CARGO: CARRYING CURVATURE ACROSS TASKS FOR CONTINUAL PRUNING OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual pruning has recently emerged as a promising direction that progressively updates pruned LLMs as new tasks arrive, while retaining knowledge from previously encountered tasks. Existing methods focus primarily on which weights to prune, leaving unexplored whether the remaining weights can be actively adapted to compensate for the information lost through continual pruning. We find that combining accumulated-importance-based weight pruning with existing single-task weight reconstruction strategies can improve final performance but leads to larger forgetting, calling for a dedicated design for continual pruning. To this end, we first formulate a progressive weight reconstruction problem for continual pruning to jointly minimize the layer-wise pruning error of the pruned model over all tasks seen so far. Under fixed layer inputs and an invertible sum of task Hessians, we show that optimal compensation for a single-weight deletion depends on historical tasks only through their accumulated Hessian, enabling efficient reconstruction without retaining past calibration data. Building on this insight, we next propose CARGO (**C**urvature **A**ccumulation and **R**egime-**G**ated **Co**nsolidation), a training-free continual pruning framework that carries curvature information across tasks to guide both weight selection and reconstruction. To robustly consolidate curvature over time, CARGO further incorporates spectral cleaning to suppress noisy low-energy curvature components and a task-correlation-aware directional gate to selectively retain historical curvature. Experiments across three model families and two task streams show that CARGO substantially improves continual pruning performance over strongest baselines.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023a) have recently reshaped the field of AI, demonstrating tremendous success across a wide range of applications. While adapting pretrained LLMs to specific domains can substantially improve their performance on downstream tasks (Gururangan et al., 2020), such adaptation is computationally expensive, and deploying large dense models incurs substantial memory and inference costs. Recent pruning methods, such as SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024), address this challenge by leveraging a small set of calibration samples from a target task to extract sparse models for pretrained LLMs without costly retraining. However, in practice, models often need to continuously adapt to a sequence of tasks. Directly pruning the model using only the current task can preserve weights important for the new task while removing those critical to previous tasks, resulting in catastrophic forgetting. This gives rise to a new research direction, i.e., continual pruning, which aims to continuously prune LLMs to adapt to new tasks while retaining knowledge from previous ones.

Recent work (Malla et al., 2024) has taken an initial step toward addressing this question, by repeatedly pruning the same dense model for each new task using task-specific calibration data. To preserve the knowledge of previous tasks while maintaining performance on the current one without access to previous tasks' calibration data, a training-free approach COPAL is introduced which accumulates per-weight importance scores across tasks to identify the weights important to all tasks encountered so far. Nonetheless, this approach focuses primarily on *which* weights to preserve or prune, leaving unexplored whether the remaining weights can be actively adapted to compensate for the information lost through continual pruning. Meanwhile, weight reconstruction has shown great potential for enhancing pruning performance in the single-task setting. Methods such as Op-

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.

timal Brain Surgeon (OBS) (Hassibi & Stork, 1992) and SparseGPT (Frantar & Alistarh, 2023) compensate the pruning-induced errors by adjusting the remaining weights based on second-order information derived from input features. Thus motivated, an interesting question to ask is: *Can weight reconstruction be leveraged for better continual pruning?*

To answer this question, a natural approach is to combine the idea of importance score accumulation with the single-task weight reconstruction strategy as in SparseGPT that uses the current task Hessian. We consider two different strategies: *Full reconstruction* which updates all surviving weights, and *Protected reconstruction* which fixes half of the surviving weights with the highest historical scores and reconstructs the rest. Figure 1 shows that both can achieve higher final average accuracy over COPAL on all three models in the Long-seq benchmark, but with a substantially lower backward transfer (BWT, i.e., larger forgetting). While these results highlight the value of weight reconstruction for compensating for pruning-induced errors, relying only on the current task Hessian can bias the pruned model toward the current task at the expense of previously gained knowledge. This points to the need of a dedicated reconstruction strategy that explicitly balances knowledge adaptation and retention in continual pruning.

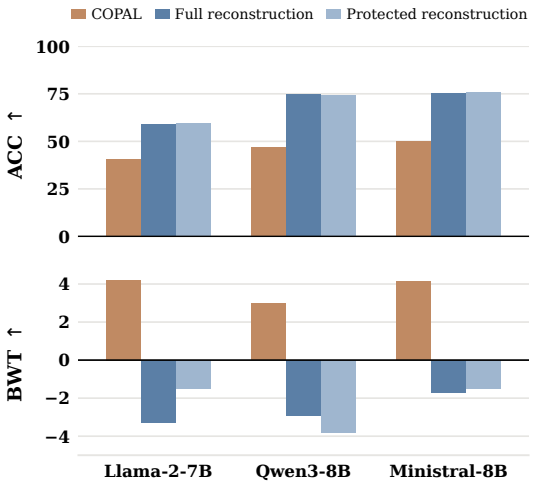

Figure 1: Long-seq ACC and BWT at 50% sparsity: COPAL (Malla et al., 2024), *Full reconstruction*, and *Protected reconstruction*.

To this end, by revisiting the single-task weight reconstruction objective in OBS (Hassibi & Stork, 1992), we first formulate a *progressive weight reconstruction* problem for continual pruning, where the goal is to find an optimal weight reconstruction that jointly minimizes the layer-wise pruning error of the current sparse model on all tasks seen so far. Interestingly, we can show that the optimal compensation for deleting any individual weight depends on previous tasks only through the sum of their task-specific Hessians. This result reveals that the curvature information required for reconstruction can be efficiently carried across tasks: the Hessian sum can be updated incrementally during continual pruning using only a single matrix for each pruned layer, that is independent of the number of tasks and does not require storing any calibration data from previous tasks.

Based on this insight, we propose CARGO, a training-free continual pruning approach that accumulates curvature information across tasks to guide weight pruning and reconstruction. To address challenges arising from curvature accumulation, CARGO further incorporates two mechanisms: 1) spectral cleaning, which suppresses noisy low-energy components in Hessians whose influence can be amplified through accumulation; and 2) task-correlation-aware directional gating, which selectively retains historical curvature directions supported by the current task while decaying it elsewhere. Extensive experiments across multiple LLM families and benchmarks demonstrate that CARGO consistently and substantially improves continual pruning performance over existing baselines.

2 Related Work

LLM Pruning. Post-training LLM pruning leverages calibration data to guide weight selection and preserve model outputs after sparsification. SparseGPT (Frantar & Alistarh, 2023) scales OBS (Hassibi & Stork, 1992) to LLMs, jointly selecting weights for pruning and compensating for their removal through layer-wise reconstruction based on second-order information. Wanda (Sun et al., 2024) scores weights by combining their magnitudes with input activation norms, while RIA (Zhang et al., 2024) further accounts for relative weight importance and permutes channels to preserve important weights. Other approaches focus on the compression structure or sparsity allocation: SliceGPT (Ashkboos et al., 2024) rotates representations to enable structured removal of rows and columns, whereas LSA (Yang et al., 2026) allocates sparsity across layers and projections based on

their minimal linear reconstruction errors. These methods use a fixed calibration set and are not explicitly designed for continual pruning.

Continual Pruning. Earlier approaches learn sparse subnetworks during continual training. For instance, SNCL (Yan et al., 2022) learns sparse neuron activations using Bayesian priors and experience replay; Continual Prune-and-Select (Dekhovich et al., 2023) trains and prunes task-specific subnetworks while preserving connections learned for previous tasks. These approaches require training on new tasks. In contrast, COPAL (Malla et al., 2024) introduces training-free continual pruning for LLMs by accumulating perturbation-based sensitivity scores across tasks. While these accumulated scores guide mask selection, COPAL does not compensate for pruning-induced errors by reconstructing the surviving weights. More recently, PruneStream (Huang et al., 2026) treats pruning masks as structural memory for heterogeneous vision tasks, using shared and task-specific LoRA adapters to estimate weight importance and refine the masks over time. Although the backbone remains frozen, both task adaptation and pruning compensation rely on training these adapters.

