LOCAL SUPPORT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We explore catastrophic forgetting in the context of large pre-trained models. By considering forgetting as a geometric problem in the input space of each weight matrix, we uncover a natural retention objective under which updates produced by gradient-based optimizers are suboptimal. Following this observation, we propose Local Support Learning (LSL), a general-purpose framework that augments gradient-based training for retention of prior capabilities without access to prior data. During a new learning phase, LSL pairs two components with distinct roles: a standard weight adapter, trained as usual to minimize the loss, and a gating function that enables the adapter only on input activations from its own training distribution, making the update *local* to that distribution. The key challenge is that this gate must route data from all learning phases while training only on data from the current one. We address this with a gate based on a Gaussian Mixture Model (GMM), whose likelihood decays rapidly away from its training data, giving it a natural tendency to stay closed on data from prior phases. We show that this post-training approach can resolve forgetting in LLMs of up to 7 billion parameters, retaining both pretrained and finetuned capabilities across multiple training phases, while being efficient in memory and compute, robust to hyperparameter choice, and showing scaling potential.

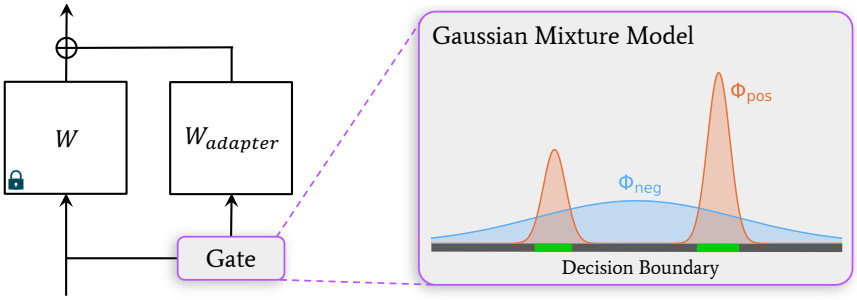


Figure 1: **Avoiding Catastrophic Forgetting with Local Support Learning.** (**Left**) We augment gradient-based training with a mechanism that retains prior capabilities without access to prior data. The key component is a gating function that enables the weight adapter only on input activations from its training distribution. Interestingly, conventional MLP classifiers are not well suited for this task. Instead, we propose a GMM-based gate that tends to stay closed on prior data it has never encountered. (**Right**) 1D illustration of the GMM-based gate. Φ_{pos} (orange) captures the distribution of the finetuning data, while the wider Φ_{neg} (blue) dominates elsewhere - ensuring that data outside the current distribution is routed only to the pretrained weights.

1 INTRODUCTION

The modern deep learning paradigm first pretrains a neural network on a large dataset and then adapts it to desired tasks and behaviors through additional finetuning phases (Ouyang et al., 2022; Wei et al., 2022; Rozière et al., 2024). However, neural networks tend to forget previously learned capabilities when adapting to new data - a phenomenon known as catastrophic forgetting (McCloskey & Cohen, 1989). Forgetting hinders multi-phase learning, as capabilities acquired during pretraining and previous finetuning phases often get overwritten (Biderman et al., 2024).

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.

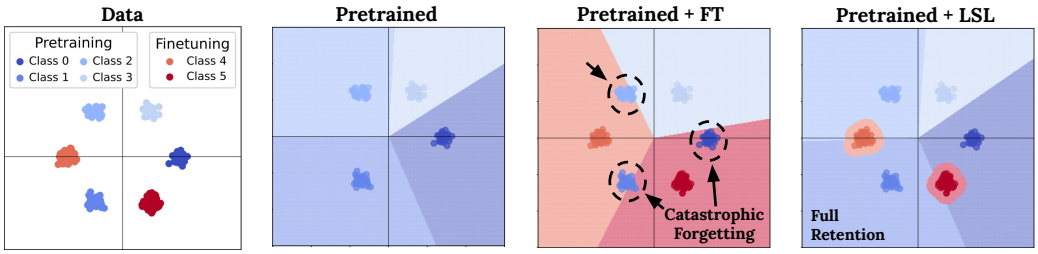


Figure 2: **Catastrophic Forgetting - Toy Example**. (**Data**) We train over six classes in two phases: "Pretraining" (points in blue), and "Finetuning" (points in red). (**Pretrained**) Classifier trained on the pretraining data. The background colors are the learned decision boundaries - depicting a perfect score on the observed distribution. (**Pretrained + FT**) We continually train the model on two other classes. The model performs well on data from the second phase (red), yet the boundaries learned during the first phase are heavily deformed - a clear case of forgetting. (**Pretrained + LSL**) Same experiment, this time with LSL. We restrict weight updates to the region of the input space that produced them, leaving model behavior unchanged elsewhere - thus achieving a perfect score.

Catastrophic forgetting is a long-standing problem, studied well before pretraining became popular. Among earlier mitigation approaches, the most successful augment the model with buffers of previous data samples (Chaudhry et al., 2019a), apply data-dependent regularization (Kirkpatrick et al., 2017), or impose optimization constraints (Farajtabar et al., 2019). However, these auxiliary mechanisms scale poorly, making them impractical in the modern paradigm, especially for retaining capabilities from the large-scale pretraining phase. Instead, mitigation approaches now aim to minimize change in pre-trained networks (Xiong & Xie, 2025; Nayak et al., 2025; Wang et al., 2023; Liang & Li, 2024), yet the exact definition of change, and how to prevent it, remain open problems.

By considering change at the level of a single weight matrix, we can analyze the effect of an update on the full input-output mapping of the matrix. In the context of large-scale pretraining, this perspective yields a natural retention objective: updates should affect only the input region required for learning, i.e., the support of the current phase's distribution. We show that updates produced by gradient-based optimizers often act outside this support: any prior input not orthogonal to the update has its mapping altered, which may lead to forgetting. Existing works that enforce orthogonality to all prior inputs work in small-scale settings (Saha et al., 2021), yet become infeasible at pretraining scale. Without assuming access to pretraining data, recent approaches either protect only previous finetuning phases, keeping updates orthogonal to them (Wang et al., 2023; Liang & Li, 2024), or approximate the orthogonal complement of the pretraining data from the weights (Xiong & Xie, 2025; Nayak et al., 2025) - an approximation we find insufficient (App. C), calling for a new approach.

Motivated by this, we propose Local Support Learning (LSL), a general-purpose framework that augments standard gradient-based training to retain prior capabilities without access to prior data. During a new learning phase, LSL pairs two components with distinct roles: a standard weight adapter (Hu et al., 2021), trained as usual to minimize the loss, and a gating function that estimates the support of the current data in the adapter's input space. At inference, the gate applies the adapter only to tokens whose activations fall within that support. It is a difficult challenge given that the gate cannot train on data from prior phases. We address this with a gate based on Gaussian Mixture Models (GMMs). The key idea is that a Gaussian's density decays exponentially with distance from the training data, providing the gate an inductive bias to stay closed on inputs it has never seen.

To demonstrate the effectiveness of our approach, we finetune LLMs of up to 7 billion parameters on diverse tasks and show that they are able to learn at full capacity without degrading capabilities acquired during pretraining including math, coding, and instruction following. This holds even across multiple finetuning phases, where LSL retains both pretrained and previously finetuned capabilities. In addition, retention is largely unaffected by the choice of hyperparameters. This is in contrast to existing methods, where the learning rate, adapter rank, and batch size induce a direct trade-off between learning and forgetting. On the practical side, LSL exhibits desirable scaling trends and is highly efficient in both memory and compute. It requires no more memory than a LoRA adapter while offering fast inference, competitive training time, and straightforward parallelization.

Our main contributions are four-fold: (i) We analyze forgetting in modern large-scale settings and uncover a natural retention objective that existing methods do not satisfy, providing a possible

explanation for their limited ability to mitigate forgetting. (ii) We propose Local Support Learning, a general-purpose framework that avoids forgetting by restricting weight updates to the region of the input space that produced them. (iii) We motivate our approach by providing a theoretical connection between optimal gates and density estimators. (iv) We evaluate LSL at scales of up to 7 billion parameters, demonstrating its effectiveness, robustness, and efficiency.

2 Related Work

Modern Learning Paradigms. Large-scale pretraining has proven transformative across modalities, enabling in-context learning in language (Brown et al., 2020), robust representations in vision (Dosovitskiy et al., 2021; Radford et al., 2021), and general-purpose features in speech (Baevski et al., 2020; Radford et al., 2022). Development typically proceeds in phases: large-scale pretraining, followed by post-training that adapts the model to desired tasks and behaviors: instruction following (Wei et al., 2022; Chung et al., 2022), alignment via RLHF (Ouyang et al., 2022), domain specialization (Rozière et al., 2024; Singhal et al., 2023), and multilingual adaptation (Üstün et al., 2024). While essential, post-training risks overwriting the very capabilities that pretraining established at enormous cost (Biderman et al., 2024). Our work directly addresses this tension.

