qwen3-8B	Qwen/Qwen3-8B-Base	4096	12288	32	8	36	26	16
Qwen3-14B	Qwen/Qwen3-14B-Base	5120	17408	40	8	40	30	–
Qwen3-30B-A3B†	Qwen/Qwen3-30B-A3B-Base	2048	768	32	4	48	36	24
Nemotron-H-4B‡	nvidia/Nemotron-H-4B-Base-8K	3072	12288	32	8	52	39	26

Table 7: Backbone configurations. We report the Hugging Face repository (HF Repo.), detailed backbone configuration, and the number of layers (denoted # L) remaining at 25% and 50% pruning ratios. †MoE backbone with 128 experts and top-8 routing. 'Inter dim.' is the per-expert intermediate dimension. ‡Hybrid backbone whose 52 layers comprise Mamba-2, attention, and MLP sublayers, and 'Inter dim.' refers to the MLP layers.

	Method	Epoch	#B	LR	Sche.	(r, α)
LLaMA2-7B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	1e-4	linear	(16, 32)
	ShortGPT	10	32	1e-4	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–
LLaMA2-13B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	6e-5	linear	(16, 32)
	ShortGPT	10	32	6e-5	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–
LLaMA3-3B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	1e-4	linear	(16, 32)
	ShortGPT	10	32	1e-4	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–
LLaMA3-8B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	1e-4	linear	(16, 32)
	ShortGPT	10	32	1e-4	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–
Qwen3-4B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	1e-4	linear	(16, 32)
	ShortGPT	10	32	1e-4	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–
Qwen3-8B	Recoveries	10	32	1e-4	cosine	(16, 32)
	LaCo	10	32	1e-4	linear	(16, 32)
	ShortGPT	10	32	1e-4	linear	(16, 32)
	Streamline	100	8	1e-4	cosine	–
	OverRep	20	8	1e-4	cosine	–

Table 8: Recovery training configurations. 'Recoveries' denotes recovery methods such as LoRA, AdaLoRA, RankAdaptor, and RestoreLCC. '#B' denotes the total batch size, and 'LR' denotes the learning rate. r and α indicate the LoRA rank and LoRA scaling, respectively.

B.1 Recovery Dataset

We use the `FineWeb-Edu` dataset (Lozhkov et al., 2024) provided by HuggingFaceFW. Specifically, we use the `sample-10BT` split and draw 120,000 training samples and 4,000 test samples with a fixed random seed of 42. We tokenize all samples with a sequence length of 1024 for memory efficiency. For pruning-criterion calibration, we randomly select 50 samples from the same training split. We use this dataset configuration consistently across all experiments, regardless of backbone or method, to ensure a fair comparison.

B.2 Training Configurations

We follow each baseline method's official training configuration as closely as possible. To ensure a fair comparison while obtaining strong performance, we use the official settings as the starting point and conduct a limited hyperparameter search around them for each method and backbone. Unless otherwise specified, all recovery methods are trained on the same recovery dataset and evaluated using the same benchmark pipeline.

For LaCo, the original recovery stage uses full fine-tuning, which is not feasible on an RTX 3090 GPU with 24GB memory in our setting. We therefore use LoRA as the recovery module for LaCo. ShortGPT performs pruning based on block influence and uses LoRA for recovery. For Streamline, we find that the official training schedule uses too few epochs to achieve stable recovery performance in our setting. Thus, we increase the number of epochs so that its training time is comparable to that of the other methods.

Tables 7 and 8 provide the hyperparameters and architectural configurations used for recovery. We apply the AdamW optimizer with $(\beta_1 = 0.9, \beta_2 = 0.95)$ in all experiments. In Tab. 8, LoRA, AdaLoRA, RankAdaptor, and RestoreLCC share the same hyperparameters and are collectively referred to as recovery methods.

	PR	BoolQ	WSC	COQA	HellaSwag	PIQA	Race-M	Race-H	MMLU	Avg.
Dense (From Streamline)	–	70.8	37.5	66.7	71.3	78.1	33.1	35.5	46.8	55.0
Dense (Ours Rep.)	–	70.4	37.5	67.4	70.9	77.2	33.1	35.5	46.7	54.8
Streamline (From Streamline)	25%	67.5	36.5	59.2	61.1	71.5	34.8	37.0	45.5	51.6
Streamline (Ours Rep.)	25%	66.0	36.5	61.9	64.4	71.9	33.4	30.3	45.8	51.3

Table 9: Reproduction verification for Streamline on LLaMA2-7B. '(From Streamline)' denotes the results reported in the original paper, while '(Ours Rep.)' denotes our reproduced models evaluated with the original Streamline evaluation framework. The close agreement supports the fidelity of our baseline reproduction.

		Reasoning Task											Generation Task			
	Method	ARC_C	ARC_E	BoolQ	Hella.	MathQA	MMLU	OBQA	PIQA	RACE	Wino.	Avg.	CoQA	GSM	Triv.QA	Avg.
25%	LoRA	0.143	1.434	1.762	0.497	1.383	1.616	1.490	1.120	1.314	1.797	0.250	2.501	1.413	0.287	1.257
	Streamline	0.896	0.657	0.517	0.248	0.379	0.517	0.014	0.248	0.143	1.518	0.162	5.858	1.004	0.014	2.287
	OverRep	0.745	0.430	0.657	0.248	0.625	0.287	0.861	0.896	1.762	0.896	0.190	1.004	0.717	0.379	0.564
50%	LoRA	1.034	0.896	2.656	0.430	0.799	0.283	2.068	0.994	1.616	1.597	0.396	2.879	0.896	0.759	1.065
	Streamline	0.517	0.287	0.574	0.379	0.287	0.378	1.034	0.574	0.379	0.379	0.102	0.287	0.517	0.248	0.126
	OverRep	1.875	0.574	0.574	0.745	0.574	0.271	2.068	0.379	1.897	1.511	0.165	1.654	0.379	0.248	0.461

Table 10: 95% confidence interval for recovery of pruned LLaMA3-3B on reasoning and generation benchmarks.

