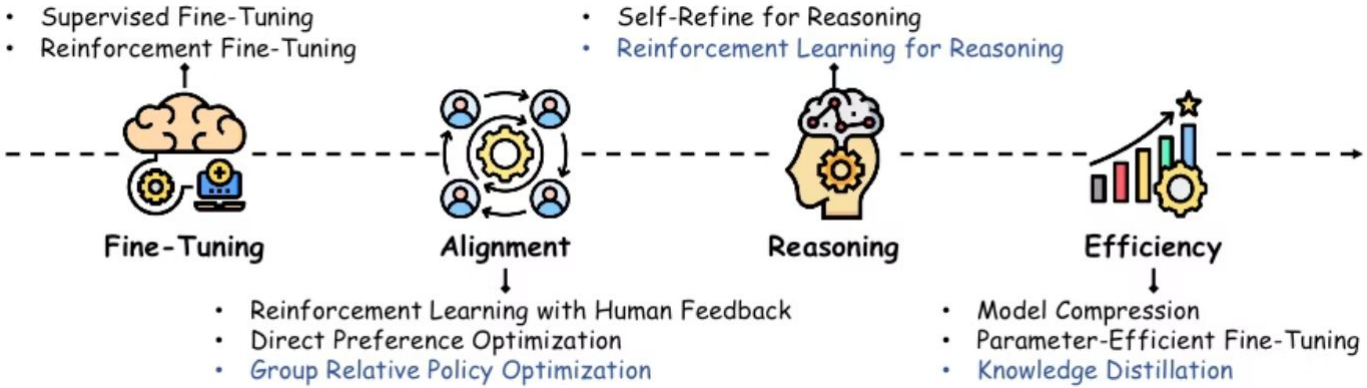
Sculpting Subspaces: Constrained Full Fine-Tuning in LLMs for Continual Learning

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, Akash Srivastava

Paper: https://openreview.net/forum?id=vQcyqsGJDw
Code: https://github.com/Red-Hat-AI-Innovation-Team/orthogonal-subspace-learning

Current Post-Training Regime

After pre-training, large language models embark on a transformative path, transitioning from initial fine-tuning with examples to honing reasoning abilities through reward-driven learning. This iterative approach continuously enhances the model's performance across varying datasets and objectives.



Sub-Optimal Sequential Training

The standard sequential SFT → RLHF/DPO approach is sub-optimal, as each stage induces forgetting of previously learned capabilities and knowledge.

The Problem of Catastrophic Forgetting in LLMs

Addressing this challenge is crucial for building robust, capable, and continually learning language models.

Knowledge Degradation

Catastrophic Forgetting causes degradation of prior capabilities during new learning phases (SFT, RLHF/DPO), creating a significant challenge in the LLM lifecycle.

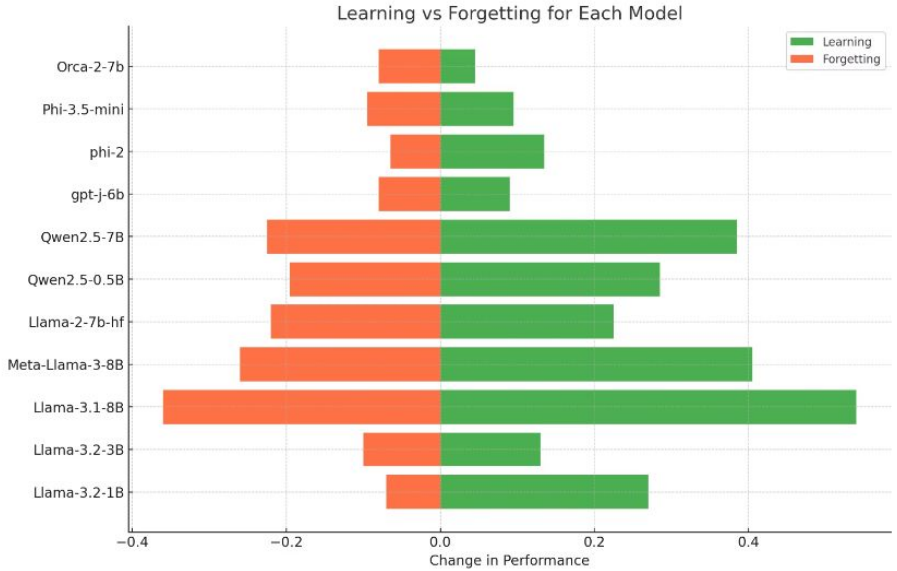
Scale Intensifies Forgetting

CF can intensify with increasing model scale (1B to 7B parameters), making it more challenging to maintain comprehensive knowledge as models grow.

Resource Wastage

Retraining large LLMs to recover lost knowledge can cost millions of dollars in compute resources.

Quantifiable Impact of Catastrophic Forgetting



Haque, N. (2025). Catastrophic Forgetting in LLMs: A Comparative Analysis Across Language Tasks. *arXiv preprint* arXiv:2504.01241.

Revisiting Catastrophic Forgetting in Large Language Model Tuning

Mitigating Forgetting in LLM Supervised Fine-Tuning and Preference Learning

An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning

Interpretable Catastrophic Forgetting of Large Language Model Fine-tuning via Instruction Vector

Extensive existing research demonstrates significant knowledge degradation across various fine-tuning stages, with models losing up to 40% of previously acquired capabilities. These findings highlight the severity of catastrophic forgetting as a critical challenge in LLM development.

Methods to Mitigate Catastrophic Forgetting in LLMs

Key approaches for preserving knowledge during continuous learning phases:

Regularization & Memory-Based Methods

Techniques that constrain weight updates or store previous examples to maintain stability while incorporating new knowledge.