Curvature and Subspace Memory in Continual Learning. From a curvature perspective, EWC (Kirkpatrick et al., 2017) uses diagonal Fisher information as a curvature approximation to impose importance-weighted quadratic penalties on parameter updates, while Online Structured Laplace (Ritter et al., 2018) maintains a Gaussian posterior whose precision accumulates layer-wise Kronecker-factored approximations of the loss Hessian. More recently, VR-MCL (Wu et al., 2024) reduces hypergradient variance to stabilize the implicit online Hessian approximation in meta-continual learning. OCAR (Urettini & Carta, 2025) uses a Kronecker-factored Fisher information matrix to precondition gradients on incoming and replayed data. From a subspace perspective, GPM (Saha et al., 2021) preserves dominant activation directions from previous tasks and projects gradients for new tasks onto their orthogonal complement. TRGP (Lin et al., 2022b) further identifies related tasks through gradient projections onto their activation subspaces and learns to scale weight projections within selected subspaces to facilitate knowledge reuse. CUBER (Lin et al., 2022a) combines task-specific activation subspaces with stored task gradients, relaxing projection constraints for positively correlated tasks to enable backward transfer. CODE-CL (Apolinario et al., 2025) forms conceptor matrices from layer-input correlations for soft gradient projection and extracts shared directions through intersections of current and historical conceptors. While these methods demonstrate the value of preserving curvature or subspace information across tasks, they primarily use such information to regulate gradient-based continual learning, rather than to enable training-free continual pruning.

3 PRELIMINARIES

Continual Pruning. Continual pruning aims to progressively prune LLMs to adapt to new tasks while retaining knowledge from previously encountered tasks without access to their data. Specifically, let W^0 denote the dense pretrained LLM. A sequence of tasks $t = 1, \ldots, T$ arrive one at a time, where T is the number of tasks and $\mathcal{D}_t$ denotes the calibration data for task t. For each task t, the pruner uses its calibration data $\mathcal{D}_t$ and produces a model $\widehat{W}_t$ from the frozen LLM W^0 at target sparsity s, which characterizes the fraction of weights set to zero. Pruning using only current-task data can remove weights important to previous tasks and cause forgetting of their knowledge.

To address this challenge, COPAL (Malla et al., 2024) accumulates weight importance scores across tasks to preserve information about previous tasks. Specifically, upon receiving the new task t, the pruner computes a perturbation-based importance score $\sigma_t(w)$ for each weight w and adds it to the previous accumulated score $I_{t-1}(w)$ from previous tasks:

$$I_t(w) \;=\; \sigma_t(w) + I_{t-1}(w), \qquad I_0 \equiv 0. \tag{1}$$

Here the importance score $\sigma_t(w)$ is captured by the product of the weight magnitude and the derivative of a perturbation loss with respect to that weight's perturbation, where the perturbation loss measures the model output change caused by small perturbations to the weight and input. Given a target sparsity, the weights with the largest accumulated scores I_t are preserved at their pretrained values. Importantly, this accumulation requires storing only a single scalar per weight, resulting in a memory cost that is independent of the number of tasks.

Layer-wise Pruning with Weight Reconstruction. SparseGPT (Frantar & Alistarh, 2023) formulates single-task pruning as layer-wise weight reconstruction using the second-order framework

of Optimal OBS (Hassibi & Stork, 1992). Weight reconstruction adjusts the surviving weights to compensate for the output error introduced by pruning. For dense and sparse layer weights $W, \widehat{W} \in \mathbb{R}^{r_{\text{out}} \times d}$, the objective is to minimize the output change on calibration inputs X:

$$\mathcal{L}(\widehat{W}) \;=\; \left\|(W - \widehat{W})\, X\right\|_F^2, \tag{2}$$

where d and r_{out} are the input and output dimensions, $X \in \mathbb{R}^{d \times n}$ collects n token-level calibration inputs, and $\|\cdot\|_F$ is the Frobenius norm. With X held fixed, the loss separates over output rows. For dense and sparse weights $w, \widehat{w}$, the change $\delta = \widehat{w} - w$ gives a loss $\delta^\top X X^\top \delta$ with Hessian $2XX^\top$, where all weights share the positive-semidefinite curvature matrix

$$H \;=\; \textstyle\sum_x x x^\top \;=\; XX^\top \in \mathbb{R}^{d \times d}. \tag{3}$$

For invertible H, OBS (Hassibi & Stork, 1992) gives the minimum reconstruction loss from deleting weight w_q as $w_q^2 / [H^{-1}]_{qq}$ (the *saliency*), attained by the reconstruction update $\delta = -\frac{w_q}{[H^{-1}]_{qq}} H^{-1} e_q$. Here e_q is the q-th coordinate vector, and $[H^{-1}]_{qq}$ is the corresponding diagonal entry of the inverse Hessian.

Both weight selection and reconstruction depend on inverse curvature. In practice, $H = XX^\top$ can be singular or ill-conditioned when calibration features are linearly dependent or strongly correlated. SparseGPT (Frantar & Alistarh, 2023) applies diagonal damping before inversion, using $H_\lambda = H + \lambda I$, where $\lambda > 0$ is the damping strength and I is the identity matrix. This shifts every eigenvalue upward by λ, making H_λ positive definite and invertible. The inverse H_λ^{-1}, computed through Cholesky factorization, replaces H^{-1} in the saliency and reconstruction updates above. SparseGPT (Frantar & Alistarh, 2023) applies these updates through a column sweep; implementation details are summarized in Appendix D.

4 Method

To enable effective weight reconstruction for continual pruning, we propose CARGO, a training-free continual pruning approach that accumulates curvature information across tasks to jointly guide weight pruning and reconstruction. Figure 2 summarizes the workflow.

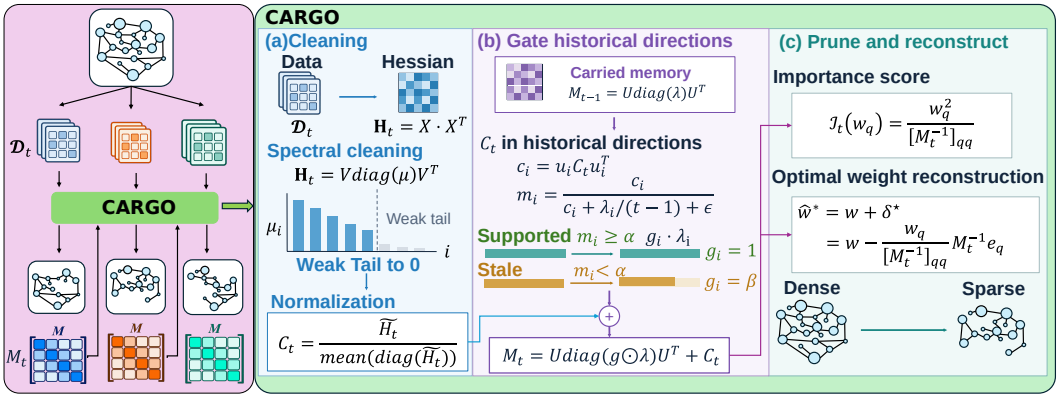

Figure 2: Overview of CARGO. Spectral cleaning and directional retention produce the accumulated curvature used for weight selection and reconstruction.

4.1 Hessian Accumulation for Joint Pruning and Reconstruction

A natural design is to combine the score accumulation of COPAL (Malla et al., 2024) with the weight reconstruction of SparseGPT (Frantar & Alistarh, 2023). At each task, we start from the same dense weights and use the OBS (Hassibi & Stork, 1992) saliencies under the current task as the per-weight importance score. The pruning mask is then determined by removing weights with the lowest accumulated scores, i.e., the smallest reconstruction losses. Given this mask, we consider two reconstruction strategies:

- *Full reconstruction* updates all surviving weights using the current task Hessian following the column-wise reconstruction procedure of SparseGPT. However, this may alter weights that are important to previous tasks but less important under the current task.
- *Protected reconstruction* aims to mitigate such interference by ranking the survivors in each linear module according to the importance scores accumulated before the current task. The higher-scoring half, regarded as more important to previous tasks, is fixed at dense values, while only the remaining half are reconstructed using the current task Hessian.