B.3 Reproduction Verification

Prior pruning studies often use different training and evaluation protocols, making direct comparison difficult. We standardize these factors in our main experiments and further verify the fidelity of our baseline reproduction by evaluating our reproduced Streamline with the original Streamline evaluation framework. In Tab. 9, the reproduced results closely match the Streamline values reported in the original paper. This supports the reliability of our reproduced baselines.

B.4 Confidence Intervals

To improve the reliability of our experiments, Tab. 10 reports the 95% confidence-interval half-widths computed over three random seeds (0, 26, 42) for pruned LLaMA3-3B on both reasoning and generation benchmarks. Each interval is computed as $t_{0.975,n-1} \cdot s/\sqrt{n}$, where $n = 3$ and s is the sample standard deviation across seeds. Thus, an entry c corresponds to a confidence interval of $\pm c$.

B.5 Recovery Training Fairness

Wall-clock fairness. OverRep introduces substantially more trainable parameters than conventional recovery methods, increasing recovery capacity while raising potential concerns about optimization cost. To address this, we use method-specific recovery epochs to keep wall-clock training time as comparable as possible across methods, rather than equalizing training tokens.

Method	Trainable Params. (M)	Epochs	Time (h)
LoRA-like	16.5	10	8.9
AdaLoRA	33.0	10	10.6
RankAdaptor	12.2	10	8.7
RestoreLCC	0.1	10	6.8
Streamline*	100.7	100	4.6
OverRep*	650.1	20	4.6

Table 11: Comparison of trainable parameters, recovery epochs, and wall-clock training time for LLaMA3-3B after 25% pruning. Wall-clock time is reported in hours.

Table 11 reports trainable parameters, recovery epochs, and wall-clock time for recovering LLaMA3-3B after 25% pruning on a single RTX 3090 GPU with 24GB memory. Here, 'LoRA-like' denotes LoRA-based methods, including LoRA, LaCo, and ShortGPT. Despite having the most trainable parameters, OverRep achieves wall-clock recovery time comparable to or lower than existing baselines. Streamline* and OverRep* use cached features, whose precomputation takes less than one hour; even including this overhead, their total recovery time remains comparable. These results show that the gains of OverRep are not due to a substantially larger wall-clock training budget.

Training budget fairness. Beyond the wall-clock comparison, we also evaluate all methods under the same 10-epoch recovery schedule to control for the number of optimization epochs and recovery samples seen during training. As shown in Tab. 12, OverRep still outperforms the baselines

	Method	Reasoning Task											Generation Task			
		ARC_C	ARC_E	BoolQ	Hella.	MathQA	MMLU	OBQA	PIQA	RACE	Wino.	Avg.	CoQA	GSM	Triv.QA	Avg.
	Dense	46.2	71.8	73.0	73.6	34.4	54.1	43.0	77.3	39.8	69.1	58.2	75.9	25.2	56.3	52.5
25%	LoRA	35.7	54.8	68.2	58.3	26.6	46.3	34.2	69.2	36.2	63.5	49.3	38.4	3.3	13.4	18.4
25%	AdaLoRA	32.3	54.1	64.6	53.9	24.1	47.6	33.2	66.2	34.3	65.8	47.6	42.6	3.0	13.2	19.6
25%	RankAdaptor	35.9	56.5	66.9	58.1	26.2	45.7	32.8	68.4	35.3	63.9	49.0	41.1	3.6	13.0	19.2
25%	RestoreLCC	32.3	49.2	70.4	50.5	24.5	46.7	31.4	64.9	33.7	65.3	46.9	48.0	2.0	8.6	19.5
25%	Streamline	36.6	59.6	71.2	55.5	25.2	54.8	32.6	68.1	37.0	64.9	50.6	34.7	1.6	9.1	15.1
25%	OverRep	37.7	65.4	64.3	58.5	25.0	54.4	35.6	69.5	35.7	67.4	51.4	54.3	2.6	10.7	22.5
50%	LoRA	26.1	41.0	49.3	36.9	23.0	23.1	29.2	60.9	29.7	52.1	37.1	9.4	1.7	3.7	4.9
50%	AdaLoRA	24.1	34.3	56.4	33.1	22.1	22.9	27.6	57.4	25.4	51.0	35.4	3.8	1.7	1.3	2.3
50%	RankAdaptor	25.2	42.8	51.6	36.5	23.4	23.2	29.4	60.8	27.6	51.8	37.2	9.6	1.3	3.1	4.7
50%	RestoreLCC	23.3	30.1	38.2	29.7	22.7	23.0	27.0	54.7	23.9	49.3	32.2	2.0	1.1	0.4	1.2
50%	Streamline	25.3	48.3	59.2	37.0	22.6	23.1	28.0	62.5	28.3	53.5	38.8	5.0	0.8	1.5	2.4
50%	OverRep	29.4	54.7	61.8	41.8	22.5	23.2	32.0	63.5	28.5	57.1	41.5	12.5	1.6	2.7	5.6

Table 12: Experimental results on reasoning and generation benchmarks for LLaMA3-3B under 10-epoch settings.