Parameter-Efficient & Subspace Approaches

Methods that update only specific parameter subsets or project updates into constrained subspaces to preserve critical knowledge.

Model Fusion & Modular Architectures

Strategies that combine multiple specialized models or utilize modular components to compartmentalize knowledge and prevent interference.

Each approach offers unique tradeoffs between performance, memory requirements, and computational efficiency in addressing the catastrophic forgetting challenge.

Regularization & Memory-Based Methods

Replay Buffers

Stores old task data; revisits it during training on new tasks

✔ **Advantage:** Reinforces prior knowledge

⚠ **Limitation:** Needs large storage, increases compute, may raise privacy concerns, and access to past fine-tuning or pre-training data is not always available.

Gradient Projections (EWC, GEM)

Identifies important parameters and penalizes changes

✔ **Advantage:** Prevents overwriting of critical representations

⚠ **Limitation:** Requires clear task boundaries, effectiveness varies

Parameter-Efficient & Subspace Approaches

LoRA (Low-Rank Adaptation)

Injects trainable matrices, freezing base model weights

✔ **Advantage:** Lightweight and forgets less due to training of less number of parameters

⚠ **Limitation:** May limit expressivity or add inference overhead

OLoRA (Orthogonal LoRA)

Ensures LoRA updates in a subspace orthogonal to past tasks

✔ **Advantage:** Minimizes interference with previously learned knowledge

⚠ **Limitation:** Can restrict learning capacity for new tasks

Model Fusion & Modular Architectures

Model Merging

Combine weights from separately fine-tuned models to unify multiple capabilities without retraining.

✔ **Advantage:** Retains diverse task knowledge

⚠ **Limitation:** Conflicting representations can cause instability

Mixture of Experts (MoE)

Use sparse expert routing where different subsets of the model specialize in different tasks or domains.

✔ **Advantage:** Mitigates forgetting by isolating learning

⚠ **Limitation:** Increased model size and routing complexity

What If We Could Have It All?

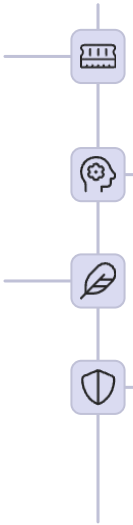
Imagine an approach to fine-tuning that delivers optimal performance with these game-changing advantages:

Minimal Catastrophic Forgetting

Retain critical knowledge from previous tasks while flawlessly integrating new capabilities

Parameter Efficiency

Maintain computational efficiency without additional parameters or increased memory footprint



Uncompromised Expressivity

Match or exceed the learning potential of traditional full fine-tuning methods

No need of Historical Data

Enable continuous learning without relying on historical data or compromising privacy standards

The Solution

Constrained Full Fine-tuning

Revisiting LoRA: Can We Do Better Without Extra Parameters?

- LoRA performs low-rank updates via new trainable matrices.

$$\Delta \mathbf{W} = \mathbf{AB}, \quad \text{where } \mathbf{A} \in \mathbb{R}^{d \times r},\ \mathbf{B} \in \mathbb{R}^{r \times k},\ r \ll \min(d, k)$$

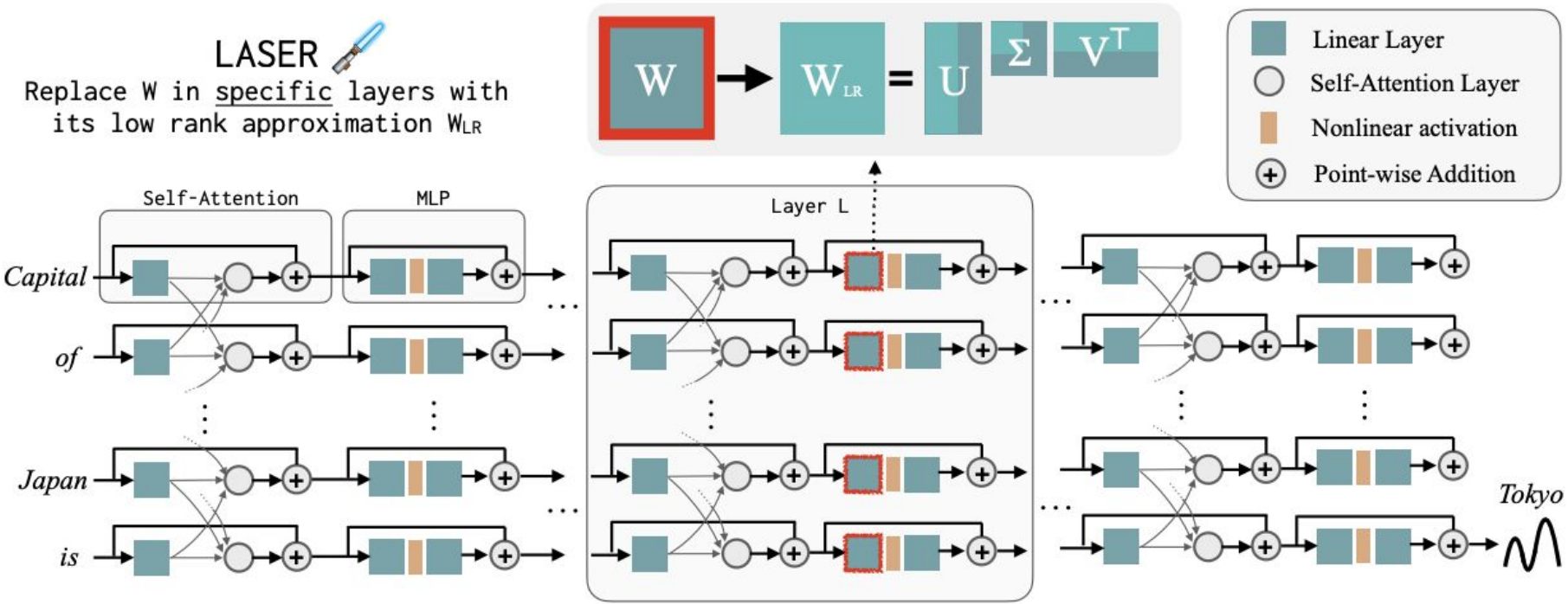
- It reduces training cost but still introduces task-specific parameters.
- We ask: can we achieve low-rank updates **within** the existing parameter space?

