GaLore-based Continuous Learning from Conversational Data: Balancing Adaptation and Retention

Anonymous Authors

Abstract

We present a memory-efficient framework for continuous learning of large language models from streaming conversational data using GaLore (Gradient Low-Rank Projection). Our approach enables online adaptation without catastrophic forgetting by projecting gradients to a low-rank subspace via SVD, reducing optimizer memory by 36% compared to full AdamW while maintaining 80% of training throughput. Surprisingly, we find that GaLore's gradient projection acts as implicit regularization: full AdamW exhibits the worst knowledge retention (42.42% MMLU) due to rapid overfitting, while GaLore preserves significantly more knowledge. Through systematic 2^3 factorial Design of Experiments analysis on Qwen2.5-0.5B, we identify that GaLore rank is the dominant hyperparameter controlling the forgetting-learning trade-off. Cross-model validation spanning four architectures from 0.5B to 8B parameters establishes a scaling rule: rank $\propto 1/\sqrt{\text{params}}$, with rank=64 for $\leq$1.1B models, rank=32 for 2–3B, and rank=8 for 8B models. Extended training at 8B scale demonstrates net positive learning: Qwen3-8B with rank=8 trained for 1000 steps achieves +0.14% MMLU improvement over baseline while reducing holdout perplexity by 8.8%—proving that the model genuinely learns from conversational data without forgetting. Critically, we show that LoRA fails at continuous learning despite superior memory efficiency (22GB vs 39GB VRAM): LoRA's frozen base weights cause catastrophic distribution shift (+530% perplexity), while GaLore's full-weight updates with implicit regularization enable genuine knowledge integration. These findings establish GaLore as the preferred approach for memory-constrained continuous learning where both efficiency and learning quality are required.

1 Introduction

1.1 Motivation

Large language models (LLMs) are typically static after pretraining, representing a snapshot of knowledge at the time of training. However, real-world applications require continuous adaptation to new data and domains. The fundamental challenge lies in balancing the acquisition of new information against the preservation of previously learned knowledge—a problem known as catastrophic forgetting.

This work addresses a core research question: *"Can a system learn over time by using conversations with humans as a new dataset?"* We investigate whether memory-efficient training methods can enable continuous learning from streaming conversational data while maintaining model capabilities.

1.2 Contributions

This work makes several contributions to the field of memory-efficient continuous learning. We develop a framework that leverages GaLore's gradient projection for adapting language models to streaming conversational data, demonstrating that memory efficiency and implicit regularization emerge jointly from the same mechanism. Through systematic experimental analysis using a 2^3 factorial design, we establish that GaLore rank is the dominant hyperparameter controlling the trade-off between knowledge acquisition and retention, with lower ranks providing stronger regularization against catastrophic forgetting. Our cross-model validation spans four models across three distinct transformer architectures—Qwen (Alibaba), Llama (Meta), and Gemma (Google)—ranging from 0.5B to 8B parameters and showing that the approach generalizes without architecture-specific tuning. From these experiments, we derive a practical scaling rule relating GaLore rank inversely to model size: rank 64 proves sufficient for models up to 1.1B parameters, while 2B+ models require rank 32 to maintain acceptable forgetting rates. The complete implementation is released for reproducibility.

2 Related Work

2.1 Continual Learning for Language Models

Continual learning addresses the challenge of adapting models to new data while preserving previously acquired knowledge. Recent surveys provide comprehensive coverage of this problem in the context of large language models [Shi et al., 2024, Wu et al., 2024], identifying three primary scenarios: continual pretraining on new corpora, continual instruction tuning on new tasks, and continual alignment with evolving human preferences. Our work focuses on the first scenario, where models must integrate knowledge from streaming conversational data.

The seminal "Don't Stop Pretraining" work by Gururangan et al. [2020] demonstrated that continued pretraining on domain-specific corpora consistently improves downstream task performance, establishing that static pretrained models leave significant adaptation potential untapped. Ke et al. [2023] extended this finding by developing strategies for continual pretraining that explicitly balance knowledge acquisition against retention. Gupta et al. [2023] investigated learning rate strategies for continual pretraining, finding that re-warming the learning rate can improve adaptation—though our experiments suggest this benefit diminishes at shorter training scales. These works motivate our investigation: if domain adaptation benefits from continued training, can models similarly benefit from continuous exposure to human conversations?

2.2 Memory-Efficient Training Methods

Full-parameter training of large language models requires substantial memory for optimizer states, which store momentum and variance estimates for each parameter. GaLore [Zhao et al., 2024a] addresses this bottleneck through gradient low-rank projection: rather than maintaining optimizer states for the full gradient matrix, GaLore projects gradients into a low-rank subspace via SVD and maintains states only for the projected representation. This approach reduces optimizer memory by up to 65% while preserving the ability to make full-rank weight updates, distinguishing it fundamentally from parameter-efficient methods.

