DYNAMICS: FINE-TUNING LLMS WITH PERFORMANCE CONSTRAINTS USING DYNAMIC MIXTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-domain fine-tuning of large language models requires improving performance on target domains while preserving previous capabilities, such as general knowledge, instruction following, or safety evaluations. Existing data mixing strategies rely on fixed heuristics or adaptive rules that cannot explicitly enforce preservation of such capabilities. We propose DYNAMICS, a dynamic mixture optimizer that casts multi-domain fine-tuning as a constrained optimization problem. At each update, DYNAMICS performs short domain-specific probing runs to estimate a slope matrix of local cross-domain effects, capturing how training on each fine-tuning dataset affects each evaluation domain. These estimates are then used to compute mixture weights through optimization over the probability simplex, with the objective of improving target-domain performance while keeping constrained-domain metrics within a specified tolerance of reference levels. Because these effects are measured by finite differences rather than gradients, targets and constraints need not be differentiable, or present in the fine-tuning data, and can be specified directly as benchmark accuracies. Across scenarios with varying numbers of target and constrained domains, and with loss- or accuracy-based objectives, DYNAMICS achieves stronger target-domain improvements and higher constraint satisfaction than static, dynamic, similarity-based and probing-based alternatives, without a reference model, per-example scoring, or manually tuned weights.

1 INTRODUCTION

Fine-tuning large language models (LLMs) on new specialization domains presents a fundamental challenge: models must acquire new capabilities, such as code generation or mathematical reasoning, while retaining previously learned abilities, such as general knowledge, instruction following, or safety compliance. This adaptation–retention tension is reflected in two related phenomena. First, *catastrophic forgetting* (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017; Shi et al., 2024a) degrades performance on domains previously mastered by the LLM. Second, because supervised fine-tuning (SFT) datasets are typically small compared to pre-training corpora, large models tend to *overfit*, adapting rapidly to the specialization domain while losing broader generalization abilities (Shi et al., 2024b; Gupta et al., 2025; Hao et al., 2025). These risks motivate fine-tuning models with *data mixtures*, where specialization data are interleaved with auxiliary or pre-training domains that can regularize adaptation and mitigate forgetting (He et al., 2019; Ibrahim et al., 2024; Bethune et al., 2025).

What mixture for fine-tuning? As models grow in size, post-training absorbs a larger share of total training computation, with SFT emerging as one of its key scaling dimensions (Tie et al., 2025; Olmo et al., 2025). In that context, the costs of picking suboptimal mixtures add up. As a result, without a principled selection method, practitioners resort to trial-and-error attempts, where meaningfully covering the simplex grows exponentially as the number of fine-tuning datasets grows. Simpler strategies such as uniform, dataset-size proportional, or temperature-based sampling are the most popular (Wang et al., 2020; Mueller et al., 2024). These methods assume fixed inter-domain relationships throughout training and, crucially, offer no way to guarantee that critical capabilities are retained. More recently, a line of work updates mixtures during training, either from curriculum or loss-based signals (Wu et al., 2024), or by modeling cross-domain interactions and reweighting online (Chen et al., 2023; 2024). These methods adapt to evolving inter-domain relationships, but

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.