Insights from LAyer-SElective Rank reduction (LASER)

- Singular Value Decomposition splits weight matrices into components ordered by singular values.
- Lower-order components (large singular values): encode important information.
- Higher-order components (small singular values): represent noise or redundant information.
- LASER discards higher-order components, giving strong performance even without fine-tuning.

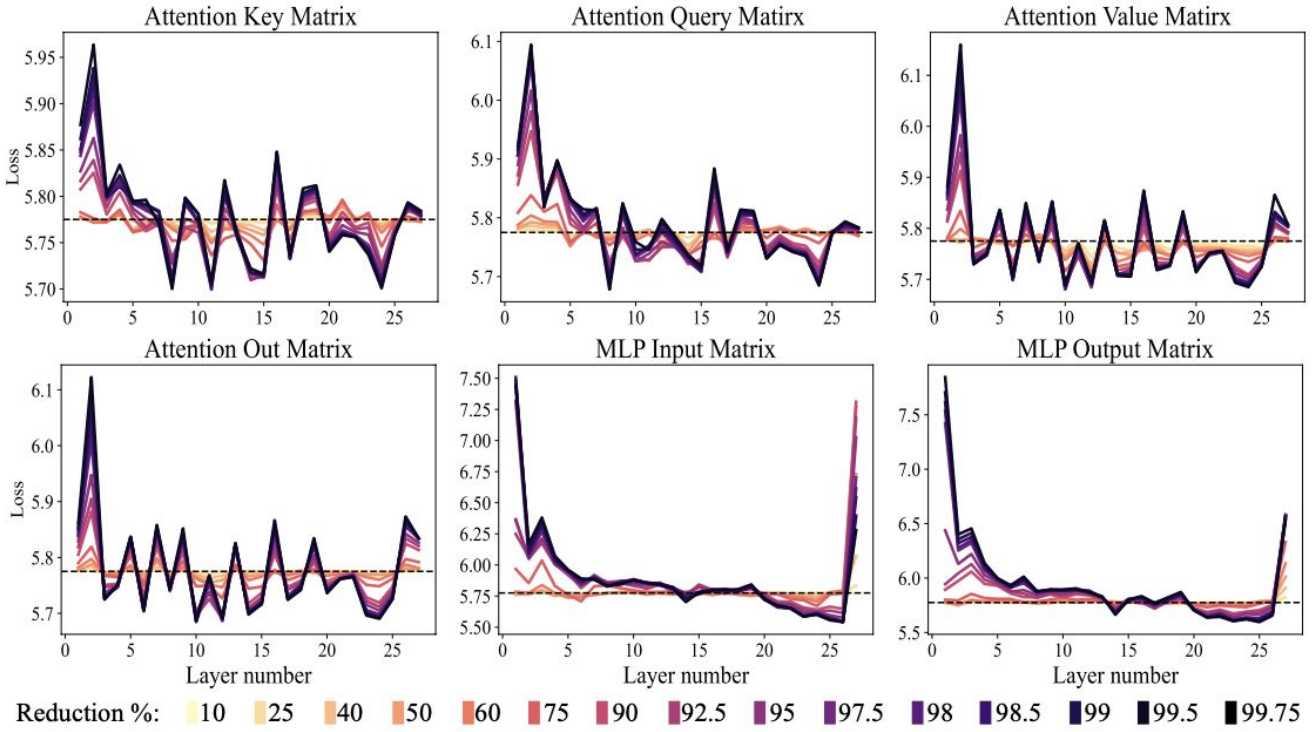
LASER: Sharma et al., "The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction", 2023.

Insights from LAyer-SElective Rank reduction (LASER)



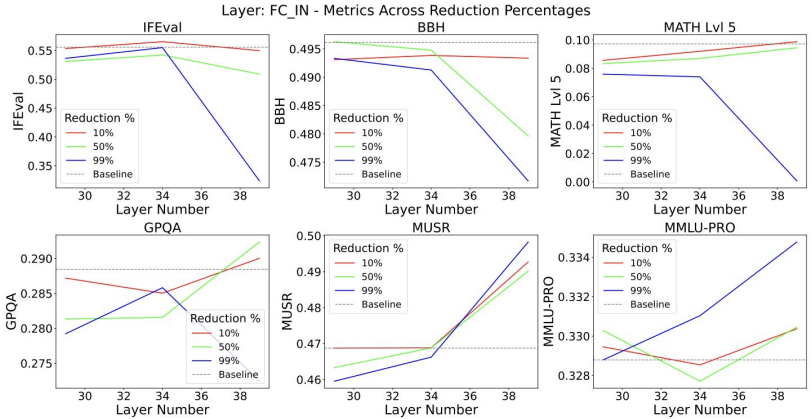
LASER: Sharma et al., "The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction", 2023.