Both designs achieve higher final average accuracy than COPAL (Malla et al., 2024) in the six 7–8B settings in Table 1, but yield substantially lower BWT, as illustrated on Long-seq in Figure 1. These results highlight the benefit of reconstruction for reducing pruning-induced errors, alongside increased forgetting of previously acquired knowledge. In both designs, historical scores guide which weights are pruned or protected, but reconstruction uses only the current task Hessian and leaves past-task errors out of the objective. To address this, we revisit the single-task objective in (2) and propose the following progressive weight reconstruction problem for continual pruning:

$$\min_{\widehat{W}} \ \mathcal{L}_{1:t}(\widehat{W}) \ = \ \sum_{k=1}^{t} \left\| (W^0 - \widehat{W})\, X_k \right\|_F^2, \tag{4}$$

where $X_k \in \mathbb{R}^{d \times n_k}$ contains the layer inputs recorded during task k's pruning pass and n_k is the number of recorded token-level inputs for task k. Intuitively, solving Eq. (4) at task t leads to a sparse model that can perform well on all tasks encountered so far. Let $H_k = X_k X_k^\top$ and $H_t^{\text{sum}} = \sum_{k=1}^{t} H_k$. As in SparseGPT (Frantar & Alistarh, 2023), fixing the layer inputs allows the joint loss in (4) to be decomposed over the rows of the layer's weight matrix, each corresponding to one output feature. For continual pruning, Proposition 1 shows that the optimal reconstruction and minimum joint loss depend on past tasks only through their accumulated Hessian.

Proposition 1 (Optimal weight reconstruction for continual pruning) *Assume H_t^{sum} is invertible given fixed layer inputs $\{X_k\}_{k=1}^t$. For each row w in the dense model W^0, upon deleting its q-th weight w_q, the minimizer $\widehat{W}^\star$ of Eq.* (4) *replaces w with $\widehat{w}^\star$ and leaves all other rows unchanged, where*

$$\mathcal{L}_{1:t}(\widehat{W}^\star) = \mathcal{L}_{1:t}^{\min}(q) = \frac{w_q^2}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}}, \ \widehat{w}^\star = w + \delta^\star = w - \frac{w_q}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}} (H_t^{\text{sum}})^{-1} e_q. \tag{5}$$

The detailed proof is in Appendix C. The minimum loss $\mathcal{L}_{1:t}(\widehat{W}^\star)$ serves as an importance score for weight selection, while $\delta^\star$ gives the optimal weight reconstruction to compensate for pruning-induced errors. Clearly, both values depend on task history only through the accumulated Hessian H_t^{sum}, which admits an efficient online update

$$H_t^{\text{sum}} = H_{t-1}^{\text{sum}} + X_t X_t^\top, \qquad H_0^{\text{sum}} = 0.$$

Instead of storing the calibration data for previous tasks, we only need to store a single $d \times d$ matrix for the accumulated Hessian for each layer and update it progressively with the current task curvature. However, direct Hessian accumulation requires controlling estimation errors and balancing historical retention with adaptation to new tasks. We address these challenges next.

4.2 Spectral Cleaning and Per-Task Normalization

With limited calibration data, weak curvature directions can be poorly estimated (Tatzel et al., 2025). Such estimation errors can have a disproportionate impact on the pruning decisions in Eq. (5), as the inverse Hessian amplifies errors along low-curvature directions. Moreover, the scales of layer inputs can vary substantially across tasks, causing tasks with large-magnitude activations to dominate the unnormalized Hessian sum. To address these problems, from the second task onward, we will truncate the spectral tail of each incoming Hessian and normalize the retained curvature.

Specifically, let the Eigen-decomposition of the positive semi-definite Hessian H_t for the current task t be $H_t = V \operatorname{diag}(\mu) V^\top$, with eigenvalues $\mu_1 \geq \cdots \geq \mu_d \geq 0$ and $\operatorname{tr}(H_t) > 0$. We define

$$\ell_t = \min\left\{ \ell : \frac{\sum_{i=1}^{\ell} \mu_i}{\sum_{i=1}^{d} \mu_i} \geq \rho \right\}, \qquad \widetilde{H}_t = V_{1:\ell_t} \operatorname{diag}(\mu_{1:\ell_t}) V_{1:\ell_t}^\top. \tag{6}$$

Here ℓ_t is the retained rank; $V_{1:\ell_t}$ and $\mu_{1:\ell_t}$ select the first ℓ_t columns and entries. In this way, $\widetilde{H}_t$ keeps the minimum number of leading directions needed to capture a fraction ρ of the total trace. We then normalize $\widetilde{H}_t$ by its mean diagonal value, $\bar{d}(A) \coloneqq \mathrm{mean}(\mathrm{diag}\, A)$, to put task contributions on a common scale before accumulation. The cleaned and normalized curvature matrix for task t is $C_t = \widetilde{H}_t / \bar{d}(\widetilde{H}_t)$.

4.3 Directional Retention

Under direct Hessian accumulation, historical curvature grows with the number of tasks, whereas each new task contributes only a single normalized Hessian. Consequently, later tasks may exert progressively less influence along directions already strongly represented by the accumulated curvature. Moreover, simple accumulation preserves curvature contributions from early tasks indefinitely, even when those directions are no longer supported by subsequent tasks. To address these, we next use recurrence as a prior for reusable structure, where historical directions retain their weight when later tasks support them and decay otherwise.

Specifically, for $t \geq 2$ and given the accumulated curvature $\mathbf{M}_{t-1}$, let the Eigen-decomposition be $\mathbf{M}_{t-1} = U\,\mathrm{diag}(\lambda)U^\top$, with the eigenvectors u_i and the corresponding eigenvalues λ_i. Given the updated and cleaned curvature matrix C_t for the current task t, we measure its curvature along historical direction u_i as

$$c_i \;=\; u_i^\top C_t\, u_i \;\geq\; 0. \tag{7}$$

We compare this curvature with the historical reference $b_i = \lambda_i/(t-1)$, using a small constant $\varepsilon > 0$ for numerical stability:

$$m_i \;=\; \frac{c_i}{c_i + b_i + \varepsilon} \;\in\; [0, 1). \tag{8}$$

The ratio follows the graded-membership form of conceptors (Jaeger, 2014; Apolinario et al., 2025) and ridge shrinkage (Hoerl & Kennard, 1970), with the reference scale b_i determined by history. For an ungated sum in a fixed direction, b_i is the historical per-task average; under gating, it measures retained curvature per elapsed task.

With support threshold α and decay factor $\beta \in (0, 1]$, the historical retention factor is

$$g_i = \begin{cases} 1, & m_i \geq \alpha \quad \text{(novel or re-supported)} \\ \beta, & m_i < \alpha \quad \text{(stale).} \end{cases} \tag{9}$$

The update rescales historical eigenvalues and adds the current contribution:

$$\mathbf{M}_t \;=\; U\,\mathrm{diag}(g \odot \lambda)\,U^\top \;+\; C_t. \tag{10}$$

Here g collects the retention factors g_i, and $\odot$ denotes element-wise multiplication. The gate makes a binary decision on historical curvature: retain it with factor 1 or decay it with factor β. The cleaned matrix C_t of the current task t enters the accumulated curvature in full, adding novel directions that have no historical weight to decay. Since C_t is positive semi-definite and $g_i \geq 0$, the update preserves positive semi-definiteness.

