ALIGNING HUMAN AND MACHINE SKILLS FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans describe learning in terms of named and often compositional skills, whereas LLM training records model change through high-dimensional gradient updates. Without a correspondence between these views, a gradient does not directly reveal what an example teaches or which capabilities later training should preserve. We introduce Gradient Skill Decomposition (GSD), a framework for human–machine skill alignment in model gradient update space. GSD extracts compact layer-wise structure from per-example gradients, aligns this structure with multi-label skill annotations, and maps the recovered skill directions back to the parameter coordinates used for fine-tuning. The resulting shared representation supports both skill interpretation and skill-guided intervention. Across SmolLM3-3B, Qwen3.5-2B, and MiniCPM5-1B, GSD predicts human-defined skill sets with Jaccard scores of 64.2%–87.1%. During continual learning, we use the aligned directions as preservation constraints by projecting fine-tuning updates away from the selected skill subspace. Compared with standard LoRA, GSD-guided projection improves protected-task accuracy across 6 independently fine-tuned tasks and 3 models, with broadly comparable new continual-task performance. After 6 sequential tasks on MiniCPM5-1B, the projection improves protected-task accuracy by 12.5–18.0 percentage points over standard LoRA across two task orders. A targeted subset ablation provides initial evidence that selecting particular skill directions can concentrate protection on the corresponding behaviors. Across both independent and sequential continual-learning runs, once the skill basis is constructed, GSD processes the same number of fine-tuning tokens as standard LoRA, with only small increases in adaptation time and peak memory. Together, these results show that GSD connects an interpretable account of what examples teach to actionable control over continual learning, turning human-interpretable skill directions into a practical mechanism for reducing forgetting across successive tasks while maintaining new-task performance.

1 INTRODUCTION

Humans describe learning in terms of named and often compositional skills, whereas LLM training records learning through high-dimensional gradient updates. These views are disconnected: a gradient alone does not tell a person which skills an example exercises or which parts of a future update would modify them. Bridging the two would make model updates expressible in a vocabulary that people can inspect and use to guide training. We refer to this correspondence between human-defined skills and model update geometry as *human–machine skill alignment*.

The value of this alignment is that it would extend interpretability from model behavior to model change. Concept-based methods already relate internal representations or predictions to human-understandable concepts (Braun et al., 2025; Bushnaq et al., 2025). Locating these concepts in update space would make it possible to describe what an example teaches, identify where that effect is represented, and designate particular skills for preservation. This capability is directly useful for continual learning, where adaptation to new data can overwrite previously learned behavior. In this setting, an aligned representation could specify what should remain stable in human terms while leaving the rest of the update space available for learning.

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

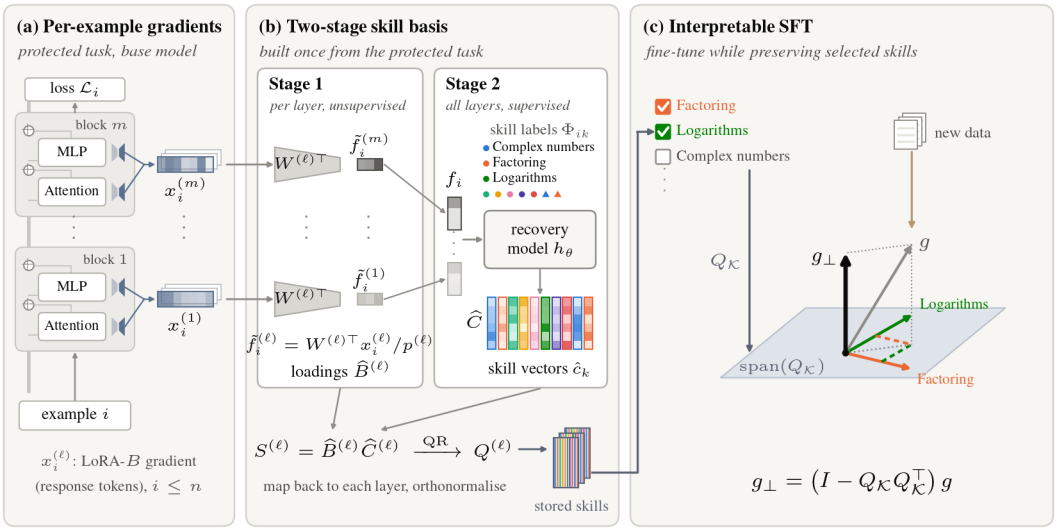

Figure 1: Overview of GSD. (a) We extract one gradient signature per protected example and layer in the trainable LoRA-B coordinates. (b) Layer-wise factor estimation compresses these signatures, and a supervised multilabel predictor aligns the concatenated factors with named skills; the resulting directions are mapped back to each layer and orthonormalized. (c) Selected skill directions define a protected basis. The panel depicts the orthogonal removal schematically; the implementation applies it to the cumulative parameter change after each optimizer step.

Using alignment for continual learning, however, requires more than predicting skill labels at a fixed version of the model before continual training. The representation must separate the contributions of overlapping skills even though their gradient signatures are high-dimensional and distributed across layers. It must then map each skill back to the coordinates used for optimization, so that a chosen set of skills defines a protected subspace and subsequent updates can be stripped of their components within it. Raw gradients do not clearly expose this structure, while a single averaged direction collapses the distinctions needed to protect skills in examples with multiple labels.

These representational and optimization requirements fall between two lines of existing work. Concept-based methods primarily interpret activations or predictions, whereas continual-learning methods protect examples, tasks, or model-defined subspaces (Farajtabar et al., 2020; Saha et al., 2021). This separation leaves open how to construct a protected update space from a human skill vocabulary. We introduce *Gradient Skill Decomposition* (GSD) to bridge this gap. GSD learns a compact, layer-wise representation of per-example gradients, aligns it with observed skill annotations, and maps the resulting skill directions back to the coordinates used for fine-tuning. The same skill basis can therefore explain what an example teaches and constrain how later examples change the model.

This dual role leads to two empirical questions: whether GSD recovers meaningful skill structure, and whether that structure remains useful for controlling later updates. To address the first, GSD predicts held-out multi-label skill sets across three LLMs with Jaccard scores of 64.2%–87.1%, while geometric analyses reveal skill organization that is difficult to discern in raw-gradient space. To address the second, we use the recovered basis for continual learning. Without replay, skill-guided projection improves protected-task accuracy over standard LoRA in all 18 independent model–task settings while maintaining broadly comparable new-task performance. To examine whether this protection persists, we reuse the same fixed basis across two six-task sequences; it improves final protected-task accuracy by 12.5–18.0 percentage points over standard LoRA. Finally, a targeted subset ablation provides initial evidence that selecting particular skill directions can concentrate protection on the corresponding behaviors.

Together, these results support three contributions:

- **An operational framework for human–machine skill alignment.** We formulate alignment in model update space and introduce GSD to connect human-defined, compositional skills with layer-wise gradient structure. This shared representation makes skills both interpretable at the example level and actionable during optimization.
- **Evidence for skill structure in gradient space.** We show that per-example gradients contain recoverable information about individual skills and their composition. Across three LLMs, GSD predicts held-out multi-label skill sets with 64.2%–87.1% Jaccard, while geometric and layer-wise analyses show how the aligned representation organizes skill information that is diffuse in the raw gradients.
- **Skill-guided continual learning.** We turn the aligned skill directions into explicit preservation constraints for subsequent fine-tuning. The intervention improves retention over standard LoRA in all 18 independent settings, remains effective through two six-task sequences, and complements replay. A restricted subset ablation provides initial evidence that protection can be directed toward selected skills, rather than only toward a task as a whole.

Overall, GSD makes human–machine skill alignment operational by connecting an interpretable account of what training examples teach to direct control over what an LLM retains during continual learning.

2 Problem Setup

The problem for skill alignment now can formulated as seeking a representation of their induced updates that both exposes shared skill structure and remains directly comparable to the gradients produced during subsequent adaptation, given protected examples with observed skill labels.

Protected-example loss. Let $(u_i, v_i, \boldsymbol{\phi}_i^{\mathrm{obs}})$ be a protected example, where u_i is the prompt,v_i is the target response, and $\boldsymbol{\phi}_i^{\mathrm{obs}} \in \{0,1\}^q$ is the observed multi-hot skill label. We extract its gradient signal at the original pretrained checkpoint $\boldsymbol{\theta}_0$. The model receives the complete prompt–response sequence, but the language-modeling loss is evaluated only at response positions:

$$\mathcal{L}_i(\boldsymbol{\theta}_0) = -\frac{1}{|R_i|} \sum_{t \in R_i} \log p_{\boldsymbol{\theta}_0}(v_{i,t} \mid u_i, v_{i,<t}) \,, \tag{1}$$

Here R_i indexes the response tokens. Thus, the prompt conditions every prediction while contributing no tokens directly to the loss.

From full gradients to LoRA coordinates. For the jth selected weight matrix in layer ℓ, define the conceptual gradient $\boldsymbol{G}_{i,j}^{(\ell)} = \nabla_{\boldsymbol{W}_j^{(\ell)}} \mathcal{L}_i(\boldsymbol{\theta}_0)$. Stacking these quantities over the model yields a full per-example gradient. Storing it for every protected example is infeasible at scale, and it includes directions that LoRA SFT cannot traverse. We therefore encode each example directly in the trainable LoRA coordinates (Hu et al., 2022), which writes $\boldsymbol{W}_j^{(\ell)} = \boldsymbol{W}_{0,j}^{(\ell)} + \Delta \boldsymbol{W}_j^{(\ell)}$, with $\Delta \boldsymbol{W}_j^{(\ell)} = \boldsymbol{B}_j^{(\ell)} \boldsymbol{A}_j^{(\ell)}$. We freeze $\boldsymbol{W}_{0,j}^{(\ell)}$ and the right factor $\boldsymbol{A}_j^{(\ell)}$, and initialize the trainable left factor $\boldsymbol{B}_j^{(\ell)} = 0$, so the model remains at $\boldsymbol{\theta}_0$. Backpropagation directly gives $\nabla_{\boldsymbol{B}_j^{(\ell)}} \mathcal{L}_i(\boldsymbol{\theta}_0) = \boldsymbol{G}_{i,j}^{(\ell)} (\boldsymbol{A}_j^{(\ell)})^\top$ and we directly flatten its gradient as $\boldsymbol{x}_i^{(\ell)} = \mathrm{vec}(\nabla_{\boldsymbol{B}^{(\ell)}} \mathcal{L}_i(\boldsymbol{\theta}_0)) \in \mathbb{R}^{p^{(\ell)}}$, where $p^{(\ell)}$ is the number of scalar parameters in $\boldsymbol{B}^{(\ell)}$.

Why the left-factor gradient. We use $\boldsymbol{X}_i^{(\ell)}$ as the layer-wise signature of example i. Its dimension is fixed, independent of input length, and describes the output-side update coordinates. Comparability requires a fixed right factor because the LoRA factorization is not unique: $\boldsymbol{B}_j^{(\ell)} \boldsymbol{A}_j^{(\ell)} = (\boldsymbol{B}_j^{(\ell)} \boldsymbol{H})(\boldsymbol{H}^{-1} \boldsymbol{A}_j^{(\ell)})$ for any invertible $\boldsymbol{H} \in \mathbb{R}^{s \times s}$. We instantiate each $\boldsymbol{A}_j^{(\ell)}$ once and reuse it for all protected examples and subsequent adaptation, placing every signature in the same coordinate system. Right-factor and full-update representations are natural alternatives that we leave to ablation. The resulting $\boldsymbol{x}_i^{(\ell)}$ is the per-example input to the layer-wise GSD analysis.