Backbone	OverRep Time (h) train + cache	LoRA Time (h) (GPUs)	Speedup w/o / w/ cache	Rea. LoRA	Rea. OverRep	Gen. LoRA	Gen. OverRep
L3-3B	4.6 + 0.8	8.9 (1 GPU)	1.9× / 1.6×	49.3	52.5	18.4	24.7
Q3-14B	12.6 + 2.1	90.2 (11.27h × 8 GPUs)	7.2× / 6.1×	51.8	52.5	30.2	40.0
Q3-30B	8.8 + 3.8	426 (53.3h × 8 GPUs)	48× / 34×	49.8	53.1	23.4	30.1

Table 13: End-to-end cost relative to LoRA at a 25% pruning ratio.

when epoch count is fixed across methods. This indicates that the advantage of OverRep is not simply due to longer training or exposure to more recovery data. Together with the wall-clock analysis, these results show that OverRep remains effective under multiple fair training-budget comparisons.

B.6 Training-cost Quantification

We quantify OverRep's cost from two perspectives: end-to-end cost relative to LoRA and intrinsic overhead over an otherwise identical Plain recovery on LLaMA3-3B (L3-3B), Qwen3-14B (Q3-14B), and Qwen3-30B-A3B (Q3-30B). 'Time' denotes wall-clock time multiplied by the number of GPUs.

End-to-end cost relative to LoRA. OverRep trains only localized recovery blocks, and reuses cached frozen-prefix activations, avoiding repeated full-model forward and backward passes. Although OverRep is trained for 20 epochs versus 10 for LoRA, it requires fewer GPU-hours because optimization is limited to the recovery blocks. We report cache precomputation separately and include it in the end-to-end comparison. As shown in Tab. 13, LoRA requires eight RTX 3090 GPUs on the two larger backbones, while OverRep recovery runs on one GPU. Including cache precomputation, OverRep is 6.1× cheaper on Qwen3-14B and 34× cheaper on Qwen3-30B-A3B.

Intrinsic overhead of the overcomplete parameterization. We isolate OverRep's additional cost by comparing it with Plain recovery. Plain recovery directly fine-tunes the same localized recovery blocks without the auxiliary $\mathbf{W}_i$ and $\mathbf{D}_i$ components or annealed activation, trained for the same 20 epochs. As shown in Tab. 14, overcomplete parameterization increases recovery time by 2.4–6.6× over Plain recovery, while even the largest run requires only 12.6 single-GPU hours. Qwen3-30B-A3B shows a smaller increase in trainable parameters because each expert has a narrower intermediate dimension than dense models.

C Generality

C.1 OverRep in Channel-wise Pruning

We extend OverRep to channel-wise pruning using the same recovery components. Channel-wise pruning reduces the intermediate width of a block, so we adapt the shapes of $\mathbf{W}_i$ and $\mathbf{D}_i$ while retaining the training formulation in Eq. (6) and the merge rule in Eq. (10). Let d_{mid} denote the intermediate width of a self-attention or feed-forward block, and let $d_{\text{ref}} < d_{\text{mid}}$ denote the reduced width after pruning. Because the block input and output dimensions remain unchanged, all other network components

Backbone	Recovery	Params. (M)	Time (h)	Rea.	Gen.
L3-3B	Plain	100.7	1.3	50.6	18.8
L3-3B	OverRep	650.1	4.6	52.5	24.7
Q3-14B	Plain	330	1.9	48.5	13.9
Q3-14B	OverRep	2596	12.6	52.5	40.0
Q3-30B	Plain	1246	3.6	49.3	16.3
Q3-30B	OverRep	1752	8.8	53.1	30.1

Table 14: Intrinsic overhead of the overcomplete parameterization at a 25% pruning ratio.

are unaffected.

Expanding projections. For projections into the intermediate space, $i \in \{q, k, v, gate, up\}$, the pretrained weight $\mathbf{P}_i \in \mathbb{R}^{d_{\text{mid}} \times d_i^{\text{in}}}$ and additive branch $\mathbf{W}_i \in \mathbb{R}^{d_{\text{mid}} \times d_i^{\text{in}}}$ retain their original shapes, while the multiplicative factor becomes rectangular, $\mathbf{D}_i \in \mathbb{R}^{d_{\text{ref}} \times d_{\text{mid}}}$. After training, the merge rule in Eq. (10) yields $\hat{\mathbf{P}}_i = \mathbf{D}_i(\mathbf{P}_i + \mathbf{W}_i) \in \mathbb{R}^{d_{\text{ref}} \times d_i^{\text{in}}}$. The merge itself reduces the dimension.

Contracting projections. Projections out of the intermediate space, $i \in \{o, down\}$, have pretrained weights $\mathbf{P}_i \in \mathbb{R}^{d_i^{\text{out}} \times d_{\text{mid}}}$ and must consume the narrowed hidden state $\mathbf{h}' \in \mathbb{R}^{d_{\text{ref}}}$. We mirror the preceding construction by placing the multiplicative factor on the input side, yielding the merged projection $\hat{\mathbf{P}}_i = (\mathbf{P}_i + \mathbf{W}_i)\mathbf{D}_i \in \mathbb{R}^{d_i^{\text{out}} \times d_{\text{ref}}}$, where $\mathbf{W}_i \in \mathbb{R}^{d_i^{\text{out}} \times d_{\text{mid}}}$ and $\mathbf{D}_i \in \mathbb{R}^{d_{\text{mid}} \times d_{\text{ref}}}$.

