# PRESERVE MORE, FREEZE LESS: LOSS-AWARE SUBSPACES FOR CONTINUAL LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Continually fine-tuning a pretrained language model requires acquiring new capabilities (plasticity) while retaining both its initial capabilities and those learned along the way (stability). Projection-based methods constrain parameter updates to protect directions associated with prior knowledge. Existing approaches select directions without measuring their importance for prior behavior, protect directions too coarsely, reducing plasticity, or store reference gradients, which does not scale to large models. We formulate an explicit preservation objective and derive a tractable approximation that identifies pairs of input and output directions in each linear transformation and estimates their importance from limited reference data. Building on this formulation, we develop SCLL, a projection-based continual learning method that protects only the most important pairs, improving the balance between preservation and adaptation. We evaluate SCLL on two continual learning benchmarks while protecting five initial capabilities. On TRACE, whose tasks differ in language, domain, and output format, SCLL improves average accuracy over the strongest projection baseline by 4 points while matching its retention and freezing fewer update directions; on Long CL, whose tasks require few new directions, it matches the strongest baseline. Protecting the selected initial capabilities also retains performance on related datasets not used for protection.

## 1 INTRODUCTION

Adapting a language model to new tasks should expand its capabilities without sacrificing those it has already learned. However, standard fine-tuning can cause catastrophic forgetting, eroding both pretrained capabilities and skills acquired through earlier adaptation. Continual learning (CL) develops methods that balance the *plasticity* to learn new capabilities with the *stability* to retain existing ones. We address this challenge when adapting a single model by updating its backbone weight matrices. We assume access to limited reference data representing the capabilities to preserve, but do not replay it during new-task training.

Projection-based CL methods constrain how model parameters change during training. They identify directions in parameter space associated with previously learned capabilities and project each update to be orthogonal to these *protected directions*, so that learning proceeds only in the remaining directions. Their effectiveness depends on which directions are protected: protecting too many limits plasticity, while protecting the wrong ones fails to prevent forgetting. Weight-based methods (Nayak et al., 2026) select directions from the singular vectors of weight matrices, without measuring their importance on reference data. Activation-based methods (Wang et al., 2021) protect input directions of reference activations across all outputs of a layer, which can be overly restrictive. Gradient-based methods (Farajtabar et al., 2020) protect reference gradients directly but store a model-sized vector per direction, which is impractical for large models. What is missing is a scalable way to protect only the directions that matter for prior behavior, leaving the rest free for new learning.

We introduce SCLL, a projection method that protects selected *input-output pairs* in each linear layer (weight matrix) of the backbone, each pair consisting of a direction in the layer's input space and a direction in its output space. Our key observation is that pairs sharing the same input direction can matter differently for prior behavior: changing the layer's output along one output direction may substantially change the reference loss, while changing it along another has little effect. Protecting only the first pair leaves the second available for new learning. SCLL estimates each pair's importance



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.

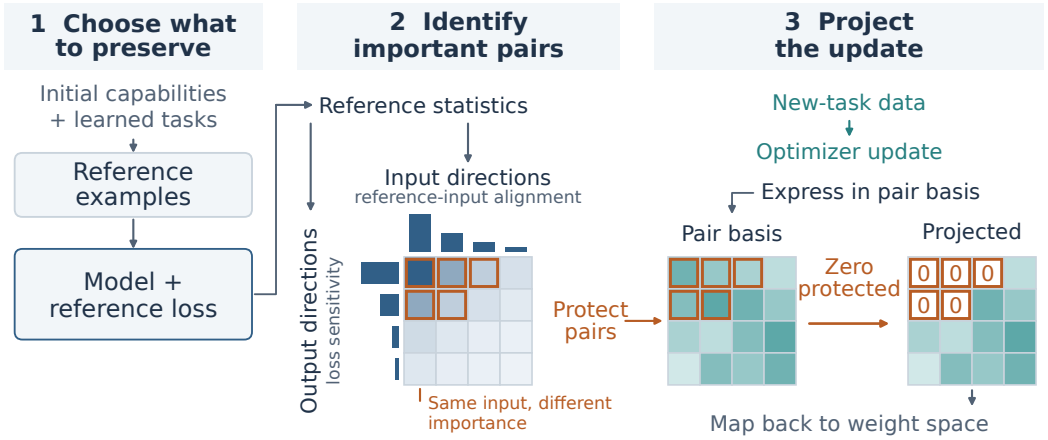


Figure 1: Overview of SCLL for one linear layer. (1) Reference examples from the initial capabilities and learned tasks define the reference loss to preserve. (2) Reference statistics give input directions, weighted by their alignment with reference inputs (top bars), and output directions, weighted by the sensitivity of the reference loss (side bars). Each cell is an input-output pair whose importance is the product of the two (shading), and the most important pairs (outlined) are protected. Pairs sharing an input direction can differ: in the first column, only the top two pairs are protected. (3) During new-task training, each optimizer update is expressed in the pair basis, the coefficients of the protected pairs are set to zero, and the result is mapped back to weight space. Grids are illustrative: protected pairs cover 75% of the energy (the total pair importance), and teal shading indicates the magnitude of update coefficients.

from reference data: a pair is important when reference inputs align strongly with its input direction and the reference loss is sensitive to changes along its output direction. Each pair defines a rank-one direction in the layer's weight space, and the directions of the selected pairs become the protected directions: during new-task training, SCLL projects each parameter update to be orthogonal to them, leaving the remaining directions free to change (Figure 1).