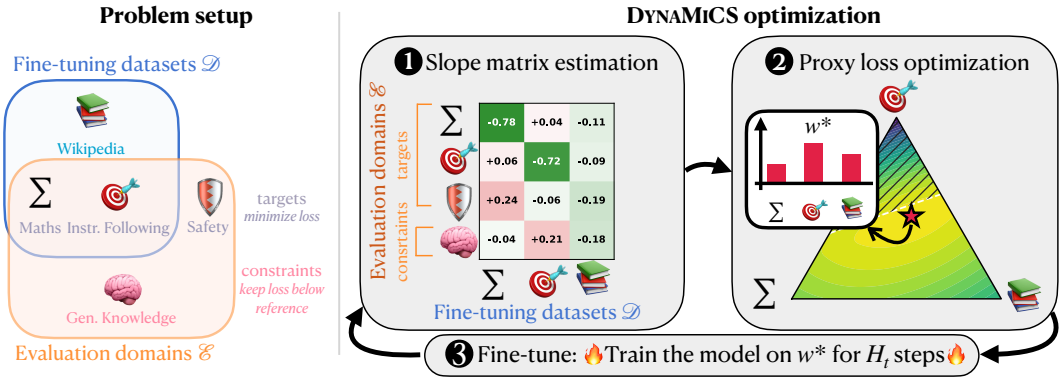


Figure 1: **Problem setup.** Fine-tuning datasets $\mathcal{D}$ provide the data for mixture selection: target datasets plus optional auxiliary ones for transfer or regularization. Evaluation domains $\mathcal{E}$ are partitioned into target domains, whose losses are minimized, and constrained domains, whose losses must remain below reference values. **DynaMiCS optimization.** At each update, DynaMiCS estimates a slope matrix $\mathbf{S}(t)$ **(1)**, where $S_{ij}(t)$ measures the local effect of D_j on evaluation loss L_i. Green/red entries denote loss decrease/increase. Given $\mathbf{S}(t)$, DynaMiCS solves a constrained optimization problem to obtain weights $\mathbf{w}^*$ **(2)**, trains with them for H_t steps **(3)**, and then repeats. The simplex illustrates the proxy objective landscape; white lines mark constraint boundaries; values are illustrative.

optimize unconstrained performance objectives: none lets a practitioner name a capability that must be preserved and state an explicit tolerance for its degradation. Knowledge preservation is therefore treated only implicitly as part of the performance trade-off. Finally, methods originally developed for pre-training rely on proxy models, making them costly to repurpose in the fine-tuning setting (Xie et al., 2023; Fan et al., 2024b; Wang et al., 2025).

Controlled Dynamic Mixtures We propose *dynamic mixtures for constraint satisfaction* (DynaMiCS), a dynamic mixture optimizer that estimates local cross-domain effects and uses them to compute constraint-aware sampling weights during training, see Figure 1. We consider three types of knowledge domains: *fine-tuning* datasets, from which the mixture is computed; and two *evaluation* domains: *target* domains, on which performance should be improved; and *constrained* domains, for which the model's performance should remain above a reference level. DynaMiCS periodically probes the impact on evaluation domains of fine-tuning for a few steps, using each of the fine-tuning datasets in isolation. The resulting loss changes across all evaluation domains form a *slope matrix*, a "finite-differences" estimate of each fine-tuning dataset's effect on targets and constraints. Using this simple model, DynaMiCS computes the optimal mixture weight that would achieve the best possible outcome (best improvement on target evaluations while remaining feasible), and commits to that mixture for a certain number of steps, before reevaluating the slope matrix. Our contributions are the following:

- **Constrained fine-tuning formulation.** We cast multi-domain fine-tuning as constrained optimization, where mixture weights are chosen to improve target domains while preserving performance on constrained capabilities.
- **Efficient constraint-aware mixture updates.** We propose DynaMiCS, which converts short domain-specific probing runs into local estimates of transfer and interference, in each task's own units (perplexity or accuracy alike, since effects are measured by finite differences rather than gradients). Building on probing mechanisms used in prior work (Chen et al., 2023; 2024), DynaMiCS turns these local measurements into constraint-aware mixture updates that explicitly trade off improvement against capability preservation.
- **Comprehensive empirical analysis.** We evaluate DynaMiCS on 50 multi-domain fine-tuning scenarios spanning 1–3 target domains and 3–10 constrained domains, testing Gemma3-12B (Gemma Team, 2025), Qwen3-8B (Qwen Team, 2025), and Qwen2.5-3B (Qwen Team, 2024); the latter, for multiple LoRA ranks as well as full-parameter fine-tuning, with objectives and constraints expressed as losses or as benchmark accuracies. Across over 8,500 fine-tuning runs, and against fixed-weight sweeps and five mixture optimization methods, DynaMiCS achieves reliable constraint satisfaction, which remains a challenge for all baselines, while reaching stronger target-domain improvements at substantially lower cost than best-of-many baseline selection.

2 Related Work

Forgetting, overfitting, and data mixtures. Fine-tuning pre-trained language models on specialized domains can degrade previously acquired capabilities, a phenomenon known as *catastrophic forgetting* (McCloskey & Cohen, 1989; Shi et al., 2024a; Kirkpatrick et al., 2017). In SFT, this risk is compounded by the small size of specialization datasets relative to pre-training corpora: models may adapt rapidly to the target domain while overfitting it and losing broader generalization ability (Shi et al., 2024b). A common mitigation is to fine-tune on *data mixtures*, interleaving target-domain data with supporting or general-domain data that can act as regularizers (Bethune et al., 2025; He et al., 2019; Ibrahim et al., 2024). This raises the central question of how to choose mixture weights to improve target domains while limiting regressions on domains that should remain stable.