Initialization and training. As in the main setting, $\mathbf{W}_i = \mathbf{0}$. Because $\mathbf{D}_i$ is rectangular, it is initialized as a partial identity, $\mathbf{D}_i = [\,\mathbf{I}_{d_{\text{ref}}} \;\; \mathbf{0}\,] \in \mathbb{R}^{d_{\text{ref}} \times d_{\text{mid}}}$ for expanding projections and its transpose for contracting projections. Training therefore begins from a width-reduced copy of the pretrained block. We keep $\mathbf{P}_i$ frozen and train only $(\mathbf{W}_i, \mathbf{D}_i)$. The annealed activation $\mathcal{A}_s$ follows Eqs. (7) and (8) without modification, and the recovery objective applies the layer-wise reconstruction loss in Eq. (2) to every pruned block. At deployment, each block folds into standard reduced-width projections without additional parameters or computation.

Comparison with LLM-Pruner. We compare against LLM-Pruner (Ma et al., 2023) on LLaMA3-3B and LLaMA3-8B at approximately 25% parameter reduction, following its evaluation protocol of nine reasoning tasks, including WSC, without the generation suite. To match the LLM-Pruner setting, these experiments use a smaller recovery budget of 24K `FineWeb-Edu` samples for 5 epochs. Their absolute results are therefore not directly comparable to those under our main protocol, which uses 120K samples for 20 epochs. As shown in Tab. 15, OverRep improves the average by 5.6 points on LLaMA3-3B and 4.2 points on LLaMA3-8B.

Backbone	Method	PR	Rea.
LLaMA3-3B	Dense	-	56.6
	LLM-Pruner	25%	40.5
	OverRep	25%	46.1
LLaMA3-8B	Dense	-	61.7
	LLM-Pruner	24%	47.5
	OverRep	24%	51.7

Table 15: Experimental results of channel-wise pruning with LLM-Pruner and OverRep.

	Backbone	Method	Rea.	Gen.	Time (h)
	Q3-30B	Dense	67.9	78.5	-
25%	Q3-30B	LoRA	49.8	23.4	53.3h × 8 GPUs
	Q3-30B	Streamline	49.5	22.2	7.3h
	Q3-30B	OverRep	53.1	30.1	8.8h
50%	Q3-30B	LoRA	43.6	9.9	35.6h × 8 GPUs
	Q3-30B	Streamline	42.9	7.2	6.0h
	Q3-30B	OverRep	44.9	14.7	9.0h
	Q3-14B	Dense	69.7	76.9	-
25%	Q3-14B	LoRA	51.8	30.2	11.3h × 8 GPUs
	Q3-14B	Streamline	51.9	32.7	11.7h
	Q3-14B	OverRep	52.5	40.0	12.6h

Table 16: Experimental results across architectures. 'Time' denotes wall-clock training time×GPUs; unless otherwise specified, results use a single GPU.

C.2 OverRep in Various Architectures

Larger and MoE architectures. A key property of OverRep is that recovery is layer-local: training memory is bounded by the two recovery blocks plus cached features, not by total model size. We scale OverRep to Qwen3-30B-A3B-Base, a recent large-scale 30.5B total-parameter MoE model, and Qwen3-14B. Each expert's FFN projections and the attention projections are expanded with their own $(\mathbf{W}_i, \mathbf{D}_i)$ and merged exactly by Eq. (10). The router is left untouched, so expert routing behavior and sparsity are preserved at deployment. As shown in Tab. 16, OverRep outperforms LoRA while using 48× fewer aggregate GPU-hours on Qwen3-30B-A3B and 7.2× fewer on Qwen3-14B.

Hybrid architecture. OverRep acts on linear projections and therefore also extends to the dominant parameterized components of Mamba-style hybrid blocks. We validate this on Nemotron-H-4B (Blakeman et al., 2025) by applying OverRep to the at-

	Method	Rea.	Gen.	Time (h)
	Dense	60.5	45.8	-
25%	LoRA	49.1	28.8	9.0
	OverRep	49.9	24.5	3.3
	OverRep-KD	50.1	31.5	5.2
50%	LoRA	39.7	11.1	6.0
	OverRep	42.9	11.2	3.3
	OverRep-KD	42.7	15.7	5.2

Table 17: Experimental results on Nemotron-H-4B.

	Backbone	Criterion	Rea.		Gen.	
			Own	OverRep	Own	OverRep
25%	L3-3B	ShortGPT	48.0	50.6	13.9	25.6
	L3-3B	Streamline	50.2	52.5	24.0	24.7
	Q3-4B	ShortGPT	46.7	45.8	15.5	22.5
	Q3-4B	Streamline	47.7	48.9	20.8	29.1
50%	L3-3B	ShortGPT	35.8	40.5	2.1	5.5
	L3-3B	Streamline	39.8	40.7	3.4	7.2
	Q3-4B	ShortGPT	36.4	35.9	1.2	0.8
	Q3-4B	Streamline	39.3	40.9	5.0	7.4

Table 18: Compatibility experiment with existing pruning criteria. 'Own' is the criterion's own recovery, and 'OverRep' is OverRep recovery on the same mask.

tention and MLP sublayers of the recovery blocks and keeping the Mamba-2 (Dao and Gu, 2024) mixer frozen. As shown in Tab. 17, at 25% pruning, OverRep modestly improves reasoning at less than half the cost, although LoRA remains stronger on generation; OverRep-KD improves both metrics. At 50%, OverRep improves reasoning by 3.2 points, and OverRep-KD improves generation by 4.6 points.

C.3 Compatibility with Existing Pruning Criteria

We apply OverRep to both contiguous Streamline masks and non-contiguous ShortGPT block influence masks. As shown in Tab. 18, across all eight settings, OverRep improves reasoning in six and generation in seven; the remaining differences are below one point. These gains require no criterion-specific tuning, supporting OverRep as a broadly compatible recovery framework.

D Detailed Experimental Results

D.1 Reasoning Performance

Tables 19 and 20 provide detailed results for reasoning tasks.