We derive both the pairs and their importance from an explicit preservation objective: the expected squared change in the reference loss over reference examples. We approximately decompose this objective into per-layer terms, each expressed through two small matrices of reference statistics: one over the layer's inputs and one over the sensitivity of the reference loss to its outputs. Their eigenvectors define the input and output directions of the pairs, and products of their eigenvalues give each pair's importance, making the intuition above precise. Because all pairs of a layer are formed from these shared directions, SCLL stores only two small factor matrices and the selected directions of each layer rather than a model-sized vector per protected direction. The derivation also applies to other preserved quantities, sensitivity approximations and ways of selecting the protected directions, providing a framework for designing projection methods beyond SCLL. In continual learning, SCLL combines statistics from the initial capabilities and each completed task, updating the protected directions as the model learns.

We evaluate SCLL on TRACE (Wang et al., 2023b) and Long CL (Wang et al., 2023a) using Qwen3.5-0.8B (Qwen Team, 2026), measuring new-task acquisition, retention of earlier tasks, and retention of five selected initial capabilities. On TRACE, whose tasks differ in language, domain, and output format, SCLL improves average accuracy over NSCL, the strongest baseline, by 4.01 points while matching its retention and freezing fewer update directions (52.1% vs. 66.5%). On Long CL, SCLL is ahead of NSCL on every task order, but by small margins; tracking the protected subspace shows that Long CL tasks add few new directions to protect, leaving little room for finer selection to help. SCLL has the higher mean initial-capability score on both benchmarks and, on five related datasets not used to construct protection, retains mean performance slightly above the pretrained model.

Our **contributions** are as follows:

- A *preservation framework* for projection-based CL methods, which derives protected directions and their importance from an explicit objective: the expected squared change in a chosen quantity over reference examples. The quantity to preserve, the approximation of the

objective and the selection of protected directions are explicit design choices, so different projection methods follow from different choices.
- *SCLL*, a projection method that protects input-output pairs selected by their importance for the reference loss, storing only two small factor matrices and the selected directions of each layer rather than a model-sized vector per protected direction.
- An *empirical evaluation* on two language-model task streams, covering earlier tasks, initial capabilities, and related datasets not used for protection, with an analysis of when pair-level protection helps and a controlled Permuted MNIST study of update freedom.

## 2 METHOD

This section derives SCLL from an explicit preservation objective, which measures how a parameter change alters the model's behavior on reference data. After introducing the setting and notation (Section 2.1), we define the objective and approximate it as a sum of terms, one per weight matrix, each expressed through two small matrices of reference statistics (Section 2.2). Their eigenvectors identify the principal input and output directions of each weight matrix, and products of their eigenvalues assign an importance to each input-output pair. Section 2.3 describes how SCLL estimates these statistics, selects the most important pairs, removes their components from each optimizer update, and updates protection along the task stream.

### 2.1 PRELIMINARIES

We study continual fine-tuning of the backbone weight matrices of a pretrained model, which must learn a sequence of tasks while retaining selected initial capabilities and previously learned tasks. Each capability or task to preserve is represented by a reference dataset $\mathcal{D}_k$ of limited size, whose examples $z$ contain the inputs and any supervision needed to compute the loss $\ell(z;\theta)$; for a learned task, $\mathcal{D}_k$ is drawn from its training data. We estimate statistics of $\mathcal{D}_k$ once, at the checkpoint $\theta_k^\star$ at which its protection begins: the pretrained checkpoint for the initial capabilities, and, for a learned task, the checkpoint reached after training on it. Reference examples are used only to estimate protection and are never replayed during new-task training.

We write the trainable parameters as weight matrices $W_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $i = 1, \dots, L$. In our language-model experiments, these are the attention and feed-forward weight matrices; all other parameters remain fixed. Each matrix forms a parameter block $\theta_i = \text{vec}(W_i)$, where vec stacks matrix columns into a vector, and $\theta$ concatenates all blocks. Let $\Delta\theta_i = \text{vec}(\Delta W_i)$ denote a parameter update, either a single optimizer step or a displacement accumulated over several steps.