Static and dynamic mixture selection. Standard mixture choices include uniform sampling, sampling proportional to dataset size, and temperature-based sampling (Wang et al., 2020; Mueller et al., 2024). More principled static approaches formulate mixture selection as an optimization problem, for example, using scaling-law predictions to estimate useful mixture weights (Li et al., 2025). However, static mixtures cannot react to evolving transfer and interference. Recent approaches adapt the mixture online, using staged curricula (Dong et al., 2023), reinforcement-learning controllers (Wu et al., 2024; Shin et al., 2025), loss-based signals (Lu et al., 2025), sample-level influence estimates (Liang et al., 2025), or explicit models of cross-domain effects (Chen et al., 2023; 2024; Wang et al., 2025) (Appendix A). Among the latter, Skill-It and Aioli are closest to DynaMiCS: Skill-It builds a skill-influence matrix, a structural precursor of our slope matrix, though non-negative and estimated once, while our local predictor has the same linear form as Aioli's dynamic mixing law. Across these methods, however, mixture selection remains unconstrained: even those that explicitly account for forgetting (Lu et al., 2025) offer no way to name a capability to preserve and bound its degradation.

Relation to pre-training mixture optimization. Mixture optimization has also been studied in pre-training, including proxy-model methods such as DoReMi (Xie et al., 2023) and DoGE (Fan et al., 2024b), benchmark-driven data curation (Li et al., 2024), scaling-law-based mixture prediction (Ye et al., 2025), online reweighting (Zhao et al., 2026), gradient-alignment methods (Fan et al., 2024a), and multi-target or domain-affinity-based reweighting frameworks (Fan et al., 2025b; Xie et al., 2025). DoReMi minimizes the worst-case excess loss across domains with Group-DRO on a small proxy model, and transfers the time-average of the resulting weights to the larger model as a single static mixture; DoGE instead combines independently estimated per-domain effects. In both cases the mixture is fixed before the final run: relative to a sweep over fixed weights, such methods contribute one additional candidate mixture, chosen without any notion of feasibility. Moreover, our setting differs from pre-training mixture optimization: in SFT, data is smaller, repeated more often, and forgetting of specific capabilities, such as alignment, is often a central concern. An extended discussion of this body of work is provided in Appendix A.

3 DynaMiCS: Dynamic, Slope-Informed Data Mixture Optimization

We consider a fine-tuning setting where data from N *fine-tuning datasets* $\mathcal{D} = \{D_1, \ldots, D_N\}$ can be accessed, and each of M *evaluation domains* can be evaluated through a loss (or negative performance metric) $L_i, i \leq M$, where $\{1, \ldots, M\}$ is split into two disjoint target and constrained sets of indices, $\mathcal{T} \cup \mathcal{C} = \{1, \ldots, M\}$. The fine-tuning datasets and evaluation domains may overlap but need not coincide: target or constrained domains may be benchmarks not included in training, such as safety or general-knowledge evaluations, while other domains may enter the mixture without being optimization targets. This includes settings in which a practitioner has one or a few small specialized datasets they want the model to learn, auxiliary corpora that may stabilize training, or held-out benchmarks on which performance should be preserved (e.g., safety or commonsense knowledge observable via a handful of evaluation batches, but for which training data may not be available).

3.1 Constrained data-mixture optimization

Training proceeds on a mixture of fine-tuning datasets with weights $\mathbf{w} = (w_1, \ldots, w_N) \in \Delta^N$, where $\Delta^N = \{\mathbf{w} \in \mathbb{R}^N_{\geq 0} : \sum_j w_j = 1\}$ is the probability simplex. At each training step t, a batch

is drawn from fine-tuning dataset D_j with proportion w_j. We formulate mixture selection as the constrained problem

$$\min_{\mathbf{w}\in\Delta^N} \sum_{i\in\mathcal{T}} \hat{L}_i(\mathbf{w}) \quad \text{s.t.} \quad \hat{L}_i(\mathbf{w}) \le L_i^{\text{ref}}, \quad \forall\, i \in \mathcal{C}, \tag{1}$$

where $\hat{L}_i(\mathbf{w})$ is the *predicted* loss on the i-th evaluation domain after training with weights $\mathbf{w}$ for a fixed horizon, and L_i^{ref} is the reference loss, computed on the base model prior to fine-tuning, for constrained domain E_i. The central challenge is estimating $\hat{L}_i(\mathbf{w})$ reliably; we address this by measuring the marginal effect of each fine-tuning dataset independently and combining these effects through a local linear approximation.