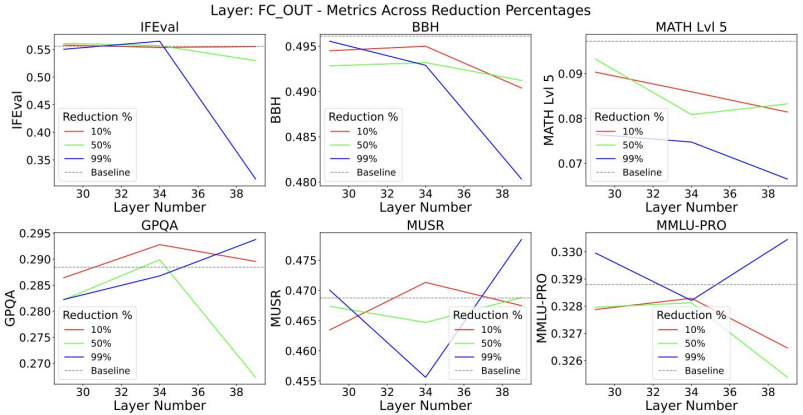
Insights from LAyer-SElective Rank reduction (LASER)

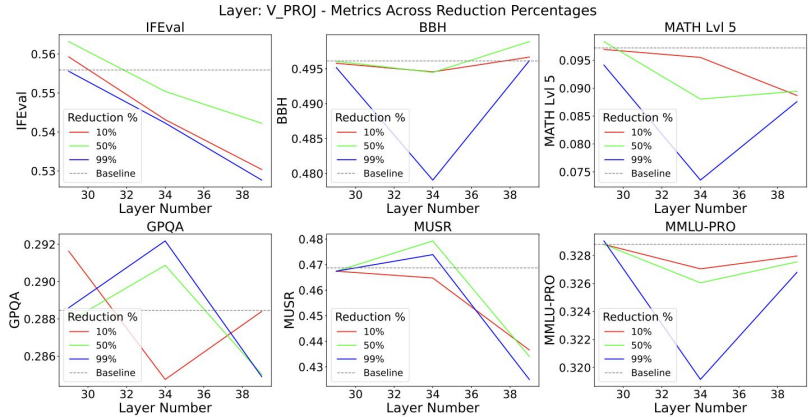


LASER: Sharma et al., "The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction", 2023.

Validating Low-Rank Approximation on Granite 8B







Reduction Percentage	Above Baseline	Below Baseline
10%	3	9
50%	4	8
90%	2	10
99%	2	10
99.75%	0	11

Insights from SPECTRUM: Fine-Tuning High-SNR Layers

- Uses Random Matrix Theory (RMT) to separate signal (large singular values) from noise (small singular values).
- Computes Signal-to-Noise Ratio (SNR) for each layer to guide selective fine-tuning:

$$\text{SNR} = \frac{\sum_{k|\sigma_k \geq \epsilon} \sigma_k}{\sum_{k|\sigma_k < \epsilon} \sigma_k}$$

  ϵ is a threshold separating signal from noise based on singular value magnitudes.

- Retrains only top 25% layers with highest SNR, achieving parameter efficient fine-tuning.

Hartford, Eric et al., "Spectrum: Targeted Training on Signal to Noise Ratio", 2024.

Insights from SPECTRUM: Memory Savings and Training Time

Table 1: Distributed Training

Model	Peak Memory Usage per GPU	% Efficiency Compared to FFT
Llama-3-8b-FFT	24.92 GB	Baseline
Llama-3-8b-QLoRA	21.25 GB	14.73%
Llama-3-8b-Spectrum-50	20.50 GB	17.72%
Llama-3-8b-Spectrum-25	19.18 GB	23.05%
Llama-3-8b-Spectrum-25+QLoRA	16.95 GB	31.99%

Model	Single GPU VRAM Usage
Llama-3-8b-FFT	N/A (out of memory)
Llama-3-8b-Spectrum-50	34.65 GB
Llama-3-8b-Spectrum-25	27.46 GB
Llama-3-8b-QLoRA	23.39 GB
Llama-3-8b-Spectrum-25+QLoRA	21.18 GB

Table 2: Single GPU VRAM Usage

Model	Training Time (8xL40S)
Llama-3-8b-FFT	1h 43m 16s
Llama-3-8b-Spectrum-50	1h 27m 17s
Llama-3-8b-QLoRA	1h 18m 14s
Llama-3-8b-Spectrum-25	1h 5m 33s
Llama-3-8b-Spectrum-25+QLoRA	54m 55s

Table 3: Training Time

Hartford, Eric et al., "Spectrum: Targeted Training on Signal to Noise Ratio", 2024.

Methodology: Problem Setup

- Model parameters: weight matrices $\theta = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$.
- Each layer $\mathbf{W}^{(l)} \in \mathbb{R}^{d_O^{(l)} \times d_I^{(l)}}$ may contain millions of parameters.
- Tasks arrive sequentially: $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$.
- Goal: **learn each task** $\mathcal{D}_t$ while preserving performance on past tasks $\mathcal{D}_{<t}$.

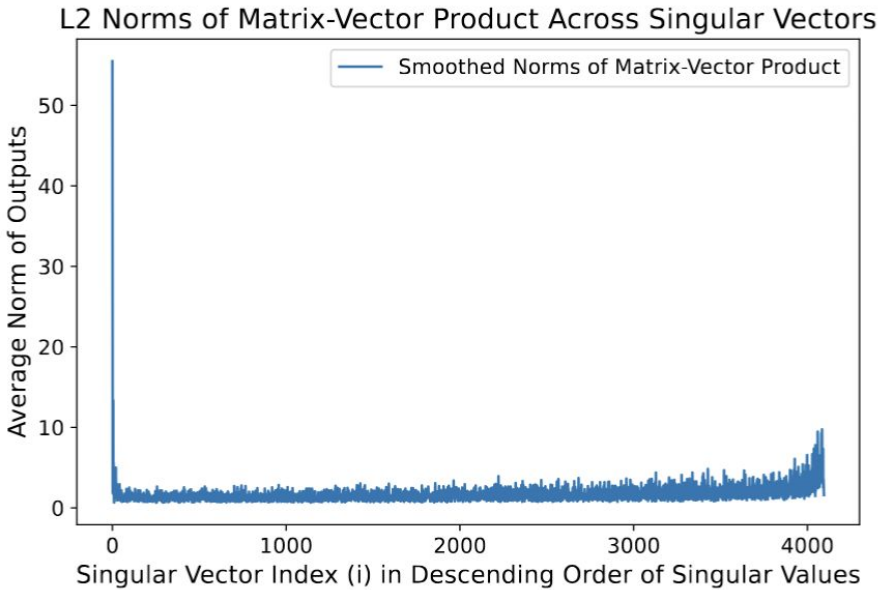
Methodology: Low-Rank and High-Rank Subspaces via SVD