For each weight matrix $W_i$, let $h_i \in \mathbb{R}^{d_{\text{in}}}$ be the input activation of its linear transformation $y_i = W_i h_i$ and $y_i \in \mathbb{R}^{d_{\text{out}}}$ its output before subsequent operations. We represent the behavior to preserve by a differentiable scalar target $f(z;\theta)$, such as the reference loss $\ell(z;\theta)$. When considering a single reference dataset, we drop its index and write $\theta^\star$ for its estimation checkpoint. The output sensitivity $s_i = \nabla_{y_i} f(z;\theta^\star) \in \mathbb{R}^{d_{\text{out}}}$ is the gradient of the target with respect to that output, computed by backpropagating through the remaining network, including downstream nonlinearities. Both $h_i$ and $s_i$ are evaluated at $\theta^\star$.

### 2.2 A FIRST-ORDER PRESERVATION FRAMEWORK

**Preservation objective.** We define the preservation objective as the expected squared change of the target under a parameter update $\Delta\theta$ from $\theta^\star$. Squaring before averaging prevents changes on different examples from canceling and penalizes increases and decreases symmetrically. Let $g(z) = \nabla_\theta f(z;\theta^\star)$ and $G = \mathbb{E}_{z\sim\mathcal{D}}[g(z)g(z)^\top]$. We first approximate the target change by $g(z)^\top \Delta\theta$, then square it. This gives

$$\mathcal{L}_{\text{pres}}(\Delta\theta) := \mathbb{E}_{z\sim\mathcal{D}}\left[\left(f(z;\theta^\star + \Delta\theta) - f(z;\theta^\star)\right)^2\right] \approx \Delta\theta^\top G\, \Delta\theta. \tag{1}$$

The approximation is local to $\theta^\star$, even when $\Delta\theta$ accumulates several optimizer steps. With the reference loss as the target, the objective controls the local change of that loss rather than the model's entire predictive distribution.

**Tractable approximation.** The gradient second moment $G$, the average outer product of reference gradients, is too large to form explicitly. We first neglect interactions between parameter blocks, adopting a block-diagonal approximation as in K-FAC (Martens & Grosse, 2015). This gives $\mathcal{L}_{\text{pres}}(\Delta\theta) \approx \sum_{i=1}^{L} \mathcal{L}_{\text{pres}}^{(i)}(\Delta\theta_i)$, where $\mathcal{L}_{\text{pres}}^{(i)}(\Delta\theta_i) = \Delta\theta_i^\top G_i \Delta\theta_i$, $G_i = \mathbb{E}[g_i(z)g_i(z)^\top]$, and $g_i$ is the gradient restricted to block $i$.

For a single application of the linear transformation with weight matrix $W_i$, the gradient is $\nabla_{W_i} f = s_i h_i^\top$, so $g_i = h_i \otimes s_i$, where $\otimes$ denotes the Kronecker product, and $G_i = \mathbb{E}[(h_i h_i^\top) \otimes (s_i s_i^\top)]$. This structure motivates approximating $G_i$ by a Kronecker product of two smaller positive semidefinite factors:

$$G_i \approx A_i \otimes B_i, \tag{2}$$

where $A_i \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}$ describes input structure and $B_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$ output-sensitivity structure. The simplest choice is the unweighted factors $A_i = \mathbb{E}[h_i h_i^\top]$ and $B_i = \mathbb{E}[s_i s_i^\top]$, as in K-FAC.

**Importance of input-output pairs.** For any such factors, write their eigendecompositions as $A_i = U_i \Lambda_i U_i^\top$ and $B_i = V_i \Sigma_i V_i^\top$. Let $u_p, v_q$ denote columns of $U_i, V_i$, with eigenvalues $\lambda_p, \sigma_q$; we omit the block index $i$ from these symbols. Substituting the decompositions into the block's approximate objective yields, in terms of $\Delta W_i$,

$$\mathcal{L}_{\text{pres}}^{(i)}(\Delta W_i) \approx \sum_{p,q} \lambda_p \sigma_q \left(v_q^\top \Delta W_i \, u_p\right)^2. \tag{3}$$

Each input-output pair $(u_p, v_q)$ defines a rank-one direction $v_q u_p^\top$ in weight space, which changes the transformation's output along $v_q$ in response to input along $u_p$. These directions are orthonormal, and $v_q^\top \Delta W_i u_p$ is the update's coefficient along $v_q u_p^\top$. Its weight $\lambda_p \sigma_q$ is the pair's estimated importance: it is large when reference inputs align with $u_p$ and the target is sensitive to output changes along $v_q$. Pairs sharing an input direction can therefore receive different importance, and pairs with large importance are candidates for protection.

**Design choices.** The framework leaves three choices open. The *target* $f$ specifies which behavior is preserved, for example the loss of the provided reference answer or of fixed predictions of the reference model. The *factors* $A_i, B_i$ determine how sensitivity is approximated, for example by summarizing activations and sensitivities separately, or by weighting inputs by the magnitude of their sensitivities. The *protected set* determines which directions are removed from updates: whole input directions across all outputs, or individual input-output pairs. SCLL instantiates these choices with the reference loss, sensitivity-weighted input factors, and individual pairs (Section 2.3); a systematic study of the alternatives is left for future work.