3.2 Objective and constraint approximation using finite-differences

To approximate $\hat{L}_i$ used in equation 1, we resort to a "finite differences" approach to compute a linear approximation. We form a *slope matrix* $\mathbf{S}(t) \in \mathbb{R}^{M\times N}$: Each entry $S_{ij}(t)$ captures the change in the metric for evaluation domain E_i induced by training on fine-tuning dataset D_j exclusively for one step. An important flexibility provided by this approach is that computing the slope matrix does *not* require evaluation domains (targets or constraints) to be represented within the fine-tuning datasets.

For each fine-tuning dataset D_j, starting from the current checkpoint (parameters and optimizer state), we perform c_t temporary probing steps using only data from D_j with the same optimizer used for the ongoing fine-tuning training. Both model and optimizer are restored after each domain's probe, so these updates serve only to estimate local loss changes.

We then measure the impact of this domain-specific probe on each evaluation domain: Negative entries indicate that probing on D_j decreases the loss on E_i, corresponding to beneficial transfer. Positive entries indicate an increase in loss, corresponding to interference or, when E_i is a constrained domain, potential forgetting.

$$S_{ij}(t) = \frac{\overbrace{L_i(\theta_j^{(c_t)})}^{\text{loss after probing}} - \overbrace{L_i(\theta_t)}^{\text{current loss}}}{c_t}. \tag{2}$$

3.3 Adaptive weight optimization

DynaMiCS leverages the slope matrix to predict the effect of candidate mixtures on the evaluation losses. Given $\mathbf{S}(t)$, we approximate the loss on evaluation domain E_i after training for H_t steps (the interval until the next weight update), which may vary across training following a schedule, with mixture weights $\mathbf{w}$ as in equation 3. Equation 3 is a local linear approximation of the loss: per-domain effects are assumed to combine additively, and loss changes are extrapolated only over the short horizon between two mixture updates H_t. Frequent re-estimation of $\mathbf{S}(t)$ ensures the reliability of this approximation, and we verify its fidelity directly (Appendix L.1). Note that we also explored a variant of this approach where we fit parametric curves to the loss trajectories observed during slope estimation, allowing for non-linear loss dynamics over the prediction horizon. This approach yields more accurate predictions but requires frequent L_i evaluations to fit such curves. This inflates the overall compute budget, without significant performance gains (Appendix L.2).

$$\hat{L}_i(\mathbf{w}) = \underbrace{L_i(\theta_t)}_{\text{current loss}} + H_t \underbrace{\sum_{j=1}^{N} S_{ij}(t) w_j}_{\text{predicted change}}. \tag{3}$$

Substituting equation 3 into equation 1 and dropping terms constant in $\mathbf{w}$ yields a linear objective minimized over the simplex, with $\mathbf{S}_{i\cdot}(t)$ denoting the i-th row of $\mathbf{S}(t)$. This optimization is a low-dimensional problem in N, the number of fine-tuning datasets. Problem equation 1 is a linear program. However, we relax the hard constraints into a squared-hinge penalty so that the optimizer returns a least-violating mixture when no feasible point exists, a practically important regime:

$$\min_{\mathbf{w}\in\Delta^N} \underbrace{\textstyle\sum_{i\in\mathcal{T}} \mathbf{S}_{i\cdot}(t)^\top \mathbf{w}}_{\text{minimize target loss}} + \lambda \underbrace{\textstyle\sum_{i\in\mathcal{C}} \max\Big(0,\ \overbrace{\hat{L}_i(\mathbf{w}) - L_i^{\text{ref}}}^{\text{constraint violation}} + \varepsilon\Big)^2}_{\text{constraint penalties}}, \tag{4}$$

where $\hat{L}_i(\mathbf{w})$ is the predicted loss from equation 3 with prediction horizon H_t, $\lambda > 0$ controls the penalty strength, and $\varepsilon \geq 0$ is a safety margin that activates the penalty before the constraint boundary is reached. At each update, we solve equation 4 over a grid of (λ, ε) values and select the feasible solution with the lowest target objective, or the least-violating solution if none is feasible (Appendix D). The cost of these computations is negligible. The resulting weights $\mathbf{w}^*$ are used as the mixture proportions until the next scheduled update (Figure 1). At designated training steps, DYNAMICS re-estimates $\mathbf{S}(t)$ and recomputes $\mathbf{w}^*$. The update schedule, defined as $\mathcal{U}$, is configurable: it can be a fixed interval, a geometric progression with doubling intervals, or any user-defined sequence. The full training loop is given in Algorithm 2; Figure 1 offers an illustration of DYNAMICS; hyperparameter choices ($\mathcal{U}$, c) and compute costs are detailed in Section 4.