D.2 Generation Performance

Tables 21 and 22 provide generation results.

	PR	Method	ARC_C	ARC_E	BoolQ	Hella.	MathQA	MMLU	OBQA	PIQA	RACE	Wino.	Avg.	RP
LLaMA2-7B	-	Dense	46.2	76.2	77.9	76.0	28.5	40.8	44.2	79.1	39.5	69.5	57.8	100.0
	25%	LoRA	37.6	65.4	77.6	66.2	25.3	24.6	37.8	72.5	39.0	66.6	51.3	88.7
		AdaLoRA	37.5	65.1	77.8	66.1	25.0	24.8	38.0	72.1	39.1	66.1	51.2	88.5
		RankAdaptor	39.2	64.4	68.9	66.9	23.4	28.1	37.6	69.6	35.9	62.8	49.7	86.0
		RestoreLCC	35.3	61.1	76.3	63.8	25.3	27.3	36.6	71.5	37.4	66.2	50.1	86.7
		LaCo	31.7	63.1	62.5	59.6	24.3	24.7	38.4	73.3	35.1	54.0	46.7	80.8
		ShortGPT	38.1	65.3	74.6	66.9	25.2	26.1	38.0	72.7	37.1	65.8	51.0	88.2
		Streamline	39.7	68.9	71.6	66.9	24.5	27.7	38.8	73.1	38.9	66.4	51.7	89.4
		OverRep	41.0	70.5	73.6	67.7	25.1	38.5	40.2	74.3	37.9	66.1	53.5	92.6
	50%	LoRA	27.8	44.3	62.4	45.1	22.0	28.7	30.4	61.0	32.6	59.1	41.3	71.5
		AdaLoRA	26.4	41.2	62.5	43.7	22.4	26.1	28.2	60.2	31.8	59.6	40.2	69.6
		RankAdaptor	27.7	44.8	62.4	45.2	22.2	28.6	30.0	61.0	32.4	58.6	41.3	71.4
		RestoreLCC	25.3	38.0	62.6	40.4	23.1	23.8	28.2	59.4	31.1	59.6	39.2	67.7
		LaCo	26.9	50.5	56.1	47.3	22.8	23.1	32.0	67.2	30.1	53.5	41.0	70.9
		ShortGPT	27.3	44.0	62.3	46.3	22.5	23.4	29.8	63.0	33.9	55.9	40.8	70.7
		Streamline	27.2	58.9	61.7	43.6	22.5	23.1	34.4	68.6	30.9	50.8	42.2	73.0
		OverRep	31.5	60.0	62.3	51.1	21.8	24.4	33.8	66.9	32.7	59.9	44.4	76.9
LLaMA2-13B	-	Dense	49.1	77.5	80.6	79.4	31.8	50.5	45.2	80.5	40.5	72.4	60.8	100.0
	25%	LoRA	46.8	73.1	78.7	73.1	25.7	49.4	43.6	75.4	39.3	69.4	57.5	94.6
		AdaLoRA	43.9	69.4	81.7	70.8	26.1	49.7	42.0	74.5	38.6	70.3	56.7	93.3
		RankAdaptor	45.9	72.3	78.6	73.0	26.0	49.5	43.8	75.3	39.3	71.0	57.5	94.6
		RestoreLCC	38.8	65.3	77.7	67.4	25.5	44.8	40.0	72.3	36.9	69.5	53.8	88.6
		LaCo	41.2	69.8	69.7	69.5	26.5	43.0	41.8	77.8	37.9	63.0	54.0	88.9
		ShortGPT	46.0	71.5	70.7	74.0	26.7	47.2	43.0	74.6	38.0	68.2	56.0	92.1
		Streamline	46.8	75.1	70.3	73.3	27.1	51.6	42.0	75.7	40.3	71.0	57.3	94.4
		OverRep	46.0	76.1	76.7	74.2	27.5	47.4	43.8	76.8	39.3	71.3	57.9	95.3
	50%	LoRA	32.2	54.8	64.3	56.6	22.9	44.7	35.4	65.3	34.7	65.0	47.6	78.3
		AdaLoRA	30.0	48.8	62.4	51.3	22.4	33.9	31.4	64.7	33.8	63.8	44.3	72.8
		RankAdaptor	32.7	55.0	64.0	56.6	22.7	44.7	35.6	65.6	35.4	65.7	47.8	78.7
		RestoreLCC	29.4	43.2	62.2	45.4	22.6	29.7	31.8	59.2	33.0	63.4	42.0	69.1
		LaCo	34.3	52.1	62.7	57.4	23.7	29.8	35.6	70.1	28.5	62.4	45.7	75.2
		ShortGPT	33.4	53.5	64.3	56.8	22.7	37.9	36.2	66.3	35.0	65.0	47.1	77.5
		Streamline	34.5	61.1	64.6	57.0	23.4	50.3	36.8	67.8	37.3	65.7	49.9	82.1