### 2.3 SCLL: Statistics, Selection, and Projection

**Preservation target.** We set $f$ to the reference loss $\ell$: the negative log-likelihood (NLL) of the provided answer, summed over its tokens.

**Sensitivity approximation.** For sequences, we treat each non-padding token position as a separate application of $W_i$: we collect $h_i$ and $s_i$ at every position and omit interactions between positions. To give more weight to inputs associated with larger sensitivities, we place the squared sensitivity magnitude in the input factor and normalize the sensitivity to retain its direction. Let $\bar{s}_i = s_i / \|s_i\|_2$ when $s_i \neq 0$, and $\bar{s}_i = 0$ otherwise. For reference dataset $k$, the factors are

$$A_{i,k} = \mathbb{E}_{k,\text{token}}[\|s_i\|_2^2 h_i h_i^\top], \qquad B_{i,k} = \mathbb{E}_{k,\text{token}}[\bar{s}_i \bar{s}_i^\top]. \tag{4}$$

Both expectations average over the observed token positions in dataset $k$. Thus, $A_{i,k}$ captures inputs weighted by loss sensitivity, while $B_{i,k}$ captures sensitivity directions. With $K$ protected datasets, we use $A_i = K^{-1} \sum_{k=1}^{K} A_{i,k}$ and $B_i = K^{-1} \sum_{k=1}^{K} B_{i,k}$, giving each dataset equal weight. Averaging the two factors separately approximates the equal-weight average of the per-dataset objectives.

**Protected directions.** We eigendecompose the averaged factors using the notation of Section 2.2: $A_i = U_i \Lambda_i U_i^\top$ and $B_i = V_i \Sigma_i V_i^\top$. Equation (3) assigns each direction $v_q u_p^\top$ the importance $\lambda_p \sigma_q$, and the total importance of the matrix, which we call its energy, is $E_i = \sum_{p,q} \lambda_p \sigma_q = \operatorname{tr}(A_i) \operatorname{tr}(B_i)$.

Within each matrix, we rank pairs by decreasing importance and protect the fewest highest-ranked pairs whose importance sums to at least $\tau E_i$, for a threshold $\tau \in [0, 1]$. Selecting the largest products reaches this fraction with the fewest protected directions. Note that $\tau$ is a fraction of estimated importance, not of the parameters frozen. We record the selected pairs in a binary mask $M_i$, with rows indexing output directions and columns indexing input directions: an entry is one for a protected pair and zero otherwise.

**Projecting updates.** For an optimizer-proposed update $\Delta W_i$, projection removes the components marked by $M_i$ in the input-output basis:

$$\Delta W_i^{\text{proj}} = \Delta W_i - V_i(M_i \odot C_i)U_i^\top, \qquad C_i = V_i^\top \Delta W_i U_i. \tag{5}$$

Here $C_i$ contains the update coefficients in the input-output basis and $\odot$ denotes elementwise multiplication. The projection zeroes the selected coefficients and maps the remaining update back to weight space. We project the complete update proposed by the optimizer (for AdamW, after preconditioning and weight decay) and apply $W_i \leftarrow W_i + \Delta W_i^{\text{proj}}$.

**Continual operation.** Before each task, SCLL eigendecomposes the averaged factors and selects the protected pairs; the bases $U_i, V_i$ and the mask $M_i$ remain fixed while the task is learned. After the task, it estimates the task's factors at the resulting checkpoint, adds them to the equal-dataset averages, and re-selects the protected pairs, so the protected subspace can grow or shrink as tasks are added. Per weight matrix, SCLL stores only the two factors, the selected eigenvectors, and a binary mask, $O(d_{\text{in}}^2 + d_{\text{out}}^2)$ values independent of the number of tasks, rather than a model-sized vector per protected direction. Algorithm 1 summarizes the procedure; implementation settings are in Appendix A.

## 3 Related Work

Continual learning is an active area of research. We focus on the methods most relevant to our work to place it in context; a full overview is beyond the scope of this paper.

**Projection-based continual learning.** Closest to our work are methods that project updates to be orthogonal to protected directions. GPM and Adam-NSCL (Saha et al., 2021; Wang et al., 2021) protect principal directions of previous-task input activations across all outputs of a layer. SCLL recovers this geometry when every output direction is protected, but instead selects individual input-output pairs. OSFT (Nayak et al., 2026) and OPLoRA (Xiong & Xie, 2026) protect the dominant singular components of the weights, which need not identify the directions a particular behavior depends on. OGD (Farajtabar et al., 2020) protects the span of stored full-parameter gradients, whereas SCLL summarizes first-order sensitivity with two small matrices per layer.