With these components in place, we now present our approach CARGO, which carries an accumulated curvature matrix for each layer across tasks to jointly guide weight selection and reconstruction. As shown in Figure 2 and Algorithm 1 in the appendix, for the first task, CARGO prunes the pretrained model based on Eq. 5 with its normalized Hessian, which also initializes the accumulated curvature matrix. For each subsequent task, CARGO first cleans and normalizes the incoming Hessian as in Section 4.2. It then retains historical curvature in directions supported by the current task, decays it elsewhere, and incorporates the new curvature contribution to obtain $\mathbf{M}_t$ as in Section 4.3. At each task, CARGO starts from the same dense pretrained model and uses $\mathbf{M}_t$ as a surrogate for H_t^{sum} to select weights for pruning and compensate for the resulting pruning error based on Proposition 1. Specifically, after applying diagonal damping to $\mathbf{M}_t$ to ensure invertibility as in SparseGPT, the importance score for deleting weight w_q and the corresponding optimal update to the row w are given by:

$$\mathcal{I}_t(w_q) \;=\; \frac{w_q^2}{[\mathbf{M}_t^{-1}]_{qq}}, \qquad \widehat{w}^\star \;=\; w - \frac{w_q}{[\mathbf{M}_t^{-1}]_{qq}}\,\mathbf{M}_t^{-1} e_q. \tag{11}$$

Table 1: Performance comparison at 50% unstructured sparsity. Cells report ACC (BWT) and the strongest results are shown in bold. A full table with error bars is in Appendix B.1.

Method	Llama-2-7B		Qwen3-8B		Ministral-8B	
ACC↑(BWT↑)	TRACE	Long-seq	TRACE	Long-seq	TRACE	Long-seq
Dense	38.80	65.62	55.66	79.62	53.82	78.56
Single-task SparseGPT	37.13	62.57	50.72	77.43	49.85	77.59
COPAL	30.69(+7.58)	40.82(+4.21)	45.69(+7.25)	47.16(+2.96)	37.45(+4.81)	50.35(+4.12)
Accum. Wanda	34.53(+0.89)	54.07(+0.43)	44.02(+2.15)	58.37(−1.43)	40.73(+2.53)	66.65(+1.31)
Full reconstruction	35.45(+0.34)	60.23(−2.62)	46.17(−3.07)	75.16(−2.55)	43.37(−7.19)	75.89(−1.38)
Protected reconstruction	34.80(−1.64)	59.88(−2.88)	45.41(−4.55)	74.83(−3.36)	44.92(−7.31)	76.65(−1.06)
CARGO	**37.55**(−0.17)	**62.21**(+1.45)	**50.99**(+0.30)	**77.53**(+0.09)	**48.37**(+0.57)	**78.03**(−0.28)
Method	Llama-2-13B		Qwen3-14B		Ministral-14B	
Dense	43.63	69.12	57.83	80.53	58.39	79.91
Single-task SparseGPT	40.69	66.94	54.76	79.95	54.32	78.55
COPAL	36.09(+8.63)	38.91(+3.74)	53.11(+7.33)	37.59(+0.85)	48.87(+9.66)	73.66(+9.13)
Accum. Wanda	37.63(+4.36)	59.36(+0.20)	53.02(+5.21)	68.87(+0.89)	50.23(+5.78)	76.47(+1.95)
Full reconstruction	36.81(−0.71)	66.66(−2.84)	53.21(−1.12)	79.42(−1.58)	48.64(−2.33)	75.80(−2.26)
Protected reconstruction	36.28(−2.70)	65.98(−2.47)	52.87(−1.19)	80.20(−0.84)	49.49(−5.53)	77.04(−1.63)
CARGO	**41.99**(+2.39)	**66.86**(+2.18)	**55.15**(+0.34)	**80.80**(+0.38)	**55.06**(+0.84)	**78.97**(+0.66)

5 Experiments

5.1 Experimental Setup

Task streams and models. We test CARGO on two continual pruning streams: **TRACE** (Wang et al., 2023) contains eight heterogeneous tasks, and **Long-seq** (Razdaibiedina et al., 2023) contains fifteen text classification tasks. Each stream is evaluated on two sizes from three model families: Llama-2-7B/13B-chat (Touvron et al., 2023b), Qwen3-8B/14B (Yang et al., 2025), and Ministral-3-8B/14B-Instruct (Liu et al., 2026). Dataset details are given in Appendix A.1.

Baselines. We compare CARGO with COPAL (Malla et al., 2024) and three continual baselines that we construct from Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023). *Accum. Wanda* sums Wanda's activation-aware importance scores across tasks and selects the mask from this sum, leaving surviving weights unchanged. *Full reconstruction* and *Protected reconstruction* use the current task's Hessian for weight reconstruction (Section 4.1). Additionally, the dense models and single-task SparseGPT (Frantar & Alistarh, 2023) serve as references, with SparseGPT pruned and evaluated independently on each task. Further details are given in Appendix A.2.1.

Protocol and metrics. For every task, all approaches start from the same dense pretrained models and prune without gradient updates; the main comparison targets 50% unstructured sparsity. We evaluate all tasks seen so far after each pruning step, using the task-specific TRACE metrics or Long-seq accuracy and normalizing each score on a $[0, 100]$ scale. Let $R_{t,k}$ be the score on task k after pruning step t in a stream of T tasks. We report final average performance (ACC) and backward transfer (BWT):

$$\mathrm{ACC} = \frac{1}{T}\sum_{k=1}^{T} R_{T,k}, \qquad \mathrm{BWT} = \frac{1}{T-1}\sum_{k=1}^{T-1}\big(R_{T,k} - R_{k,k}\big). \tag{12}$$

ACC measures performance at the end of the stream, while BWT measures the average performance change on earlier tasks relative to their initial post-pruning scores. Higher values are better for both metrics. Hyperparameters and evaluation settings are given in Appendices A.3 and A.2.2.

5.2 Main Results

At 50% unstructured sparsity, CARGO achieves the highest final ACC among continual pruning approaches across all twelve model–stream settings (Table 1), outperforming the strongest continual baseline in each setting by 0.20–4.83 points. In particular, CARGO significantly outperforms COPAL, which does not perform weight reconstruction, demonstrating the benefit of incorporating appropriate reconstruction into continual pruning. Remarkably, CARGO also surpasses the independently pruned single-task SparseGPT (Frantar & Alistarh, 2023) reference in nine of the twelve settings, suggesting that carrying information across tasks can enable beneficial knowledge transfer during continual pruning. Despite pruning 50% of the weights, CARGO remains close to the dense

models across all settings, retaining 94.0% and 97.9% of their ACC on TRACE and Long-seq. Finally, compared with *Full reconstruction* and *Protected reconstruction*, which reconstruct weights using only the current task's Hessian and exhibit the most negative BWT overall, CARGO substantially reduces forgetting by better balancing the retention of previously acquired knowledge with adaptation to new tasks.

5.3 Ablation

In what follows, we conduct various studies to better understand the performance of CARGO under different scenarios. More results about hyperparameter sensitivity and computational costs are in the appendix due to the space limit.

Current-task performance. We first evaluate the adaptation capability by measuring performance on each task immediately after pruning with its calibration data. As shown in Figure 3, CARGO consistently achieves strong current-task performance compared to the baselines under different sparsity levels, while COPAL performs the worst, highlighting the importance of weight reconstruction for adapting to incoming tasks. For instance, on Llama-2-7B-chat across the eight TRACE tasks, CARGO achieves an average current-task performance of 39.51 at 50% sparsity, closely approaching the dense model performance of 40.46. The advantage becomes more pronounced at 70% sparsity, where CARGO substantially outperforms all continual baselines, suggesting that it can effectively identify and preserve curvature information critical to the current task even under aggressive pruning.