These comparable signatures provide the interface between examples and skills: GSD identifies shared, named directions in this gradient space and returns them to the same coordinates for intervention (Figure 1).

3 GRADIENT SKILL DECOMPOSITION (GSD)

GSD uses the layer-wise per-example gradient representations $\boldsymbol{x}_i^{(\ell)}$ defined above to recover skill labels and construct explicit skill directions. Stage 1 estimates a low-dimensional gradient subspace independently for each layer (Section 3.1). Stage 2 maps the resulting coordinates to the observed multi-hot labels $\boldsymbol{\phi}_i^{\text{obs}}$ and derives one direction for each skill from the trained predictor (Section 3.2). These directions are later mapped back to the same LoRA-B coordinates for continual-learning intervention (Section 4.3). Algorithm 1 summarizes fitting and prediction; estimation details appear in Appendix A.

3.1 STAGE 1: LAYER-WISE SKILL SUBSPACE ESTIMATION

We estimate the shared structure one layer at a time and then aggregate—the distributed-eigenspace strategy of Fan et al. (2019). For n protected training examples, we stack the layer-ℓ gradient vectors row-wise as $\boldsymbol{X}^{(\ell)} = [\boldsymbol{X}_1^{(\ell)}, \ldots, \boldsymbol{X}_n^{(\ell)}]^\top \in \mathbb{R}^{n \times p^{(\ell)}}$ and posit the working factor model $\boldsymbol{X}^{(\ell)} = \boldsymbol{F}^{(\ell)} \boldsymbol{\Lambda}^{(\ell)\top} + \boldsymbol{U}^{(\ell)}$. Here $\boldsymbol{F}^{(\ell)} \in \mathbb{R}^{n \times \bar{r}^{(\ell)}}$ and $\boldsymbol{\Lambda}^{(\ell)} \in \mathbb{R}^{p^{(\ell)} \times \bar{r}^{(\ell)}}$ denote the layer factors and GSD loadings, respectively; $\boldsymbol{\Lambda}^{(\ell)}$ is the residual that is unrelated to the LoRA factors $\boldsymbol{A}_j^{(\ell)}$ and $\boldsymbol{B}_j^{(\ell)}$. For each layer, we construct a diversified projection $\boldsymbol{D}^{(\ell)} \in \mathbb{R}^{p^{(\ell)} \times \bar{r}^{(\ell)}}$ from the leading $\bar{r}^{(\ell)}$ eigenvectors of its sample covariance (Fan & Liao, 2022; Fan & Gu, 2024). The surrogate coordinates are $\widetilde{\boldsymbol{f}}_i^{(\ell)} = \frac{1}{p^{(\ell)}} \boldsymbol{D}^{(\ell)\top} \boldsymbol{x}_i^{(\ell)}$. The training projection $\boldsymbol{D}^{(\ell)}$ remains fixed when constructing validation and test coordinates, where the construction of $\boldsymbol{D}^{(\ell)}$ and its regularity conditions are given in Appendix A Concatenating across the m layers gives the factor vecctor $\widetilde{\boldsymbol{f}}_i = [\widetilde{\boldsymbol{f}}_i^{(1)}; \ldots; \widetilde{\boldsymbol{f}}_i^{(m)}] \in \mathbb{R}^r$, where $r = \sum_{\ell=1}^m \bar{r}^{(\ell)}$; stacking these vectors row-wise gives $\widetilde{\boldsymbol{F}} \in \mathbb{R}^{n \times r}$. This stage is unsupervised and parallel over layers. We will use this dimension-reduced factor $\widetilde{\boldsymbol{f}}_i$ to serve as a compressed gradient update matrix analog with low dimensionality to proceed with the next stage for skill recovery learning.

3.2 STAGE 2: MLP-BASED SKILL RECOVERY

We train a two-layer multilabel MLP to map the Stage-1 coordinates $\widetilde{\boldsymbol{f}}_i \in \mathbb{R}^r$ to the observed skill labels $\boldsymbol{\phi}_i^{\text{obs}} \in \{0,1\}^q$. Feature standardization is fitted on training factors and included in $h_{\boldsymbol{\psi}} : \mathbb{R}^r \to \mathbb{R}^q$. The skill logits and probabilities are

$$\boldsymbol{z}_i = h_{\boldsymbol{\psi}}(\widetilde{\boldsymbol{f}}_i), \qquad \boldsymbol{\pi}_i = \sigma(\boldsymbol{z}_i),$$

where σ is the elementwise sigmoid. We train with class-balanced binary cross-entropy, select the checkpoint by validation loss, and select per-skill thresholds τ_k by validation sample Jaccard. The predicted multi-hot label $\widehat{\boldsymbol{\phi}}_i \in \{0,1\}^q$ has entries $\widehat{\phi}_{ik} = \mathbf{1}\{\pi_{ik} \geq \tau_k\}$. The oracle diagnostic instead selects the known number of active skills with the largest probabilities. We approximate the direction of skill k by averaging its logit gradient over positive training examples:

$$\widehat{\boldsymbol{c}}_k = \frac{1}{n_k^+} \sum_{i:\phi_{ik}^{\text{obs}}=1} \nabla_{\widetilde{\boldsymbol{f}}_i} z_{ik}, \qquad \widehat{\boldsymbol{C}} = [\widehat{\boldsymbol{c}}_1, \ldots, \widehat{\boldsymbol{c}}_q] \in \mathbb{R}^{r \times q},$$

where n_k^+ is the number of positive training examples for skill k. The derivative includes the standardization map, so each direction is expressed in raw factor coordinates. Because $\widetilde{\boldsymbol{f}}_i$ concatenates layer-wise coordinates, we partition each skill direction conformably as $\widehat{\boldsymbol{c}}_k = [\widehat{\boldsymbol{c}}_k^{(1)}; \ldots; \widehat{\boldsymbol{c}}_k^{(m)}]$. Its image in the LoRA-$B$ gradient coordinates of layer ℓ is $\widehat{\boldsymbol{s}}_k^{(\ell)} = \widehat{\boldsymbol{\Lambda}}^{(\ell)} \widehat{\boldsymbol{c}}_k^{(\ell)}$. By doing this two-stage dimensional reduction learning, we finally learn the skill vector that corresponds to human-understood skills for each question.

3.3 SKILL-ORTHOGONAL SUPERVISED FINE-TUNING

The recovered skill directions define a control surface for supervised fine-tuning. Let $\widehat{\boldsymbol{C}}^{(\ell)} \in \mathbb{R}^{\bar{r}^{(\ell)} \times q}$ be the row block of $\widehat{\boldsymbol{C}}$ associated with layer ℓ. We collect the mapped directions for that layer as

$$\widehat{\boldsymbol{S}}^{(\ell)} = \widehat{\boldsymbol{\Lambda}}^{(\ell)} \widehat{\boldsymbol{C}}^{(\ell)} = [\widehat{\boldsymbol{s}}_1^{(\ell)}, \ldots, \widehat{\boldsymbol{s}}_q^{(\ell)}] \in \mathbb{R}^{p^{(\ell)} \times q},$$

Let $\rho_\ell = \operatorname{rank}(\widehat{\boldsymbol{S}}^{(\ell)}) \leq q$. We obtain an orthonormal basis $\boldsymbol{Q}^{(\ell)} \in \mathbb{R}^{p^{(\ell)} \times \rho_\ell}$ for this column space, which is the protected subspace in exactly the layer-wise LoRA-B coordinates of $\boldsymbol{x}_i^{(\ell)}$. During SFT on a new dataset, vanilla LoRA uses the raw adapter gradient $\boldsymbol{g}$. Skill-orthogonal SFT instead removes the component of this gradient lying in the protected subspace,

$$\boldsymbol{g}_\perp = (\boldsymbol{I} - \boldsymbol{P}_{\hat{S}})g, \qquad \boldsymbol{P}_S = \widehat{\boldsymbol{S}}(\widehat{\boldsymbol{S}}^\top \widehat{\boldsymbol{S}})^{-1} \widehat{\boldsymbol{S}}^\top = \boldsymbol{Q}^{(\ell)} (\boldsymbol{Q}^{(\ell)})^\top.$$

This intervention tests whether the recovered skill vectors are not only predictive, but also writable. If forgetting occurs through movement along these directions, removing that component should preserve the old skill while still allowing learning through the remaining gradient directions. Algorithm 2 gives the training loop with optional replay; Appendix D gives the implementation and evaluation protocol.

4 EXPERIMENTAL SETUP AND MAIN RESULTS

4.1 EXPERIMENTAL SETUP

Protected dataset by construction. We construct Math Gen, a synthetic algebra dataset of 22,232 questions covering ten skills, with one to three target skills per question. Following the skill-labeling and semantic-grouping approach of Didolkar et al. (2024), Gemini 3.1 Pro annotates problems from the original MATH dataset (Hendrycks et al., 2021) and groups similar skill labels into a consolidated vocabulary. Using Gemini, we select ten moderately scoped Algebra skills and form 30 generation groups: the ten individual skills, ten randomly selected two-skill combinations, and ten randomly selected three-skill combinations. We request 1,000 questions with worked solutions per group from Qwen3.5-27B (Team, 2026), for 30,000 generation requests in total. Generation prompts target MATH level 3–5 difficulty and require multi-skill questions to integrate all designated skills in one solution. Each question's target skills define its multi-hot label, and its saved solution supplies the completion used for gradient extraction and protected-task replay. Output validation and within-category deduplication yield the final 22,232 questions. We use fixed, stratified training, validation, and test splits of approximately 70/10/20% across all models and methods. Appendix B details skill selection, generation settings, filtering, and split construction. The supplementary material includes the Math Gen dataset as a ZIP archive.

Continual-learning datasets. For continual learning, we fine-tune on widely used existing datasets. The task lists and evaluation protocols appear in Appendix C.

LLM models for continual learning. We evaluate MiniCPM5-1B (Team et al., 2025), SmolLM3-3B (Bakouch et al., 2025), and Qwen3.5-2B (Team, 2026).

We extract gradients from each model's original checkpoint without prior fine-tuning on Math Gen. We use LoRA (Hu et al., 2022) for adaptation, keeping the pretrained weights and LoRA-A matrices frozen and updating only LoRA-B. Skill directions are constructed separately for each model using Math Gen gradients in the same LoRA coordinates used for continual learning.

4.2 SKILL PREDICTION USING GSD

Following Section 3, we compute per-example gradients of the saved reference solution's negative log-likelihood with respect to LoRA-B, masking prompt tokens from the loss. The gradient representation, feature standardization, and MLP recovery model are fitted using training data only. We select the MLP checkpoint by validation loss and choose non-oracle per-skill thresholds by validation sample Jaccard. All fitted components and thresholds remain fixed during test evaluation.