Algorithm 1 Slope matrix estimation

Require: Current parameters θ_t; optimizer state; fine-tuning datasets $\mathcal{D}$; evaluation domains $\mathcal{E}$; probing budget c_t
Ensure: Slope matrix $\mathbf{S}(t) \in \mathbb{R}^{M \times N}$

Evaluate current losses: $L_i^{(0)} \leftarrow L_i(\theta_t)$ for all $E_i \in \mathcal{E}$
for each fine-tuning dataset $D_j \in \mathcal{D}$ **do**
  Restore θ_t and optimizer state *// temp probe*
  Fine-tune c_t steps on D_j only, yielding $\theta_j^{(c_t)}$
  Fill column j in S:
  $S_{ij}(t) \leftarrow \dfrac{L_i(\theta_j^{(c_t)}) - L_i^{(0)}}{c_t} \quad \forall E_i \in \mathcal{E}$
end for
return $\mathbf{S}(t)$

Algorithm 2 Adaptive mixture optimization

Require: Initial model θ_0; fine-tuning datasets $\mathcal{D}$; eval domains $\mathcal{E}$; target set $\mathcal{T}$; constrained set $\mathcal{C}$; total steps T_{total}; schedule $\mathcal{U} = \{0, t_1, \ldots\}$; budget c
Ensure: Fine-tuned model

Evaluate reference losses: $L_i^{\text{ref}} \leftarrow L_i(\theta_0)\ \forall\, i \in \mathcal{C}$
for $t = 0, \ldots, T_{\text{total}} - 1$ **do**
  if $t \in \mathcal{U}$ **then**
    $\mathbf{S}(t) \leftarrow$ Alg. $1(\theta_t, \mathcal{D}, \mathcal{E}, c_t)$
    $\mathbf{w}^* \leftarrow \arg\min$ of equation 4 using $\mathbf{S}(t)$
  end if
  Train one step on mixture $(\mathcal{D}, \mathbf{w}^*)$ to obtain θ_{t+1}
end for
return $\theta_{T_{\text{total}}}$ *// post-hoc: select best*

4 EXPERIMENTAL SETUP

4.1 EVALUATION SCENARIOS

We evaluate DYNAMICS across 50 scenarios designed to cover increasingly challenging constrained fine-tuning settings. Each scenario defines a set of fine-tuning datasets, here split into target domains whose loss is explicitly minimized and non-target domains that provide additional data for mixture selection. It also defines constrained domains, which are used only for evaluation and whose loss must remain below the pre-fine-tuning reference. Targets span mathematical reasoning, code generation, general instruction following, medical QA, function calling, and multilingual tasks. Constrained domains represent capabilities to preserve, such as safety, commonsense reasoning, or general knowledge. Non-target domains range from generic corpora such as Wikipedia (Wikimedia, 2024) to domain-specific datasets whose transfer relationship to the targets is not known a priori. The number of targets ranges from 1 to 3 and the number of constraints from 3 to 10. Scenario definitions and dataset descriptions are given in Appendices B and C.

Since DYNAMICS estimates domain effects through finite differences to actual fine-tuning steps rather than through gradients, its applicability is not restricted to differentiable objectives. This same property also lets a practitioner designate as a target a capability for which *no training data is available*, since targets need not be represented among the fine-tuning datasets. We exploit both forms of flexibility in variants of the same 50 scenarios: promoting a held-out benchmark to target and scoring it by accuracy, which stresses the two at once since no training data optimizes it directly (Section 5); and expressing a constraint in accuracy rather than loss (Appendix O). In both, a scenario mixes objectives, with loss used for some domains and accuracy for others.

4.2 MODELS AND TRAINING

We evaluate DYNAMICS across three models, spanning 3B to 12B parameters. We fine-tune models in the following setups:

- **Qwen2.5-3B** (Qwen Team, 2024) with LoRA (Hu et al., 2022) ranks $r \in \{8, 32, 64\}$, as well as full-parameter fine-tuning, to test sensitivity to the number of trainable parameters.
- **Qwen3-8B** (Qwen Team, 2025) at $r = 32$.
- **Gemma3-12B** (Gemma Team, 2025) at $r = 32$.

Note that DYNAMICS is not tied to LoRA: all our optimization steps are equally applicable to full fine-tuning, and we verify this empirically on all 50 scenarios (Appendix I). We use LoRA as it provides a practical instantiation of fine-tuning, allowing for controlling adapter capacity. We train all models with Adam for 2,048 steps with batch size 8, sequence length 512, and evaluations every 64 steps on 200 batches. Training runs for a fixed budget without early stopping; the best feasible checkpoint is selected post-hoc (Section 4.6). The learning rate is selected per model from a six-candidate sweep: 10^{-4} for Qwen2.5-3B (all ranks), 5×10^{-5} for Qwen3-8B and Gemma3-12B, and 10^{-6} for full-parameter fine-tuning of Qwen2.5-3B. Further details on learning rate sweep, dataset sizes, token counts, and epoch behavior are provided in Appendix D.