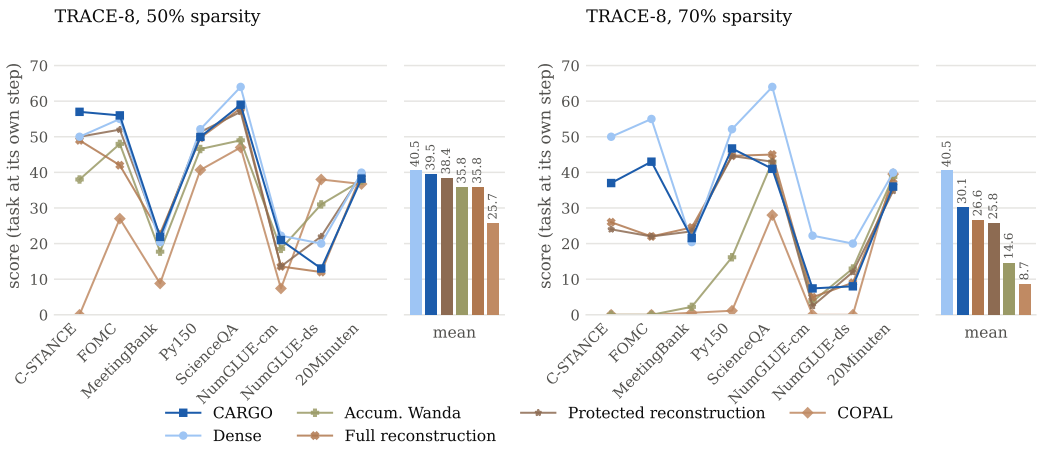

Figure 3: Current-task scores $R_{k,k}$ evaluated immediately after pruning Llama-2-7B-chat for that task on TRACE at 50% and 70% sparsity. Curves show scores by task; bars show their task averages.

Impact of each component. Table 2 evaluates three variants: (1) *ACC. Hessian* directly accumulates task Hessians without cleaning or gating; (2) CARGO *(no gate)* applies Hessian cleaning and normalization but disables directional retention by setting $\beta = 1$; (3) CARGO *(no denoise)* retains directional gating but removes spectral cleaning. It is clear that incorporating Hessian cleaning and directional gating in CARGO can significantly improve both ACC and BWT over ACC.Hessian across all six settings. For example, on TRACE with Llama-2-7B, cleaning and normalization improve ACC from 35.40 to 37.70, while further introducing directional gating increases it to 39.27. Moreover, CARGO consistently improves ACC over the no-gate variant across all settings, highlighting the benefits of selectively retaining historical curvature along directions supported by the current task. Spectral cleaning also improves performance in most settings, supporting its role in suppressing noisy weak curvature directions during curvature accumulation.

Sparsity sweep. We also evaluate all approaches at different sparsity levels beyond the 50% setting considered in Table 1. As demonstrated in Table 3, CARGO achieves the highest ACC among continual pruning baselines at both 30% and 70% sparsity. In particular, aligning with current-task performance in Figure 3, the improvement of CARGO over baselines is even more significant at higher sparsity, where accurately identifying critical weights and effectively reconstructing the surviving weights are increasingly important for preserving model performance. At low sparsity

Table 2: Impacts of different components at 50% unstructured sparsity. Cells report ACC (BWT); best ACC is shown in bold.

Variant	Llama-2-7B		Qwen3-8B		Ministral-8B	
ACC↑ (BWT↑)	TRACE	Long-seq	TRACE	Long-seq	TRACE	Long-seq
ACC.Hessian	35.40 (−1.68)	59.79 (−0.64)	50.41 (−0.22)	77.21 (−0.51)	46.50 (−2.82)	77.62 (+0.04)
CARGO (no gate)	37.70 (−1.70)	60.64 (+0.06)	48.73 (−0.79)	76.56 (−0.41)	48.12 (−0.14)	77.53 (+0.33)
CARGO (no denoise)	37.10 (−1.20)	59.86 (+0.09)	50.84 (−0.02)	77.00 (+0.18)	**49.12** (+0.36)	77.61 (+0.04)
CARGO	**39.27** (−0.28)	**62.05** (+2.26)	**51.45** (+1.26)	**77.31** (+0.03)	48.69 (−0.84)	**77.86** (+0.11)

Table 3: ACC and BWT at 30% and 70% unstructured sparsity with Llama-2-7B-chat. Best ACC per column is bold.

	TRACE				Long-sequence			
	30%		70%		30%		70%	
Method	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT
Dense	40.46		40.46		63.62		63.62	
Single-task SparseGPT	40.30		32.79		63.86		51.72	
COPAL	39.75	−0.40	10.51	+2.13	60.64	+2.33	5.87	+6.06
Accum. Wanda score	39.78	+0.79	20.94	+7.24	61.78	−0.04	20.61	+2.66
Full reconstruction	39.97	+0.88	29.86	+3.68	63.30	−1.34	35.47	−16.71
Protected reconstruction	39.73	−0.87	22.99	−3.20	62.96	−0.96	38.71	−10.13
CARGO	**40.65**	+0.63	**34.29**	+4.80	**63.85**	+0.13	**49.12**	+0.05

levels such as 30% sparsity, most pretrained weights preserved, resulting in smaller differences among the pruned models produced by different approaches.

Table 4: ACC and BWT under 2:4 semi-structured sparsity. Best ACC per column is bold.

	Llama-2-7B				Qwen3-8B				Ministral-8B			
	TRACE		Long-seq		TRACE		Long-seq		TRACE		Long-seq	
Method	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT	ACC	BWT
Dense	40.46		63.62		54.35		79.51		52.90		78.89	
Single-task SparseGPT	35.21		58.19		45.25		73.27		46.91		73.56	
COPAL	29.23	+11.35	36.67	−0.57	37.72	+6.82	43.96	+2.22	30.92	+11.69	33.25	+1.04
Accum. Wanda	30.53	+1.08	40.77	+2.38	36.97	+4.00	40.48	−10.24	29.78	−0.02	33.64	−6.24
Full reconstruction	32.63	−2.25	50.16	−8.72	40.41	−5.92	63.96	−6.64	35.16	−8.37	65.60	−7.58
Protected reconstruction	30.12	−5.74	48.44	−9.70	41.11	−6.33	63.39	−9.80	30.70	−14.02	65.89	−6.90
CARGO	**35.43**	+3.14	**58.73**	+2.71	**44.53**	−0.57	**71.27**	+0.29	**43.20**	+1.78	**73.82**	+1.67

Semi-structured 2:4 sparsity. The advantage of CARGO extends beyond unstructured pruning to hardware-friendly 2:4 sparsity, which retains two weights in every group of four and enables accelerated matrix multiplication on NVIDIA Sparse Tensor Cores (Mishra et al., 2021). As shown in Table 4, CARGO achieves the highest ACC across all six model-stream settings, outperforming the strongest competing method in each setting by 2.80–8.57 points. These gains show that cross-task curvature accumulation remains effective under hardware-oriented constraints on weight selection and reconstruction. Compared with both reconstruction baselines, CARGO achieves higher BWT in all six settings. Its positive BWT in five settings indicates improved average performance on earlier tasks after subsequent pruning.

6 Conclusion

In this paper, we propose CARGO, a training-free continual pruning framework that carries curvature across tasks to guide weight selection and reconstruction. By introducing the progressive weight reconstruction problem for continual pruning, we find that the accumulated Hessian across tasks is the key to account for pruning errors on past tasks without retaining their calibration data. Building on this insight, CARGO combines spectral cleaning to suppress poorly estimated curvature directions with directional retention to selectively preserve historical curvature. Experiments across three LLM families on TRACE and Long-seq show that CARGO consistently and significantly improves final average performance over the evaluated continual pruning baselines, with particularly strong gains at high sparsity levels and under 2:4 semi-structured pruning. A limitation of CARGO is the memory and computational overhead of storing and decomposing full curvature matrices, which grows with layer width. Future work could explore more efficient and compact curvature representations to reduce this overhead.

AI USE STATEMENT

Generative AI tools were used to assist with code implementation and checking, and to support manuscript organization, drafting, language editing, and paper polishing. The authors reviewed and verified all AI-assisted text and code changes. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Implementation details of CARGO are provided in Section 4 and Appendix D. The datasets, baseline implementations, hyperparameters, computational setup, and evaluation protocol are described in Section 5.1 and Appendix A. Our implementation code is provided in the supplementary material.

REFERENCES

Marco P.E. Apolinario, Sakshi Choudhary, and Kaushik Roy. CODE-CL: Conceptor-based gradient projection for deep continual learning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 775–784. IEEE, 2025.