LoRA [Hu et al., 2022] takes an orthogonal approach by freezing pretrained weights entirely and injecting trainable low-rank matrices. While highly effective for task adaptation, LoRA's architectural constraint—that weight updates must lie in a fixed low-rank subspace—limits its applicability to scenarios requiring genuine weight modification. Recent work by Biderman et al. [2024] demonstrates that LoRA exhibits reduced plasticity compared to full fine-tuning, learning less new information while also forgetting less—a trade-off that proves problematic for continuous learning where knowledge acquisition is the primary goal. QLoRA [Dettmers et al., 2023] extends LoRA with 4-bit quantization for further memory reduction. Alternative memory-efficient approaches include LISA [Pan et al., 2024], which randomly freezes layers during training, and BAdam [Luo et al., 2024], which applies block-coordinate descent to achieve full-parameter updates with reduced memory. Hao et al. [2024] provide theoretical analysis showing that LoRA can be viewed as a form of gradient compression, offering insight into the relationship between adapter-based and gradient-projection methods. For continuous learning from conversational data, where the model must integrate new knowledge throughout its weight matrices rather than merely adapt surface behaviors, GaLore's full-rank update capability offers a theoretical advantage we empirically validate.

2.3 Catastrophic Forgetting

Catastrophic forgetting—the tendency of neural networks to abruptly lose previously learned knowledge when trained on new data—was first formalized by McCloskey & Cohen [1989] and subsequently reviewed by French [1999]. Goodfellow et al. [2014] provided empirical characterization showing that forgetting correlates with the dissimilarity between old and new task distributions. This phenomenon poses a fundamental challenge for continuous learning systems.

Several mitigation strategies have been developed. Elastic Weight Consolidation (EWC) [Kirkpatrick et al., 2017] penalizes changes to weights deemed important for previous tasks, estimated via the Fisher information matrix. Experience replay methods [Rolnick et al., 2019] maintain a buffer of previous examples to interleave with new training data; LAMOL [Sun et al., 2020] extends this to language models by using the model itself to generate pseudo-replay samples. These approaches require either explicit importance estimation or memory overhead for replay buffers.

Recent work has specifically analyzed forgetting in LLMs. Luo et al. [2023] demonstrated that forgetting during continual fine-tuning follows predictable patterns related to learning rate and training duration. Kotha et al. [2024] provided theoretical grounding through implicit inference analysis, showing that forgetting can be understood as the model shifting its implicit prior over tasks. Parameter-efficient methods have also been adapted for continual learning: O-LoRA [Wang et al., 2023] maintains orthogonality between task-specific adapters to prevent interference, achieving state-of-the-art results on continual learning benchmarks. Our work contributes to this literature by demonstrating that GaLore's gradient projection provides implicit regularization against forgetting without requiring explicit mitigation mechanisms or task-specific adapters.

2.4 Conversational Datasets

Large-scale conversational datasets enable training and evaluation of dialogue systems. LMSYS-Chat-1M [Zheng et al., 2023] provides one million real-world conversations collected from the Chatbot Arena platform, spanning 154 languages and interactions with 25 different LLMs. Its scale and diversity make it well-suited for continuous learning experiments. OpenAssistant [Köpf et al., 2023] offers 161K human-annotated messages organized in tree-structured conversations, with explicit quality ratings. WildChat [Zhao et al., 2024b] contains one million ChatGPT interaction logs with rich metadata including user demographics and conversation topics.

We select LMSYS-Chat-1M as our primary training corpus due to its scale, linguistic diversity, and representation of authentic human-LLM interactions across multiple model architectures.

2.5 Evaluation Benchmarks

Measuring catastrophic forgetting requires benchmarks that assess retained capabilities. MMLU [Hendrycks et al., 2021] evaluates multitask language understanding across 57 subjects spanning STEM, humanities, and social sciences, making it sensitive to broad capability degradation. We adopt MMLU as our primary forgetting metric, using 5-shot evaluation to assess whether continuous learning preserves the general knowledge acquired during pretraining.

The lm-evaluation-harness [Gao et al., 2024] provides a unified framework for reproducible LLM evaluation, which we use for all benchmark assessments. Perplexity on held-out conversational data complements MMLU by measuring learning efficacy: decreasing holdout perplexity indicates successful adaptation to the conversational domain.

3 Method

3.1 Problem Formulation

The continuous learning framework presented in this work addresses the challenge of adapting large language models to streaming conversational data under memory constraints. We formulate this as an online semi-supervised learning problem in which the model receives two concurrent data streams: an unlabeled stream of conversational text for next-token prediction, and a labeled stream of instruction-response pairs for supervised fine-tuning.

At each training step, a batch of unlabeled examples is drawn from the conversational corpus, and the model computes a standard language modeling loss over next-token predictions. Periodically, labeled examples consisting of user prompts paired with assistant responses are sampled, and a supervised fine-tuning loss is computed with prompt tokens masked from the loss calculation to focus learning on response generation. The total objective combines both losses:

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda \cdot \mathcal{L}_{\mathrm{SFT}} \tag{1}$$

where λ controls the relative contribution of instruction tuning to the overall training signal.