4.3 DYNAMICS CONFIGURATIONS

Each of the model configurations is evaluated on all 50 scenarios with 3 random seeds. In the main results, all optimizer hyper-parameters are held fixed; we vary only the update schedule $\mathcal{U}$, comparing denser versus sparser re-estimation of mixture weights in the early training phase. Our hypothesis is that early updates can help the optimizer gather information quickly when the loss landscape changes fastest; later, as the model stabilizes, less frequent updates suffice. Sensitivity to overall tightness of $\mathcal{U}$ and to the probe length is analyzed separately in Appendix G.

All three update at step 0 and share the same doubling tail —updates at steps 64, 128, 256, 512, 1024— but differ in how aggressively they front-load early updates:

- *No warmup*: no additional updates (6 updates total).
- *Light warmup*: additional updates at steps 8, 16, 32 (9 updates total).
- *Dense warmup*: additional updates at steps 2, 4, 8, 16, 32 (11 updates total).

At each update step t, the number of probing steps c_t used to estimate the slope matrix is set to $\min(H_t, c_{\max})$, with $c_{\max} = 128$. In other words, early updates with short intervals use short probes (e.g. $c_t = 2$ at the first dense-warmup update), while later updates use a full budget of 128 steps.

4.4 BASELINES

Fixed-weight grid search. Our primary baseline holds mixture weights fixed for the entire run. A practitioner facing this problem must choose a weight allocation without knowing how each dataset will interact with the targets and constraints, so a competitive fixed mixture can require an expensive sweep over candidate target/additional-domain trade-offs, a well-known bottleneck in data mixture optimization (Xie et al., 2023; Ye et al., 2025; Kendall et al., 2018). We represent this class of strategies with two allocation schemes that cover the most natural choices. *Uniform* splits a target mass w equally across all target domains in the fine-tuning set, and the remaining $1 - w$ equally across all non-target domains in the fine-tuning set. *Proportional*, instead, distributes w across target datasets in proportion to their sample counts (giving more weight to larger datasets), while still splitting $1 - w$ uniformly across non-target datasets. We sweep $w \in \{0.0, 0.2, 0.5, 0.8, 1.0\}$ and run both schemes, yielding up to 9 baseline runs per scenario. Proportional runs are skipped when they coincide with uniform, e.g., at $w = 0$ where all weight goes to non-target domains regardless of scheme. We report results of this baseline throughout the paper.

Additional baselines. We further compare DYNAMICS against DoReMi (Xie et al., 2023), the probing-based methods Skill-It (Chen et al., 2023) and Aioli (Chen et al., 2024), a gradient-alignment baseline using instantaneous gradient dot products (Ren et al., 2018; Yu et al., 2020; Fan et al., 2024a), and an embedding-similarity variant that replaces probing with cosine similarities between domains. None of these methods was originally designed to incorporate knowledge preservation constraints; we extend them to the constrained setting where possible. Overall, these methods fall short for one of two reasons: some rely on signals that are not calibrated in loss units and so cannot anticipate where a constraint will land, while others optimize an aggregate objective that does not distinguish

target improvement from knowledge preservation, leaving them no way to trade one against the other. Details and results are in Appendix M.

4.5 COMPUTE COST

We express all costs as multiples of a single baseline run ($1\times$), so that DYNAMICS's overhead can be compared directly to the cost of running additional fixed-weight trials. DYNAMICS shares the training loop of the baselines but adds a per-update overhead: at each scheduled update step $t \in \mathcal{U}$, it probes each of the N fine-tuning datasets in isolation. This involves c_t probing steps per dataset, followed by a reduced evaluation (in our experiments, 50 batches per domain vs. 200 in the main evaluations) on the evaluation domains. The overhead thus scales with the number of candidate fine-tuning sources, not with the number of targets or constraints: an additional evaluation domain adds a few evaluation batches after each probe, but no further probing. Pre-probe losses need not be recomputed either: they are recycled from the main evaluation loop whenever an update aligns with a scheduled evaluation.

$$\beta_t = \frac{N \times (\overbrace{c_t}^{\text{probing}} + \underbrace{C_{\text{eval,red}}}_{\text{reduced eval}})}{\rho_t} \tag{5}$$

For an update interval of length H_t with a baseline compute cost of ρ_t, the relative per-update overhead ratio β_t is given by equation 5, where $C_{\text{eval,red}}$ is the cost of one reduced evaluation. Under the geometric schedule (Section 4.3), the adaptive rule $c_t = \min(H_t, c_{\max})$ ties probing to interval length: early short-interval updates use few probing steps, while later long intervals make the capped $c_{\max} = 128$ negligible relative to ρ_t.