Catastrophic Forgetting. The degradation of previously learned capabilities when learning from new data has been studied as a sequential task learning problem since its identification (McCloskey & Cohen, 1989). Earlier approaches either augment the model with buffers of previous data samples (Chaudhry et al., 2019a; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019b), apply data-dependent regularization techniques (Kirkpatrick et al., 2017; Aljundi et al., 2018; Zenke et al., 2017), or optimize within subspaces orthogonal to gradients or data from previous tasks (Farajtabar et al., 2019; Saha et al., 2021). However, modern practice introduces a fundamental asymmetry: the first 'task' is pretraining on trillions of tokens, followed by several smaller finetuning phases. This renders earlier approaches impractical: a replay buffer representative of pretraining is infeasible to curate, and the data is often unavailable. Still, these works pinpoint key mechanisms of forgetting; in particular, the analysis of gradient interference (Zeng et al., 2019; Wang et al., 2021; Saha et al., 2021) shows how a weight update to a single matrix affects all possible inputs to that matrix. In Sec. 3 we revisit this analysis under modern constraints and show that updates produced by gradient-based methods can hurt retention of prior capabilities when applied to all inputs.

Catastrophic Forgetting in Large Pretrained Models. More modern methods try to minimize change in pretrained models. A common approach restricts change in weight space: Low-Rank Adapters (LoRA) (Hu et al., 2021) accumulate updates in a small subspace, forgetting less than full fine-tuning but still forgetting, with the price of reduced learning capacity (Biderman et al., 2024). Other works constrain updates to subspaces orthogonal to the top-k singular vectors of the pretrained weights (Xiong & Xie, 2025; Nayak et al., 2025), though the choice of k is heuristic and the remaining vectors may still carry important information. A related line restricts different adapters to orthogonal subspaces (Wang et al., 2023; Liang & Li, 2024), mitigating forgetting between finetuning phases but not of pretraining capabilities. Learning without Forgetting (LwF) (Li & Hoiem, 2017) takes a different approach, defining change as a shift in the model's functional behavior. This is enforced by penalizing deviation from the base model's outputs on the finetuning data via knowledge distillation (Hinton et al., 2015). What unites these methods is that they apply gradient updates to all inputs, which, as shown in Sec. 3, causes forgetting. We avoid this by augmenting the gradient update with a gate that restricts its effect to activations from its training distribution. Lastly, due to the abundance of relevant work, we refer the reader to an extended discussion in App. A.

3 Formulation and Problem Investigation

Formulation. We formalize catastrophic forgetting over a sequence of P learning phases. The evaluation is the average performance on all tasks from all phases. Within each phase, the model has IID access to the corresponding dataset; across phases, learning must proceed in a streaming manner, and the memory retained per phase must be bounded. This memory constraint is essential: without it, continual learning admits the trivial solution of storing all previously seen data in a replay buffer and repeatedly training on it, which is impractical at pretraining scale and circumvents the

core challenge of learning continually with the model alone. In practice, a small per-phase memory footprint is what would allow scaling to many phases.

Reproducing Catastrophic Forgetting in a Controlled Setup. To motivate our approach, we study a toy classification problem with six classes over a two dimensional input space $x \in \mathbb{R}^2$. This analysis builds on the per-matrix view of interference (Saha et al., 2021), described in Sec. 2. As shown in Fig. 2 ("Data"), our setup includes two learning phases - "pretraining" and "finetuning"[1]. In the pretraining phase the model was trained over IID samples from four classes - colored in different shades of blue. In the finetuning phase the model trains on samples from two additional classes - colored in different shades of red. We denote both distributions as $\mathcal{D}_{pre}$ and $\mathcal{D}_{ft}$, respectively. Our model is a linear classifier $W \in \mathbb{R}^{6\times 2}$ trained with a cross-entropy loss:

$$v = Wx, \qquad \hat{P}(y \mid x) = \mathrm{Softmax}(v)_y, \qquad \mathcal{L}_i = -\mathbb{E}_{x,y\sim\mathcal{D}_i}\Big[\log \hat{P}(y \mid x)\Big] \tag{1}$$

As shown in Figure 2, during the first learning phase ("Pretrained") the model learns a perfect classifier over D_{pre}. After the finetuning phase ("Pretrained + FT") it correctly classifies the new distribution D_{ft}; however, the decision boundaries from the first phase are heavily deformed, resulting in severe test loss on $\mathcal{D}_{pre}$ - a clear case of catastrophic forgetting.

Global-Support Updates Cause Forgetting. To better understand this phenomenon we revisit the weight update rule. For training step s and learning rate α we have:

$$\Delta W^{(s)} = -\alpha \cdot \frac{d\mathcal{L}}{dW}^{(s)} = -\alpha \cdot \frac{d\mathcal{L}}{dv}^{(s)} x^{(s)T} \tag{2}$$

Assume two new test samples $x_{pre} \in \mathcal{D}_{pre}$ and $x_{ft} \in \mathcal{D}_{ft}$, and that the training sample $x^{(s)}$ was seen during the finetuning phase, i.e. $x^{(s)} \in \mathcal{D}_{ft}$. The change in logits for each sample is:

$$\Delta v_{pre} = \Delta W^{(s)} x_{pre} = -\alpha \cdot \frac{d\mathcal{L}}{dv}^{(s)} x^{(s)T} x_{pre}, \quad \Delta v_{ft} = \Delta W^{(s)} x_{ft} = -\alpha \cdot \frac{d\mathcal{L}}{dv}^{(s)} x^{(s)T} x_{ft} \tag{3}$$

Notice that the change in logits for x_{pre} depends on the dot product $x^{(s)T} x_{pre}$. While $\Delta W^{(s)}$ is useful for x_{ft}, it is not relevant for most samples in $\mathcal{D}_{pre}$. However, unless x_{pre} is orthogonal to $x^{(s)}$, this dot product is nonzero, and the update alters the logits of x_{pre} - causing interference. This holds for all updates produced by gradient-based optimizers: Adam rescales gradients by their second moment and Muon orthogonalizes them, but both still produce a matrix update $\Delta W^{(s)}$ that operates on all inputs - making these updates suboptimal.

Local-Support Updates via Support Estimation. Saha et al. (2021) approaches this problem by estimating a linear subspace in which the prior data (here, $\mathcal{D}_{pre}$) lies, and restricting new updates to its orthogonal complement. This works well in small-scale settings, yet becomes infeasible when the prior data includes a full pretraining corpus, which might not even be available. Therefore, a practical choice is to assume nothing about the prior data - implying that adversarial pretraining samples could exist at any region outside the current data's support. Concretely, we now diverge from Saha et al. (2021) and instead propose to estimate the region where the current data is, denoted $\mathcal{M}$, and restrict the update to that region alone. As shown in Fig. 1 (Left), we realize this by placing a gate $g(x) = x \cdot \mathbb{I}\{x \in \mathcal{M}\}$ before the new phase's weight adapter. Locality is thus controlled by the gate: an update is local if the gate opens only on inputs that are close to the current phase's training data. Standard finetuning corresponds to an always-open gate, which makes the update global.

Theoretical Motivation: A density estimator can recover an optimal gate. As detailed in App. F, against the worst-case test distribution, a gate's error decomposes into two terms, *deficit* and *excess*. The gate's training objective optimizes the deficit directly but is completely blind to the excess. The excess quantifies forgetting: it is the region outside the training distribution on which the gate erroneously stays open. While no objective on the current data alone can act on it, we show that other factors such as the hypothesis class can control it under explicit conditions. We study several hypothesis families, finding that density estimators can recover optimal gates, at costs we analyze.

[1] Interference can arise between any two learning phases; without loss of generality, we adopt the terminology of pretraining and finetuning since retaining pretraining capabilities is the hardest case.

Following this motivation we solve the toy problem with a local estimator (App. F.3): $\mathcal{M} = \bigcup_{i=1}^{N} \mathcal{B}(x_i, r)$, where $\mathcal{B}(x, r)$ is a ball with center x and radius r, and $\{x_i\}_{i=1}^{N}$ are the training points from the current phase's data. In Fig. 2 (right) we repeat the experiment with this approach, which we call Local Support Learning (LSL - detailed in Sec. 4). The LSL-tuned model fully adapts to $\mathcal{D}_{ft}$'s data while having zero interference with $\mathcal{D}_{pre}$, resulting with a perfect classifier. In addition, the decision boundary changed only near the support of $\mathcal{D}_{ft}$, and has not changed elsewhere. The above gate addresses the problem's core challenge - classifying data from all phases while training only on data from the current one. Yet, it is highly inefficient since its complexity scales with N, the number of tokens it encountered. In Sec. 4, we propose a much more efficient parametric gate that removes this dependence, with complexity that scales only with the number of phases P.

4 METHOD

Gate Design. We design our gate to be applied individually to each weight matrix. Each gate fits a Gaussian Mixture Model (GMM) to the current phase's input distribution:

$$\Phi(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k), \qquad \mathcal{M} = \{x \mid \Phi_{pos}(x) - \Phi_{neg}(x) > 0\} \tag{4}$$

where each gaussian $\mathcal{N}$ has mean μ_k, covariance matrix Σ_k, and mixing weight π_k. The GMM is fit via the Expectation-Maximization (EM) algorithm. We compute the gating decision by checking whether the GMM density is above some threshold TH. Conceptually, the support is then $\mathcal{M} = \{x \mid \Phi(x) - \text{TH} > 0\}$. This approach requires tuning the threshold, complicating the method. Instead, in Eq. 4 (right) we propose an input-dependent thresholding approach involving two GMMs. The first GMM Φ_{pos} is fit over the current phase's distribution and the second GMM Φ_{neg} is fit over a *small* sample from a *generic* pre-training dataset. Now, each input x is classified according to the GMM that gives it a larger likelihood. We emphasize that this reference dataset need not have any relation to the data the model was originally pretrained on, and empirically, does not require consisting more than 1M tokens (for reference, this constitutes about 0.0000067% of Qwen's pretraining set).