		OverRep	36.7	65.1	70.9	59.1	23.7	50.3	39.0	69.8	37.8	67.0	51.9	85.5
LLaMA3-3B	-	Dense	46.2	71.8	73.0	73.6	34.4	54.1	43.0	77.3	39.8	69.1	58.2	100.0
	25%	LoRA	35.7	54.8	68.2	58.3	26.6	46.3	34.2	69.2	36.2	63.5	49.3	84.7
		AdaLoRA	32.3	54.1	64.6	53.9	24.1	47.6	33.2	66.2	34.3	65.8	47.6	81.8
		RankAdaptor	35.9	56.5	66.9	58.1	26.2	45.7	32.8	68.4	35.3	63.9	49.0	84.1
		RestoreLCC	32.3	49.2	70.4	50.5	24.5	46.7	31.4	64.9	33.7	65.3	46.9	80.5
		LaCo	34.0	62.0	62.5	54.8	26.6	32.5	32.4	69.9	37.5	58.0	47.0	80.7
		ShortGPT	34.6	53.7	70.3	56.2	25.1	41.4	32.2	67.6	35.3	64.0	48.0	82.5
		Streamline	36.4	63.7	63.1	58.4	25.4	48.2	34.8	69.7	36.8	65.0	50.2	86.1
		OverRep	39.2	66.0	67.9	59.5	25.7	54.9	36.6	70.5	38.0	66.8	52.5	90.2
	50%	LoRA	26.1	41.0	49.3	36.9	23.0	23.1	29.2	60.9	29.7	52.1	37.1	63.8
		AdaLoRA	24.1	34.3	56.4	33.1	22.1	22.9	27.6	57.4	25.4	51.0	35.4	60.8
		RankAdaptor	25.2	42.8	51.6	36.5	23.4	23.2	29.4	60.8	27.6	51.8	37.2	63.9
		RestoreLCC	23.3	30.1	38.2	29.7	22.7	23.0	27.0	54.7	23.9	49.3	32.2	55.3
		LaCo	23.9	42.3	59.4	33.6	21.8	23.0	27.6	60.6	26.6	50.5	36.9	63.4
		ShortGPT	22.6	40.7	48.5	34.8	22.1	23.9	27.8	58.9	26.9	51.8	35.8	61.5
		Streamline	25.3	53.5	57.0	36.3	23.7	23.0	32.8	65.3	27.7	53.7	39.8	68.4
		OverRep	29.1	55.4	61.4	42.6	23.7	24.8	32.6	64.5	30.4	56.5	42.1	72.3
LLaMA3-8B	-	Dense	53.2	81.2	82.0	79.0	39.7	63.1	44.8	81.1	38.9	74.2	63.7	100.0
	25%	LoRA	41.3	61.7	71.7	68.5	28.1	54.0	37.6	71.5	36.8	63.3	53.5	83.9
		AdaLoRA	41.1	64.9	67.6	63.7	29.7	56.5	36.8	71.0	37.9	69.5	53.9	84.5
		RankAdaptor	43.3	67.3	75.6	68.6	29.7	55.5	38.6	72.5	37.8	66.2	55.5	87.1
		RestoreLCC	38.1	60.0	62.4	57.6	28.1	36.0	33.4	69.2	36.2	68.3	48.9	76.8
		LaCo	40.4	61.5	66.5	67.4	28.5	41.0	39.2	75.6	36.2	64.0	52.0	81.7
		ShortGPT	41.7	67.6	76.2	68.4	30.2	57.7	38.4	72.9	39.1	69.8	56.2	88.2
		Streamline	45.5	73.6	64.8	69.0	29.5	61.5	39.4	74.0	38.1	70.8	56.6	88.9
		OverRep	47.4	74.8	75.4	69.3	29.2	60.7	40.8	74.8	38.5	70.9	58.2	91.3
	50%	LoRA	27.3	42.3	60.7	42.4	21.9	23.0	32.4	62.4	31.2	57.7	40.1	63.0
		AdaLoRA	25.0	37.8	62.0	38.3	21.7	22.9	28.4	59.7	27.9	56.1	38.0	59.6
		RankAdaptor	26.8	41.5	59.9	42.6	21.9	23.1	32.4	62.3	31.0	56.6	39.8	62.5
		RestoreLCC	24.5	33.4	58.1	31.8	21.4	22.9	26.4	53.7	23.0	51.5	34.7	54.4
		LaCo	25.4	46.2	61.1	41.7	22.9	23.0	28.2	61.5	30.0	56.2	39.6	62.2
		ShortGPT	26.8	48.0	62.0	44.6	22.8	24.8	32.0	63.1	32.1	58.1	41.4	65.0
		Streamline	26.7	55.9	62.1	40.1	23.0	23.0	34.0	67.2	29.8	51.8	41.4	64.9
		OverRep	33.8	58.5	62.0	48.4	22.4	23.3	34.8	67.2	32.2	60.5	44.3	69.5