Table 1: Skill prediction (%) on 4,446 Math Gen test questions. The precision, recall, and Jaccard metrics use validation-selected thresholds. Jaccard averages the intersection-over-union of predicted and annotated skill sets; Oracle J. selects the highest-scoring skills using the true label count and serves only as a diagnostic.

Model	Precision	Recall	Jaccard	Oracle J.
MiniCPM5-1B	71.4	77.9	64.2	74.1
SmolLM3-3B	91.5	93.2	87.1	94.1
Qwen3.5-2B	82.3	84.5	75.0	84.6

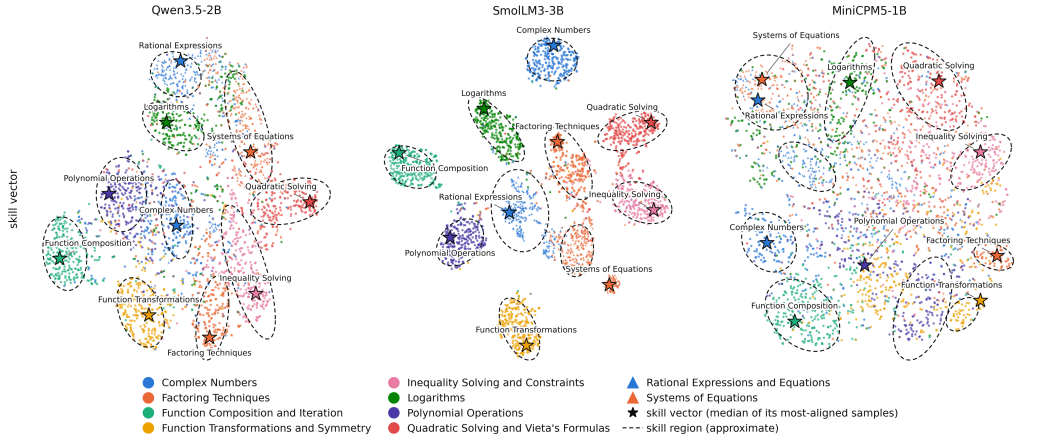

Figure 2: Two-dimensional t-SNE visualizations of matched Math Gen examples in the learned skill-aligned representation. Columns show Qwen3.5-2B, SmolLM3-3B, and MiniCPM5-1B; colors and markers denote annotated skills. Stars mark the locations of skill directions estimated from their most-aligned examples, and dashed contours serve only as visual guides. Axes are local to each panel. Appendix Figure 5 shows the corresponding raw-gradient embeddings.

Skill recovery. GSD recovers annotated Math Gen skills from compact gradient features across all three evaluated models (Table 1). With thresholds selected on validation data, test Jaccard reaches 64.2% for MiniCPM5-1B, 87.1% for SmolLM3-3B, and 75.0% for Qwen3.5-2B. These results demonstrate that the gradient representation retains information useful for supervised skill prediction on held-out questions.

Geometry of the aligned representation. Figure 2 complements the held-out prediction metrics by visualizing matched examples in the learned skill representation. Examples sharing a skill form coherent regions around the corresponding skill directions. The separation is clearest for SmolLM3-3B, which also attains the highest test Jaccard in Table 1; MiniCPM5-1B remains more intermixed, consistent with its lower recovery score. The corresponding raw-gradient embeddings are substantially more mixed (Appendix Figure 5), providing qualitative evidence that GSD organizes gradient variation around the annotated skill vocabulary. Appendix B.2 further shows that the recovered directions are distributed across layers rather than localized to a single block with especail emphasis on lateral layers.

4.3 Skill Protection for Continual Learning

We test whether the recovered skill directions preserve mathematical ability during adaptation to new tasks. We report independent fine-tuning on multiple continuation tasks here and defer the accumulated forgetting experiment across six sequential tasks to Appendix E. We evaluate two variants of our method: *Ours*, skill projection without replay, and *Replay+Ours*, skill projection combined with replay.

Protocol and baselines. Each independent run starts from the original checkpoint and trains for one epoch. The protected subspace is estimated using Math Gen training gradients and frozen before continuation training. We compare against three baselines: **Vanilla**, standard LoRA fine-tuning; **mean-gradient projection (Mean-Grad)**, which applies projection using one normalized average Math Gen training-gradient direction per layer; and **Replay**, which mixes Math Gen training completions with continuation-task data using a protected-to-new-task completion-token ratio of 0.20 and token-proportional loss. Our two variants are **Ours**, which removes the protected-subspace component of the cumulative LoRA-B parameter change relative to the stage's starting adapter after each optimizer step; and **Replay+Ours**, which applies the same skill projection with the replay mixture defined above. All methods keep LoRA-A frozen. Appendices C and D provide the task metrics and training details.

Independent continual learning. For each model, one of our two variants achieves the highest six-task average Math Gen accuracy among the five compared methods (Table 2). Ours improves over Vanilla in all 18 model–task settings, with average gains of 4.5, 2.7, and 6.0 percentage points for MiniCPM5-1B, SmolLM3-3B, and Qwen3.5-2B, respectively. It also exceeds Mean-Grad and Replay on the six-task average within each model.

Our method can further improve retention in settings where Replay already helps. For MiniCPM5-1B on FinQA, Replay raises Math Gen accuracy from Vanilla's 29.2% to 32.4%, and Replay+Ours raises it further to 34.4%. For Qwen3.5-2B on LIMA, the corresponding accuracies are 24.6%, 38.8%, and 56.2%. Averaged over the six continuation tasks, Replay+Ours improves over Replay by 3.0, 1.7, and 10.5 points for MiniCPM5-1B, SmolLM3-3B, and Qwen3.5-2B, respectively. These results show that our skill projection improves retention over standard fine-tuning and adds further value when combined with replay. Table 6 in Appendix G reports the corresponding continuation-task test performance. Appendix E shows that the same fixed skill basis also improves retention across two six-task orders and reports new-task performance immediately after the corresponding training stages.

5 Ablation Studies

5.1 Skill-targeted protection

In this section, we conduct experiments to test whether protecting on specific skills can be effective in continual learning. To test whether GSD can target individual skills rather than only preserve Math Gen as a whole, we fine-tune SmolLM3-3B on HellaSwag and construct each $\boldsymbol{Q}^{(\ell)}$ using only the mapped direction for Factoring Techniques, Systems of Equations, or both (Figure 3a). We compare each arm with the same unprotected Vanilla model on the same 500 Math Gen questions, stratified by their annotated skills, and estimate 95% confidence intervals by paired bootstrap. Protecting Factoring raises accuracy on Factoring questions from 41.4% to 46.6% ($+5.2$ points), but also raises accuracy on the remaining questions by 3.1 points (Figure 3b). Systems protection is more selective: Systems accuracy rises from 46.7% to 52.6% ($+5.9$ points), while accuracy on the remaining questions falls by 2.7 points (Figure 3c). Protecting both directions raises accuracy on questions requiring both skills from 35.3% to 41.2% ($+5.9$ points), while accuracy on questions requiring neither changes by only $+0.8$ points (Figure 3d). These point estimates suggest that selecting columns of the learned basis can target the corresponding skills, which may serve as a steerable protection mechanism in continual learning that human can assist selecting continue learning target directions by composition.

5.2 Compute and adaptation cost

Projection is most useful as a continual-learning mechanism if its reusable protection does not impose the recurring cost of replay. We therefore compare processed input tokens, peak allocated GPU memory, and code-level adaptation time for Vanilla, Mean-Grad, Ours, Replay, and Replay+Ours (Appendix F, Table 5). Both projection methods process exactly the same number of tokens as Vanilla. Across the independent runs, Mean-Grad adds at most 0.06 GiB of peak memory and 0.1 minutes, whereas Ours adds at most 0.48 GiB and 0.2 minutes; across the two six-stage sequences, the corresponding maxima are 0.12 GiB and 0.2 minutes for Mean-Grad and 0.54 GiB and 0.1 min-

Table 2: Math Gen retention after independent continuation-task fine-tuning. Entries are boxed-answer accuracy (%) on the final test panel; higher is better. Each task starts from the original checkpoint. "Six-task average" reports equally weighted accuracy across the six independent fine-tuning tasks. "Mean-Grad" denotes the mean-gradient projection baseline. "Ours" and "Replay+Ours" are the two variants of our method, using skill projection without and with replay, respectively. Base accuracies are 40.8% for MiniCPM5-1B, 47.6% for SmolLM3-3B, and 62.2% for Qwen3.5-2B. MiniCPM5-1B and SmolLM3-3B use sampled decoding and Math-Verify; the reported Qwen3.5-2B runs use greedy decoding and a local boxed-answer scorer. Comparisons therefore concern methods within each model. Bold marks the six-task average rows. Underlining marks the highest accuracy in each row at the displayed precision, including ties.

Continuation task	Vanilla	Mean-Grad	Replay	Ours	Replay+Ours
MiniCPM5-1B					
FinQA	29.2	30.6	32.4	31.4	34.4
GPQA	35.4	38.0	40.0	37.6	35.6
SQuAD	24.4	22.2	28.0	31.2	30.0
HellaSwag	27.2	28.4	29.4	33.0	33.4
LIMA	23.4	22.2	23.0	31.0	33.0
UltraChat	26.2	23.2	21.6	28.6	26.0
Six-task average	**27.6**	**27.4**	**29.1**	**32.1**	**32.1**
SmolLM3-3B					
FinQA	45.4	45.4	47.6	51.0	46.6
GPQA	49.2	48.8	45.0	49.6	46.2
SQuAD	48.6	45.6	50.4	50.6	50.8
HellaSwag	48.6	51.8	48.6	50.4	51.0
LIMA	43.6	44.4	45.0	47.2	51.4
UltraChat	45.6	45.8	43.8	48.0	44.6
Six-task average	**46.8**	**47.0**	**46.7**	**49.5**	**48.4**
Qwen3.5-2B					
FinQA	59.6	60.4	54.8	60.4	57.4
GPQA	59.6	63.2	61.6	61.8	64.4
SQuAD	53.6	60.0	38.0	57.6	47.2
HellaSwag	50.2	57.4	43.8	55.6	58.0
LIMA	24.6	27.8	38.8	36.2	56.2
UltraChat	32.8	38.4	36.2	44.4	52.8
Six-task average	**46.7**	**51.2**	**45.5**	**52.7**	**56.0**

utes for Ours. Replay instead processes 36–45% more tokens in the independent runs and 20% more over the sequences, increasing adaptation time by 23–44% and 15–19%, respectively. Adding Ours to Replay requires no additional tokens or adaptation time at the reported precision and adds at most 0.54 GiB of peak memory. Mean-Grad stores one direction per layer in 19.5–59.7 MiB, while the richer skill basis requires 165.0–492.0 MiB but provides stronger retention in the main experiments (Tables 2 and 4). Because reliable basis-construction time was not recorded, these measurements establish the marginal cost of reusing GSD during adaptation, not its total end-to-end setup cost.

6 Related Work

Gradient-based skill discovery. QDG clusters next-token gradients (Michaud et al., 2023); NPEFF factors per-example Fisher matrices and supports parameter interventions (Matena & Raffel, 2026). APD and SPD target mechanism discovery (Braun et al., 2025; Bushnaq et al., 2025). Our method learns a supervised gradient representation at a single checkpoint, recovering named skill combinations and deriving their protection directions for continued training.