Intuition for Φ_{neg}. It may seem unlikely that an exceedingly small, generic, pretraining sample can capture the support of the full pretraining distribution. The key insight is that the gate need not capture the full support, but only the width of the distribution, which manifests even in a small sample. We illustrate this in 1D in Figure 1 (Right): Gaussians fit to a narrow distribution Φ_{pos} have high likelihood only in their vicinity, while the wider Φ_{neg} dominates elsewhere. In high dimensions, Φ_{neg} need not be wider than Φ_{pos} in every direction, and constructing an ideal threshold, one that is exceeded exactly on the current phase's data, is not trivial. Φ_{neg} is a practical approximation: the gate still opens on about 16% of tokens from out-of-distribution pretraining tasks (5% with temporal smoothing), yet retains 96.6% of pretrained performance (98.8% with smoothing; Fig. 10).

Algorithm 1 LSL: Training

Require: Model M, datasets $\{\mathcal{D}_i\}_{i=1}^{P}$, num epochs E, reference data $\mathcal{D}_{\text{ref}}$
Ensure: $M \oplus \bigcup_{i=1}^{P} \{\mathbf{W}_{\text{adapter}}^{(i)}, \mathbf{\Phi}_{pos}^{(i)}, \mathbf{\Phi}_{neg}\}$
▷ $\oplus$ denotes mounting onto the base model
$\mathbf{\Phi}_{neg} \leftarrow \text{FITGMMS}(M, \mathcal{D}_{\text{ref}})$
for $i = 1, \ldots, P$ **do**
  $\mathbf{W}_{\text{adapter}}^{(i)} \leftarrow \text{FITADAPTERS}(M, \mathcal{D}_i, E)$
  $\mathbf{\Phi}_{pos}^{(i)} \leftarrow \text{FITGMMS}(M \oplus \mathbf{W}_{\text{adapter}}^{(i)}, \mathcal{D}_i)$
  $M \leftarrow M \oplus \{\mathbf{W}_{\text{adapter}}^{(i)}, \mathbf{\Phi}_{pos}^{(i)}, \mathbf{\Phi}_{neg}\}$
end for
return M

Algorithm 2 LSL: Inference (Per-Module)

Require: Input x, model M, smoothing states $\{s^{(i)}\}_{i=1}^{p}$ and coefficient α.
▷ each $s^{(i)}$ initialized to 0 if not set.
$v \leftarrow W_{\text{pretrained}}\, x$
for $i = 1, \ldots, p$ **do**
  $g \leftarrow \mathbb{I}\{\mathbf{\Phi}_{pos}^{(i)}(x) - \mathbf{\Phi}_{neg}(x) > 0\}$
  $s^{(i)} \leftarrow \alpha \cdot g + (1 - \alpha) \cdot s^{(i)}$
  $v \leftarrow v + \mathbb{I}\{s^{(i)} > 0.5\} \cdot W_{\text{adapter}}^{(i)}\, x$
end for
return v, $\{s^{(i)}\}_{i=1}^{p}$

Figure 3: **Local Support Learning.** We describe the training and inference algorithms. Inference is presented per-module (a weight matrix in some layer) and training is presented for the full model. See Alg. 3 (FitGMMs) and Alg. 4 (FitAdapters) for full details. $\mathbb{I}\{\cdot\}$ is the indicator function.

Temporal Smoothing. The GMM gates each token independently, with no notion of temporal continuity. We address this with a simple exponential moving average that smooths decisions over time, leveraging the fact that task boundaries do not change at a per-token frequency. Importantly, the core mechanism accounting for the vast majority of retention is the GMM, while the smoothing aggregator is only a lightweight complement. We support this with detailed ablation studies showing that the GMM improves retention from 76.6% to 96.6%, and the aggregator improves retention further from 96.6% to 98.8% (Fig. 10 and Sec. 5.2).

Local Support Learning. The full training algorithm can be found in Alg. 1. We first fit Φ_{neg}. Then, in each phase i we fit the i-th weight adapter via gradient descent, followed by fitting its respective Φ_{pos} GMM. Notice that adapters and GMMs from previous phases are active during the current training phase. While FitGMMs (Alg. 3) is straightforward algorithmically, it introduces some technical challenges, addressed in Appendix E. FitAdapters (Alg. 4) is identical to standard adapter finetuning (e.g., LoRA); additional design choices are discussed in the same appendix.
The inference algorithm is provided in Alg. 2, presented per-token. Importantly, it is fully parallelizable during prefill, just like a standard MLPs. We first compute the output for the pretrained weights. Then, for each previously seen phase i, we compute gate i's decision and apply temporal smoothing. The routing decision of adapter i is based on the smoothed value. See complexity analysis in App. B

5 Experimental Results

5.1 Post-Training with Local Support Learning

In this section we thoroughly evaluate our approach. Implementation details can be found in App. D.

Baselines. We finetune Qwen2.5-7B-Instruct (Qwen et al., 2025), a strong base model with established math, coding, and instruction-following capabilities. We compare LSL against two weight-based retention techniques: (i) LoRA (Hu et al., 2021) which constrains updates to a low-dimensional subspace, and (ii) OP-LoRA (Xiong & Xie, 2025), which minimizes interference between the update and the dominant singular directions of the pretrained weights. We also examine one regularization-based approach: (iii) Learning without Forgetting (LwF) (Li & Hoiem, 2017), which penalizes change in the model's output relative to the base model, using only finetuning data. We exclude methods that do not target retention of pretraining capabilities, such as O-LoRA, and methods that are impractical in pretraining settings, such as replay buffers (see Sec. 2 and App. A). In the hyperparameter sweep, we optimize for new-task validation performance; retention benchmarks are never used for selection. This compares all methods at their full learning capacity, where the trade-off between learning and forgetting is most pronounced (Fig. 7)

Finetuning tasks. We evaluate three post-training settings: (i) cybersecurity instruction tuning (Trendyol Security Team, 2025), (ii) low-resource translation from English to Igbo (Nwafor & Nguyen, 2025), and (iii) chemistry instruction tuning (Zhang et al., 2024). These tasks are chosen to be diverse and challenging, testing not only retention of pretrained capabilities but also sufficient adaptation to each new domain: English→Igbo is an open-ended generation task into a low-resource language; cybersecurity consists of instruction-response pairs requiring specialized domain knowledge; and chemistry demands understanding of molecular structure and interactions, expressed via a blend of natural language and SMILES molecular notation. Samples from each dataset are provided in Fig. 12. English→Igbo translation is evaluated using chrF (Popović, 2015), a character-level metric suited for low-resource languages. Chemistry and cybersecurity are evaluated on ChemBench (Zhang et al., 2024) and SecEval (Li et al., 2023), respectively, both multi-choice QA tasks.

Pretrained capability retention. To measure forgetting, we evaluate on three diverse tasks reflecting the base model's capabilities: GSM8K for math reasoning (Cobbe et al., 2021), HumanEval for code generation (Chen et al., 2021), and IFEval for instruction following (Zhou et al., 2023).
Our results are reported in Fig. 4. Across all benchmarks, LSL achieves strong performance on the new task while maintaining close to optimal retention. LoRA exhibits strong adaptation, yet suffers from forgetting in all settings. OP-LoRA performs very similarly to the LoRA baseline. We find that this is because forgetting is only weakly tied to the top-k singular-value subspace of the pretrained weights (see App. C). The function-regularization technique, LwF, performs better than the weight-based retention techniques, yet still falls behind LSL across all tasks. Common to all baselines is that their gradient updates are applied to all inputs, which, according to the analysis in Sec. 3, leaves them prone to interference with prior data. We study this in detail in the ablations section (Sec. 5.2).

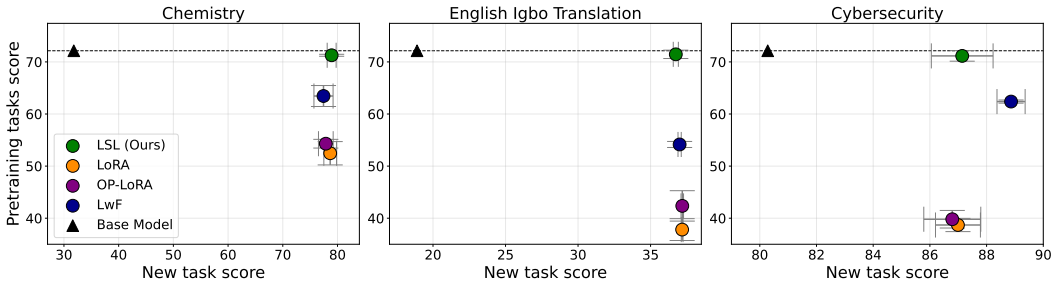


Figure 4: **Post-training with LSL**. We evaluate forgetting on a Qwen2.5-7B-Instruct model for three diverse downstream tasks. The x-axis measures the new task's performance and the y-axis measures capability retention - the average score across three pretraining benchmarks. The upper-right corner reflects stronger performance. LSL learns the downstream task while achieving near-optimal retention across all settings. This is in contrast to the baselines, which apply new updates to all inputs and thus are prone to interference on prior task data

Multiple Post-Training Phases. In Fig. 5, the model is trained on all finetuning tasks sequentially. We compare LSL to a LoRA baseline that allocates a new adapter per phase and merges it into the model before the next phase begins. In the leftmost plot, LSL retains pretrained capabilities across all phases, whereas the baseline degrades severely already after the first phase. The other plots show performance on each finetuning task. With LSL, performance on a task is preserved after its training phase ends, showing that interference between different finetuning phases is also avoided. In contrast, LoRA forgets once the corresponding phase ends: task 1 (translation) drops by 8% after two phases, and task 2 (chemistry) by 5% after one. Notably, after task 1, the zero-shot performance on task 3 (cybersecurity) falls by 11% - a form of forgetting that LSL again avoids.