**Other continual learning approaches.** Replay methods (Chaudhry et al., 2019) mix stored examples into new-task training, which adds forward and backward passes at every step and makes the preservation signal depend on the sampled minibatch. SCLL instead amortizes preservation: it processes reference data once per task and applies the resulting constraint consistently to every update. Functional regularization methods (Li & Hoiem, 2017) seek to preserve model outputs directly; SCLL shares this behavioral view but translates it into a parameter-space constraint enforced by projection. Parameter-importance regularization methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018) typically use diagonal approximations that ignore correlations between parameters, so they cannot distinguish harmful from harmless combinations of changes. Kronecker-factored Laplace approximations (Ritter et al., 2018) capture within-layer correlations with a factorization similar to SCLL's, but derive it from a second-order expansion around a local optimum of previous tasks' loss, where the loss gradient vanishes. SCLL's first-order objective requires no such point and applies to any differentiable preserved quantity $f$: a different choice of $f$ yields different factors, and hence different protected directions. Moreover, these methods keep the full factors in memory and use them as a quadratic penalty on all parameter changes, whereas SCLL's projection uses only the eigenvectors of the selected pairs and removes them from each update. Parameter-efficient methods (Hu et al., 2022; Wang et al., 2023a) and sparse fine-tuning (Ansell et al., 2022) restrict which parameters or coordinates change, limiting capacity. SCLL fine-tunes the backbone weights and restricts only directions, so a weight can serve both protected and new computation. Model merging (Ilharco et al.,

2023; Yadav et al., 2023) resolves conflicts between independently trained models after training, whereas SCLL avoids them during optimization, without storing task-specific models.

# 4 EXPERIMENTS

We evaluate SCLL on two language-model continual-learning benchmarks, TRACE and Long CL, measuring acquisition of each new task, retention of earlier tasks, and retention of initial capabilities that are protected without replaying their data. We also perform a controlled Permuted MNIST study to examine how protecting input-output pairs, rather than whole input directions, preserves the freedom to learn over a long task sequence (Section 4.6).

## 4.1 BENCHMARKS AND EVALUATION PROTOCOL

Following OSFT (Nayak et al., 2026), we evaluate on two task streams. **TRACE** (Wang et al., 2023b) is an eight-task instruction-tuning benchmark covering domain-specific stance and policy classification, meeting summarization, code completion, science question answering, numerical reasoning, and multilingual simplification, in one fixed order. **Long CL** (Wang et al., 2023a) consists of fifteen text-classification tasks in three released orders: AG News, Amazon Reviews, DBpedia, and Yahoo Answers (Zhang et al., 2015), together with tasks from GLUE (Wang et al., 2019b), SuperGLUE (Wang et al., 2019a), and IMDb (Maas et al., 2011). Every method fine-tunes the same selected attention and feed-forward weight matrices of Qwen3.5-0.8B (Qwen Team, 2026) (OSFT's selection) from the same pretrained checkpoint, with benchmark-specific optimizer settings shared across methods and a single training seed; standard deviations on Long CL therefore reflect variation across task orders. Before continual learning, methods that use reference data receive up to $5{,}000$ training examples from each of five datasets representing initial capabilities to protect ($D_P$): reading comprehension (BoolQ; Clark et al., 2019), arithmetic reasoning (GSM8K; Cobbe et al., 2021), commonsense (HellaSwag; Zellers et al., 2019), broad knowledge (MMLU; Hendrycks et al., 2021), and compositional science question answering (QASC; Khot et al., 2020). SCLL and NSCL use these examples only to construct protection, never replay them, and update protection after each task. Retention is measured on held-out examples from the same datasets. Task inventories, orders, and training details are in Appendix B.

**Metrics.** Following TRACE and OSFT, we report acquisition (Acq), average accuracy (AA), and backward transfer (BWT), given by

$$\text{Acq} = \frac{1}{T}\sum_{j=1}^{T} a_{j,j}, \qquad \text{AA} = \frac{1}{T}\sum_{j=1}^{T} a_{T,j}, \qquad \text{BWT} = \frac{1}{T-1}\sum_{j=1}^{T-1}\left(a_{T,j} - a_{j,j}\right), \tag{6}$$

where $a_{i,j}$ is the score on task $j$ after training on task $i$ in a sequence of $T$ tasks, using each task's native evaluation metric on a 0–100 scale. Acquisition measures how well each task is learned, and average accuracy measures performance after the full sequence. Negative BWT indicates forgetting, and positive BWT indicates improvement on earlier tasks. For the protected capabilities, we report the mean held-out score over the five $D_P$ datasets after the sequence ($D_P$) and its change from the pretrained model ($\Delta D_P$). All averages weight tasks or datasets equally.