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aleksandr Dekhovich, David MJ Tax, Marcel HF Sluiter, and Miguel A Bessa. Continual prune-and-select: class-incremental learning with specialized subnetworks. *Applied Intelligence*, 53 (14):17849–17864, 2023.

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.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 8342–8360, 2020.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Ying-Hua Huang, Rui Fang, Hsi-Wen Chen, and Ming-Syan Chen. Mask as memory: Streaming multi-task pruning for vision transformers. In Michelangelo Ceci, Dragi Kocev, Anna Monreale, Celine Vens, Peter A. Flach, Giuseppe Manco, Myra Spiliopoulou, and Elio Masciari (eds.), *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2026, Naples, Italy, September 7-11, 2026, Proceedings, Part IV*, volume 16944 of *Lecture Notes in Computer Science*, pp. 391–409. Springer, 2026. doi: 10.1007/978-3-032-37667-1_23. URL https://doi.org/10.1007/978-3-032-37667-1_23.

Herbert Jaeger. Controlling recurrent neural networks by conceptors. *arXiv preprint arXiv:1403.3369*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. Beyond not-forgetting: Continual learning with backward knowledge transfer. *Advances in Neural Information Processing Systems*, 35:16165–16177, 2022a.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. *arXiv preprint arXiv:2202.02931*, 2022b.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Srikanth Malla, Joon Hee Choi, and Chiho Choi. COPAL: Continual pruning in large language generative models. *arXiv preprint arXiv:2405.02347*, 2024.

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.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*, 2023.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured Laplace approximations for overcoming catastrophic forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

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.

Lukas Nicola Tatzel, Bálint Mucsányi, Osane Hackel, and Philipp Hennig. Debiasing mini-batch quadratics for applications in deep learning. In *International Conference on Learning Representations*, volume 2025, pp. 41298–41337, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

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.

Edoardo Urettini and Antonio Carta. Online curvature-aware replay: Leveraging $\mathbf{2^{nd}}$ order information for online continual learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60590–60609. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/urettini25a.html.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023.

Yichen Wu, Long-Kai Huang, Renzhen Wang, Deyu Meng, and Ying Wei. Meta continual learning revisited: Implicitly enhancing online Hessian approximation via variance reduction. In *International Conference on Learning Representations*, volume 2024, pp. 2689–2718, 2024.

Qingsen Yan, Dong Gong, Yuhang Liu, Anton Van Den Hengel, and Javen Qinfeng Shi. Learning Bayesian sparse networks with full experience replay for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 109–118. IEEE, 2022.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhiguo Yang, Changjian Deng, Qinke Chen, Zijing Zhou, and Jian Cheng. LSA: Layer-wise sparsity allocation for large language model pruning based on minimal linear reconstruction error. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 50490–50508, 2024.

A Experiment Details

A.1 Datasets

TRACE. TRACE (Wang et al., 2023) comprises eight tasks covering Chinese, English, German, and Python code. Table 5 gives their content and metrics. Calibration uses each task's training split; evaluation uses the 500-example test split.

Table 5: TRACE tasks.

Dataset	Task content	Metric
C-STANCE	Identify whether Chinese text supports, opposes, or is neutral toward a specified target.	Accuracy
FOMC	Classify monetary-policy statements as dovish, hawkish, or neutral.	Accuracy
MeetingBank	Summarize English meeting transcripts.	ROUGE-L
Py150	Complete Python code from its preceding context.	Edit similarity
ScienceQA	Answer science questions from the supplied question and context.	Accuracy
NumGLUE-cm	Combine commonsense knowledge with arithmetic to produce numerical answers.	Accuracy
NumGLUE-ds	Combine domain-specific knowledge with arithmetic to produce numerical answers.	Accuracy
20Minuten	Simplify German news text while preserving its meaning.	SARI

Long-seq. Table 6 describes the fifteen classification tasks from Razdaibiedina et al. (2023). All tasks use short textual class labels and are scored by accuracy. We sample a fixed subset of up to 500 test examples per task and share it across methods; calibration uses the current task's training split.

Table 6: Long-seq tasks.

Dataset	Task content
Yelp	Classify the sentiment of Yelp reviews into five ratings.
Amazon	Classify the sentiment of product reviews into five ratings.
MNLI	Classify premise–hypothesis pairs as entailment, neutral, or contradiction.
CB	Infer entailment, contradiction, or neutrality for a hypothesis in context.
COPA	Select the more plausible cause or effect from two alternatives.
QQP	Determine whether two questions have the same meaning.
RTE	Determine whether a premise entails a hypothesis.
IMDB	Classify movie reviews as positive or negative.
SST-2	Classify the sentiment of movie-review sentences as positive or negative.
DBpedia	Assign an encyclopedic text to one of fourteen topic categories.
AGNews	Classify news text into four topic categories.
Yahoo	Assign question-and-answer text to one of ten topic categories.
MultiRC	Judge whether a candidate answer is correct given a passage and question.
BoolQ	Answer a yes/no question using a supporting passage.
WiC	Determine whether a word has the same sense in two sentences.

A.2 Implementation Details

A.2.1 Baseline Implementations

Score accumulation. COPAL (Malla et al., 2024) uses its per-tensor mask rule. Accum. Wanda (Sun et al., 2024) sums activation-aware importance scores across tasks and selects the mask from this sum, leaving surviving weights unchanged. For *Full reconstruction* and *Protected reconstruction*, we compute OBS (Hassibi & Stork, 1992) saliencies at the dense weights using the current task's regularized Hessian $H_{t,\lambda}$:

$$\sigma_t(w_{rq}^0) = \frac{(w_{rq}^0)^2}{[H_{t,\lambda}^{-1}]_{qq}}, \qquad S_t = S_{t-1} + \sigma_t, \qquad S_0 = 0.$$

Here w^0_{rq} is the dense weight in row r and column q, and S_t is the matrix of accumulated scores, updated elementwise. Scores are set to zero on input coordinates with zero calibration activation. Both variants select pruning masks from S_t. For unstructured sparsity, the mask removes the lowest-scoring weights in each output row; for 2:4 sparsity, it removes the two lowest-scoring weights in each group of four.

Full reconstruction. The full variant applies the SparseGPT (Frantar & Alistarh, 2023) column sweep with the accumulated-score mask and the current task's damped Hessian. Deletion errors are propagated within each row to columns that have not yet been processed.

Protected reconstruction. After selecting the pruning mask from S_t, we rank only the surviving weights in each linear module by S_{t-1}. The half with the highest historical scores is fixed at its dense values; the lower-scoring half is allowed to change. The number of adjustable weights is rounded down when the survivor count is odd; ties do not increase this count. For one output row, let $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{F}$ index the pruned, adjustable, and protected weights, respectively. Here w^0 contains the dense weights of this row, $\delta = \widehat{w} - w^0$, and subscripts select the corresponding coordinates or matrix blocks. We minimize the damped current-task reconstruction loss $\delta^\top (H_t + \lambda I)\delta$ subject to $\delta_\mathcal{P} = -w^0_\mathcal{P}$ and $\delta_\mathcal{F} = 0$. The adjustable weights are jointly updated by solving

$$\big[(H_t)_{\mathcal{A}\mathcal{A}} + \lambda I\big]\delta_\mathcal{A} = (H_t)_{\mathcal{A}\mathcal{P}} w^0_\mathcal{P}, \qquad \widehat{w}_\mathcal{A} = w^0_\mathcal{A} + \delta_\mathcal{A}.$$

We solve this system by Cholesky factorization, with positive diagonal damping increased if needed for numerical stability. At the first task, no historical scores are available, so the protected variant uses full reconstruction.

Curvature accumulation. ACC.Hessian stores a running average of task Hessians to control numerical scale. Multiplying curvature by a positive scalar rescales all OBS (Hassibi & Stork, 1992) saliencies uniformly and cancels from the weight reconstruction update, so the running average and sum produce the same rankings and reconstructed weights. The CARGO (no gate) control sets $\beta = 1$ and follows the same initialization and cleaning schedule as CARGO.