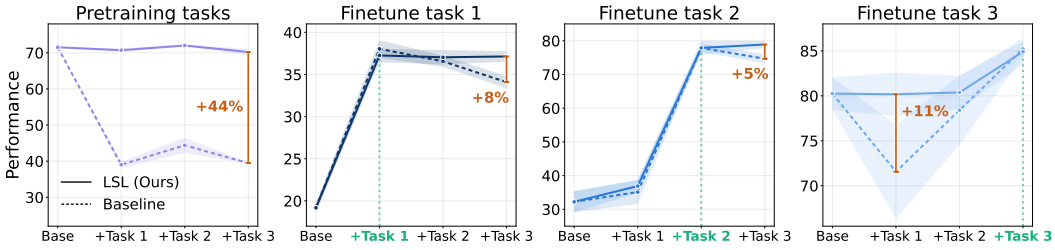


Figure 5: **Post-Training with Multiple Phases**. Each plot tracks the evaluation score of one task throughout training; the phase whose training data matches the evaluated task is highlighted in green. The training order is (1) Igbo translation, (2) Chemistry, (3) Cybersecurity. In the leftmost plot, LSL retains pretrained capabilities throughout, while the baseline drops by 44% already after one phase. On the three finetuning benchmarks, LSL retains its gains after each task's phase ends, whereas the baseline's performance declines steadily once the peak is reached.

Scaling Behavior. In Fig. 6 we test LSL across different model sizes. Each point shows the retention of pretraining tasks, defined as the ratio of performance before and after finetuning. LSL works at all scales tested (1.5B to 7B), and its performance improves with scale. LoRA also improves with scale, but far more slowly, remaining well below LSL at every size. This highlights LSL's scaling potential.

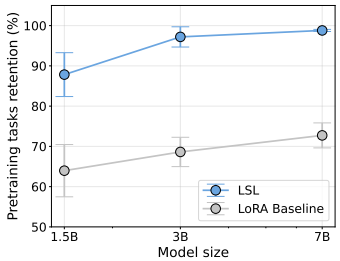


Figure 6: **Scaling behavior.** We measure the retention of pretraining tasks, defined as the ratio of performance before and after finetuning. LSL exhibits desirable scaling trends.

Robustness to Hyperparameters. By avoiding interference, LSL naturally decouples learning from forgetting. A direct benefit is that learning hyperparameters - traditionally considered to trade off with retention (Biderman et al., 2024) - no longer affect forgetting. In Fig. 7 we sweep the learning rate, adapter rank, and batch size for both LSL and LoRA. LSL learns at full capacity without sacrificing pretraining capabilities across all sweeps, while the traditional approach faces the aforementioned trade-off, limiting the model from reaching its full potential.

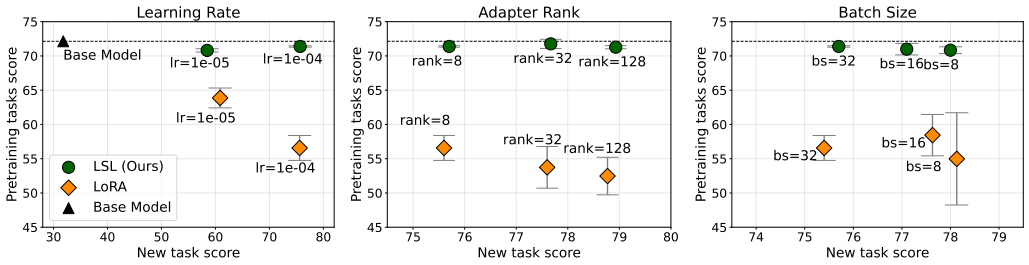


Figure 7: **Robustness to Hyperparameters**. In each plot, we sweep one of the hyperparameters. While conventional fine-tuning (♦) trades off learning against forgetting, our method (●) decouples the two, allowing the model to reach its peak capability.

Efficiency In Fig. 8 we benchmark LSL on a Nvidia B200 GPU with the Qwen2.5-7B-Instruct model. We highlight three practical results. First, LSL requires only 13.8MB of additional memory for the GMMs (Φ_{pos}, Φ_{neg}). Since Φ_{neg} is shared, this results with a negligible overhead of 6.9MB per learning phase. Second, a LSL adapter adds only 86ms of inference latency per forward pass, already competitive with standard LoRA baselines. Third, adapter training time on 1M tokens is comparable to a LoRA baseline; adding the GMM fitting step increases total training time by $\times 1.64$. We exclude Φ_{neg} fitting from this comparison as it is a one-time cost independent of the finetuning data. We note that neither inference nor training were fully optimized. The key takeaway is that LSL's overhead is on par with established baselines, while achieving superior retention.

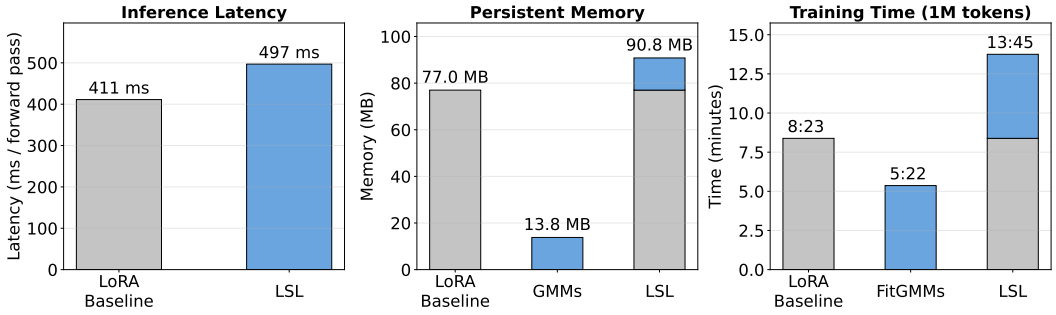


Figure 8: **Efficiency benchmarks** on Nvidia B200 with Qwen2.5-7B-Instruct. LSL achieves strong retention with memory, inference, and training overhead on par with established baselines that do not achieve comparable retention.

5.2 Ablations

Importance of Locality. An important question is whether the retention gains stem from the GMM itself, or from a more general property - local support of the update. To disentangle the two, we construct two alternative support estimators and use them to ablate the GMM. The first is Union of Spheres (UoS), defined as $\mathcal{M} = \bigcup_{i=1}^{M} \mathcal{B}(x_i, r)$, where $\mathcal{B}(x, r)$ is a ball with center x and radius r, and $\{x_i\}_{i=1}^{M}$ are representative points from the current phase's data. By construction, this gate's support is local. The gate is optimized greedily: each new point encountered during training is discarded if it falls within $\mathcal{M}$, and added to the representative set otherwise. The radius r is selected per gate via a calibration process detailed in Appendix G. The second is an MLP classifier trained on the same data used to fit the GMM (Φ_{pos} and Φ_{neg}). Importantly, the MLP's discriminative objective does not guarantee that the gate will be closed outside its training data. In Fig. 9 (right), we repeat the experiment from Sec. 5.1 with the new gates. The GMM and UoS gates achieve close to optimal retention scores while learning at full capacity. In contrast, the MLP lags behind in retention. This supports our hypothesis that local-support updates are the key ingredient for retention, rather than a specific architecture. The main difference between the UoS and GMM gates is efficiency: the UoS gate is a lookup table that stores many samples, requiring about 5.67GB of memory. In contrast, the parametric GMM learns the support with far less memory - 13.8MB in total, comparable to the LoRA adapter. For reference, the base model weights require about 14GB (in bfloat16), making the GMM overhead negligible - three orders of magnitude smaller.

Gate Behavior on OOD Data. Since the gate has no access to data from prior learning phases, it must behave correctly on out-of-distribution (OOD) data, i.e., stay closed. In Fig. 9 (Left) we compare the classification accuracy of the GMM and MLP gates. While the MLP has a slight edge on in-distribution data, it degrades significantly on all three OOD pretraining tasks. This further confirms that update locality, rather than in-domain expressiveness, is key for retention - suggesting that the gate's behavior on OOD data is the main driver for the performance observed in Sec. 5.1.