Table 19: Experimental results of reasoning tasks on LLaMA2 and LLaMA3 reported as accuracy and RP.

	PR	Method	ARC_C	ARC_E	BoolQ	Hella.	MathQA	MMLU	OBQA	PIQA	RACE	Wino.	Avg.	RP
Qwen3-4B	-	Dense	51.7	78.8	83.2	73.6	54.0	71.2	41.2	78.2	40.7	70.5	64.3	100.0
	25%	LoRA	37.5	59.5	63.9	53.9	26.1	36.9	33.0	66.1	36.0	63.6	47.7	74.1
		AdaLoRA	32.3	53.8	71.7	49.8	26.5	32.0	29.8	63.9	34.3	63.7	45.8	71.2
		RankAdaptor	36.6	57.9	65.0	53.9	25.7	36.3	33.2	65.4	34.4	63.0	47.1	73.3
		RestoreLCC	31.2	40.8	70.1	40.5	23.5	23.2	31.6	59.4	26.8	60.4	40.8	63.4
		LaCo	35.3	63.0	68.2	51.3	26.2	31.7	34.0	68.8	34.5	57.3	47.0	73.1
		ShortGPT	36.8	66.8	62.2	53.9	27.0	25.2	34.6	71.1	33.1	56.1	46.7	72.6
		Streamline	35.5	64.1	63.0	57.2	25.2	23.0	37.0	71.7	35.5	65.0	47.7	74.2
		OverRep	39.1	67.3	64.3	57.1	27.5	24.3	35.6	70.5	35.7	62.9	48.4	75.3
	50%	LoRA	26.3	44.0	54.3	33.1	21.6	23.0	27.2	60.0	26.5	51.6	36.8	57.2
		AdaLoRA	24.6	37.8	45.5	31.1	20.6	22.9	28.4	57.5	24.9	51.8	34.5	53.7
		RankAdaptor	26.9	45.2	55.7	33.1	21.7	23.0	27.8	58.8	25.9	50.8	36.9	57.4
		RestoreLCC	27.0	28.3	60.4	27.6	19.6	22.9	28.0	52.2	21.5	49.3	33.7	52.4
		LaCo	24.1	41.4	60.4	30.4	21.6	23.0	27.6	57.5	26.2	52.3	36.5	56.7
		ShortGPT	24.1	47.7	48.3	33.7	22.1	23.0	28.2	58.9	25.6	52.5	36.4	56.6
		Streamline	26.6	54.1	55.1	36.2	23.4	23.0	29.8	62.8	27.0	55.0	39.3	61.1
		OverRep	27.2	56.5	61.9	38.4	22.5	23.0	32.2	64.5	28.8	53.5	40.9	63.5
Qwen3-8B	-	Dense	56.9	82.0	83.1	78.6	54.2	74.7	42.2	79.5	42.2	72.2	66.6	100.0
	25%	LoRA	40.8	64.0	69.8	60.0	31.0	65.9	33.0	67.6	36.2	63.2	53.2	79.9
		AdaLoRA	35.3	57.7	62.3	56.2	29.0	52.3	32.2	67.2	35.9	65.7	49.4	74.2
		RankAdaptor	41.0	64.5	67.6	58.7	30.7	67.6	34.0	66.7	39.2	63.9	53.4	80.2
		RestoreLCC	32.9	44.4	62.2	45.2	23.8	63.9	34.2	62.1	30.0	63.2	46.2	69.4
		LaCo	38.6	66.2	63.1	56.8	28.1	34.1	35.4	70.7	36.3	59.5	48.9	73.4
		ShortGPT	39.8	71.3	52.5	60.6	28.8	25.5	40.4	74.8	33.4	56.3	48.3	72.6
		Streamline	39.5	72.1	62.2	61.7	29.7	23.0	39.8	75.8	35.8	55.4	49.5	74.4
		OverRep	41.1	69.5	62.7	60.8	29.1	64.4	36.2	71.3	36.4	65.0	53.7	80.6
	50%	LoRA	27.0	48.1	61.3	35.4	21.1	22.9	29.0	60.7	26.1	51.7	38.3	57.6
		AdaLoRA	25.3	39.5	62.0	32.5	21.0	22.9	30.8	59.6	25.6	49.4	36.9	55.4
		RankAdaptor	27.5	48.0	61.6	35.7	21.6	22.9	30.0	60.8	26.6	50.4	38.5	57.9
		RestoreLCC	27.5	27.1	38.0	28.9	18.7	22.8	30.0	54.9	22.6	51.5	32.2	48.4
		LaCo	24.9	42.4	62.2	32.5	20.7	23.0	28.4	59.1	27.2	51.6	37.2	55.9
		ShortGPT	23.0	48.0	52.1	35.5	21.9	23.0	28.4	62.2	24.8	52.2	37.1	55.8
		Streamline	26.2	58.2	62.1	39.4	22.5	22.9	32.0	66.0	28.3	53.0	41.1	61.7
		OverRep	28.9	58.0	59.5	41.5	23.1	23.0	36.4	66.4	29.7	53.3	42.0	63.1

Table 20: Experimental results of reasoning tasks on Qwen3 reported as accuracy and RP.