Reference models. The dense reference uses the unpruned pretrained model. Single-task SparseGPT (Frantar & Alistarh, 2023) prunes a fresh copy of the dense model on each task and evaluates it on that task. These two references have no backward-transfer score.

A.2.2 Implementation and Evaluation Settings

Pruning. Calibration uses 128 sequences of $2{,}048$ tokens drawn from the current task's training split. We prune the attention projections and MLP matrices in every transformer block, while keeping the embeddings and language-model head dense. Methods with curvature memory carry their per-module state across tasks. Beyond the main 50% comparison, we evaluate 30% and 70% unstructured sparsity and 2:4 sparsity. For 2:4 sparsity, each method prunes two weights in every group of four consecutive weights using its own selection criterion. For variants with spectral cleaning, we use $\rho = 0.99$; the gate parameters (β, α) are listed in Appendix A.3. Damping and weight reconstruction are summarized in Appendix D.

Evaluation. We evaluate each task on the test samples described in Appendix A.1. The prompt for each test example includes up to six input–output pairs from the task's training split as in-context demonstrations. We generate responses using sampling with a temperature of 0.1. Generation is capped at 512 tokens on TRACE and 16 tokens on Long-seq. We retain the full task$\times$step score matrix. TRACE accuracy and ROUGE-L are multiplied by 100 before aggregation; edit similarity and SARI are already on the $[0, 100]$ scale. Long-seq accuracy is also expressed as a percentage. We compute ACC and BWT using (12), with $T = 8$ for TRACE and $T = 15$ for Long-seq. For single-task SparseGPT (Frantar & Alistarh, 2023), ACC averages the scores obtained by pruning and evaluating each task independently.

Computation. We use bf16 inference, accumulate Hessians in fp32, and store the carried curvature on the CPU. The curvature state occupies approximately 28 GB for Llama-2-7B; the total

memory requirement sums $O(d^2)$ storage over the linear modules. For Llama-2-7B-chat, the curvature matrices of a transformer block together require about $1.1\times$ the storage of its per-weight score arrays at the same precision. Experiments use single-GPU jobs with 128 GB of host memory.

A.3 Hyperparameters

CARGO retains the full curvature of historical directions whose support meets the threshold α and scales the curvature in the remaining directions by β. We choose $\beta \in \{0.7, 0.8, 0.9, 0.95\}$ and $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$, while fixing $\rho = 0.99$ whenever spectral cleaning is enabled. Table 7 lists the (β, α) settings used for the performance comparisons.

Table 7: Gate parameters (β, α) by model, task stream, and sparsity.

Model	Sparsity	TRACE	Long-seq
Llama-2-7B-chat	30%	$(0.7, 0.5)$	$(0.7, 0.3)$
Llama-2-7B-chat	50%	$(0.9, 0.3)$	$(0.7, 0.3)$
Qwen3-8B	50%	$(0.95, 0.5)$	$(0.9, 0.3)$
Ministral-3-8B-Instruct	50%	$(0.9, 0.5)$	$(0.9, 0.5)$
Llama-2-13B-chat	50%	$(0.7, 0.5)$	$(0.8, 0.7)$
Qwen3-14B	50%	$(0.95, 0.3)$	$(0.9, 0.4)$
Ministral-3-14B-Instruct	50%	$(0.9, 0.5)$	$(0.9, 0.5)$
Llama-2-7B-chat	70%	$(0.7, 0.5)$	$(0.95, 0.5)$
Llama-2-7B-chat	2:4	$(0.9, 0.5)$	$(0.95, 0.7)$
Qwen3-8B	2:4	$(0.8, 0.5)$	$(0.9, 0.4)$
Ministral-3-8B-Instruct	2:4	$(0.8, 0.3)$	$(0.95, 0.3)$

B More Results

B.1 Final Average Accuracy with Error Bars

Table 8 reports the mean and standard deviation of ACC and BWT over three runs with different random seeds for the 7–8B models in Table 1.

Table 8: Results at 50% unstructured sparsity, mean ± standard deviation, with ACC above BWT. The best performance is bold.

Method	Llama-2-7B		Qwen3-8B		Ministral-8B	
ACC↑ / BWT↑	TRACE	Long-seq	TRACE	Long-seq	TRACE	Long-seq
Dense	38.80 ±1.50	65.62 ±1.89	55.66 ±1.14	79.62 ±0.31	53.82 ±1.09	78.56 ±0.79
Single-task SparseGPT	37.13 ±1.46	62.57 ±1.36	50.72 ±0.55	77.43 ±0.34	49.85 ±0.99	77.59 ±0.39
COPAL	30.69 ±2.76	40.82 ±2.76	45.69 ±0.78	47.16 ±2.22	37.45 ±1.24	50.35 ±3.04
	+7.58 ±1.55	+4.21 ±1.54	+7.25 ±1.55	+2.96 ±3.73	+4.81 ±1.10	+4.12 ±3.23
Accum. Wanda	34.53 ±1.13	54.07 ±0.88	44.02 ±0.36	58.37 ±0.09	40.73 ±1.54	66.65 ±0.66
	+0.89 ±1.84	+0.43 ±1.21	+2.15 ±1.43	−1.43 ±0.82	+2.53 ±1.41	+1.31 ±1.07
Full reconstruction	35.45 ±1.93	60.23 ±1.00	46.17 ±1.34	75.16 ±0.65	43.37 ±0.36	75.89 ±0.48
	+0.34 ±1.84	−2.62 ±0.59	−3.07 ±2.36	−2.55 ±0.34	−7.19 ±1.62	−1.38 ±0.66
Protected reconstruction	34.80 ±2.49	59.88 ±0.83	45.41 ±1.26	74.83 ±0.82	44.92 ±2.00	76.65 ±0.78
	−1.64 ±0.86	−2.88 ±1.59	−4.55 ±1.08	−3.36 ±0.50	−7.31 ±1.92	−1.06 ±0.67
CARGO	**37.55** ±1.51	**62.21** ±0.55	**50.99** ±0.40	**77.53** ±0.52	**48.37** ±0.43	**78.03** ±0.30
	−0.17 ±0.52	+1.45 ±0.81	+0.30 ±1.57	+0.09 ±0.09	+0.57 ±1.24	−0.28 ±0.34

B.2 Robustness to the gate setting

We evaluate the sensitivity of CARGO to the decay factor β and support threshold α using Qwen3-8B at 50% sparsity on TRACE and Long-seq. We vary $\beta \in \{0.7, 0.8, 0.9, 0.95\}$ and $\alpha \in \{0.3, 0.4, 0.5, 0.7\}$. As shown in Figure 4, final ACC varies by 2.4 points on TRACE and

1.1 points on Long-seq across the evaluated settings, indicating that CARGO remains stable over the tested range of gate parameters.

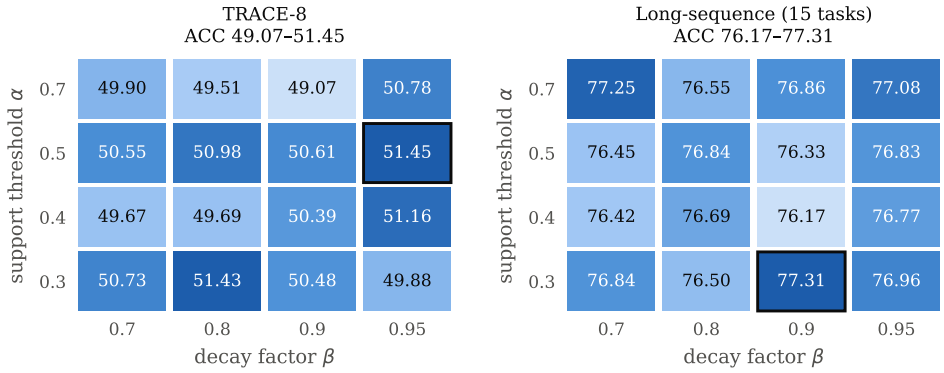

Figure 4: Sensitivity to the gate parameters (β, α) on Qwen3-8B at 50% sparsity. Cells show final ACC; darker cells indicate higher ACC.