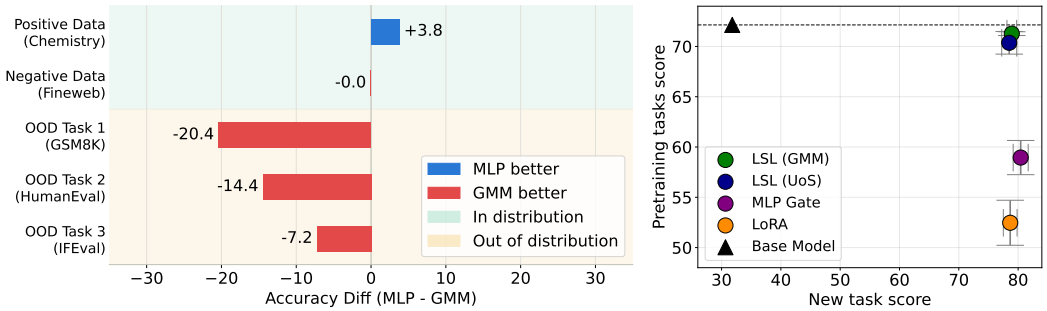


Figure 9: **Gate Ablations.** (**Left**) Classification accuracy of the MLP and GMM gates at an intermediate layer. The GMM tends to stay closed on data from previous learning phases, a property that enables stronger retention. (**Right**) Retention gains stem from support locality rather than a specific architecture. Replacing the GMM with a different local gate, Union of Spheres (UoS), yields equivalent performance, whereas a non-local MLP gate (purple) degrades retention significantly.

Number of GMM Components and Temporal Smoothing In Fig. 10 (Appendix) we ablate the remaining components of our method. We find that LSL is largely insensitive to the number of GMM components - achieving near-optimal retention with as few as two - highlighting the quality of the base model's embeddings. Temporal smoothing further improves retention from a score of 69.8 to 71.1 (96.6% to 98.8% with respect to the base model's performance), primarily by further reducing the hit rate on pretraining tasks. Yet, the dominant contribution comes from the GMM, which improves retention from 55.3 to 69.8 (76.6% to 96.6% with respect to the base model).

6 Conclusion

We studied catastrophic forgetting in large pretrained models through the geometry of each weight matrix's input space. This view yields a natural retention objective: updates should be restricted to activations from their training distribution. This reveals that updates produced by gradient-based optimizers are suboptimal, since the matrix update they produce acts on all possible inputs. Building on this analysis, we proposed Local Support Learning, which pairs a standard adapter with a gating function fit by a tailored Gaussian Mixture Model. The gate's local support is what allows it to route inputs from all phases while training only on the current one. On LLMs of up to 7 billion parameters, LSL learns at full capacity while retaining pretrained capabilities and previously finetuned skills across multiple sequential phases. It is also robust to hyperparameter choice, incurs a small memory footprint independent of the number of training tokens, and shows promising scaling behavior.

Limitations. LSL's adapters cannot be merged into the base weights, adding inference cost that grows linearly with the number of phases P. However, this computation is highly parallelizable, allowing to trade compute for inference time. Note that P counts phases, not tasks - many tasks can be learned jointly within a phase. In addition, our theoretical motivation concerns a single weight matrix; its extension to the full multi-layer network is supported empirically but not formally. Lastly, LSL relies on the quality of the base model's representations, since the gate assumes each phase's activations can be captured by a GMM. This is reflected in the scaling experiment, where larger models achieve better retention (Fig. 6).

Future research includes extending LSL to larger models, other modalities, reinforcement learning objectives, and hundreds of phases. The latter would open the door to test-time training, a form of learning currently avoided in practice due to the unpredictability of forgetting, with the potential to store new knowledge in the weights rather than in context, at a fraction of the cost. Lastly, LSL itself can be further explored, e.g., through other local gating functions and improved acceleration.

AI USE STATEMENT

In this work, we used generative AI tools for code implementation, for identifying additional related work, for checking the fluency and formatting of the manuscript, and for refining mathematical claims and proofs. We have not used generative AI tools for generating data or running experiments. We have reviewed all AI-assisted work. Code was written with an AI coding assistant in a mode where every proposed edit was reviewed and approved by an author before being applied, and its correctness was verified by the authors as well as through internal unit testing. Related work surfaced by AI tools was read and verified by the authors before citation. Manuscript suggestions were limited to wording and formatting. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work analyzes and proposes a method to overcome catastrophic forgetting, which is crucial when deploying LLMs in real-world systems. This improvement is anticipated to have a positive impact on the use of LLMs in society, particularly by increasing model efficiency and reducing energy consumption. However, we acknowledge that continual learning methods, including ours, inherit the limitations of the underlying pretrained model and training data. We emphasize the necessity of careful evaluation before deployment beyond the research environment.

REPRODUCIBILITY STATEMENT

First, we provide the source code used for the key experiments. Second, in Appendix D we provide full configurations for all experiments including instructions on how to train and evaluate the models. We also explicitly state all used datasets, models and hardware, and describe the exact metrics used in each experiment.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018. URL https://arxiv.org/abs/1711.09601.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. URL https://arxiv.org/abs/2006.11477.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less, 2024. URL https://arxiv.org/abs/2405.09673.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem, 2019a. URL https://arxiv.org/abs/1812.00420.

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, 2019b. URL https://arxiv.org/abs/1902.10486.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting, 2026. URL https://arxiv.org/abs/2510.18874.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin, 2024. URL https://arxiv.org/abs/2312.09979.

Zhen Fang, Yixuan Li, Jie Lu, Jiahua Dong, Bo Han, and Feng Liu. Is out-of-distribution detection learnable?, 2023. URL https://arxiv.org/abs/2210.14707.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. *CoRR*, abs/1910.07104, 2019. URL http://arxiv.org/abs/1910.07104.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors, 2023. URL https://arxiv.org/abs/2211.11031.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal, 2024. URL https://arxiv.org/abs/2403.01244.

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*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Guancheng Li, Yifeng Li, Wang Guannan, Haoyu Yang, and Yang Yu. Seceval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. https://github.com/XuanwuAI/SecEval, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017. URL https://arxiv.org/abs/1606.09282.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning, 2024. URL https://arxiv.org/abs/2404.00228.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Hao Wang, Aldo Pareja, Oleg Silkin, Mustafa Eyceoz, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in llms for continual learning, 2025. URL https://arxiv.org/abs/2504.07097.

Ebelechukwu Nwafor and Minh Phuc Nguyen. Fostering digital inclusion for low-resource Nigerian languages: A case study of Igbo and Nigerian Pidgin. In Atul Kr. Ojha, Chao-hong Liu, Ekaterina Vylomova, Flammie Pirinen, Jonathan Washington, Nathaniel Oco, and Xiaobing Zhao (eds.), *Proceedings of the Eighth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2025)*, pp. 44–53, Albuquerque, New Mexico, U.S.A., May 2025. Association for Computational Linguistics. ISBN 979-8-89176-230-5. doi: 10.18653/v1/2025.loresmt-1.6. URL https://aclanthology.org/2025.loresmt-1.6/.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina (eds.), *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049/.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning, 2021. URL https://arxiv.org/abs/2103.09762.

Jacob Schreiber. Pomegranate: fast and flexible probabilistic modeling in python, 2018. URL https://arxiv.org/abs/1711.00137.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Fine-tuned language models are continual learners, 2022. URL https://arxiv.org/abs/2205.12393.

Clayton D. Scott and Robert D. Nowak. Learning minimum volume sets. *Journal of Machine Learning Research*, 7(24):665–704, 2006. URL http://jmlr.org/papers/v7/scott06a.html.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral, Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. Towards expert-level medical question answering with large language models, 2023. URL https://arxiv.org/abs/2305.09617.

Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6(8):211–232, 2005. URL http://jmlr.org/papers/v6/steinwart05a.html.

Trendyol Security Team. Trendyol cybersecurity defense instruction-tuning dataset v2.0, 7 2025.

Ahmet Üstün, Viraat Aryabumi, Zheng Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. Aya model: An instruction finetuned open-access multilingual language model. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15894–15939, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.845. URL https://aclanthology.org/2024.acl-long.845/.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models, 2024. URL https://arxiv.org/abs/2405.14768.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning, 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, 2023. URL https://arxiv.org/abs/2310.14152.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652.

Yifeng Xiong and Xiaohui Xie. Oplora: Orthogonal projection lora prevents catastrophic forgetting during parameter-efficient fine-tuning, 2025. URL https://arxiv.org/abs/2510.13003.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, August 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0080-x. URL http://dx.doi.org/10.1038/s42256-019-0080-x.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017. URL https://arxiv.org/abs/1703.04200.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, Dongzhan Zhou, Shufei Zhang, Mao Su, Han-Sen Zhong, and Yuqiang Li. Chemllm: A chemical large language model, 2024. URL https://arxiv.org/abs/2402.06852.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Extended Related Work

Replay-based methods mix pretraining-like data into the finetuning set, whether from external corpora (Scialom et al., 2022), generated by the model itself (Huang et al., 2024), or routed to dedicated adapters (Dou et al., 2024) that re-learn information that may be forgotten; however, approximating the pretraining distribution during fine-tuning is impractical, and these methods can only protect capabilities represented in the replayed samples.