3.2 GaLore Optimizer

Central to our approach is the GaLore (Gradient Low-Rank Projection) optimizer, which enables full-parameter training on consumer hardware by exploiting the low-rank structure of gradient matrices. For each two-dimensional parameter tensor (*i.e.*, linear layer weights), the optimizer maintains a projector matrix P obtained via truncated singular value decomposition of the gradient G.

Specifically, we compute P using randomized SVD to extract the top-r left singular vectors. The full gradient is then projected into this compact subspace:

$$R = P^T G \tag{2}$$

yielding a reduced representation of shape $r \times n$ rather than $m \times n$. The Adam optimizer moments (first and second moment estimates) are maintained exclusively for these projected gradients, reducing memory requirements from $\mathcal{O}(mn)$ to $\mathcal{O}(rn + rm)$ per parameter matrix.

After computing the Adam update in the projected space, the parameter update is recovered via back-projection:

$$\Delta W = P \cdot \mathrm{update}(R) \tag{3}$$

The projection matrix is recomputed periodically (every 50 steps by default) to track the evolving gradient geometry, while intermediate steps reuse the cached projector. Parameters with fewer than 128 elements in their smallest dimension, as well as one-dimensional parameters such as biases and layer norms, receive standard AdamW updates without projection.

3.3 Training Procedure

Training proceeds in a streaming fashion with gradient accumulation to simulate larger effective batch sizes. Gradients are accumulated over multiple forward-backward passes before being clipped to a maximum norm of 1.0 and applied via an optimizer step. This combination of gradient accumulation and clipping stabilizes training on noisy conversational data while maintaining memory efficiency.

An optional cosine learning rate schedule with linear warmup may be applied, though our empirical results suggest this provides minimal benefit at shorter training horizons (Section 5).

3.4 Evaluation Framework

Model evaluation encompasses both learning efficacy and catastrophic forgetting. To measure learning, we compute held-out perplexity on four evaluation sets: a deterministically sampled holdout partition of LMSYS-Chat-1M (in-distribution), OpenAssistant conversations (out-of-distribution due to differing annotation conventions), ShareGPT transcripts (cross-model generalization), and C4 web text (pretraining distribution baseline).

Forgetting is assessed through the MMLU benchmark using 5-shot evaluation, which measures retention of world knowledge and reasoning capabilities acquired during pretraining. Together, these metrics quantify the trade-off between acquiring new conversational abilities and preserving general-purpose language understanding.

4 Experiments

4.1 Experimental Setup

All experiments use Qwen2.5-0.5B-Instruct [Qwen Team, 2024a] as the primary base model, trained on LMSYS-Chat-1M [Zheng et al., 2023] conversational data. Training runs for 300 steps with sequence length 512 and batch size 4 on a single NVIDIA A100-80GB GPU. This configuration enables rapid experimental iteration while remaining representative of practical single-GPU deployment scenarios.

4.2 Phase 1: Initial Validation

The initial experiments (exp001–002) compared OpenAssistant and LMSYS-Chat-1M as conversation sources for the unlabeled training stream. LMSYS-Chat-1M provided substantially better convergence, with supervised loss reaching 0.81 compared to 1.45 for OpenAssistant. We attribute this difference to LMSYS's cleaner single-turn format and more consistent conversation structure, which reduces noise in the training signal.

4.3 Phase 2: Overfitting Detection

Extended training to 500 steps (exp003) revealed clear signs of overfitting. MMLU accuracy dropped from the pretrained baseline of 48.26% to 43.45%, representing a degradation of 4.8 percentage points. Simultaneously, holdout perplexity increased by 11–22% across all four evaluation datasets, despite training loss continuing to decrease. This divergence between training and evaluation metrics—where the model improves on in-distribution training data while degrading on held-out evaluations—motivated systematic hyperparameter analysis to identify configurations that mitigate overfitting.

4.4 Phase 3: Design of Experiments

We conducted a 2^3 factorial experiment to systematically evaluate three hyperparameters suspected of influencing the forgetting-learning trade-off. The first factor (f1) reduced GaLore rank from the default 128 to 64, hypothesizing that lower rank would constrain gradient updates and reduce overfitting. The second factor (f2) introduced learning rate warmup over 30 steps followed by cosine decay, a common practice in longer training runs. The third factor (f3) increased weight decay from 0.1 to 0.2, providing explicit L2 regularization.

All eight configurations spanning the full factorial design were trained for 300 steps and evaluated on MMLU and holdout perplexity across four datasets. The complete results are presented in Section 5, with main effects analysis identifying which factors most strongly influence outcomes.

4.5 Phase 4: Cross-Model Validation

To validate that our findings generalize beyond a single architecture, we applied the optimal configuration from Phase 3 to two additional model families.

TinyLlama-1.1B [Zhang et al., 2024], representing the Llama architecture family, demonstrated remarkable stability under continuous learning with rank 64. MMLU accuracy declined by only 0.02 percentage points, from 24.96% to 24.94%, effectively zero within measurement precision. This result confirms that the rank-64 configuration successfully prevents catastrophic forgetting for models in the 1B parameter range.