Skill labeling and generation. Math Gen follows the LLM annotation and grouping approach of Didolkar et al. (2024). Instruct-SkillMix and STAT guide generation and training with skill

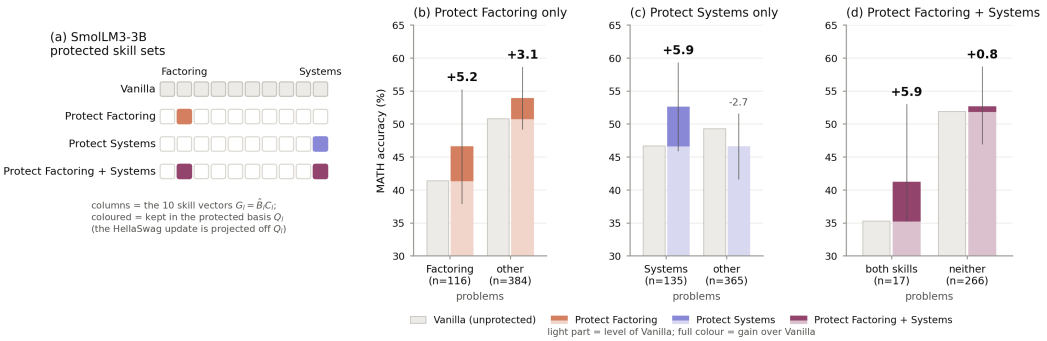

Figure 3: Skill-targeted protection for SmolLM3-3B during HellaSwag fine-tuning. (a) Directions included in each protected basis. (b–d) Math Gen accuracy on skill-defined subsets of the same test dataset. Gray bars show Vanilla; colored bars show the restricted-basis arms, with saturated shading marking gains over Vanilla.

descriptions (Kaur et al., 2025; He et al., 2026). Our method makes skill annotations actionable as protection targets in the model's update space.

Gradient and concept interpretability. TCAV probes specified concepts (Kim et al., 2018). GradSAE and Jacobian SAEs analyze influence and computations through derivatives (Shu et al., 2025; Farnik et al., 2025). SAE-SSV supports supervised steering in sparse activation spaces (He et al., 2025). Our method places named skills in parameter-gradient coordinates, where their directions constrain fine-tuning updates.

Continual learning. Replay rehearses prior examples (Rolnick et al., 2019; Scialom et al., 2022). GEM and A-GEM constrain updates with task-wise and averaged memory gradients, respectively (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019). OGD and GPM protect important subspaces (Farajtabar et al., 2020; Saha et al., 2021). O-LoRA, InfLoRA, and GORP constrain adaptation through subspace structure (Wang et al., 2023; Liang & Li, 2024; Wang et al., 2025). Our method makes the protected subspace selectable by named skill, improving retention both on its own and with replay.

7 Conclusion

We introduced Gradient Skill Decomposition (GSD), a framework that aligns human-defined, compositional skills with low-dimensional structure in per-example LoRA gradients. GSD recovers held-out skill labels and maps the learned skill directions back to LoRA parameter space, enabling selective protection during continual learning. Across multiple models and tasks, this approach improves retention over standard LoRA while maintaining comparable new-task performance and complementing replay. These results establish gradient space as a practical interface for understanding what examples teach and controlling what subsequent fine-tuning preserves.

AI use statement

In this work, we used generative AI tools for the following tasks: generate synthetic data sets, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We have not used generative AI tools for the following tasks: help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments. The following tasks are not applicable to this work: assist in the writing of proofs, assist with translation.

Additionally, we used generative AI tools for the following tasks: create or modify scientific figures or images, create or edit software code, draft parts of a research paper, summarize or analyse existing

literature, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature.

We have reviewed all AI-assisted work. Two co-authors checked and edited AI-assisted outputs and used test runs to check the research code. The authors reviewed and corrected interpretations and determined the final conclusions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

The supplementary material includes the Math Gen dataset as a ZIP archive. Appendix B documents dataset construction, preprocessing, data splits, and skill-recovery settings, and Appendix H reports the data-construction prompts. Model configurations and experimental protocols are reported in Section 4 and Appendices C and D. Implementation details and pseudocode for skill prediction and continual learning are provided in Appendices A and D. If the paper is accepted, we will release our experimental code in a GitHub repository with the camera-ready version.

References

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition, 2025. URL https://arxiv.org/abs/2501.14926.

Lucius Bushnaq, Dan Braun, and Lee Sharkey. Stochastic parameter decomposition, 2025. URL https://arxiv.org/abs/2506.20790.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke, Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Sanjeev Arora. Metacognitive capabilities of llms: An exploration in mathematical problem solving. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 19783–19812. Curran Associates, Inc., 2024. doi: 10.52202/079017-0623. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2318d75a06437eaa257737a5cf3ab83c-Paper-Conference.pdf.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Jianqing Fan and Yihong Gu. Factor augmented sparse throughput deep relu neural networks for high dimensional regression. *Journal of the American Statistical Association*, 119(548):2680–2694, 2024.

Jianqing Fan and Yuan Liao. Learning latent factors from diversified projections and its applications to over-estimated and weak factors. *Journal of the American Statistical Association*, 117(538): 909–924, 2022.

Jianqing Fan, Dong Wang, Kaizheng Wang, and Ziwei Zhu. Distributed estimation of principal eigenspaces. *Annals of statistics*, 47(6):3009, 2019.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

Lucy Farnik, Tim Lawson, Conor Houghton, and Laurence Aitchison. Jacobian sparse autoencoders: Sparsify computations, not just activations. 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. 16178–16236. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/farnik25a.html.

Yinghui He, Abhishek Panigrahi, Yong Lin, and Sanjeev Arora. Stat: Skill-targeted adaptive training. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 14359–14387, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/17d60fef592086d1a5cb136f1946df59-Paper-Conference.pdf.

Zirui He, Mingyu Jin, Bo Shen, Ali Payani, Yongfeng Zhang, and Mengnan Du. SAE-SSV: Supervised steering in sparse representation spaces for reliable control of language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2207–2236, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.112. URL https://aclanthology.org/2025.emnlp-main.112/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Simran Kaur, Simon Park, Anirudh Goyal, and Sanjeev Arora. Instruct-skillmix: A powerful pipeline for llm instruction tuning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 28842–28983, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/473a9a75edc46eff5ff224d53d5f7294-Paper-Conference.pdf.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2668–2677. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/kim18d.html.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23638–23647. IEEE, 2024.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Michael S Matena and Colin Raffel. Uncovering language model processing strategies with non-negative per-example fisher factorization. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=UjeDVujI8q.

Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28699–28722. Curran Associates, Inc., 2023. doi: 10.52202/075280-1248. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5b6346a05a537d4cdb2f50323452a9fe-Paper-Conference.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=3AOj0RCNC2.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6107–6122, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.410. URL https://aclanthology.org/2022.emnlp-main.410/.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Mengnan Du, and Ninghao Liu. Beyond input activations: Identifying influential latents by gradient sparse autoencoders. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1673–1682, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.87. URL https://aclanthology.org/2025.emnlp-main.87/.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, Ning Ding, Shengda Fan, Yewei Fang, Zixuan Fu, Wenyu Guan, Yitong Guan, Junshao Guo, Yufeng Han, Bingxiang He, Yuxiang Huang, Baoxi Ji, Cunliang Kong, Qiuzuo Li, Siyuan Li, Wenhao Li, Xin Li, Yanghao Li, Yishan Li, Zhen Li, Dan Liu, Biyuan Lin, Yankai Lin, Xiang Long, Quanyu Lu, Yaxi Lu, Peiyan Luo, Hongya Lyu, Litu

Ou, Yinxu Pan, Lushi Pu, Zekai Qu, Qundong Shi, Zijun Song, Jiayuan Su, Zhou Su, Ao Sun, Xianghui Sun, Peijun Tang, Fangzheng Wang, Feng Wang, Shuo Wang, Yudong Wang, Zheng Wang, Yesai Wu, Zhenyu Xiao, Jie Xie, Zihao Xie, Xiaoyue Xu, Yukun Yan, Jiarui Yuan, Jinqian Zhang, Kaihuo Zhang, Lei Zhang, Linyue Zhang, Xueren Zhang, Yudi Zhang, Hengyu Zhao, Weilin Zhao, Weilun Zhao, Yuanqian Zhao, Zhi Zheng, Chuyue Zhou, Ge Zhou, Jie Zhou, Wei Zhou, Yanghao Zhou, Zihan Zhou, Zixuan Zhou, Zhiyuan Liu, Guoyang Zeng, Chao Jia, Dahai Li, and Maosong Sun. Minicpm4: Ultra-efficient llms on end devices, 2025. URL https://arxiv.org/abs/2506.07900.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Chenxu Wang, Yilin Lyu, Zicheng Sun, and Liping Jing. Continual gradient low-rank projection fine-tuning for LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14815–14829, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.721. URL https://aclanthology.org/2025.acl-long.721/.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55006–55021. Curran Associates, Inc., 2023. doi: 10.52202/075280-2400. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ac662d74829e4407ce1d126477f4a03a-Paper-Conference.pdf.

APPENDIX

A IMPLEMENTATION DETAILS

We give the full estimator behind the two stages of Section 3. Algorithm 1 combines representation fitting, skill-direction extraction, and prediction for a new prompt–response pair.

Algorithm 1 Learning skill directions and predicting skill labels

Require: Base checkpoint $\boldsymbol{\theta}_0$; labeled prompt–response pairs $\mathcal{D}_{\mathrm{tr}}, \mathcal{D}_{\mathrm{val}}$; fixed LoRA-$A$ matrices; layer ranks $\{\bar{r}^{(\ell)}\}_{\ell=1}^{m}$.

Ensure: Frozen factor maps $\{\boldsymbol{D}^{(\ell)}\}$, predictor $h_{\boldsymbol{\psi}}$, thresholds $\{\tau_k\}$, and skill directions $\{\widehat{\boldsymbol{S}}^{(\ell)}\}$ for Algorithm 2.