A recent line of work has shown that on-policy reinforcement learning is able to mitigate forgetting of pretraining capabilities (Shenfeld et al., 2025; 2026; Chen et al., 2026). These methods show promise, yet are confined to RL-based settings, rely on in-context learning capabilities and necessitate a full rollout at each learning step - requirements that our method does not impose. Furthermore, in those settings, these methods are complementary to ours since they target the training objective and data rather than the architecture and optimizer.

In lifelong model editing, several works restrict new parameters to the activations they were trained on. GRACE (Hartvigsen et al., 2023) caches layer activations as keys with a deferral radius and applies an edit only when an input falls within it, a local-support gate closely related to our Union-of-Spheres ablation (Sec. 5.2), while WISE (Wang et al., 2024) routes tokens between a side memory and the original weights by thresholding an activation score. Both target the editing of individual facts at a single chosen layer. LSL, in contrast, is a general-purpose learning framework: it trains the whole model on full finetuning datasets.

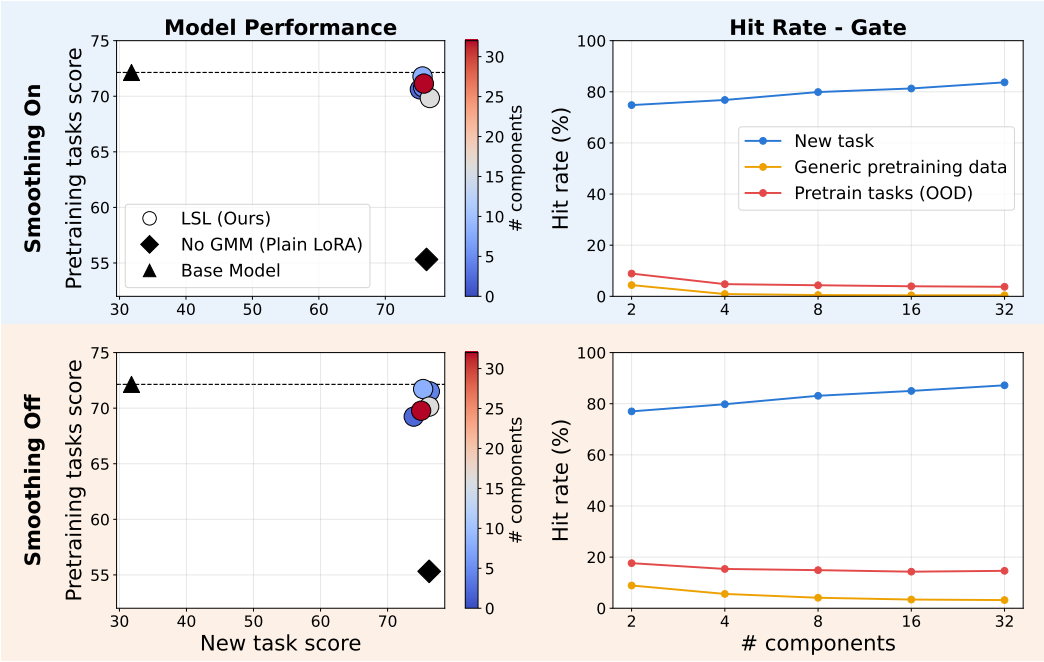


Figure 10: **Ablating the Number of GMM Components and Temporal Smoothing**. We measure performance via overall model performance (left column) and average hit rate across gates (right column). An optimal gate achieves 100% hit rate on the new task (blue) and 0% on pretrain data (yellow, red). Note that each plot sweeps the number of GMM components. We find that LSL is largely insensitive to the number of GMM components. In addition, the GMM accounts for the vast majority of retention, while temporal smoothing adds the final increment, mainly by further reducing the hit rate on pretraining tasks.

B Complexity and Acceleration

We analyze the overhead relative to typical adapter-based finetuning. LSL adds one GMM per weight matrix per phase; a naive implementation requires $O(PLKD^2)$ persistent memory and time complexity of $O(PBLKD^2)$ per forward-pass. This is for P phases, K GMM components, L

layers, hidden dimension D, and batch size B. Crucially, the memory cost per phase is independent of the number of tokens, unlike replay buffers or in-context memory, which grow with the data. Inference time, however, is also linear in P: unlike traditional finetuning, the adapters cannot be merged into the weights. The naive implementation's D^2 dependence is undesirable and hinders scaling. We alleviate it through two key accelerations that reduce the memory complexity to $O(PLKE + LDE)$ and time complexity to $O(PBL(DE + KE))$, where $E \ll D$. These are achieved by: (1) applying the Johnson–Lindenstrauss lemma to compress gate inputs to dimension E, and (2) using diagonal covariance GMMs, which we find sufficient. Training cost of the GMMs is comparable to typical adapter-based finetuning. In addition, the GMM fitting step is lightweight (Expectation-Maximization), but scaling to large finetuning datasets may require further acceleration. Another practical improvement is sharing gates across matrices with identical inputs (e.g., W_q, W_k, W_v). This does not change the asymptotic complexity, yet reduces the number of gates by 43% in a standard Qwen/LLaMA architecture. Also, since different gates train independently, parallelizing their training is highly straightforward. GMM training, as detailed in Alg. 3, is performed module-by-module; therefore, beyond the typical complexity of fitting each GMM, the only overhead is a single forward pass to generate the inputs—which is completely negligible. Empirical efficiency evaluations are in Sec. 5.1.

C Explaining OP-LoRA's Performance

In Sec. 5.1 we found that OP-LoRA performs very similarly to the LoRA baseline. To understand why, we refer to Table 1, which reports the ρ_k metric from the OP-LoRA paper - measuring how much energy of a standard LoRA adapter falls within the subspace of the top-k singular values of the pretrained weights.

k	512	128	64	16	4	1
ρ_k	1.70%	0.44%	0.27%	0.12%	0.06%	0.02%

Table 1: **Evaluating the effectiveness of OP-LoRA.** We report the ρ_k metric from the OP-LoRA paper, which measures how much energy of a standard LoRA adapter falls within the subspace of the top-k singular values of the pretrained weights (trained on the chemistry dataset). We find that forgetting is only weakly related to the preservation of the top-k singular-value subspace, explaining why OP-LoRA performs similarly to the LoRA baseline.

Across all values of k, ρ_k peaks at roughly 1.7%, meaning at least 98.3% of the update's energy already lies in the orthogonal subspace. This indicates that forgetting is only weakly tied to the top-k singular-value subspace of the pretrained weights.

D Implementation Details

Each experiment is averaged over three seeds. Hyperparameters are chosen after a sweep of learning rate $\in \{3\text{e}{-6}, 1\text{e}{-5}, 3\text{e}{-5}, 1\text{e}{-4}, 3\text{e}{-4}\}$, batch size $\in \{8, 16, 32, 64\}$, and adapter rank $\in \{8, 32, 128, 512\}$, optimizing performance on the finetuning task. Since the lora-learning phase is identical both in the GMM and LoRA baselines, we use the optimal LoRA parameters for the GMM runs as well. These are displayed in Table 2 and Table 3. For the GMM we sweep $K \in \{16, 32\}$ and smoothing coefficient $\alpha \in \{0.1, 0.3, 1.0\}$

Our GMM implementation relies on the Pomegranate library (Schreiber, 2018), while the rest of the code is written in PyTorch (Paszke et al., 2019).

For OP-LoRA we also sweep the Top-k value for $k \in \{16, 128, 512\}$ and select the best one.

LwF is implemented via a KL-divergence regularization term. At each training step, an additional forward pass acquires the base model's logits, which are compared to the logits produced with the adapter weights. We sweep the regularization coefficient over $\{0.1, 0.3, 1.0, 3.0, 10.0\}$.

The MLP gate has two hidden layers of dimension 256 and 128. The MLP optimizes a typical binary cross entropy loss, where the model tries to classify whether the current token embedding belongs the current distribution or to the small generic pretraining dataset - similar to the GMM gate. In

Table 2: Shared hyperparameters across all tasks.

Hyperparameter	**Value**
Optimizer (AdamW)	
Weight decay	0.1
$\beta_1,\ \beta_2$	0.9, 0.95
Gradient clip norm	1.0
Schedule	
Warmup ratio	0.1
Decay ratio	0.1
Min learning rate	$\mathrm{lr}/100$
GMM	
JL projection dimension	256
GMM covariance type	Diagonal
Other	
Sequence length	1024
Gradient checkpointing	Yes
LoRA α	$2r$

Table 3: Per-task hyperparameters.

	Chemistry	**Igbo Translation**	**Cybersecurity**
LoRA Adapter			
Learning rate	1e−4	1e−4	1e−4
Batch size	32	16	32
Adapter rank r	128	128	512
Epochs	3	3	1
GMM Gate			
GMM positive components	32	16	32
GMM negative components	32	32	32
Smoothing α	0.1	0.3	0.3

contrast to the GMM, a batch of MLP training data includes data from both labels (the GMM learns one distribution at a time). The gate's loss is added as an auxiliary loss to the network's main objective with a scaling factor. Similar to the GMM we perform a hyperparameter sweep including the learning rate of the MLP $\{1e-3, 1e-4, 1e-5\}$, the loss' scaling factor $\{10.0, 1.0, 0.1\}$, and the smoothing filter's coefficient $\{0.1, 0.3, 1.0\}$.