Gemma-2-2B [Gemma Team, 2024], representing Google's architecture, presented an instructive failure case with rank 64. MMLU dropped from 56.69% to 51.02%, a decline of 5.67 percentage points that exceeds our 5% acceptability threshold. We hypothesized that the larger model's increased parameter count provides more degrees of freedom for overfitting, requiring stronger regularization. Reducing the rank to 32 confirmed this hypothesis: MMLU improved to 52.84%, reducing degradation to 3.85 percentage points and bringing performance within acceptable bounds.

4.6 Phase 5: 8B Scale Validation

To validate the scaling rule at a practically important model size, we conducted experiments on Qwen3-8B [Qwen Team, 2025], an 8.2 billion parameter model. Based on the pattern observed at smaller scales, we hypothesized that 8B models would require even lower rank to prevent forgetting, testing rank values of 32, 16, and 8.

All three configurations achieved acceptable forgetting rates, with rank 8 emerging as optimal. At rank 32, MMLU declined from the baseline of 74.93% to 74.23%, a drop of 0.70 percentage points. Rank 16 performed similarly at 74.21% (0.72% drop). Rank 8 achieved the best retention at 74.49%, reducing MMLU degradation to just 0.44 percentage points. This counterintuitive result—that lower rank yields less forgetting—reinforces our finding that GaLore's gradient projection provides implicit regularization proportional to the rank reduction.

Memory efficiency at 8B scale proved particularly compelling. All GaLore configurations consumed approximately 39GB of peak VRAM, compared to an estimated 80GB for full AdamW optimization on the same model. Training throughput ranged from 463 to 494 tokens per second on A100 GPUs. These results demonstrate that GaLore enables continuous learning on 8B models using single-GPU configurations that would otherwise require multi-GPU setups or model parallelism.

4.7 Phase 5.2: Extended Training

To validate that the minimal forgetting observed in Phase 5 persists under extended training, we continued training Qwen3-8B with rank 8 for 1000 steps—more than three times the initial 300-step runs. This extended experiment tests whether the model can continue learning without eventually succumbing to catastrophic forgetting.

The results exceeded expectations. MMLU accuracy improved from the pretrained baseline of 74.93% to 75.07%, a gain of 0.14 percentage points. Simultaneously, holdout perplexity on LMSYS conversations decreased from 5.34 to 4.87, representing an 8.8% improvement in language modeling on held-out conversational data. This dual improvement—better benchmark performance and lower held-out perplexity—provides strong evidence that the model is genuinely learning from the conversational data stream while retaining, and even slightly enhancing, its pretrained knowledge.

Training completed in 5 hours 40 minutes at 775 tokens per second, with peak VRAM consumption remaining at 38.73GB. The stability of memory usage across both 300-step and 1000-step runs confirms that GaLore's memory efficiency scales to longer training horizons without accumulating overhead.

4.8 Phase 5.3: GaLore versus LoRA Comparison

A natural question arises: how does GaLore compare to LoRA [Hu et al., 2022], the dominant parameter-efficient fine-tuning method? To address this, we trained Qwen3-8B with LoRA rank 8 for 1000 steps using identical data and hyperparameters, enabling direct comparison of learning quality and efficiency.

The efficiency comparison favored LoRA substantially. LoRA consumed only 22.07GB of peak VRAM compared to GaLore's 38.73GB—a 43% reduction—and achieved 27% higher throughput (982 versus 775 tokens per second). Both methods reached similar final training losses, suggesting comparable optimization on the training distribution.

However, the learning quality comparison revealed a critical distinction. While GaLore achieved the 8.8% holdout perplexity improvement described above (5.34 to 4.87), LoRA exhibited catastrophic failure: holdout perplexity exploded from 5.34 to 33.68, a 530% increase. Despite converging on training data, the LoRA-trained model completely failed to generalize to held-out conversations.

This failure stems from LoRA's architectural design. By freezing base model weights and only updating low-rank adapter matrices, LoRA can adapt surface-level behaviors but cannot integrate new knowledge into the model's core representations. The adapters learn to "patch" outputs for the training distribution, but this patching does not transfer to held-out samples from the same domain. In contrast, GaLore projects gradients through a low-rank bottleneck but still updates all model weights, enabling genuine knowledge integration throughout the parameter space.

5 Results

We present our experimental findings in three parts: memory and throughput characterization comparing GaLore against baseline methods, factorial analysis identifying optimal hyperparameters, and cross-model validation demonstrating generalization across architectures.

5.1 Memory Efficiency and Training Throughput

To establish the practical viability of GaLore for continuous learning, we conducted controlled experiments comparing four training methods on Qwen2.5-0.5B over 100 steps using identical LMSYS-Chat-1M data. Full AdamW optimization, which maintains first and second moment estimates for all parameters, consumed 10.05 GB of peak VRAM and achieved a throughput of 1,425 tokens per second. GaLore with rank 128 reduced memory consumption to 6.74 GB, representing a 33% reduction, while GaLore with rank 64 achieved 6.43 GB, a 36% reduction from the full AdamW baseline. Both GaLore configurations exhibited throughput of approximately 1,135 tokens per second, corresponding to 80% of AdamW's speed due to the computational overhead of periodic SVD recomputation for gradient projection. For comparison, LoRA with rank 64 achieved the lowest memory footprint at 5.26 GB (48% reduction) and the highest throughput at 2,718 tokens per second, though this method differs fundamentally from GaLore in that it freezes the base model weights and only trains low-rank adapter matrices.