We decompose each weight matrix using Singular Value Decomposition once per task.

$$\mathbf{W}^{(l)} = \mathbf{U}^{(l)}\Sigma^{(l)}(\mathbf{V}^{(l)})^{\top}$$

Core Assumption:

- Data resides in a subspace spanned by vectors corresponding to high singular values.
- Neural network weights are redundant - small singular values have minimal impact.
- Large singular values encode critical knowledge to be preserved.



Methodology: Determining Layer Importance via Input–Output Similarity

- Inspired by **AdaSVD***, we quantify a layer's importance by measuring the **cosine similarity** between it's input activations and it's linear outputs.
- The **importance score** for each layer is computed over a subset of data samples from the previous task:

$$I^{(l)} = \frac{1}{N} \sum_{i=1}^{N} \text{cosine\_similarity}(\mathbf{X}_i^{(l)}, \mathbf{Y}_i^{(l)})$$

- **High similarity** implies the layer is preserving input directions rather than transforming activation representations, which suggests that it encodes **task-general features** that should be retained.
- Scores are **normalized** across layers to maintain relative importance: $\frac{1}{L} \sum_{l=1}^{L} I^{(l)} = 1$

* Li, Zhiteng, et al., "AdaSVD: Adaptive Singular Value Decomposition for Large Language Models.", 2025.

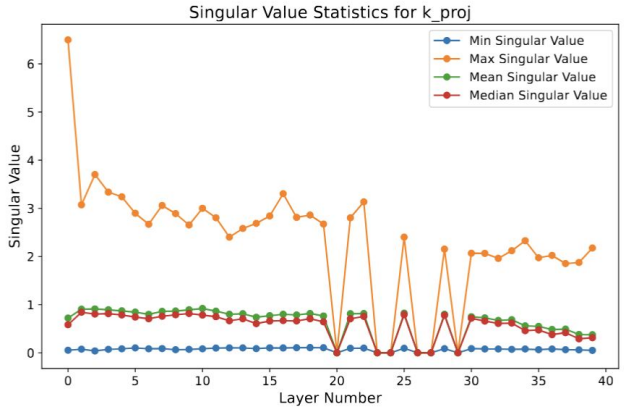
Methodology: Adaptive Effective Rank Selection

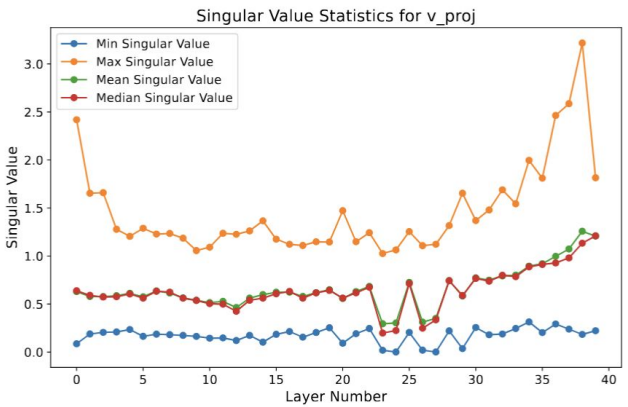
- Layer importance score guides how many singular vectors to retain per layer.
- Two hyperparameters control this:
  - **Minimum Retention Ratio (mrr):** minimal essential retention even for the least critical layers.
  - **Target Retention Ratio (trr):** the upper retention bound for highly critical layers.
- The **retention ratio** is computed as: $r^{(l)} = \mathrm{mrr} + I^{(l)}(\mathrm{trr} - \mathrm{mrr})$
- Based on **retention ratio**, singular vectors are split into **high-rank** and **low-rank** subspaces for selective adaptation.

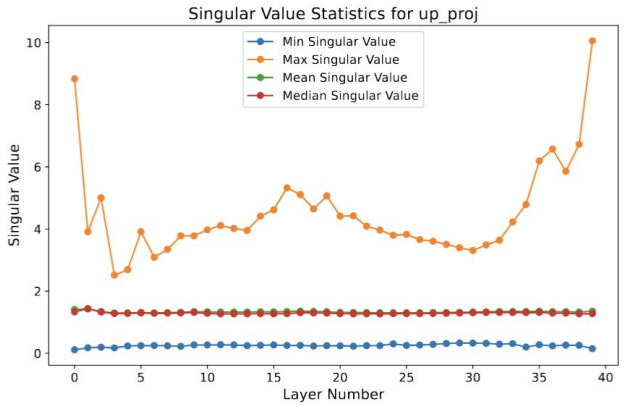
Li, Zhiteng, et al., "AdaSVD: Adaptive Singular Value Decomposition for Large Language Models.", 2025.