During multiphase learning the models use a learning rate of $1e-4$, batch size of 32, and adapter rank of 128. LSL used 16 components for Φ_{pos} and 32 components for Φ_{neg}, as well as a smoothing coefficient of 0.1. Like in the single-task finetuning experiment we report the average performance over 3 seeds, igbo translation and chemistry are trained over 3 epochs each and cybersecurity over 1 epoch. Due to the long runtimes of this experiment we cap the amount of eval samples of each benchmark to 300.

All model checkpoints are taken from the Huggingface Model Hub[2]:

- `Qwen/Qwen2.5-7B-Instruct`
- `Qwen/Qwen2.5-3B-Instruct`
- `Qwen/Qwen2.5-1.5B-Instruct`

Our code is based on the official Huggingface implementation.

[2]https://www.huggingface.co/models

E FITTING THE GMM AND THE WEIGHT ADAPTER

While fitting a single GMM is conceptually simple (standard EM), fitting more than one hundred GMMs (one per adapter) in high dimensions over a large dataset is computationally demanding. Furthermore, unlike gradient descent, conventional EM requires access to the full dataset at each step. To alleviate this, we fit GMMs module-by-module, as described in Algorithm 3: since each GMM depends only on its own module's inputs, this computation is exact and allows us to hold inputs for only one module at a time, discarding them before proceeding to the next.

Algorithm 3 FitGMMs

Require: Model M with L layers and J modules per layer, dataset $\mathcal{D}$
Ensure: Trained GMMs $\mathbf{\Phi} = \{\Phi^{(l,j)}\}_{l\in[L],j\in[J]}$

```
Load full dataset D
for l = 1, ..., L do
    for j = 1, ..., J do
        Compute all inputs to module-(l, j); discard previous inputs
        Fit Φ^(l,j) on inputs via EM
    end for
end for
return Φ
```

Fitting the weight adapter is identical to standard adapter finetuning. The key insight is that during training, all data is from the new task's distribution, so the gate should remain open throughout the training phase. This design avoids the need for bootstrapping - without it, Φ_{pos} would need to be initialized before the adapter has learned the new representation. We note that this introduces a train-inference mismatch (the gate may not have perfect accuracy on the new phase's data), but we find empirically that this does not affect performance.

Algorithm 4 FitAdapters

Require: Model M, fine-tuning data $\mathcal{D}_i$, number of epochs E
Ensure: Adapter weights $\mathbf{W}_{\text{adapter}}$

```
Apply new adapter weights W_adapter to M
for e = 1, ..., E do
    for each batch B ∈ D_i do
        Forward pass through M                ▷ Gate always open during training
        Compute loss L
        Update W_adapter via backpropagation
    end for
end for
return W_adapter
```

F THEORETICAL MOTIVATION - FULL DERIVATION

F.1 FORMULATION

Setting. Similar to the problem formulation in Sec. 3, we study a learning problem where tasks arrive sequentially, and analyze a single weight matrix. Earlier tasks had inputs from distributions $Q \in \mathcal{Q}$; during training an update is learned for the current task's input distribution P; during test time the system is evaluated on inputs from all of $\mathcal{Q}$ and P. As in Fig. 1 (Left), a gate g decides whether to apply the update. It is trained on samples from P and possibly some small buffer Q_0; the earlier distributions $\mathcal{Q}$ are unknown at training time. We assume that the domain of the input features is bounded. This is a realistic assumption, since almost any weight matrix processes inputs

that went through a layer normalization block.[3] We assume a task's features occupy a proper subset of the domain and that within that subset P's density is bounded below by $c > 0$; all densities are bounded above by M. Beyond this we assume nothing about P, and nothing about $\mathcal{Q}$ or Q_0. The same symbol denotes a distribution and its density.

Target. Apply the update where the input looks like the data it was fitted on:

$$A^* = \operatorname{supp} P, \qquad g^* = \mathbf{1}[x \in A^*].$$

where $\operatorname{supp} P$ is the support of P. This is the minimum-volume set carrying all of P's mass - the $\alpha = 1$ case of Scott & Nowak (2006). A^* is unique.[4] In case of an overlap between the distributions P and Q, the policy is that P wins: a point in A^* receives the update whichever distribution produced it, since later updates carry more recent information.

Error. A gate is its accept set $\hat{A}$; its error on Q is $Q(\hat{A} \triangle A^*)$, where $\hat{A} \triangle A^* = (\hat{A} \setminus A^*) \cup (A^* \setminus \hat{A})$ is the symmetric difference - the set of points on which the gate and the target disagree. Since $\mathcal{Q}$ is unknown, the gate is judged against the worst admissible test distribution, and that worst case is exact:

$$\sup_{Q \le M} \operatorname{err}_Q(\hat{A}) \;=\; \min\{1,\; M \operatorname{vol}(\hat{A} \triangle A^*)\},$$

the supremum being attained by a Q of density M concentrated on the disagreement set. Minimizing the volume of disagreement with A^* is therefore the minimax objective, not merely an upper bound.

F.2 The gate's training objective is blind to the excess

The symmetric difference has two parts. The *deficit* is the part of A^* the gate rejects. Notice that covering P in population covers every Q there too; any fitting procedure handles this and we do not discuss it further. The *excess* is what the gate accepts outside A^*:

$$E(\hat{A}) = \hat{A} \setminus A^*, \qquad \sup_{Q \le M} Q\big(E(\hat{A})\big) = \min\{1,\; M \operatorname{vol}(E(\hat{A}))\},$$

We are mainly interested in the excess because P has no mass there, so any objective that is a P-weighted average of a per-point loss (e.g. a training loss) is blind to it (Steinwart et al., 2005).

Intuition for a problematic scenario (not a proof): take any trained gate from a "sufficiently expressive" hypothesis class and change it to accept an arbitrary region outside A^*. It has the exact same P-averaged objective value (the changed region has no P-mass), yet a Q concentrated on this region is now accepted in full. Since the objective cannot tell the two gates apart, which of them training returns is decided by something other than the data.

F.3 Controlling what the objective cannot control

From the above, the training objective of the gate supplies no control over the excess. Whatever controls it therefore comes from elsewhere: the hypothesis class, the estimation procedure, or the optimizer and its regularization. Estimation families supply this control in different ways and at different costs; the following is not exhaustive.

Local estimators. Balls of radius r around the samples, or a kernel density thresholded. Every sample lies in A^*, so a union of balls has excess confined to an r-shell around A^* with no assumption on the shape of P (containment in the shell, not a rate). The costs are memory growing with n and coverage: gaps inside A^* are rejected until samples fill them, which in high dimension requires many samples or a large r, widening the shell.

[3] Normalized features occupy a lower-dimensional set inside the box; volume is taken on that domain. The bounds do not depend on this choice, but uniform negatives must be drawn on the actual feature domain, not the ambient box, which they would almost surely miss.

[4] point-wise, every input either has density at least c or density zero, nothing in between, so there is no fringe to trade at the margin.

Density estimation. Accept where an estimated density exceeds a threshold, $\hat{A} = \{\hat{P} \geq \tau\}$ with $0 < \tau < c$[5]. Split the L^1 error over the excess $E = \hat{A} \setminus A^*$, the deficit $D = A^* \setminus \hat{A}$, and the rest:

$$\|\hat{P} - P\|_1 = \int_E |\hat{P} - P| + \int_D |\hat{P} - P| + \int_{\text{rest}} |\hat{P} - P|.$$

On E the gate accepts, so $\hat{P} \geq \tau$, while $P = 0$: the integrand is at least τ, giving $\int_E |\hat{P} - P| \geq \tau \,\mathrm{vol}(E)$. On D the gate rejects, so $\hat{P} < \tau$, while $P \geq c$: the integrand is at least $c - \tau$, giving $\int_D |\hat{P} - P| \geq (c - \tau)\,\mathrm{vol}(D)$. From the split, keeping E and D in the same inequality:

$$\|\hat{P} - P\|_1 \;\geq\; \tau\,\mathrm{vol}(E) + (c - \tau)\,\mathrm{vol}(D) \;\geq\; \min\{\tau, c - \tau\}\big(\mathrm{vol}(E) + \mathrm{vol}(D)\big),$$

hence $\mathrm{vol}(\hat{A} \triangle A^*) \leq \|\hat{P} - P\|_1 / \min\{\tau, c - \tau\}$. The first inequality is the one that matters - the room left for an adversarial Q is $M\,\mathrm{vol}(E) \leq M\|\hat{P} - P\|_1/\tau$. An L^1-consistent density estimate therefore gives a consistent gate.

In contrast to the union of spheres estimator, a fixed-size parametric family (e.g. a Gaussian mixture) has constant memory. However, this has a price - consistency requires $\|\hat{P} - P\|_1 \to 0$, which the family must allow; when it cannot, the excess persists with unlimited data (misspecified "two-mode" example below).