A surprising finding emerged from the MMLU evaluation following these 100-step training runs. Full AdamW achieved the lowest MMLU score of 42.42%, representing the greatest degradation from the pretrained baseline of 48.26%. In contrast, GaLore rank 128 achieved 45.22% and GaLore rank 64 achieved 45.73%, while LoRA rank 64 obtained the highest score of 46.78%. This counterintuitive result reveals that GaLore's gradient projection acts as implicit regularization during training. By constraining weight updates to lie within a low-rank subspace,

Table 1: Memory and throughput comparison across training methods (Qwen2.5-0.5B, 100 steps).

Method	Peak VRAM	Tokens/sec	MMLU	vs AdamW
Full AdamW	10.05 GB	1,425	42.42%	baseline
GaLore r=128	6.74 GB	1,137	45.22%	−33% mem
GaLore r=64	6.43 GB	1,135	45.73%	−36% mem
LoRA r=64	5.26 GB	2,718	46.78%	−48% mem

GaLore limits the model's capacity to memorize training samples, thereby preventing the rapid overfitting that occurs with full-parameter optimization.

5.2 Factorial Analysis of Hyperparameters

Having established GaLore's memory efficiency, we sought to identify optimal hyperparameters through a systematic 2^3 factorial design of experiments. Three factors were evaluated: f1, reducing GaLore rank from 128 to 64; f2, adding learning rate warmup over 30 steps followed by cosine decay; and f3, increasing weight decay from 0.1 to 0.2. All eight configurations were trained for 300 steps and evaluated on MMLU and holdout perplexity across four datasets.

Main effects analysis revealed that GaLore rank reduction (f1) was the dominant factor, yielding an average improvement of 1.1 percentage points on MMLU and reducing LMSYS holdout perplexity by 0.7 points. Every configuration incorporating rank 64 achieved MMLU scores between 45.17% and 45.30%, compared to 43.90% to 44.28% for rank 128 configurations. The learning rate scheduler (f2) proved detrimental at this training scale, decreasing MMLU by 0.2 percentage points on average and increasing perplexity, likely because warmup and decay are calibrated for longer training runs. Weight decay at 0.2 (f3) showed negligible effect. The optimal configuration was exp003_1, which applied only rank reduction and achieved 45.26% MMLU with an LMSYS holdout perplexity of 5.57, actually improving upon the pretrained baseline perplexity of 5.60 while limiting MMLU degradation to 3.0 percentage points.

5.3 Cross-Model Validation

To validate that our findings generalize beyond Qwen2.5-0.5B, we applied the rank 64 configuration to two additional architectures: TinyLlama-1.1B representing the Llama family, and Gemma-2-2B representing Google's architecture. TinyLlama demonstrated remarkable stability under continuous learning, with MMLU declining by only 0.02 percentage points from 24.96% to 24.94%, essentially zero forgetting within measurement precision.

Gemma-2-2B presented a different challenge. With rank 64, MMLU dropped from 56.69% to 51.02%, a decline of 5.67 percentage points that exceeded our 5% acceptability threshold. We hypothesized that larger models, having more degrees of freedom, require stronger regularization. Reducing the rank to 32 confirmed this hypothesis: MMLU improved to 52.84%, reducing the degradation to 3.85 percentage points, now within acceptable bounds.

Extending these experiments to 8B scale revealed that the inverse relationship between model size and optimal rank continues. Qwen3-8B achieved its best retention with rank 8, reducing MMLU degradation to just 0.44 percentage points while consuming only 39GB of VRAM—approximately half that of full AdamW optimization. The complete scaling rule that emerges from our experiments follows an approximate rank $\propto 1/\sqrt{\text{params}}$ relationship: rank 64 for models up to 1.1B, rank 32 for 2–3B models, and rank 8 for 8B models. This pattern suggests that larger models require proportionally stronger regularization through gradient projection to prevent overfitting during continuous learning.

Table 2: Cross-model validation results. Models trained on LMSYS-Chat-1M with rank scaled by model size.

Model	**Size**	**Rank**	**Steps**	**MMLU Change**	**VRAM**	**Status**
Qwen2.5-0.5B	0.5B	64	300	−3.0%	6.4 GB	Pass
TinyLlama-1.1B	1.1B	64	300	−0.02%	–	Best
Gemma-2-2B	2.6B	32	300	−3.85%	–	Pass
Qwen3-8B	8.2B	8	300	−0.44%	38.7 GB	Pass
Qwen3-8B	8.2B	8	1000	**+0.14%**	38.7 GB	**Best**

The most compelling result emerged from extended training at 8B scale. When training Qwen3-8B with rank 8 for 1000 steps, MMLU accuracy actually improved by 0.14 percentage points over the pretrained baseline (from 74.93% to 75.07%), while holdout perplexity on LMSYS conversations decreased by 8.8% (from 5.34 to 4.87). This dual improvement demonstrates that with appropriate rank scaling, continuous learning can enhance rather than degrade model capabilities.