Methodology: Alternative Effective Rank Approximation Methods

- **LASER's** fixed-rank strategy fails to reflect layer-wise variability, resulting in suboptimal retention–adaptation trade-offs.
- **SPECTRUM's** random-matrix thresholding (using Marchenko–Pastur distribution) is unstable under sequential tasks with diverse distributions.
- **Shannon's entropy** of matrix elements to compute effective rank.
- Allocating a **predetermined budget** based on number of tasks in the sequence, works surprisingly well in practice.







Methodology: Orthogonal Gradient Updates in Low-Rank Subspace

- We use **projected gradient descent** to constrain updates within the low-rank subspace.
- Updates are orthogonal to high-rank directions to prevent overwriting critical knowledge.
- Projection is computed as:

$$\nabla \mathbf{W}_{\text{proj}}^{(l)} = \nabla \mathbf{W}^{(l)} - \mathbf{U}_{\text{high}}^{(l)} (\mathbf{U}_{\text{high}}^{(l)})^\top \nabla \mathbf{W}^{(l)} \mathbf{V}_{\text{high}}^{(l)} (\mathbf{V}_{\text{high}}^{(l)})^\top$$

- This promotes **knowledge retention** while allowing effective task-specific adaptation.

Methodology: Algorithm Summary

Algorithm 1 Adaptive Low-Rank Continual Learning via SVD

Require: Initial parameters $\theta = \{\mathbf{W}^{(l)}\}_{l=1}^{L}$, tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, hyperparameters mrr, trr.

for task $t = 1, \ldots, T$ **do**

  Compute importance $I^{(l)}$ from layer activations (Eq. (2)); normalize across layers.

  for layer $l = 1, \ldots, L$ **do**

    Compute SVD: $\mathbf{W}^{(l)} = \mathbf{U}^{(l)}\Sigma^{(l)}(\mathbf{V}^{(l)})^{\top}$.

    Retain top $r^{(l)} = \text{mrr} + I^{(l)}(\text{trr} - \text{mrr})$ singular vectors.

  end for

  while not converged on task $\mathcal{D}_t$ **do**

    Sample mini-batch, compute loss $\mathcal{L}_t(\theta)$, gradients $\nabla\mathbf{W}^{(l)}$.

    Project gradients onto low-rank subspace via:

$$\nabla\mathbf{W}_{\text{proj}}^{(l)} = \nabla\mathbf{W}^{(l)} - \mathbf{U}_{\text{high}}^{(l)}(\mathbf{U}_{\text{high}}^{(l)})^{\top}\nabla\mathbf{W}^{(l)}\mathbf{V}_{\text{high}}^{(l)}(\mathbf{V}_{\text{high}}^{(l)})^{\top}$$

    Update parameters with projected gradients.

  end while

end for

Ensure: Parameters θ updated continually without significant forgetting.

Illustrative Diagram

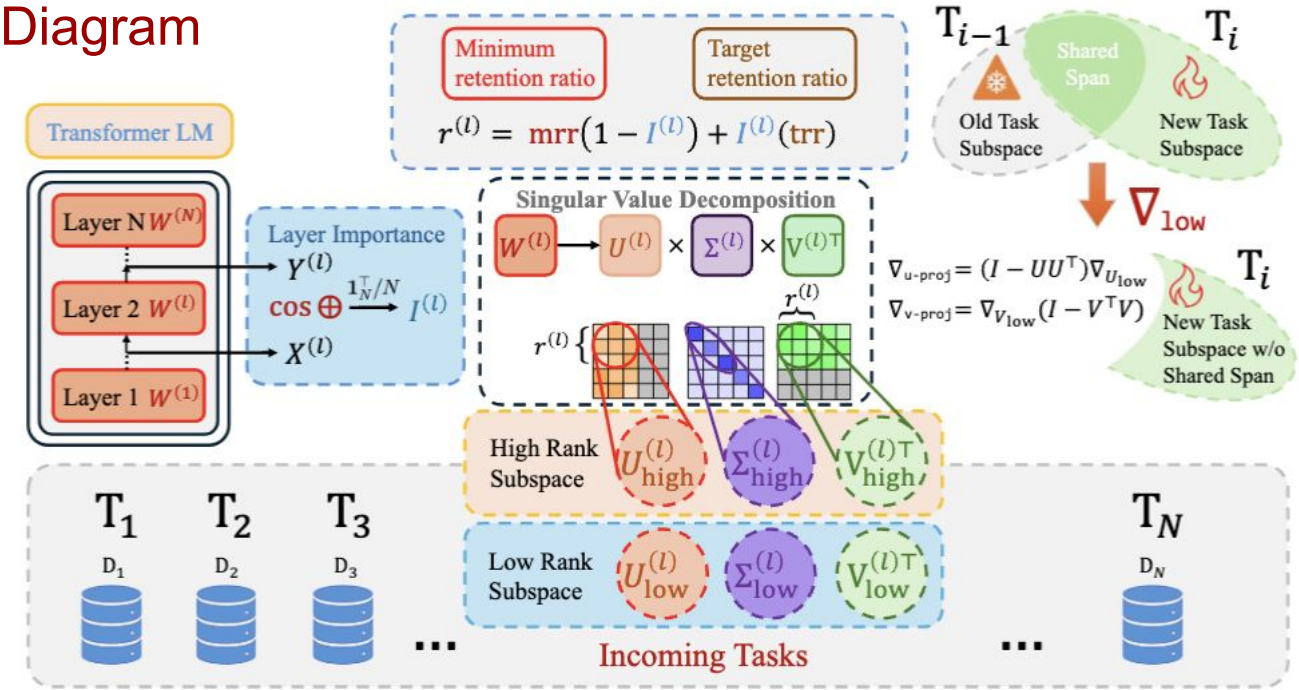


Figure 1: **Overview of our Adaptive SVD-based Continual Fine-tuning Method.** For each parameter matrix in the network, we perform SVD decomposition to identify high-rank components (associated with larger singular values) that encode crucial knowledge from previous tasks, and low-rank components (associated with smaller singular values) that contribute minimally to model performance. When learning a new task, gradient updates are projected onto the low-rank subspace orthogonal to previous task representations, allowing full parameter updates while minimizing catastrophic forgetting.