Freeze pretrained weights and LoRA-A; initialize every LoRA-B to zero.
for each $(u_i, v_i, \boldsymbol{\phi}_i^{\mathrm{obs}})$ in $\mathcal{D}_{\mathrm{tr}} \cup \mathcal{D}_{\mathrm{val}}$ **do**
    Clear gradients and compute the mean response-token loss $\mathcal{L}_i(\boldsymbol{\theta}_0)$.
    Store $\boldsymbol{x}_i^{(\ell)} \leftarrow \mathrm{vec}(\nabla_{\boldsymbol{B}^{(\ell)}} \mathcal{L}_i)$ for every layer; take no optimizer step.
end for
for each layer ℓ independently **do**
    Estimate the centered covariance on a training subsample.
    $\boldsymbol{D}^{(\ell)} \leftarrow \sqrt{p^{(\ell)}}[\widehat{\boldsymbol{v}}_1^{(\ell)}, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}^{(\ell)}}^{(\ell)}]$ using its leading eigenvectors.
    $\widetilde{\boldsymbol{f}}_i^{(\ell)} \leftarrow \boldsymbol{D}^{(\ell)\top} \boldsymbol{x}_i^{(\ell)} / p^{(\ell)}$ for training and validation examples.
    Fit $\widehat{\boldsymbol{\Lambda}}^{(\ell)}$ by least squares on all training $(\widetilde{\boldsymbol{f}}_i^{(\ell)}, \boldsymbol{x}_i^{(\ell)})$ pairs.
end for
Concatenate layer factors to obtain $\widetilde{\boldsymbol{f}}_i$.
Fit a standardizer T on training factors; define $h_{\boldsymbol{\psi}}(\widetilde{\boldsymbol{f}}) = \mathrm{MLP}_{\boldsymbol{\psi}}(T(\widetilde{\boldsymbol{f}}))$.
Train with class-balanced binary cross-entropy; select the checkpoint by validation loss.
Set the selected predictor to evaluation mode.
Select per-skill thresholds $\{\tau_k\}$ by validation sample Jaccard.
for each skill $k = 1, \ldots, q$ **do**
    $\widehat{\boldsymbol{c}}_k \leftarrow \frac{1}{n_k^+} \sum_{i \in \mathcal{D}_{\mathrm{tr}}: \phi_{ik}^{\mathrm{obs}}=1} \nabla_{\widetilde{\boldsymbol{f}}_i} h_{\boldsymbol{\psi},k}(\widetilde{\boldsymbol{f}}_i)$, including the derivative of T.
    Partition $\widehat{\boldsymbol{c}}_k$ by layer and set $\widehat{\boldsymbol{s}}_k^{(\ell)} \leftarrow \widehat{\boldsymbol{\Lambda}}^{(\ell)} \widehat{\boldsymbol{c}}_k^{(\ell)}$ for each ℓ.
end for
Freeze $\{\boldsymbol{D}^{(\ell)}\}$, $h_{\boldsymbol{\psi}}$, $\{\tau_k\}$, and $\widehat{\boldsymbol{S}}^{(\ell)} = [\widehat{\boldsymbol{s}}_1^{(\ell)}, \ldots, \widehat{\boldsymbol{s}}_q^{(\ell)}]$.
Prediction for a new prompt–response pair (u, v)**:**
Extract $\{\boldsymbol{x}^{(\ell)}\}$ at the same fixed checkpoint and LoRA initialization.
$\widetilde{\boldsymbol{f}} \leftarrow \mathrm{concat}_\ell(\boldsymbol{D}^{(\ell)\top} \boldsymbol{x}^{(\ell)} / p^{(\ell)})$; $\boldsymbol{\pi} \leftarrow \sigma(h_{\boldsymbol{\psi}}(\widetilde{\boldsymbol{f}}))$.
return $\widehat{\phi}_k = \mathbf{1}\{\pi_k \geq \tau_k\}$ for $k = 1, \ldots, q$.

A.1 FROM A COMPOSITIONAL EXAMPLE TO SKILL DIRECTIONS

Figure 4 gives a concrete view of the full construction. The example solution combines Factoring Techniques with Function Transformations and Symmetry, so its response gradient contains evidence for both labels. GSD first represents that gradient with layer-wise factors, then the supervised recovery model associates the concatenated factors with the two annotated skills. Although the contribution of each layer varies, aggregating across depth yields positive scores for both skills. The example illustrates why the method estimates structure layer by layer but recovers a compositional skill set from the joint representation.

A.2 PER-EXAMPLE GRADIENT EXTRACTION

We extract gradients at the original checkpoint with all pretrained parameters and LoRA-A matrices frozen, all LoRA-B matrices initialized to zero, and only LoRA-B requiring gradients. Each exam-

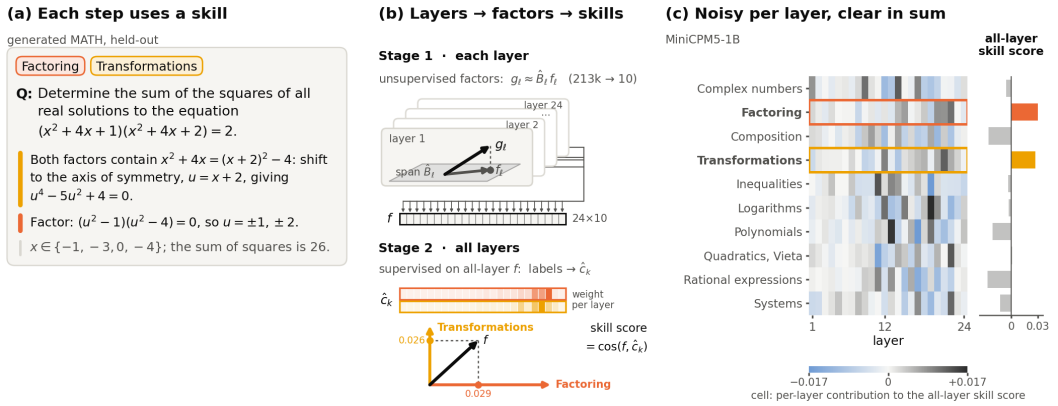

Figure 4: A compositional Math Gen example. (a) A held-out problem and solution requiring Factoring Techniques and Function Transformations and Symmetry. (b) GSD converts the response gradient into layer-wise factors and then into skill scores. (c) MiniCPM5-1B assigns different contributions across layers, while their aggregate supports both annotated skills.

ple is processed separately using the mean loss over unmasked response tokens; the prompt remains in the input as context. We directly flatten each layer's LoRA-B gradient block using the fixed parameter order of $\boldsymbol{x}_i^{(\ell)}$. Extraction uses no optimizer step, gradient clipping, post hoc normalization, or averaging across examples. The implementation differentiates directly with respect to LoRA-B and never materializes the conceptual full gradients $\boldsymbol{G}_{i,j}^{(\ell)}$.

A.3 Diversified Projection Construction (Stage 1)

For each layer $\ell = 1, \ldots, m$, fix a training subsample $\mathcal{I}'_\ell \subset \{1, \ldots, n_{\text{train}}\}$ of size n'_ℓ (a seeded 10% random draw from the training split) and form the centered sample covariance

$$\bar{\boldsymbol{x}}'_\ell = \frac{1}{n'_\ell} \sum_{i \in \mathcal{I}'_\ell} \boldsymbol{x}_i^{(\ell)}, \qquad \widehat{\boldsymbol{\Sigma}}'_\ell = \frac{1}{n'_\ell} \sum_{i \in \mathcal{I}'_\ell} \left(\boldsymbol{x}_i^{(\ell)} - \bar{\boldsymbol{x}}'_\ell\right)\left(\boldsymbol{x}_i^{(\ell)} - \bar{\boldsymbol{x}}'_\ell\right)^\top.$$

Let $\widehat{\boldsymbol{v}}_1^{(\ell)}, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}^{(\ell)}}^{(\ell)}$ be the leading $\bar{r}^{(\ell)}$ eigenvectors of $\widehat{\boldsymbol{\Sigma}}'_\ell$ and set

$$\boldsymbol{D}^{(\ell)} = \sqrt{p^{(\ell)}}\left[\widehat{\boldsymbol{v}}_1^{(\ell)}, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}^{(\ell)}}^{(\ell)}\right].$$

The same $\boldsymbol{D}^{(\ell)}$ constructs factors on both the train and test splits, so the projection is fixed from training. It satisfies the boundedness, exogeneity, and significance conditions of Fan & Gu (2024, Def. 3), under which the working number of factors $\bar{r}^{(\ell)} \geq r^{(\ell)}$ may overestimate the true count without bias.

B Protected Dataset and Skill-Recovery Protocol

This appendix describes the Generated-MATH protected dataset, its frozen splits, and the gradient-based skill-recovery protocol used in Section 4. Generated-MATH provides explicit, compositional skill labels and an objective answer-accuracy metric. The protected training split is used to estimate the skill representation, and held-out examples are used for skill prediction and post-training protection evaluation as specified below.

Skill vocabulary and selection. Following the LLM-based skill-labeling and semantic-grouping approach of Didolkar et al. (2024), we first use Gemini 3.1 Pro to annotate problems in the original MATH dataset (Hendrycks et al., 2021) with mathematical skill labels. We then use Gemini 3.1 Pro to group similar labels into clusters, forming a consolidated skill vocabulary. From the Algebra subject's vocabulary of 40 topics, Gemini 3.1 Pro selects ten skills with moderate scope and broad

coverage, preserving their names from the vocabulary. The selected skills are Complex Numbers; Factoring Techniques; Function Composition and Iteration; Function Transformations and Symmetry; Inequality Solving and Constraints; Logarithms; Polynomial Operations; Quadratic Solving and Vieta's Formulas; Rational Expressions and Equations; and Systems of Equations. These skills are fixed before generating the Math Gen questions.

Question generation. The exact selection and generation prompts appear in Section H. We form 30 generation groups: the ten individual skills, ten randomly selected pairs, and ten randomly selected triples. Pair and triple selection uses seed 20260506. For each group, we request 1,000 questions from Qwen3.5-27B (Team, 2026), giving 30,000 generation requests. Each prompt specifies the target skills and asks for a contest-style Algebra question at MATH level 3–5 difficulty, a worked solution, a single gradable final answer, and an explanation of how the target skills are used. For multi-skill groups, the prompt requires an integrated solution that uses all designated skills. The model also returns a difficulty annotation and five quality-check flags. Generation uses thinking mode with temperature 1.0, top-p 0.95, top-k 20, and a maximum of 32,768 output tokens per request.

Validation, deduplication, and labels. Structured-output validation checks the requested subject and skill list, nonempty question, solution, and answer fields, and valid difficulty and quality-check fields. The retained outputs have all five model-reported quality flags set to true. This produces 22,386 accepted records. Deduplicating question text within each of the single-, two-, and three-skill categories leaves 7,773, 6,840, and 7,619 questions, respectively, for a total of 22,232. Each question's generation target list defines its ten-dimensional multi-hot label, with one, two, or three positive entries. The saved Qwen3.5-27B solution provides the reference completion used for gradient extraction and protected-task replay.

Frozen data splits. We stratify the data by skill-count category and target group. A seed-0 split reserves 20% for testing; a seed-1 split assigns 12.5% of the remaining training partition to validation. The final partitions contain 15,562 training, 2,224 validation, and 4,446 test examples, approximately 70/10/20% of the dataset. We freeze these indices across models and methods. Skill-representation fitting and replay use the training partition; checkpoint and threshold selection use validation; skill prediction is reported on the full test split. Continual-learning evaluations use fixed subsets of that test split, as specified in Section 4.3.

Use in skill recovery and continual learning. All ten selected skills occur in at least $1{,}000$ examples across the full dataset. This dataset tests compositional skill recovery rather than only problem-level classification. Because a problem may require several algebra skills, its multi-hot annotation specifies which skills should be recovered from the response gradient. The same annotations define the protected skill space used during continual learning; they are not used to score the final answer.

Each problem is paired with its saved `Qwen/Qwen3.5-27B` solution. That response is the gradient target for every evaluated base model: it is not regenerated by MiniCPM5-1B, SmolLM3-3B, or Qwen3.5-2B and is not the original MATH reference solution. Gradients are computed only on assistant-response tokens. Skill protection in the independent runs is measured by boxed-answer accuracy on a fixed 500-example subset of the frozen test split, sampled with seed 0. Thus, skill prediction uses the problem's multi-hot labels, while protection uses answer correctness on unseen problems.