5.4 GaLore versus LoRA for Continuous Learning

To contextualize GaLore's effectiveness, we conducted a direct comparison with LoRA at 8B scale. Both methods were trained on Qwen3-8B for 1000 steps with rank 8, enabling fair comparison of efficiency and learning quality.

Table 3: GaLore versus LoRA comparison on Qwen3-8B (1000 steps, rank=8).

Method	**Peak VRAM**	**Tokens/sec**	**Post-train PPL**	**PPL Change**
GaLore r=8	38.73 GB	775	**4.87**	**−8.8%**
LoRA r=8	**22.07 GB**	**982**	33.68	+530%

LoRA achieved substantially better efficiency—43% lower memory and 27% higher throughput—due to its design of freezing base weights and training only lightweight adapter matrices. However, this efficiency came at a catastrophic cost to learning quality. While GaLore reduced holdout perplexity by 8.8%, LoRA increased it by 530%, indicating complete failure to generalize to held-out conversational data despite converging on the training set.

This result has significant implications for continuous learning system design. LoRA's adapter-based approach, while highly effective for task-specific fine-tuning where the base model's knowledge is sufficient, fundamentally cannot support genuine continuous learning. The frozen base weights prevent integration of new knowledge into the model's core representations, causing the adapters to learn distribution-specific patches that fail to transfer. GaLore's approach of projecting gradients while updating all weights enables the knowledge integration necessary for continuous learning.

6 Discussion

Our experimental findings reveal that GaLore's gradient low-rank projection serves a dual purpose beyond memory efficiency: it provides an implicit regularization mechanism that proves critical for preserving pretrained knowledge during continuous learning. The most striking evidence for this claim emerges from our memory comparison experiment, where full AdamW optimization achieved the worst MMLU performance (42.42%) despite having access to full-rank gradient updates, while GaLore with rank-64 projection preserved significantly more knowledge (45.73%). This counterintuitive result suggests that constraining gradient updates to a low-dimensional subspace limits the model's capacity to memorize individual training samples, forcing it instead to learn more generalizable representations. The low-rank projection effectively acts as a capacity bottleneck, conceptually similar to dropout or noise injection, preventing the optimization trajectory from navigating toward sharp minima that overfit the training distribution. When gradients are projected through SVD to rank-r space, only the r most significant directions of change are preserved, discarding the numerous small but collectively harmful updates that would otherwise accumulate into memorization of training-specific patterns.

The relationship between model scale and optimal GaLore rank emerges as a practical finding with clear theoretical underpinnings. Larger models possess more parameters and thus more degrees of freedom through which overfitting can manifest. Our experiments demonstrate this directly: Gemma-2-2B exhibited unacceptable forgetting (5.67% MMLU drop) with rank-64 that was entirely acceptable for the smaller Qwen2.5-0.5B and TinyLlama-1.1B models. Reducing the rank to 32 brought Gemma-2-2B within acceptable bounds (3.85% MMLU drop). At 8B scale, Qwen3-8B required rank-8 to achieve optimal retention (0.44% MMLU drop), with higher ranks yielding slightly worse performance. This yields a practical scaling rule following an approximate rank $\propto 1/\sqrt{\text{params}}$ relationship: rank-64 for models up to 1.1B, rank-32 for 2–3B models, and rank-8 for 8B models. The pattern suggests that regularization strength must scale with model capacity to maintain the forgetting-learning balance.

The comparison between GaLore and LoRA reveals a fundamental distinction in their suitability for continuous learning. Our 8B-scale experiments provide definitive evidence: while LoRA achieved 43% lower memory usage (22GB versus 39GB) and 27% higher throughput (982 versus 775 tokens per second), it completely failed at continuous learning—holdout perplexity exploded by 530% despite convergence on training data. In contrast, GaLore achieved an 8.8% perplexity improvement on held-out conversations while simultaneously improving MMLU by 0.14%.

This dramatic difference stems from the architectural designs of these methods. LoRA freezes all base model weights and trains only low-rank adapter matrices that are added to selected layers. This design excels at task adaptation, where the pretrained model already contains the necessary knowledge and adapters merely steer its behavior. However, for continuous learning where new knowledge must be integrated into the model's representations, LoRA's frozen weights become a fatal limitation. The adapters learn to produce appropriate outputs for the training distribution by "patching" the frozen model's behavior, but this patching is highly distribution-specific and fails to generalize to held-out samples from the same domain.

GaLore, by contrast, projects gradients through a low-rank bottleneck but still updates all model weights. This means new information can be distributed throughout the model's parameter space rather than being confined to adapter matrices. The low-rank projection provides implicit regularization that prevents overfitting, while full-weight updates enable genuine knowledge integration. This combination—regularized updates to all parameters—appears to be the key enabler of continuous learning.