## 4.2 BASELINES

We compare SCLL with six groups of continual-learning approaches, all of which update the same weight matrices as SCLL: (i) **sequential fine-tuning (Naive)**, which trains on each task in turn without protection and is a lower bound prone to catastrophic forgetting; (ii) **weight-based projection** methods such as OSFT-D and OPLoRA, our dense-update variants of OSFT (Nayak et al., 2026) and OPLoRA (Xiong & Xie, 2026), which derive protected directions from the singular vectors of the weights; (iii) **activation-based projection** methods such as Adam-NSCL (Wang et al., 2021), which protects input directions estimated from the same reference data as SCLL; (iv) **regularization** methods such as EWC (Kirkpatrick et al., 2017), initialized from the $D_P$ reference data, and LwF (Li & Hoiem, 2017), which distills from the model at the start of each task; (v) **experience replay (ER)**, which replays stored $D_P$ and past-task examples with a $10\%$ larger update budget and, unlike the other methods, trains on previous data; and (vi) **model merging** methods such as TIES (Yadav et al.,

2023), which merges independently fine-tuned per-task experts rather than updating a single model sequentially.

OSFT-D and NSCL run in the same framework as SCLL: the same trainable matrices, the same optimizer, and projection of the same optimizer update. The projection methods therefore differ only in the subspace they protect. OSFT-D computes OSFT's protected subspaces as described in the original paper, but it projects dense weight updates instead of splitting each matrix into frozen and trainable SVD components (Appendix B). Implementation details and hyperparameters for all baselines are in Appendix B.

### 4.3 Results on TRACE

TRACE is the more demanding of the two streams. Long CL consists of classification tasks whose answers are short labels, so consecutive tasks share an output format. The TRACE tasks (Section 4.1) instead differ in language, domain, and output format. They were selected to be challenging for current language models, and Wang et al. (2023b) show that sequential training on them erodes the general abilities of aligned models. The stream therefore resembles post-training a deployed model on specialized skills: each task requires substantial parameter change, while the model is expected to keep its general capabilities. Table 1 reports the results. SCLL obtains the highest acquisition and average accuracy, with little forgetting of earlier tasks ($-0.34$ BWT), and ends $1.58$ points above the pretrained model on the protected capabilities. Compared with NSCL, the strongest baseline, SCLL matches retention (BWT $-0.34$ vs. $-0.21$; $\Delta D_P$ $+1.58$ vs. $+1.37$) while improving average accuracy by $4.01$ points, so its gain comes from learning new tasks rather than from stronger protection. Weight-based projection does not identify the directions that matter for prior behavior: OSFT-D forgets almost as much as Naive, and OPLoRA loses $5.80$ points on the protected capabilities. SCLL, NSCL, and ER also exceed the mean of independently fine-tuned per-task models (Standalone). The size of each final projector is consistent with these results. At the end of TRACE, SCLL freezes $52.11\%$ of the update directions, compared with $66.46\%$ for NSCL, $58.64\%$ for OPLoRA, and $17.21\%$ for OSFT-D. Frozen is the parameter-weighted share of update directions in the 114 selected matrices blocked by the final projector. SCLL therefore protects fewer directions than NSCL while retaining as much, and OPLoRA freezes more than SCLL yet retains less. Retention depends on which directions are protected, not on how many: selecting input-output pairs by their estimated effect on the reference loss lets SCLL preserve more while freezing less.

### 4.4 Protecting Capabilities Beyond the Task Sequence

The protected-capability columns above measure retention on held-out examples from the datasets used to construct protection. We also test whether protection extends to related capabilities beyond the datasets used to construct it. Each protected dataset is paired with one related evaluation dataset ($D_R$): BoolQ with RACE-High (Lai et al., 2017), GSM8K with ASDiv (Miao et al., 2020), HellaSwag with PIQA (Bisk et al., 2020), MMLU with MMLU-Pro (Wang et al., 2024), and QASC with ARC-Challenge (Clark et al., 2018). The $D_R$ datasets are used only for evaluation, including few-

Table 1: TRACE results (Qwen3.5-0.8B, one order). Metrics follow Eq. (6); $D_P$ is the mean score on the five protected datasets after the sequence and $\Delta D_P$ its change from the pretrained model. Higher is better. Standalone: mean of independent per-task models (Acq only).

| | Continual-learning tasks | | | Protected capabilities | |
|---|---|---|---|---|---|
| Method | Acq | AA | BWT | $D_P$ | $\Delta D_P$ |
| Standalone | 50.13 | — | — | — | — |
| Naive | 48.16 | 31.18 | −19.41 | 34.34 | −12.20 |
| EWC | 42.30 | 48.07 | **+6.59** | 42.53 | −4.01 |
| LwF | 45.58 | 23.74 | −24.96 | 33.36 | −13.18 |
| ER (10%) | 53.18 | 51.62 | −1.79 | 44.92 | −1.62 |
| TIES | 45.56 | 43.09 | −2.83 | 42.37 | −4.17 |
| OSFT-D | 46.39 | 27.63 | −21.44 | 36.38 | −10.16 |
| OPLoRA | 48.67 | 45.01 | −4.18 | 40.74 | −5.80 |
| NSCL | 52.72 | 52.53 | −0.21 | 47.91 | +1.37 |
| SCLL (ours) | **56.84** | **56.54** | −0.34 | **48.12** | **+1.58** |

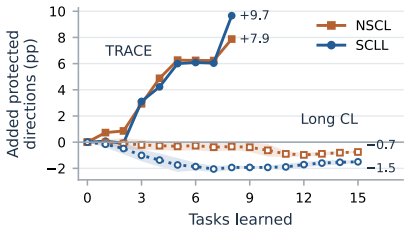


Figure 2: Change in the protected share of the update space after each task, relative to the initial $D_P$-only protection (percentage points). TRACE (solid): one order; Long CL (dotted): mean ± std over three orders.