B.1 Raw-Gradient Geometry

Figure 5 visualizes the same matched examples and models as Figure 2, but before skill alignment. The annotated skills overlap substantially in raw-gradient space, with the strongest mixing for MiniCPM5-1B. Compared with the more coherent regions in the learned representation, these qualitative visualizations indicate that the skill organization is difficult to read directly from raw gradients and becomes more explicit after GSD alignment.

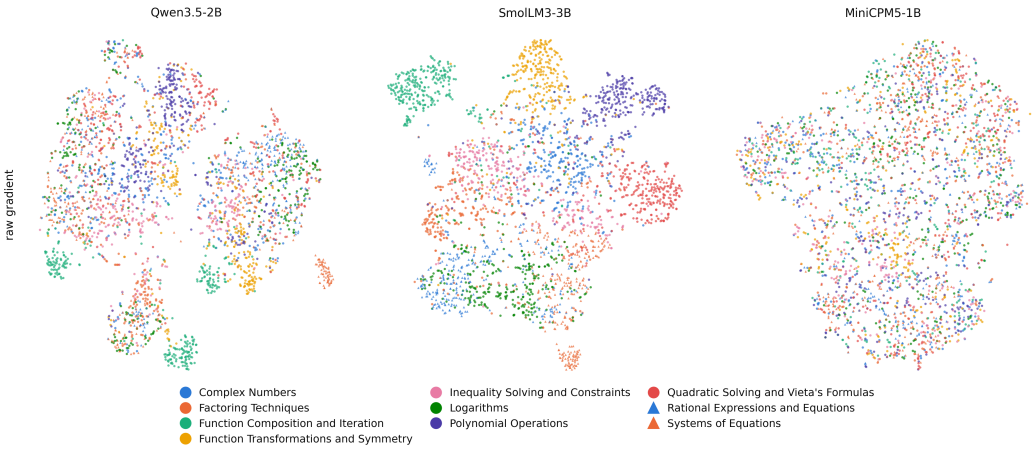

Figure 5: Two-dimensional t-SNE visualizations of matched Math Gen examples in raw-gradient coordinates. Columns show Qwen3.5-2B, SmolLM3-3B, and MiniCPM5-1B; colors and markers denote annotated skills. The examples match those in the skill-aligned visualization in Figure 2. Axes are local to each panel.

B.2 Layer-wise Distribution of Skill Directions

The recovered directions are distributed across depth but place their largest shares in upper layers (Figure 6). The three largest shares occur at layers 21–23 for Qwen3.5-2B, 30–32 for SmolLM3-3B, and 20–22 for MiniCPM5-1B, while earlier layers retain nonzero weight. This pattern supports estimating factors layer-wise and aggregating across all layers rather than choosing a single block in advance.

C Continual-SFT Datasets

This section identifies the continuation datasets used in the experiments. Every source is converted to the experiment's common instruction–response representation. Classification and question-answering datasets use the registered answer as the response target, while instruction and dialogue datasets use the supplied assistant response.

Generated-MATH protection panel. The six continuation datasets are:

- **FinQA** (Chen et al., 2021) contains questions requiring numerical reasoning over financial reports and tables. Continuation performance is answer F1.
- **GPQA** (Rein et al., 2024) contains expert-written graduate-level multiple-choice questions in biology, physics, and chemistry. Continuation performance is answer accuracy.
- **HellaSwag** (Zellers et al., 2019) is a multiple-choice commonsense sentence-completion task. Continuation performance is answer accuracy.
- **LIMA** (Zhou et al., 2023) contains curated instruction–response demonstrations. Continuation performance is held-out assistant-response loss.
- **SQuAD** (Rajpurkar et al., 2016) is an extractive reading-comprehension dataset. Continuation performance is answer F1.
- **UltraChat** (Ding et al., 2023) contains multi-turn instructional conversations. Continuation performance is held-out assistant-response loss.

For each continuation task, seed 0 shuffles one held-out evaluation pool, reserves its first half for validation and learning-rate selection, and uses the disjoint second half for final testing. The final test sizes are 250 for FinQA, HellaSwag, SQuAD, and UltraChat; 45 for GPQA; and 103 for LIMA. Within each task, all evaluated methods use the same final-test records. The 500-example cap does

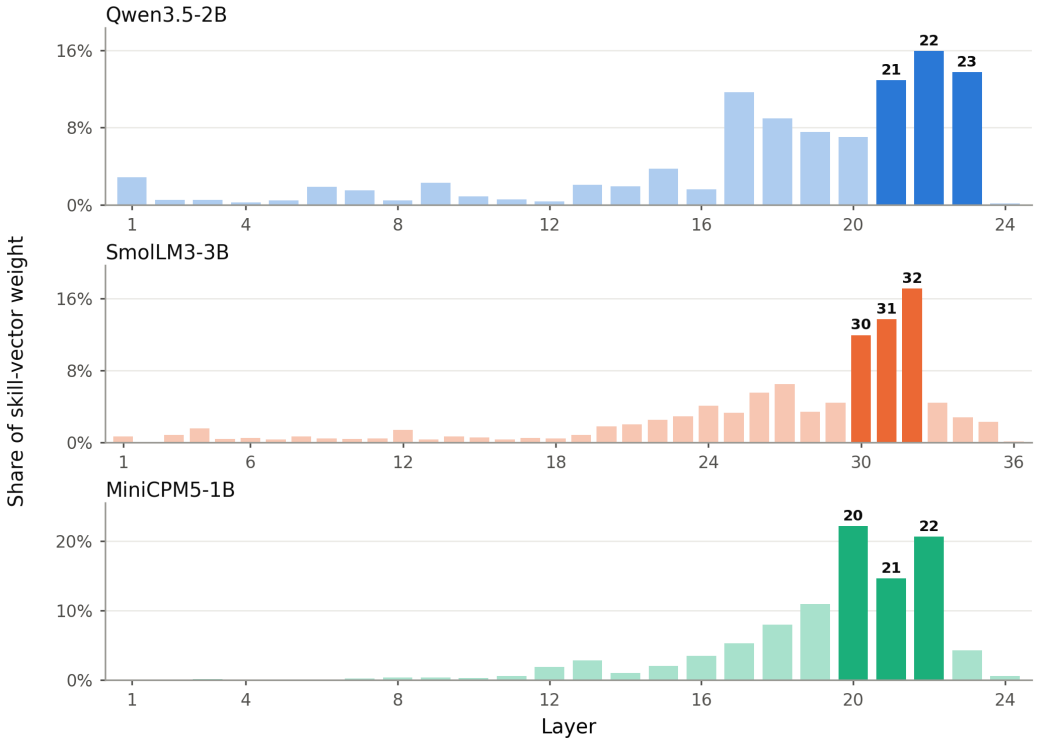

Figure 6: Distribution of recovered skill-vector weight across transformer layers. Bars report each layer's normalized share for Qwen3.5-2B, SmolLM3-3B, and MiniCPM5-1B; darker bars identify the three largest shares within each model.

not truncate any of these test halves. MiniCPM5-1B uses a 512-token generation cap for FinQA and SQuAD; SmolLM3-3B and Qwen3.5-2B use a 64-token cap.

Table 3 gives the evaluated-record hashes used to verify that the methods compare on the same final-test examples.

Replay construction and evaluation alignment. Vanilla, Mean-Grad, and Ours train only on continuation examples. Replay and Replay+Ours use the same 20% protected-token replay mixture; protected validation and test rows are never replayed. The replay target construction is identical for Replay and Replay+Ours. Ours and Replay+Ours are the two variants of our skill-projection method.

Protected-task and continuation-task results for each model–dataset setting come from the same final checkpoint. All methods within a setting use the same selected learning rate. Protected methods use the same protected final-test split, and continuation-task methods use test partitions with identical fingerprints. Reported independent results come from final-test artifacts; Test-200 monitoring and sequential-stage validation values are not substituted. Accuracy and F1 are expressed as percentages, while loss remains on its native scale. LIMA and UltraChat loss is mean response-token negative log-likelihood, so comparisons of its raw value are most meaningful within a model.

D Continual Fine-Tuning Protocol

This appendix specifies the continual-learning pipeline of Section 3.3: how the frozen skill basis is built, how the projection is applied during SFT, and how retention and acquisition are measured. Algorithm 2 uses the skill directions returned by Algorithm 1. Setting the replay ratio to zero or 0.20 yields Ours or Replay+Ours, respectively.

Table 3: Continuation-task record fingerprints. SHA-256 hash chunks in each cell are concatenated from top to bottom.

Continual task	Pool	Records	Metric	SHA-256
FinQA	500	250	F1	22280da0b85d6ed9 4d5caaba33f8cec1 ae7b85e4207fb98b ef76e69b4a3b9536 23a330d81ce05379 b82e6b5d7120d24b
GPQA	89	45	Accuracy	6fe2e6eb8127f0ae ea90ae641eed37e9 f037b40c60da3acb 6bc32982bcc45f5d 7bf6b1b416a45227
HellaSwag	500	250	Accuracy	8c09c61b5a7a11b4 825fabe41d076723 0f71349286acf4de 5ea51b41370c06df 846988db5178c6cf
LIMA	206	103	Loss	bf8987fb121065fc fe7e9604248a68c0 4336d15d1d2e4342 cdaeda782acd62d2 3ab5765107804b80
SQuAD	500	250	F1	833a28e0f50f0dde 0cf4269f92eefdbb 0f9274a57ea0de3e
UltraChat	500	250	Loss	

D.1 Frozen Skill Basis (Stage 2)

The projection basis is built once from the cached Stage-1 loadings and Stage-2 skill directions, then frozen. Because the factors and loadings are estimated per layer (Appendix A), we map each factor-space skill direction into the matching LoRA-B coordinates. For layer ℓ we form

$$\widehat{\boldsymbol{S}}^{(\ell)} = \widehat{\boldsymbol{\Lambda}}^{(\ell)} \widehat{\boldsymbol{C}}^{(\ell)} \in \mathbb{R}^{p^{(\ell)} \times q},$$

where $\widehat{\boldsymbol{C}}^{(\ell)}$ contains the rows of $\widehat{\boldsymbol{C}}$ associated with layer ℓ. For a selected skill set $\mathcal{K}$, retain the corresponding columns of $\widehat{\boldsymbol{S}}^{(\ell)}$. A thin singular value decomposition, retaining its numerical rank ρ_ℓ, gives $\boldsymbol{Q}^{(\ell)} \in \mathbb{R}^{p^{(\ell)} \times \rho_\ell}$ with orthonormal columns spanning the protected subspace at that layer. The main retention experiments protect all q skills; selected subsets use the same construction with restricted $\mathcal{K}$. The bases are frozen throughout continuation training.