B.3 Pruning Cost

Table 9 reports the total runtime of the complete experiment, including pruning and evaluation, and the corresponding final ACC under identical settings for Qwen3-8B on TRACE at 50% sparsity. CARGO achieves 51.45 ACC in 16.0 hours. Compared with COPAL (Malla et al., 2024) and Accum. Wanda (Sun et al., 2024), which do not reconstruct weights, CARGO takes approximately 6.0% and 8.8% longer while improving ACC by 12.6% and 16.9%, respectively. Relative to Full reconstruction, CARGO increases runtime by 2.6% and improves ACC by 10.1%. These relative performance gains justify the modest additional cost of maintaining and adapting curvature across tasks.

Table 9: Total runtime and final ACC on TRACE with Qwen3-8B at 50% sparsity on one L40S.

Method	Time (h)↓	ACC↑
COPAL	15.1	45.69
Accum. Wanda	14.7	44.02
Full reconstruction	15.6	46.71
Protected reconstruction	19.9	45.79
ACC.Hessian	15.6	50.41
CARGO (no gate)	15.7	48.73
CARGO	16.0	**51.45**

C Proof of Proposition 1

Fix the dense weights $W = W^0$ and the recorded layer inputs $X_k \in \mathbb{R}^{d \times n_k}$ for tasks $k = 1, \ldots, t$, where n_k is the number of recorded inputs for task k. The layer loss separates over output rows, so represent one row of W^0 by a column vector $w \in \mathbb{R}^d$ and write its reconstruction update as $\delta = \widehat{w} - w$. For task k, the reconstruction loss is

$$E_k(\widehat{w}) = \left\|(\widehat{w} - w)^\top X_k\right\|_2^2 = \delta^\top X_k X_k^\top \delta. \tag{13}$$

Summing over all seen tasks gives

$$\begin{aligned} \ell_{1:t}(\widehat{w}) &= \sum_{k=1}^{t} \delta^\top X_k X_k^\top \delta \\ &= \delta^\top \Big[\sum_{k=1}^{t} X_k X_k^\top\Big] \delta = \delta^\top H_t^{\text{sum}} \delta. \end{aligned} \tag{14}$$

The accumulated Hessian admits the recurrence

$$H_0^{\text{sum}} = 0, \qquad H_t^{\text{sum}} = \sum_{k=1}^{t} X_k X_k^\top = H_{t-1}^{\text{sum}} + X_t X_t^\top. \tag{15}$$

This recurrence carries the joint reconstruction objective forward using one matrix and the current task's inputs, so previous calibration inputs can be discarded. Since $\ell_{1:t}(w) = 0$, the loss increase is exactly $\delta^\top H_t^{\text{sum}} \delta$, a quadratic expression in $\widehat{w}$ that holds without a Taylor approximation.

Deleting weight w_q requires $\widehat{w}_q = 0$, equivalently $e_q^\top \delta + w_q = 0$. For a fixed candidate q, minimizing the joint reconstruction loss for this row gives the equivalent problems

$$\min_{\widehat{w}:\, \widehat{w}_q = 0} \ell_{1:t}(\widehat{w}) = \min_{\delta:\, e_q^\top \delta + w_q = 0} \delta^\top H_t^{\text{sum}} \delta. \tag{16}$$

Introduce a scalar Lagrange multiplier ν:

$$\mathcal{J}(\delta, \nu) = \delta^\top H_t^{\text{sum}} \delta + \nu(e_q^\top \delta + w_q). \tag{17}$$

Differentiating with respect to δ gives

$$2H_t^{\text{sum}} \delta + \nu e_q = 0, \qquad \delta = -\frac{\nu}{2}(H_t^{\text{sum}})^{-1} e_q. \tag{18}$$

Substituting into the deletion constraint yields

$$-\frac{\nu}{2}\left[(H_t^{\text{sum}})^{-1}\right]_{qq} + w_q = 0, \qquad \nu = \frac{2w_q}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}}. \tag{19}$$

Substituting this value of ν into (18) gives the optimal reconstruction update:

$$\delta^\star = -\frac{w_q}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}} (H_t^{\text{sum}})^{-1} e_q. \tag{20}$$

Under the invertibility assumption, the positive-semidefinite matrix H_t^{sum} is positive definite, making $\delta^\star$ the unique minimizer under the deletion constraint. The reconstructed row is

$$\widehat{w}^\star = w + \delta^\star = w - \frac{w_q}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}} (H_t^{\text{sum}})^{-1} e_q.$$

Construct $\widehat{W}^\star$ by replacing the selected row of W^0 with $(\widehat{w}^\star)^\top$ and retaining all other rows at their dense values. Those rows incur zero reconstruction loss, so $\widehat{W}^\star$ minimizes the layer loss under the same deletion constraint, with

$$\begin{aligned} \mathcal{L}_{1:t}(\widehat{W}^\star) = \mathcal{L}_{1:t}^{\min}(q) &= (\delta^\star)^\top H_t^{\text{sum}} \delta^\star \\ &= \frac{w_q^2}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}^2} e_q^\top (H_t^{\text{sum}})^{-1} e_q \\ &= \frac{w_q^2}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}}. \end{aligned} \tag{21}$$

The reconstructed weights and the minimum loss in (21) establish (5) in Proposition 1. Minimizing this loss over q selects the next weight to prune. The update adjusts weights throughout the same output row: for any coordinate j,

$$\delta_j^\star = -\frac{w_q}{\left[(H_t^{\text{sum}})^{-1}\right]_{qq}} \left[(H_t^{\text{sum}})^{-1}\right]_{jq}. \tag{22}$$

This update uses the full inverse accumulated Hessian, including its off-diagonal entries. □

D ALGORITHM

Algorithm 1 summarizes the update and pruning of one layer at task t. The carried state is the pair $(\mathbf{M}_{t-1}, t-1)$: one matrix per module and a scalar step count. Its storage is independent of stream length. We use $\varepsilon = 10^{-12}$ in the support ratio (8). In the algorithm, X_t collects the layer inputs obtained from calibration data $\mathcal{D}_t$, and EIGH denotes symmetric eigendecomposition.

Algorithm 1 CARGO: one layer, one task t

Require: dense W^0; carried $(\mathbf{M}_{t-1},\ t-1)$; sparsity s; parameters ρ, α, β

collect $H_t = X_t X_t^\top$ from the layer's calibration inputs
if $t = 1$ **then**
  $\mathbf{M}_1 \leftarrow H_1/\bar{d}(H_1)$ if $\bar{d}(H_1) > 0$, else 0
else
  $C_t \leftarrow$ clean and normalize H_t (Section 4.2)
  $(\lambda, U) \leftarrow \text{EIGH}(\mathbf{M}_{t-1})$; $c_i \leftarrow u_i^\top C_t u_i$
  $m_i \leftarrow c_i/(c_i + \lambda_i/(t-1) + \varepsilon)$
  $g_i \leftarrow 1$ if $m_i \geq \alpha$ else β
  $\mathbf{M}_t \leftarrow U\,\mathrm{diag}(g \odot \lambda)U^\top + C_t$
end if
reset layer to W^0; prune to sparsity s using the importance score and weight reconstruction in (11) with the regularized $\mathbf{M}_t$
carry $(\mathbf{M}_t,\ t)$ to the next task

At each task, the model starts from W^0 and processes transformer blocks in order. Current calibration inputs provide H_t for each linear module; CARGO updates $\mathbf{M}_t$ and prunes the module using this curvature. The pruned block then supplies the calibration inputs for the next block.

Before inversion, we follow SparseGPT (Frantar & Alistarh, 2023) to handle zero-diagonal coordinates and add positive diagonal damping to $\mathbf{M}_t$. This makes the matrix positive definite, and we compute its inverse through Cholesky factorization, increasing the damping if needed. We use this curvature for weight selection and reconstruct unprocessed weights in the same row as each column is pruned.