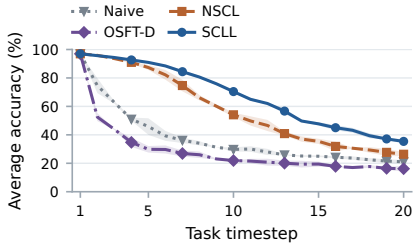


Figure 3: Permuted MNIST: average accuracy over the tasks seen so far (mean ± one sample standard deviation, three seeds). SCLL uses unweighted factors.

shot demonstrations, and never for constructing protection. The evaluation protocol is given in Appendix B.1.

Table 2 compares Naive and SCLL after the same TRACE sequence with the original pretrained model. This comparison tests whether protection estimated from the five reference datasets also preserves performance on related evaluation datasets. Closeness to Original measures retention, while the comparison with Naive measures the effect of adding protection. The pairings describe related capabilities; they do not isolate the contribution of any single reference dataset. Averaged over the five datasets, SCLL scores 50.27, slightly above the pretrained model (49.46; +0.82 points), whereas Naive drops to 28.61 (−20.85 points).

Table 2: Retention on the related datasets ($D_R$) after TRACE (%, ↑; strict exact match for ASDiv). Naive and SCLL use the final checkpoints of Table 1; Original is the pretrained model. $D_R$ is not used to construct protection.

| Dataset | Naive | SCLL | Original |
|---|---|---|---|
| RACE-High | 29.88 | 35.16 | 33.01 |
| ASDiv | 0.20 | 88.09 | 86.52 |
| PIQA | 65.62 | 69.73 | 70.51 |
| MMLU-Pro | 16.27 | 22.82 | 21.23 |
| ARC-Challenge | 31.06 | 35.58 | 36.01 |

### 4.5 Results on Long CL

Table 3 reports acquisition and average accuracy together with forgetting and retention of the protected capabilities on Long CL, a fifteen-task benchmark. Of the methods reported, only NSCL and SCLL both learn and retain: each keeps BWT within one standard deviation of zero and ends above the pretrained model on the protected capabilities. SCLL exceeds NSCL in average accuracy on every task order (81.44 vs. 81.12 on average; Appendix C) and in protected-capability retention (+0.67 vs. +0.42), but the margins are small. With Qwen3.5-2B, the gap grows to 2.0 points, and SCLL leads on every order (Appendix C.3).

Table 3: Long CL results (fifteen tasks): mean over orders 4–6, with population standard deviation in small type; $\Delta D_P$ is the mean change. Per-order AA is in Appendix C. TIES merges one shared expert bank, so its final model is identical across orders. Long CL revisits BoolQ, which is included in $D_P$.

| Method | Acq | AA | BWT | $D_P$ | $\Delta D_P$ |
|---|---|---|---|---|---|
| Naive | 73.83±0.49 | 8.86±3.59 | −69.61±3.86 | 21.57±2.19 | −24.97 |
| EWC | 65.83±2.70 | 65.62±3.12 | −0.22±3.76 | 42.20±1.86 | −4.34 |
| LwF | 70.18±1.16 | 11.42±1.80 | −62.96±2.70 | 22.34±2.16 | −24.20 |
| ER (10%) | 74.22±1.75 | 70.50±0.38 | −3.99±1.51 | 31.83±0.78 | −14.71 |
| TIES | 61.46±2.19 | 62.23 | **+0.82**±2.35 | 40.52 | −6.02 |
| OSFT-D | 75.26±0.66 | 9.40±3.44 | −70.57±3.58 | 21.79±2.47 | −24.75 |
| OPLoRA | 80.52±0.32 | 14.56±2.10 | −70.67±2.47 | 26.55±0.57 | −19.99 |
| NSCL | **81.12**±0.20 | 81.12±0.05 | −0.01±0.23 | 46.96±0.17 | +0.42 |
| SCLL (ours) | 81.08±0.41 | **81.44**±0.15 | +0.39±0.52 | **47.21**±0.24 | **+0.67** |

One likely reason for the small margins is that Long CL requires little new protection as tasks are added. After each task, both methods compute that task's reference statistics and re-select the fewest

directions that cover the energy threshold. Figure 2 tracks the resulting protected share of the update space. On TRACE, this share grows by 7.9 (NSCL) and 9.7 (SCLL) percentage points over eight tasks: each task's important directions lie largely outside those already protected. On Long CL, it stays flat or shrinks slightly ($-0.7$ and $-1.5$ points over fifteen tasks), showing that each task's important directions largely coincide with those already protected. With few new directions to protect, learning and retention rarely conflict, and both methods acquire the tasks equally well (Acq 81.08 for SCLL vs. 81.12 for NSCL). Selecting individual input-output pairs therefore has little to gain on Long CL; it pays off on TRACE, where each task needs new directions.