D.2 Skill Projection during LoRA-SFT

We freeze LoRA-A and the pretrained parameters and train LoRA-B. At the beginning of each continuation stage, we save the flattened adapter parameters $\boldsymbol{b}^{(\ell)}_{\text{start}}$. After gradient accumulation and each optimizer step, let $\boldsymbol{\Delta}^{(\ell)} = \boldsymbol{b}^{(\ell)}_{\text{cand}} - \boldsymbol{b}^{(\ell)}_{\text{start}}$ be the cumulative change from that fixed stage anchor. We assign

$$\boldsymbol{b}^{(\ell)} \leftarrow \boldsymbol{b}^{(\ell)}_{\text{start}} + \boldsymbol{\Delta}^{(\ell)} - \boldsymbol{Q}^{(\ell)} \big(\boldsymbol{Q}^{(\ell)\top} \boldsymbol{\Delta}^{(\ell)} \big).$$

The projection strength is one. Projection occurs once per optimizer step, after accumulation over four micro-batches. Each layer costs $O(p^{(\ell)} \rho_\ell)$ for the projection. Replay+Ours applies the same operation after optimizing the combined new-task and replay loss. Independent runs start from the original checkpoint; sequential runs carry the adapter across tasks and reset the projection anchor at each stage.

Training and evaluation protocol. The independent experiments evaluate MiniCPM5-1B, SmolLM3-3B, and Qwen3.5-2B with seed 0. For each model–dataset setting, Vanilla continuation-

Algorithm 2 Continual learning with skill projection and optional replay

Require: Base checkpoint $\boldsymbol{\theta}_0$; skill directions $\{\widehat{\boldsymbol{S}}^{(\ell)}\}$ from Algorithm 1; protected skills $\mathcal{K}$; continuation tasks $\mathcal{D}_1, \ldots, \mathcal{D}_T$; the same LoRA-$A$ matrices; optional protected training buffer $\mathcal{R}$ and replay token ratio β.
Ensure: Adapted LoRA-B parameters satisfying the selected subspace constraints.

```
Freeze pretrained weights and LoRA-A; initialize LoRA-B to zero.
for each layer ℓ do
    Q^(ℓ) ← orth(Ŝ^(ℓ)_{:,K}) using a thin SVD and retaining its numerical rank.
end for
Freeze {Q^(ℓ)} for all continuation stages.
for each task t = 1, ..., T do
    Save b^(ℓ)_start ← vec(B^(ℓ)) for every layer.
    for each optimizer step on D_t do
        Draw new-task micro-batches and, if β > 0, replay from R under the protected-to-new
completion-token budget β.
        Accumulate gradients of the mean completion-token loss over new-task and replay to-
kens.
        Apply the optimizer step to LoRA-B, producing {B^(ℓ)_cand}.
        for each layer ℓ do
            Δ^(ℓ) ← vec(B^(ℓ)_cand) − b^(ℓ)_start.
            b^(ℓ) ← b^(ℓ)_start + Δ^(ℓ) − Q^(ℓ)(Q^(ℓ)⊤ Δ^(ℓ)).
            Reshape b^(ℓ) into B^(ℓ) in the same fixed parameter order.
        end for
    end for
end for
return the adapted model.
```

Variants. Ours uses $\beta = 0$; Replay+Ours uses $\beta = 0.20$ with token-proportional loss. Independent runs invoke the algorithm separately for each task from the original checkpoint. Sequential runs carry the adapter across tasks while retaining the same frozen bases.

task validation selects a learning rate from $\{10^{-5}, 5\times10^{-5}, 10^{-4}\}$; that selected rate is shared by all five methods. Each continuation task starts independently from the raw checkpoint. The separate bidirectional sequential experiment uses MiniCPM5-1B, half-sized task-training samples, and a fixed learning rate of 5×10^{-5}. Its intermediate stage scores use Validation-200 and its final scores use Test-200, so they are not combined with the independent Test-500 scores.

Protected-task decoding. Math Gen evaluation uses each model's native chat template with thinking disabled. MiniCPM5-1B uses temperature 0.7, top-p 0.95, and an 8,192-token output cap; SmolLM3-3B uses 0.6, 0.95, and the same cap. These models use Math-Verify to check boxed-answer equivalence. The reported Qwen3.5-2B runs use greedy decoding with a 16,384-token cap and the earlier local boxed-answer scorer.

E Sequential Continual Learning

We additionally evaluate MiniCPM5-1B by fine-tuning the same adapter across all six tasks at a fixed learning rate of 5×10^{-5} while keeping the protected skill basis fixed. Figure 7 tracks Math Gen accuracy through forward and reverse task orders, and Table 4 reports final test accuracy. On the separate Test-200 panel, Ours raises final accuracy from 14.5% to 27.0% in the forward order and from 4.0% to 22.0% in reverse, relative to Vanilla. Replay+Ours reaches 32.0% and 28.0%, exceeding Replay by 9.0 and 14.0 points, respectively, and attaining the highest final accuracy in both orders. Meanwhile, the displayed new-task metrics remain close to the normalized Vanilla reference immediately after training (approximately 95–100%; Figure 7, lower panel). Thus, the retention gains do not arise from a broad failure to acquire the displayed continuation tasks.

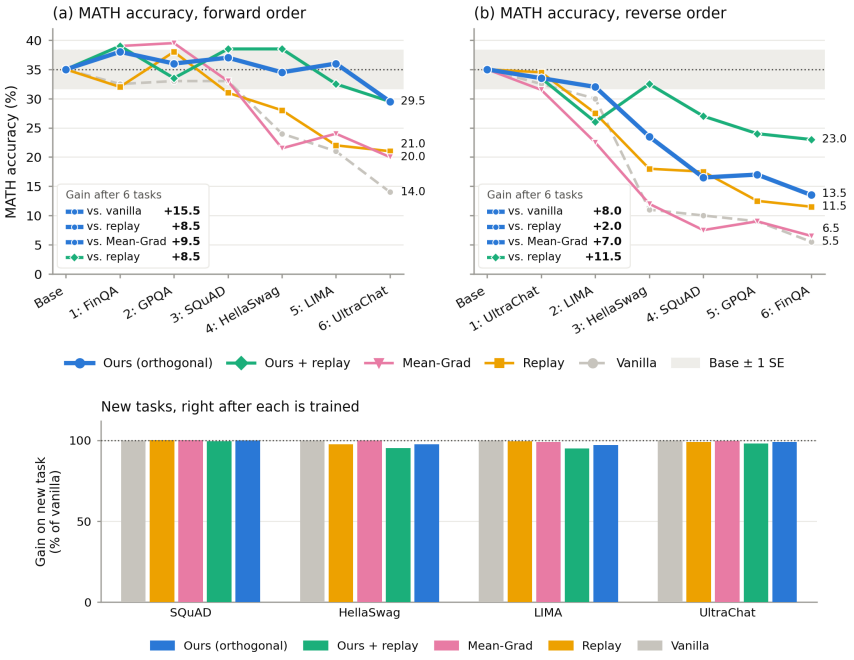

Figure 7: Protected Math Gen retention and new-task acquisition during sequential continual learning on MiniCPM5-1B. Panels (a) and (b) show boxed-answer accuracy (%) on the fixed Math Gen Validation-200 panel after each stage of the forward and reverse task orders. "Base" is the original checkpoint; the dotted line and shaded band show its accuracy and ± 1 standard error. The forward order is FinQA, GPQA, SQuAD, HellaSwag, LIMA, and UltraChat; the reverse order inverts it. The lower panel shows each displayed new-task metric immediately after its training stage as a percentage of the corresponding Vanilla score, normalized to 100%. Table 4 reports final protected accuracy on the separate Test-200 panel.

Table 4: Final Math Gen accuracy (%) for MiniCPM5-1B after six sequential tasks; higher is better. Task orders follow Figure 7. Evaluation uses Test-200, separate from the validation trajectories and independent test evaluation; base accuracy is 32.5%. Mean-Grad is mean-gradient projection. Ours and Replay+Ours use skill projection without and with replay, respectively. Underlining marks each row's best result.

Order	Vanilla	Mean-Grad	Replay	Ours	Replay+Ours
Forward	14.5	16.0	23.0	27.0	<u>32.0</u>
Reverse	4.0	9.0	14.0	22.0	<u>28.0</u>

F Detailed Compute and Adaptation Cost

Table 5 reports adaptation costs and reusable basis storage.

G Continuation-Task Performance

Table 6 reports performance on each continuation dataset after independent fine-tuning. These scores come from the same final checkpoints as the Math Gen retention results in Table 2. Each model–task setting uses the learning rate selected by Vanilla's continuation-task validation score, shared across all five methods. The held-out test partitions and metrics are described in Section C.

Table 5: Compute, memory, and time comparison. Panel A reports medians over the six independent continuation tasks. Panel B sums processed tokens and adaptation time over six sequential stages and reports the maximum peak memory across stages. Panel C reports the one-time persistent basis for each projection method; the basis for Ours is also shared with Replay+Ours. Tokens count non-padding positions processed by forward/backward passes and are a workload proxy rather than FLOPs. Times include training, projection, and replay-schedule construction, but exclude basis construction, evaluation, model loading, and checkpoint saving; they should be compared only within a model and campaign.

Setting	Method	Tokens (M)	Peak GiB	Time (min) ↓
A. Independent adaptation (median across six tasks)				
Qwen3.5-2B	Vanilla	0.354	14.07	7.8
	Mean-Grad	0.354	14.10	7.9
	Ours	0.354	14.32	8.0
	Replay	0.512	18.21	11.2
	Replay+Ours	0.512	18.33	11.1
SmolLM3-3B	Vanilla	0.451	11.72	3.0
	Mean-Grad	0.451	11.78	3.0
	Ours	0.451	12.20	3.0
	Replay	0.615	13.88	3.7
	Replay+Ours	0.615	14.36	3.6
MiniCPM5-1B	Vanilla	0.354	6.96	2.0
	Mean-Grad	0.354	6.97	2.1
	Ours	0.354	7.05	2.0
	Replay	0.508	9.00	2.5
	Replay+Ours	0.508	9.08	2.5
B. SmolLM3-3B six-stage adaptation				
Forward	Vanilla	2.694	19.26	9.9
	Mean-Grad	2.694	19.38	10.1
	Ours	2.694	19.80	10.0
	Replay	3.235	23.02	11.8
	Replay+Ours	3.235	23.56	11.8
Reverse	Vanilla	2.694	19.26	10.1
	Mean-Grad	2.694	19.32	10.0
	Ours	2.694	19.74	10.1
	Replay	3.235	23.02	11.6
	Replay+Ours	3.235	23.50	11.6
C. Reusable one-time setup				
Model	Method	Setup examples	Gradient rows	Basis (MiB)
Qwen3.5-2B	Ours	15,562	22,232	255.5
	Mean-Grad	15,562	15,562	29.7
SmolLM3-3B	Ours	15,562	22,232	492.0
	Mean-Grad	15,562	15,562	59.7
MiniCPM5-1B	Ours	15,562	22,232	165.0
	Mean-Grad	15,562	15,562	19.5

Table 6: Continuation-task test performance after independent fine-tuning. F1 and accuracy are percentages (higher is better); loss is mean assistant-response-token negative log-likelihood (lower is better). Ours and Replay+Ours are our skill-projection variants without and with replay. Underlining marks the best value in each row at the displayed precision, including ties. Metrics are reported separately because their scales differ. FinQA and SQuAD use generation caps of 512 tokens for MiniCPM5-1B and 64 tokens for SmolLM3-3B and Qwen3.5-2B; method comparisons use the same evaluation settings within each model.