	PR	Method	CoQA	GSM8K	TriviaQA	Avg.	RP
LLaMA2-7B	-	Dense	75.4	13.9	59.9	49.7	100.0
	25%	LoRA	54.2	2.0	20.2	25.5	51.2
		AdaLoRA	54.4	2.4	19.9	25.6	51.4
		RankAdaptor	47.5	2.1	18.9	22.8	45.9
		RestoreLCC	62.8	1.7	17.9	27.5	55.2
		LaCo	19.4	0.6	37.0	19.0	38.2
		ShortGPT	49.8	2.0	19.5	23.8	47.8
		Streamline	70.9	2.7	16.4	30.0	60.3
		OverRep	72.4	2.4	17.6	30.8	61.9
	50%	LoRA	22.8	1.6	4.1	9.5	19.1
		AdaLoRA	20.6	1.5	3.1	8.4	16.9
		RankAdaptor	25.4	2.4	3.9	10.6	21.2
		RestoreLCC	26.7	1.4	2.5	10.2	20.5
		LaCo	18.5	0.7	12.8	10.7	21.5
		ShortGPT	29.7	1.0	7.0	12.6	25.2
		Streamline	8.9	1.7	9.2	6.6	13.3
		OverRep	36.7	0.9	6.2	14.6	29.4
LLaMA2-13B	-	Dense	76.5	24.9	66.2	55.9	100.0
	25%	LoRA	62.2	5.6	33.8	33.9	60.6
		AdaLoRA	61.1	3.1	29.0	31.1	55.6
		RankAdaptor	61.4	6.4	33.7	33.8	60.6
		RestoreLCC	73.1	1.7	20.9	31.9	57.1
		LaCo	48.0	3.6	52.3	34.6	62.0
		ShortGPT	54.1	0.0	26.6	26.9	48.1
		Streamline	77.6	7.1	23.1	35.9	64.3
		OverRep	76.7	9.4	26.6	37.6	67.2
	50%	LoRA	50.3	1.6	9.9	20.6	36.9
		AdaLoRA	38.4	1.3	7.3	15.7	28.0
		RankAdaptor	48.0	2.2	9.8	20.0	35.8
		RestoreLCC	36.8	0.6	1.9	13.1	23.4
		LaCo	20.6	1.4	19.4	13.8	24.6
		ShortGPT	47.8	0.0	10.4	19.4	34.7
		Streamline	39.9	1.8	8.7	16.8	30.1
		OverRep	59.6	2.3	8.7	23.5	42.1
LLaMA3-3B	-	Dense	75.9	25.2	56.3	52.5	100.0
	25%	LoRA	38.4	3.3	13.4	18.4	35.0
		AdaLoRA	42.6	3.0	13.2	19.6	37.4
		RankAdaptor	41.1	3.6	13.0	19.2	36.7
		RestoreLCC	48.0	2.0	8.6	19.5	37.2
		LaCo	34.5	2.0	14.0	16.8	32.0
		ShortGPT	26.0	4.5	11.1	13.9	26.4
		Streamline	56.3	2.0	13.6	24.0	45.7
		OverRep	58.9	2.6	12.7	24.7	47.1
	50%	LoRA	9.4	1.7	3.7	4.9	9.4
		AdaLoRA	3.8	1.7	1.3	2.3	4.3
		RankAdaptor	9.6	1.3	3.1	4.7	8.9
		RestoreLCC	2.0	1.1	0.4	1.2	2.2
		LaCo	5.6	0.0	4.1	3.2	6.2
		ShortGPT	3.6	1.5	1.3	2.1	4.1
		Streamline	8.3	1.7	0.2	3.4	6.5
		OverRep	12.3	1.1	3.4	5.6	10.7
LLaMA3-8B	-	Dense	79.3	49.5	66.5	65.1	100.0
	25%	LoRA	44.8	14.8	18.3	26.0	39.9
		AdaLoRA	58.4	16.8	18.1	31.1	47.8
		RankAdaptor	45.3	18.9	19.4	27.9	42.8
		RestoreLCC	52.5	4.9	13.6	23.7	36.4
		LaCo	56.8	5.3	30.8	31.0	47.6
		ShortGPT	37.6	7.8	17.4	20.9	32.1
		Streamline	66.3	8.6	16.5	30.5	46.8
		OverRep	68.8	9.9	18.9	32.5	50.0
	50%	LoRA	18.2	1.4	4.0	7.9	12.1
		AdaLoRA	9.2	2.7	2.4	4.8	7.3
		RankAdaptor	18.5	1.4	4.1	8.0	12.3
		RestoreLCC	2.3	1.4	0.2	1.3	2.0
		LaCo	22.3	0.9	8.8	10.7	16.4
		ShortGPT	16.3	0.2	7.2	7.9	12.1
		Streamline	8.8	1.4	0.9	3.7	5.7
		OverRep	21.8	1.7	5.3	9.6	14.7

Table 21: Experimental results of generation tasks on LLaMA2 and LLaMA3 reported using task-specific metrics and RP.

	PR	Method	CoQA	GSM8K	TriviaQA	Avg.	RP
Qwen3-4B	-	Dense	83.2	84.7	50.6	72.8	100.0
	25%	LoRA	33.6	1.4	8.2	14.4	19.8
		AdaLoRA	37.9	2.1	8.5	16.2	22.2
		RankAdaptor	30.1	2.1	6.9	13.0	17.9
		RestoreLCC	23.7	1.1	3.8	9.5	13.1
		LaCo	25.6	0.8	13.2	13.2	18.1
		ShortGPT	24.9	0.5	21.0	15.5	21.2
		Streamline	39.3	2.4	20.7	20.8	28.6
		OverRep	57.4	2.5	11.6	23.8	32.7
	50%	LoRA	7.7	1.7	3.3	4.2	5.8
		AdaLoRA	4.4	1.1	0.7	2.1	2.8
		RankAdaptor	8.1	1.6	3.6	4.4	6.1
		RestoreLCC	3.2	0.5	1.2	1.6	2.2
		LaCo	4.2	0.0	1.2	1.8	2.5
		ShortGPT	3.3	0.0	0.3	1.2	1.6
		Streamline	8.6	0.4	6.1	5.0	6.9
		OverRep	16.2	1.1	5.0	7.4	10.2
Qwen3-8B	-	Dense	84.4	85.4	61.6	77.1	100.0
	25%	LoRA	36.2	5.8	13.2	18.4	23.9
		AdaLoRA	48.3	3.0	10.5	20.6	26.7
		RankAdaptor	45.7	6.1	13.5	21.8	28.2
		RestoreLCC	31.5	2.1	10.2	14.6	18.9
		LaCo	31.2	0.8	18.0	16.7	21.6
		ShortGPT	22.5	0.9	33.9	19.1	24.8
		Streamline	27.2	1.9	42.6	23.9	31.0
		OverRep	56.5	2.4	13.8	24.2	31.4
	50%	LoRA	11.7	1.2	4.2	5.7	7.4
		AdaLoRA	5.4	1.1	0.9	2.5	3.2
		RankAdaptor	11.5	0.8	4.5	5.6	7.3
		RestoreLCC	4.2	0.9	1.0	2.0	2.6
		LaCo	2.7	0.0	1.9	1.5	2.0
		ShortGPT	2.4	0.0	0.0	0.8	1.0
		Streamline	10.6	1.4	5.1	5.7	7.4
		OverRep	12.6	0.1	7.5	6.7	8.7

Table 22: Experimental results of generation tasks on Qwen3 reported with task-specific metrics and RP.