### 4.6 Analysis: Update Freedom on Permuted MNIST

Projection methods trade plasticity for stability: every protected direction is unavailable for learning later tasks. We use Permuted MNIST (PMNIST) (LeCun et al., 1998; Kirkpatrick et al., 2017) as a controlled setting in which this trade-off accumulates over a long sequence. A single network learns 20 digit-classification tasks, each with a different fixed pixel permutation. Figure 3 tracks the average accuracy over all tasks seen so far. SCLL keeps more update directions available than NSCL and achieves higher accuracy on newly learned tasks, so its average accuracy stays higher on later tasks. Both methods still lose performance over the long sequence. This study uses unweighted SCLL factors, so it isolates the effect of selecting input-output pairs rather than the sensitivity weighting used in the language-model experiments. The protocol, capacity diagnostics, and acquisition and forgetting results are in Appendix D.

## 5 Conclusion

We studied continual fine-tuning of language models through the question of which update directions must be protected. Starting from an explicit preservation objective, we showed that its Kronecker-factored approximation assigns an importance to each input-output pair of a linear layer, and that it recovers the geometry of input-level protection such as NSCL when every output direction is protected. SCLL protects only the most important pairs. On TRACE, it improves average accuracy over NSCL by 4.01 points at matched retention while freezing fewer directions, and it preserves performance on related datasets not used for protection; on Long CL, where tasks require few new directions, the two perform similarly. We believe this perspective extends beyond our method. As a model is adapted to more tasks, the question is not whether to constrain its updates but which directions to constrain: every protected direction is unavailable for future learning, so protection should follow from the behavior being preserved rather than from proxies such as the weight spectrum or input statistics alone. Our results suggest that deriving protection from such an objective lets a model preserve more while freezing less, and the same framework offers a principled basis for designing future projection methods.

## References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning what (not) to forget. In *European Conference on Computer Vision*, 2018. URL https://arxiv.org/abs/1711.09601.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable Sparse Fine-Tuning for Cross-Lingual Transfer. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://arxiv.org/abs/2110.07560.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1911.11641.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning. *arXiv preprint arXiv:1902.10486*, 2019. URL https://arxiv.org/abs/1902.10486.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill

Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal Gradient Descent for Continual Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773, 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

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://arxiv.org/abs/2106.09685.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing Models with Task Arithmetic. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2212.04089.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A Dataset for Question Answering via Sentence Composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1910.11473.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017. URL https://arxiv.org/abs/1612.00796.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082/.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://bottou.org/papers/lecun-98h.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015/.

James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the International Conference on Machine Learning*, 2015. URL https://arxiv.org/abs/1503.05671.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Oleg Silkin, Mustafa Eyceoz, Hao Wang, Aldo Pareja, and Akash Srivastava. Sculpting Subspaces: Constrained Full Fine-Tuning in LLMs for Continual Learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/22620cccae29bbb3f18d226ea0320ff5-Abstract-Conference.html.

Qwen Team. Qwen3.5-0.8B Model Card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-0.8B. Model revision 2fc06364715b967f1860aea9cf38778875588b17.

Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. In *Advances in Neural Information Processing Systems*, 2018. URL https://arxiv.org/abs/1805.07810.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient Projection Memory for Continual Learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2103.09762.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems*, 2019a. URL https://arxiv.org/abs/1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, 2019b. URL https://arxiv.org/abs/1804.07461.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training Networks in Null Space of Feature Covariance for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. URL https://arxiv.org/abs/2103.07113.

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 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A Comprehensive Benchmark for Continual Learning in Large Language Models. *arXiv preprint arXiv:2310.06762*, 2023b. URL https://arxiv.org/abs/2310.06762.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. *arXiv preprint arXiv:2406.01574*, 2024. URL https://arxiv.org/abs/2406.01574.

Yifeng Xiong and Xiaohui Xie. OPLoRA: Orthogonal Projection LoRA Prevents Catastrophic Forgetting During Parameter-Efficient Fine-Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34088–34096, 2026. doi: 10.1609/aaai.v40i40.40703. URL https://ojs.aaai.org/index.php/AAAI/article/view/40703.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html.

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/.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, 2017. URL https://arxiv.org/abs/1703.04200.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, 2015. URL https://arxiv.org/abs/1509.01626.