Task	Metric	Vanilla	Mean-Grad	Replay	Ours	Replay+Ours
MiniCPM5-1B						
FinQA	F1 ↑	3.6	3.2	2.8	3.2	2.4
GPQA	Accuracy ↑	22.2	22.2	20.0	20.0	22.2
SQuAD	F1 ↑	78.3	78.7	78.0	77.3	77.4
HellaSwag	Accuracy ↑	48.4	48.4	48.4	48.4	48.0
LIMA	Loss ↓	2.427	2.428	2.428	2.429	2.431
UltraChat	Loss ↓	1.203	1.203	1.204	1.204	1.205
SmolLM3-3B						
FinQA	F1 ↑	2.6	2.6	2.2	2.8	4.2
GPQA	Accuracy ↑	24.4	20.0	20.0	24.4	26.7
SQuAD	F1 ↑	84.6	85.0	84.9	85.5	84.9
HellaSwag	Accuracy ↑	74.4	74.0	74.0	73.6	74.0
LIMA	Loss ↓	1.941	1.941	1.941	1.941	1.943
UltraChat	Loss ↓	0.947	0.947	0.947	0.947	0.948
Qwen3.5-2B						
FinQA	F1 ↑	2.8	2.8	2.0	2.8	2.0
GPQA	Accuracy ↑	28.9	28.9	28.9	31.1	28.9
SQuAD	F1 ↑	88.3	88.9	88.0	88.5	87.4
HellaSwag	Accuracy ↑	62.4	62.4	62.8	62.4	62.4
LIMA	Loss ↓	2.038	2.037	2.036	2.039	2.037
UltraChat	Loss ↓	1.004	1.004	1.004	1.004	1.004

H Data Construction Prompts

This section records the prompts used for Math Gen skill selection and question and reference-solution generation. The blocks reproduce saved prompt text or instantiated templates; example skills and sample indices are identified below. Section B describes how their outputs are used to construct the dataset.

H.1 Skill Vocabulary Selection

After the original MATH problems are annotated and similar labels are grouped by Gemini, the following saved prompt selects ten skills from the resulting 40-topic Algebra vocabulary. It is sent to Gemini 3.1 Pro (`gemini-3.1-pro-preview`) with a JSON response schema and an 8,192-token output limit, without an additional system message. The saved prompt record covers this selection step; it does not contain the earlier annotation and grouping prompts.

Gemini: Algebra skill selection

```
You are a mathematics education expert selecting representative
   skills for a math-question generation/evaluation pipeline.

Subject: Algebra
Number of available skills: 40
Target number to select: exactly 10
```

```
Select exactly 10 skills that are at a useful middle level of
    specificity.

Definitions:
- Too general: broad umbrella labels that could cover many
    unrelated problem types, prerequisite arithmetic only, or
    generic actions such as "Algebraic Modeling" when the
    subject has more informative alternatives.
- Too specific: narrow one-off techniques, contest tricks, or
    tiny subcases that would not support a varied set of
    generated problems.
- Good middle level: a recognizable mathematical technique,
    theorem family, object class, or problem type that is
    specific enough to guide question generation but broad
    enough to support multiple distinct problems.

Selection constraints:
- Copy selected skill names exactly as written in the input list
    .
- Select exactly 10 distinct skills.
- Prefer coverage across the subject rather than ten near-
    duplicates.
- Do not invent, rename, merge, split, or paraphrase any skill.
- Do not select a skill only because it is familiar; judge the
    granularity of the skill name in this subject.

Return JSON with:
- selected_skills: exactly 10 objects, each with "skill" and a
    one-sentence "rationale".
- rejected_too_general: a few examples from the input list that
    you rejected as too broad.
- rejected_too_specific: a few examples from the input list that
     you rejected as too narrow.
- selection_notes: one brief note.

Input skill list:
1. Basic Arithmetic Operations
2. Fraction and Rational Number Arithmetic
3. Algebraic Isolation and Simplification
4. Algebraic Expansion and Identities
5. Factoring Techniques
6. Systems of Equations
7. Function Evaluation and Notation
8. Function Composition and Iteration
9. Inverse Functions
10. Function Transformations and Symmetry
11. Domain and Range
12. Exponents and Powers
13. Radicals and Square Roots
14. Logarithms
15. Sequences
16. Series and Summations
17. Linear Equations and Modeling
18. Parallel and Perpendicular Lines
19. Intercepts
20. Quadratic Features and Optimization
21. Quadratic Solving and Vieta's Formulas
22. Absolute Value and Distance
23. Polynomial Operations
```

```
24. Rational Expressions and Equations
25. Complex Numbers
26. Inequality Solving and Constraints
27. Modeling and Mathematical Logic
28. Ratios and Proportions
29. Rates and Work
30. Percentage and Conversions
31. Floor and Ceiling Functions
32. Circle Geometry
33. Coordinate and Rectilinear Geometry
34. Custom Operations
35. Number Theory
36. Combinatorics and Probability
37. Financial Modeling
38. Estimation and Verification
39. Special Algebraic Properties
40. Unit Conversions
```

The selected names are listed in Section B. For generated questions, multi-hot labels are copied directly from `target_skills`; no separate model call assigns labels after question generation. The gradient-based skill predictor is trained on these labels and does not use a natural-language classification prompt.

H.2 Synthetic Questions and Reference Solutions

Qwen3.5-27B receives a single user message, without a separate system message, and produces the question, worked solution, and final answer in one response. The following two blocks instantiate the generation template with representative skills and sample index 0. Each actual request substitutes its own target skills and sample index in both the displayed skill list and the required JSON values. The final JSON-only instruction is part of every request.

Qwen3.5-27B: single-skill generation

```
You are writing one original contest-style mathematics problem
    for a dataset.

Target subject: Algebra
Target skills:
- Factoring Techniques
Sample index for diversity: 0

Write exactly one MATH-dataset-style problem whose main solution
     path requires the target skills. The problem should be self
    -contained, mathematically well-posed, and have a single
    unambiguous final answer.

Difficulty target:
- Match the style and difficulty range of the MATH dataset: high
    -school competition mathematics, not routine classroom drill
    .
- The problem should usually require a non-obvious idea, a
    careful setup, or a multi-step chain of reasoning.
- Aim for roughly MATH level 3-5. Avoid trivial plug-and-chug
    questions, but also avoid research-level or olympiad-final
    problems that require obscure theorems.
- The final answer should usually be a number, expression,
    ordered pair, polynomial, set, probability, or other concise
     exact mathematical object.
```

```
- It is fine for the solution to use standard contest techniques
    , but all needed definitions and conditions must appear in
    the problem.

Prompt design requirements:
- The listed target skill must be central to the solution. A
    solver should naturally use it, not merely encounter it as
    decoration.
- Keep the problem within high-school contest mathematics
    appropriate for the subject. Avoid graduate-level material.
- Avoid copying standard textbook examples or famous contest
    problems verbatim.
- Use concrete numbers and conditions. Avoid vague requests like
     "prove a property" unless the answer can still be graded
    cleanly.
- Prefer problems that can be solved in 4-10 meaningful steps.
- Ensure all computations are exact and the final answer is
    clean.
- Do not mention target skill names in the problem text unless
    that terminology is normally part of the mathematical
    statement.
- Do not ask for multiple unrelated quantities.
- Do not include diagrams or references to missing figures.

Return a JSON object with exactly these fields:
- subject: copy the target subject string.
- target_skills: copy the target skills array exactly and in the
     same order.
- question: the problem statement only.
- solution: a clear worked solution.
- final_answer: the final answer as a concise string.
- skill_usage: one or two sentences explaining where each target
     skill is required.
- difficulty: one of "easy", "medium", "hard".
- quality_checks: an object with boolean fields "
    uses_all_target_skills", "skills_are_integrated", "
    single_final_answer", "self_contained", and "
    no_missing_diagram".

The JSON values for subject and target_skills must exactly match
    :
subject = "Algebra"
target_skills = ["Factoring Techniques"]


Return ONLY the JSON object -- no surrounding prose, no markdown
     fences, no commentary before or after. Start your response
    with `{` and end with `}`.
```

For two or three target skills, the integration requirement asks that all skills interact in one coherent problem. The following two-skill example gives the complete prompt. Three-skill requests use the same wording, with a third skill in the list and in the JSON array.

Qwen3.5-27B: multi-skill generation

```
You are writing one original contest-style mathematics problem
    for a dataset.

Target subject: Algebra
```

```
Target skills:
- Systems of Equations
- Function Transformations and Symmetry
Sample index for diversity: 0

Write exactly one MATH-dataset-style problem whose main solution path requires the target skills. The problem should be self-contained, mathematically well-posed, and have a single unambiguous final answer.

Difficulty target:
- Match the style and difficulty range of the MATH dataset: high-school competition mathematics, not routine classroom drill.
- The problem should usually require a non-obvious idea, a careful setup, or a multi-step chain of reasoning.
- Aim for roughly MATH level 3-5. Avoid trivial plug-and-chug questions, but also avoid research-level or olympiad-final problems that require obscure theorems.
- The final answer should usually be a number, expression, ordered pair, polynomial, set, probability, or other concise exact mathematical object.
- It is fine for the solution to use standard contest techniques, but all needed definitions and conditions must appear in the problem.

Prompt design requirements:
- Every listed target skill must be central to the solution. The skills should interact in one coherent problem, not appear as disconnected subproblems.
- Keep the problem within high-school contest mathematics appropriate for the subject. Avoid graduate-level material.
- Avoid copying standard textbook examples or famous contest problems verbatim.
- Use concrete numbers and conditions. Avoid vague requests like "prove a property" unless the answer can still be graded cleanly.
- Prefer problems that can be solved in 4-10 meaningful steps.
- Ensure all computations are exact and the final answer is clean.
- Do not mention target skill names in the problem text unless that terminology is normally part of the mathematical statement.
- Do not ask for multiple unrelated quantities.
- Do not include diagrams or references to missing figures.

Return a JSON object with exactly these fields:
- subject: copy the target subject string.
- target_skills: copy the target skills array exactly and in the same order.
- question: the problem statement only.
- solution: a clear worked solution.
- final_answer: the final answer as a concise string.
- skill_usage: one or two sentences explaining where each target skill is required.
- difficulty: one of "easy", "medium", "hard".
- quality_checks: an object with boolean fields "
  uses_all_target_skills", "skills_are_integrated", "
```

```
    single_final_answer", "self_contained", and "
    no_missing_diagram".

The JSON values for subject and target_skills must exactly match
    :
subject = "Algebra"
target_skills = ["Systems of Equations", "Function
    Transformations and Symmetry"]


Return ONLY the JSON object -- no surrounding prose, no markdown
     fences, no commentary before or after. Start your response
    with `{` and end with `}`.
```

Generation enables thinking and uses temperature 1.0, top-p 0.95, top-k 20, min-p 0, presence penalty 1.5, repetition penalty 1.0, and at most 32,768 output tokens. The quality-check fields are generator-reported flags used during filtering; the validation and deduplication procedure is described in Section B. The saved solution supplies the reference completion for gradient extraction and replay.