The practical implication is clear: LoRA should be preferred for task-specific fine-tuning where efficiency is paramount and the base model's knowledge is sufficient, while GaLore should be preferred for continuous learning scenarios where genuine knowledge acquisition is required. The computational overhead of GaLore (roughly 75% more memory and 20% slower) is justified when the alternative is complete failure to learn.

One of the more encouraging aspects of our results is the architecture-agnostic nature of the approach. Without any model-specific hyperparameter tuning beyond rank selection, we achieved acceptable forgetting rates across four models spanning three transformer architecture families: Qwen (Alibaba), Llama (Meta), and Gemma (Google), with model sizes ranging from 0.5B to 8.2B parameters. The SVD-based gradient projection operates on the linear layer weight matrices that are fundamentally similar across all modern transformer implementations, suggesting that our findings should transfer to other architectures as well. The remarkably low forgetting observed in both TinyLlama-1.1B (0.02% MMLU drop) and Qwen3-8B (0.44% drop) indicates that with appropriate rank scaling, the approach can achieve near-zero knowledge degradation across a wide range of model sizes.

Several limitations constrain the generalizability of our conclusions. While our extended 1000-step experiment at 8B scale demonstrates that the approach remains effective beyond short training windows, significantly longer training runs (tens of thousands of steps) may reveal different dynamics between rank settings and forgetting rates. Behavior on even larger models (70B+) and the potential need for rank values below 8 remain unexplored. Additionally, our experiments did not incorporate replay buffers or other explicit forgetting mitigation strategies, which could potentially allow higher ranks (and thus more expressive updates) while maintaining knowledge retention. Future work should address these gaps, including very long training runs with periodic checkpointing, multi-GPU scaling for models beyond 8B, and dynamic rank adaptation based on online forgetting metrics.

7 Conclusion

We demonstrate that GaLore-based continuous learning enables LLMs to adapt to streaming conversational data without catastrophic forgetting—and under extended training, with net positive knowledge transfer. Our systematic DOE analysis reveals that GaLore rank is the dominant hyperparameter controlling the forgetting-learning trade-off, with lower rank providing stronger implicit regularization against overfitting. Cross-model validation spanning four models from 0.5B to 8.2B parameters establishes a scaling rule: rank $\propto 1/\sqrt{\text{params}}$, with rank-64 for models up to 1.1B, rank-32 for 2–3B, and rank-8 for 8B models.

Extended training at 8B scale provides the strongest evidence for our claims. Qwen3-8B with rank-8 trained for 1000 steps achieved +0.14% MMLU improvement over baseline while reducing holdout perplexity by 8.8%—demonstrating that the model genuinely learns from conversational data while retaining, and even enhancing, its pretrained knowledge. This result transcends mere forgetting mitigation: it shows that continuous learning can improve model capabilities.

Critically, we establish that GaLore is uniquely suited to continuous learning among memory-efficient methods. LoRA, despite superior efficiency (43% less memory, 27% faster), catastrophically fails at continuous learning with +530% perplexity increase on held-out data. This failure stems from LoRA's frozen base weights, which prevent genuine knowledge integration. GaLore's design—projecting gradients through a low-rank bottleneck while updating all model weights—provides the implicit regularization necessary for continuous learning while enabling full-weight updates that integrate new knowledge throughout the model.

These findings answer our research question affirmatively: **yes, a system can learn over time by using conversations with humans as training data**. GaLore with appropriately scaled rank enables genuine continuous learning where other memory-efficient methods fail.

References

Biderman, D., Portes, J., Gonzalez Ortiz, J. J., et al. (2024). LoRA Learns Less and Forgets Less. arXiv:2405.09673.

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *NeurIPS 2023*. arXiv:2305.14314.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4), 128–135.

Gao, L., Tow, J., et al. (2024). A framework for few-shot language model evaluation. *Zenodo*. doi:10.5281/zenodo.10256836.

Gemma Team & Google DeepMind. (2024). Gemma 2: Improving Open Language Models at a Practical Size. arXiv:2408.00118.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2014). An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv:1312.6211.

Gupta, K., Tay, Y., et al. (2023). Continual Pre-Training of Large Language Models: How to (re)warm your model? arXiv:2308.04014.

Gururangan, S., Marasović, A., et al. (2020). Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *ACL 2020*. arXiv:2004.10964.

Hao, Y., Cao, Y., & Mou, L. (2024). Flora: Low-Rank Adapters Are Secretly Gradient Compressors. arXiv:2402.03293.

Hendrycks, D., Burns, C., et al. (2021). Measuring Massive Multitask Language Understanding. *ICLR 2021*. arXiv:2009.03300.

Hu, E. J., Shen, Y., et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models. *ICLR 2022*. arXiv:2106.09685.

Ke, Z., Shao, Y., Lin, H., et al. (2023). Continual Pre-training of Language Models. *ICLR 2023*. arXiv:2302.03241.