Results: Benchmarks and Evaluation Protocol

- **Two Continual Learning (CL) benchmarks** used:
    - **5-Task Standard CL Benchmark**: AG News, Amazon Reviews, Yelp Reviews, DBpedia, Yahoo Answers
    - **15-Task Extended CL Benchmark**: includes GLUE (MNLI, QQP, RTE, SST-2), SuperGLUE (WiC, CB, COPA, MultiRC, BoolQ), IMDB, 5-task Standard CL Benchmark
- Evaluated on **T5-Large** (encoder-decoder) and **LLaMA-2 7B** (decoder-only) architectures.
- Metric: **Average Accuracy (AA)** — accuracy averaged over all tasks after training on the final task.
- Results averaged over **3 random task sequences** to ensure robustness of results.
- Compared against **current SoTA**, including **O-LoRA***.

* Wang, Xiao, et al., "Orthogonal Subspace Learning for Language Model Continual Learning", 2023.

Results: Baseline Methods

We benchmark against diverse continual learning paradigms:

- **Sequential full-model fine-tuning (SeqFT)**: serves as a lower-bound baseline, prone to catastrophic forgetting.
- **Parameter-efficient LoRA** variants including SeqLoRA, IncLoRA, and the recent SoTA, O-LoRA, which utilize low-rank adapters.
- **Replay-based approaches**, such as standard replay buffers.
- **Regularization methods**, including Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF).
- **Prompt-based techniques**, including L2P and ProgPrompt.
- **Model Merging**: SLERP, TIES.
- **PerTaskFT**: trains a separate model per task, offering strong performance but requiring extensive computational resources and storage.
- **Multi-task Learning (MTL)**: trains a single model simultaneously on all tasks, representing an ideal upper bound by relaxing continual learning constraints.

Main Results

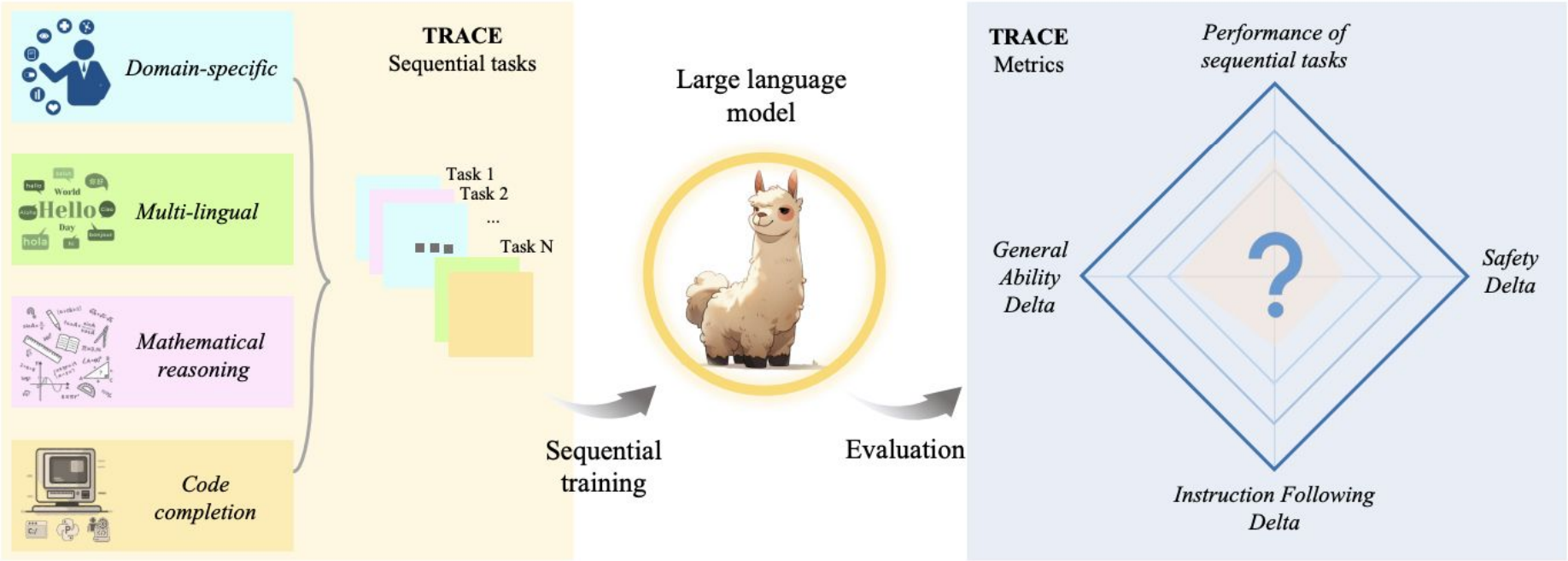
Table 1: Comparison of Average Accuracy (%) across standard continual learning benchmarks

Method	T5-Large (5 tasks)	T5-Large (15 tasks)
SeqFT	28.5	7.4
SeqLoRA	43.7	1.6
IncLoRA	66.4	61.2
Replay	57.8	54.2
EWC	48.7	45.1
LwF	52.3	46.9
L2P	60.7	56.1
LFPT5	72.7	69.2
O-LoRA	75.8	69.6
Ours (Adaptive SVD)	**75.9**	**71.3**
SLERP (Full Model Merge)	43.1	2.2
TIES (LoRA Adapter Merge)	37.1	6.9
ProgPrompt	75.1	77.9
PerTaskFT	70.0	78.1
MTL (Upper Bound)	80.0	76.5

Table 6: Ablation results on the LLaMA-2 7B model using the standard 5-task continual learning benchmark.