We exclude the cost of solving equation 4: a low-dimensional simplex optimization on CPU, negligible next to GPU training and evaluation. Consequently, while a small grid search of 9 independent mixtures costs $9\times$ a single baseline run, DYNAMICS totals only $1.4\times$ to $1.7\times$. These are means over all 50 scenarios, which span 1–3 target and 3–10 constrained domains, hence up to 13 evaluation domains. The detailed cost derivation, a per-schedule breakdown, and projected costs for larger numbers of sources and evaluation domains are provided in Appendix E.

4.6 METRICS

Feasibility. Every evaluation domain is scored by a quantity for which lower is better: a loss, or, for benchmark-based domains, an error rate ($1 -$ accuracy); a single scenario may mix the two. A run is *feasible* if it produces at least one checkpoint where the score on *every* constrained domain remains at or below its reference L_i^{ref} and the sum of target scores has decreased relative to step 0. This means that, effectively, fine-tuning led to learning without forgetting, and that a run achieving feasibility by suppressing target learning would score close to zero under the metric defined below. We study this trade-off directly in Appendix N.

Constrained performance improvement. Our primary metric is the improvement on the target domains, at the best feasible checkpoint, defined as the one minimizing evaluation-set target scores while satisfying all constraints. For loss-based targets this is the relative perplexity reduction (geometric mean over multiple targets) reported on a held-out test set to prevent overfitting to the selection criterion. Targets may also be scored by accuracy, in which case we report the gain on the target benchmark in percentage points. Infeasible runs contribute no improvement (0%): if no checkpoint satisfies the constraints, the best a practitioner can do is keep the pre-fine-tuning model.

Baselines: best of k. To place baselines on the same cost axis, we ask: for a given budget of k fixed-weight runs (each costing $1\times$), how much improvement (perplexity reduction or accuracy gain) can a practitioner expect from the best of those k attempts? For each scenario, we estimate the expected best-of-k by sampling without replacement from the scenario's baseline pool, then report the mean across scenarios.

5 RESULTS

We report here the results, across all 50 scenarios, of DYNAMICS runs under the three geometric schedules described in Section 4.3 comparing against fixed-weight baselines, for 3 random seeds