Kirkpatrick, J., et al. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13), 3521–3526. arXiv:1612.00796.

Köpf, A., Kilcher, Y., et al. (2023). OpenAssistant Conversations – Democratizing Large Language Model Alignment. arXiv:2304.07327.

Kotha, S., Springer, J. M., & Raghunathan, A. (2024). Understanding Catastrophic Forgetting in Language Models via Implicit Inference. *ICLR 2024*. arXiv:2309.10105.

Luo, Y., Yang, Z., et al. (2023). An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. arXiv:2308.08747.

Luo, Q., Yu, H., & Li, X. (2024). BAdam: A Memory Efficient Full Parameter Optimization Method for Large Language Models. arXiv:2404.02827.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24, 109–165.

Pan, R., Liu, X., Diao, S., et al. (2024). LISA: Layerwise Importance Sampling for Memory-Efficient Large Language Model Fine-Tuning. arXiv:2403.17919.

Qwen Team. (2024). Qwen2.5 Technical Report. Alibaba Group. arXiv:2412.15115.

Qwen Team. (2025). Qwen3 Technical Report. Alibaba Group. arXiv:2505.09388.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., & Wayne, G. (2019). Experience Replay for Continual Learning. *NeurIPS 2019*. arXiv:1811.11682.

Sun, F.-K., Ho, C.-H., & Lee, H.-Y. (2020). LAMOL: LAnguage MOdeling for Lifelong Language Learning. *ICLR 2020*. arXiv:1909.03329.

Shi, H., Xu, Z., Wang, H., et al. (2024). Continual Learning of Large Language Models: A Comprehensive Survey. arXiv:2404.16789.

Wang, X., Chen, T., Ge, Q., et al. (2023). Orthogonal Subspace Learning for Language Model Continual Learning. *EMNLP 2023 Findings*. arXiv:2310.14152.

Wu, T., Luo, L., et al. (2024). Continual Learning for Large Language Models: A Survey. arXiv:2402.01364.

Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). TinyLlama: An Open-Source Small Language Model. arXiv:2401.02385.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., & Tian, Y. (2024). GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. *ICML 2024*. arXiv:2403.03507.

Zhao, W., Ren, X., et al. (2024). WildChat: 1M ChatGPT Interaction Logs in the Wild. arXiv:2405.01470.

Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). LMSYS-Chat-1M: A Large-Scale Real-World LLM Conversation Dataset. arXiv:2309.11998.

A Hyperparameter Configurations

Default configuration for experiments:

```
model_name: Qwen/Qwen2.5-0.5B-Instruct
training:
  steps: 300
  lr: 1e-4
  weight_decay: 0.1
  grad_accum_steps: 4
  max_grad_norm: 1.0
galore:
  rank: 64  # 32 for 2-3B, 8 for 8B models
  update_proj_gap: 200
  scale: 1.0
data:
  seq_len: 512
  batch_size: 4
```

B Full DOE Results Table

C Cross-Model Results

Table 4: Complete 2^3 factorial DOE results. f1=rank64, f2=scheduler, f3=wd0.2. Baseline MMLU: 48.26%, LMSYS PPL: 5.60.

Exp	**f1**	**f2**	**f3**	**MMLU**	**LMSYS**	**OA**	**ShareGPT**	**C4**
3.0	–	–	–	44.15%	5.82	7.52	6.56	24.40
3.1	✓	–	–	**45.26%**	**5.57**	**6.98**	**6.20**	**23.40**
3.2	–	✓	–	44.17%	6.54	7.41	6.54	24.41
3.3	–	–	✓	44.28%	5.84	7.53	6.57	24.40
3.4	✓	✓	–	45.27%	6.00	7.03	6.22	23.44
3.5	✓	–	✓	45.30%	6.50	6.97	6.21	23.39
3.6	–	✓	✓	43.90%	6.68	7.42	6.54	24.36
3.7	✓	✓	✓	45.17%	6.44	7.01	6.22	23.44

Table 5: Complete cross-model validation results with holdout perplexity and memory usage.

Model	**Rank**	**Steps**	**Base**	**Post**	**Δ**	**LMSYS PPL**	**VRAM**
Qwen2.5-0.5B	64	300	48.26%	45.26%	−3.0%	5.57	6.4 GB
TinyLlama-1.1B	64	300	24.96%	24.94%	−0.02%	4.53	–
Gemma-2-2B	32	300	56.69%	52.84%	−3.85%	5.03	–
Qwen3-8B	8	300	74.93%	74.49%	−0.44%	–	38.7 GB
Qwen3-8B	8	1000	74.93%	**75.07%**	**+0.14%**	**4.87**	38.7 GB

Table 6: GaLore vs LoRA comparison at 8B scale (Qwen3-8B, 1000 steps, rank=8).

Method	**VRAM**	**Tok/s**	**Base PPL**	**Post PPL**	**Δ PPL**	**MMLU**
GaLore r=8	38.73 GB	775	5.34	**4.87**	**−8.8%**	**75.07%**
LoRA r=8	**22.07 GB**	**982**	5.34	33.68	+530%	–