Method	Average Accuracy (%)
Ours (Adaptive SVD)	79.6
(1) Halved mrr/trr (aggressive effective rank approximation)	51.5
(2) No projection (unconstrained low-rank updates)	31.2

TRACE Benchmark



Wang, Xiao, et al., "TRACE: A Comprehensive Benchmark for Continual Learning in Large Language Models", 2023.

TRACE: Datasets and Evaluation

📘 Table 1: Sequential Training Tasks in TRACE

Dataset	Task Type	Tests For
ScienceQA	Multi-hop QA	Science knowledge, reasoning
FOMC	Classification	Financial domain understanding
MeetingBank	Summarization	Long-context understanding
C-STANCE	Stance classification	Chinese reasoning and cross-lingual ability
20Minuten	Text simplification	German text generation/simplification
Py150	Code completion	Code modeling, long context handling
NumGLUE-cm	Arithmetic reasoning	Numerical reasoning
NumGLUE-ds	Arithmetic reasoning	Numerical reasoning

📊 Table 2: Evaluation Benchmarks

Dataset	Task Type	Tests For
MMLU	QA	Factual knowledge
BBH	Reasoning	General reasoning
GSM	Math	Mathematical reasoning
TyDiQA	QA	Multilingual QA
PIQA	QA	Commonsense reasoning
BoolQ	QA	Reading comprehension
Self-Instruct	Instruction-following	Instruction adherence
LIMA	Instruction-following	Instruction adherence
CoNa	Safety evaluation	Toxicity and alignment safety

Wang, Xiao, et al., "TRACE: A Comprehensive Benchmark for Continual Learning in Large Language Models", 2023.

Results: Sequential Training Tasks

- **Average Accuracy**: Accuracy on each task, measured after training on the final task, then averaged across all tasks.
- **Backward Transfer**: Measures how well the model retains performance on earlier tasks — evaluates **catastrophic forgetting**.
- Our method achieves higher average accuracy and backward transfer compared to **O-LoRA**.
- Demonstrates our method's ability to **effectively retain and transfer knowledge** across tasks, and in its **robustness to forgetting**.

Table 2: TRACE benchmark performance using LLaMA-2-7B-Chat.

Method	Average Accuracy (%)	Backward Transfer (%)
SeqFT	23.0	-8.3
O-LoRA	41.3	6.2
Ours (Adaptive SVD)	**48.4**	**7.1**
PerTaskFT	57.6	NA
MTL	52.3	NA

Results: General Ability, Instruction Following, Safety

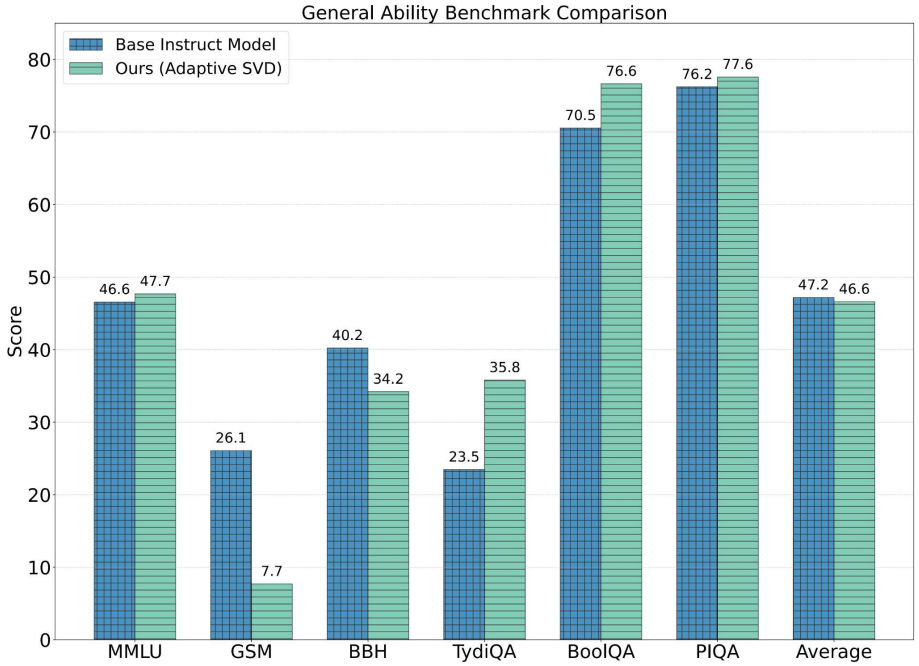


Table 4: Win / Tie / Lose breakdown (%) for instruction-following and safety evaluations against the LLaMA-2-7B-Chat base model.

Method	Instruction (Helpfulness)			Safety		
	Win	Tie	Lose	Win	Tie	Lose
Replay	10	18	72	0	88	12
LoRASeqFT	3	4	94	0	86	14
SeqFT	14	34	53	0	98	2
Ours (Adaptive SVD)	**24**	**56**	**20**	**18**	**78**	**4**

Toward the Utopia of Continual Learning

Vision

- A future where models learn continuously with **near-zero catastrophic forgetting**.
- **Real-world goals**: robust performance on tasks like QuALITY (long-form QA) and reasoning.

Limitations & Future Work

- **Rank Sensitivity**: Needs more principled, data-driven methods to determine effective rank.
- **Dynamic Allocation**: Avoid static subspace budgets—scale with task horizon.
- **SVD Overhead**: Focus decomposition on key matrices like attention to improve efficiency.