Density with an input-dependent threshold. The same argument holds for $\hat{A} = \{\hat{P}(x) \geq \tau(x)\}$ with $\tau_{\min} = \inf_x \tau(x)$ and $\tau_{\max} = \sup_{x \in A^*} \tau(x)$: the integrand is at least $\tau_{\min}$ on E and at least $c - \tau_{\max}$ on D, so $\mathrm{vol}(E) \leq \|\hat{P} - P\|_1 / \tau_{\min}$ whenever $\tau_{\min} > 0$, and $\mathrm{vol}(D) \leq \|\hat{P} - P\|_1 / (c - \tau_{\max})$ if also $\tau_{\max} < c$ (a constant τ recovers the bound above). LSL is this gate with $\hat{P} = \Phi_{\text{pos}}$ and $\tau = \Phi_{\text{neg}}$. Choosing $\tau(x)$ so that the bound is tight is not trivial; Φ_{neg} is a practical approximation that satisfies $\tau_{\min} > 0$ by construction, since a Gaussian mixture is positive on the bounded domain. Empirically, the resulting gate opens on about 16% of tokens from out-of-distribution pretraining tasks (5% with temporal smoothing) while retaining 96.6% of pretrained performance (98.8% with smoothing; Fig. 10).

Classifier. Train a classifier to separate P (positives) from a buffer Q_0 (negatives). Under equal priors its population accept set is $\{P \geq Q_0\}$ - wherever Q_0 has mass it rejects unless P is denser, and wherever neither has mass the objective is indifferent and the decision is left to other factors. Which set this is depends entirely on Q_0.

- *Generic buffer* (some data). Outside A^* it rejects where the buffer sits and is unconstrained everywhere else: that region is exposed. Inside A^*, the classifier withholds the update wherever $Q_0 > P$, against the policy. This is the class of the MLP gate tested in Sec. 5.2, which we found empirically to be suboptimal.
- *Buffer drawn from earlier tasks.* The exposed region shrinks as the buffer covers more of $\mathcal{Q}$, and vanishes when the buffer covers every earlier distribution outside A^*. This requires both access $\mathcal{Q}$ and being able to train on all of $\mathcal{Q}$ at each phase - two assumptions that are impractical for large-scale pretraining settings.
- *Uniform over the domain.* assuming $Q_0 \equiv u$ with $u = 1/\mathrm{vol}(\text{domain}) < c$: the accept set is $\{P \geq u\} = A^*$ exactly. This is minimum-volume estimation - reject uniform points subject to covering P - the target's own definition (Scott & Nowak, 2006, §8) and the learner of Fang et al. (2023, Thm. 9). It needs a hypothesis class that approximates A^* with controlled complexity, and in high dimension few uniform points land near a small A^*, requiring a very large sample. In effect it is support estimation by an expensive route.

Classification against an arbitrary reference distribution therefore does not, by itself, guarantee support recovery. The input-dependent-threshold bound does not apply to a general classifier, whose score need not estimate P and whose threshold need not be bounded away from zero.

Examples. Below are three one-dimensional cases that make the failure modes concrete: a parametric family that cannot represent P and so accepts a gap forever; two densities that likelihood

[5]The true density jumps from 0 outside A^* to at least c inside, so any τ in the gap gives the same set; it only sets the margin for estimation error on each side (a bump of $\hat{P}$ above τ outside is accepted, a dip below τ inside is rejected). Any fixed τ strictly inside $(0, c)$ is consistent.

cannot tell apart but whose gates differ; and a classifier with zero training error that accepts a region of volume ten outside the support.

- *Misspecified.* $P = \frac{1}{2}\mathrm{Unif}[-3,-2] + \frac{1}{2}\mathrm{Unif}[2,3]$; a single Gaussian fitted by maximum likelihood converges to $\mathcal{N}(0, 19/3)$, and any level set covering both modes covers the gap. For $Q = \mathrm{Unif}[-1,1]$ the excess is 1 with unlimited data.
- *Unidentified.* $P = \mathrm{Unif}[0,1]$, threshold 0.05, and a class of two densities $f_0 = 0.9 \cdot \mathbf{1}_{[0,1]} + 0.025 \cdot \mathbf{1}_{[2,6]}$, $f_1 = 0.9 \cdot \mathbf{1}_{[0,1]} + 0.1 \cdot \mathbf{1}_{[7,8]}$. They have identical likelihood on every sample from P, yet $\{f_0 \geq 0.05\} = [0,1]$ and $\{f_1 \geq 0.05\} = [0,1] \cup [7,8]$: likelihood fixes how much mass sits off the data, not where. (The class is misspecified - the true density beats both - so this concerns flexible off-data mass, not maximum likelihood in a correct family.)
- *Classifier.* $P = \mathrm{Unif}[0,1]$ against a buffer $Q_0 = \mathrm{Unif}[2,3]$ on the domain $[-10,10]$. Every threshold in $(1,2)$ separates the two perfectly, so the classifier is unidentified among them; and every one of them accepts $[-10,0]$, outside $\mathrm{supp}\, P$ - excess of volume 10 from a classifier with zero training error.

Conclusion. While every training objective on P is blind to the excess, we show that the disagreement volume can be controlled by something one can actually minimize: L^1 density error, or coverage plus volume. Every family that controls the excess supplies that missing quantity from outside the data, and pays for it. A parametric density supplies it through its family, at constant memory, and pays in assumptions: when the family cannot represent P, the excess persists. A local estimator supplies it through its radius with no assumption on P, and pays in memory that grows with the data. A uniform buffer supplies it through a volume count, and pays in reference samples that grow with the dimension.

G Union of Spheres - Radius Calibration

The support $\mathcal{M}$ is defined using a predefined radius r with respect to the representative subset. We find that a global r for all UoS gates does not perform well due to the different dimensions (e.g. the dimension of the down projection matrix of the MLP is about 4 times larger than the dimension of the other matrices), and different input distributions between layers. To find the optimal r per gate we propose a one-time calibration process, detailed below. The algorithm relies on two sets of points: A small set from the current phase's data, and a small random sample from the generic pre-training data. We first record the activations of the pre-trained model for both sets. Then, per gate, we compute a ROC curve for different values of r (see example in Fig. 11). Notice that there are several curves per gate - each one is for a different compaction ratio - namely, different values for the upper limit of number of representatives M. The optimal r is the one that gives the maximal compaction rate under the constraint that it is above a global TPR threshold and below a global FPR threshold. For our experiment we found that $\mathrm{TPR}_{\mathrm{TH}} = 0.17, \mathrm{FPR}_{\mathrm{TH}} = 0.03$ gives optimal results.

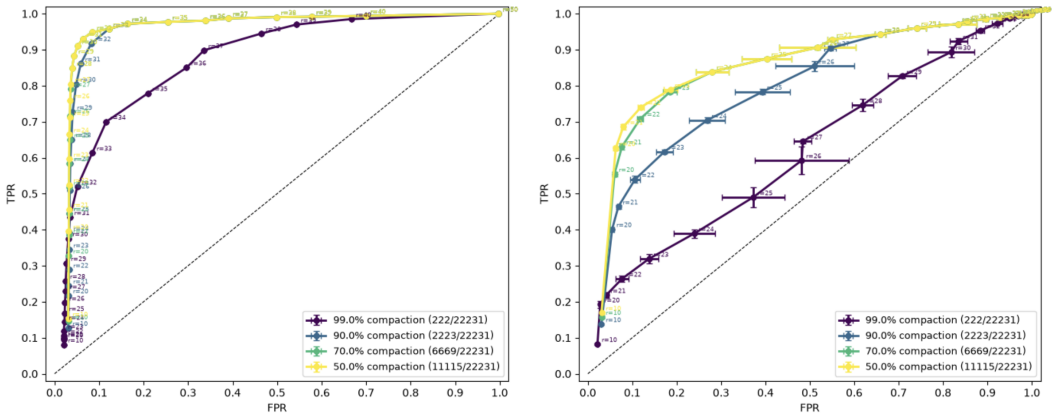


Figure 11: **ROC Curves for UoS Radius Calibration**. We show two representative curves, one for the gate projection (left) and one for the down projection (right), both in layer 17. Each curve color indicates different compaction rates (dictated by the maximal value of representative points M). The selected radius is the one with the maximal compaction rate, subject to being within the allowed TPR,FPR region (defined globally for all gates).

Cybersecurity

Input	Describe an approach to enhance Static Analysis for Dormant Malware Signatures on resource-constrained IoT endpoints with a constrained SOC headcount?
Output	*Enhancing static analysis for dormant malware signatures on resource-constrained IoT endpoints requires a multi-layered strategy that optimizes computational efficiency while maintaining detection efficacy. The NIST Cybersecurity Framework's Detect function (DE.CM-1) emphasizes continuous monitoring, which is particularly challenging in constrained environments ...*

English → Igbo Translation

Input	The story goes that this incident occurred at the residence of the twin brothers in Rumukpakani on May 17, 2018.
Output	*Akụkọ kwuru na ihe a mere n'akụkụ ụlọọrụ ndị uweojii dị na Rumukpakani n'ụbọchị 17 Mee, 2018.*

Chemistry

Input	Assuming absolute perfection, what is the projected yield for the chemical transformation represented by the SMILES code ClCCOC(c1ccccc1)c1ccc(Cl)cc1.C1CNCCN1>>Clc1ccc(C(OCCN2CCNCC2)c2ccccc2)cc1?
Output	*In an ideal setting, ClCCOC(c1ccccc1)c1ccc(Cl)cc1, C1CNCCN1 are transformed into Clc1ccc(C(OCCN2CCNCC2)c2ccccc2)cc1 in the reaction, with an expected yield of 73.0.*

Figure 12: **Samples from the finetuning datasets**.