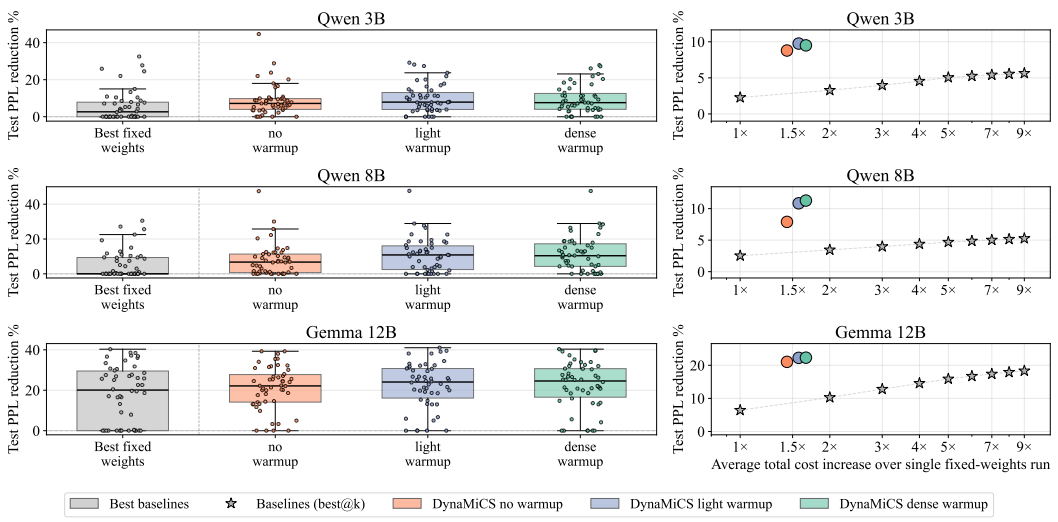


Figure 2: **Model comparison.** Test perplexity reduction (%) across 50 scenarios for different model families (Qwen2.5-3B, Qwen3-8B, Gemma3-12B), averaged over three seeds. Scenarios with no feasible checkpoint count as 0%. **Left (boxplots):** Per-scenario PPL reduction, one dot per scenario; for the baselines we report the oracle best across all fixed-weight configurations. **Right (scatter plots):** Mean PPL reduction across scenarios vs. compute cost (relative to a single fixed-weights run). Each DynaMiCS colored circle corresponds to one warmup schedule and shows the mean reduction of a single adaptive run at that schedule's cost. Baseline stars show $best@k$: the best mean reduction obtained by k fixed-weight configurations, at the corresponding budget. Higher is better.

(Figure 2). Recall that infeasible runs, those where no fine-tuning checkpoint satisfies all constraints, contribute 0% perplexity reduction, so the distributions reflect both the quality of feasible solutions and how often each method finds one. Feasibility rates are reported separately in Appendix F.

DynaMiCS consistently outperforms baselines. Across all model configurations, DynaMiCS with *any* schedule achieves substantially higher mean test perplexity reduction than fixed-weight baselines. On Qwen2.5-3B, the light and dense warmup schedules reach a mean reduction of 9.5–9.8% at $\sim$1.7$\times$ the cost of a single run; at a comparable budget of two fixed-weight attempts, baselines reach 3.3%. On Qwen3-8B, the gap widens: DynaMiCS achieves 11.3% with dense warmup, against 3.5% at two attempts. Even on Gemma3-12B, where baselines are considerably stronger, DynaMiCS adds a further 4 percentage points over the entire grid of fixed-weight runs (22.3% versus 18.3%). Appendix M compares DynaMiCS against five alternative mixture-selection methods on the same 50 scenarios (gradient alignment, embedding similarity, DoReMi, Skill-It and Aioli). None reaches a mean test perplexity reduction above 3.5%, with low feasibility rates; see Table 9.

Beyond perplexity: non-differentiable, data-free targets. Because DynaMiCS estimates domain effects by finite differences rather than gradients, it benefits from a flexibility that gradient-based mixture optimizers structurally lack: it can steer fine-tuning toward *any* target signal. We demonstrate this with an experiment that stresses *two* forms of this flexibility at once: (i) the target is scored by *accuracy*, a non-differentiable metric; and (ii) the target is *unseen*, i.e. never part of the fine-tuning mixture, so no training data directly optimizes it. We rerun all 50 scenarios with Qwen2.5-3B (LoRA rank 32), designating in each scenario one held-out benchmark as an accuracy target (measured as the length-normalized log-likelihood of the choice text). Even in this harder setup, a single DynaMiCS run raises held-out accuracy more than the best fixed-weight

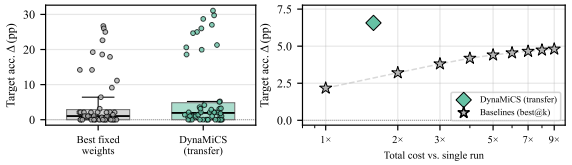


Figure 3: **Accuracy target on an unseen benchmark (Qwen2.5-3B, r=32).** Change in target-benchmark accuracy (pp) across the 50 scenarios. Format as in Fig. 2. Higher is better.

configuration (Figure 3). The advantage holds for each target benchmark (see Appendix O, which also covers accuracy used as a *constraint*).

Results generalize across ranks, architectures, scales, and fine-tuning regimes. Varying the LoRA rank on Qwen2.5-3B from 8 to 64 has minimal effect on mean PPL reduction (8.7–9.8% with warmup schedules). An extended sweep over ranks 2–128 confirms that rank 32 sits at the knee of the capacity curve: lower ranks underperform while higher ranks offer no added benefit. DYNAMICS also transfers to full-parameter fine-tuning (Appendix H and I).

Early updates matter. Warmup schedules consistently improve perplexity reduction over the no-warmup schedule, most clearly on Qwen3-8B (7.9% to 10.8% with light warmup). The difference between light and dense warmup is small in all configurations, suggesting that a moderate number of early updates suffices. On Qwen2.5-3B, denser warmup improves target quality but slightly reduces the feasibility rate (Appendix F; see Appendix G for sensitivity to the overall update frequency).

DYNAMICS degrades gracefully. As constraints grow, DYNAMICS remains largely stable while baselines degrade steadily (Figure 4); with 10 constraints, both methods struggle, although DYNAMICS still performs better for Gemma-3-12B and Qwen3-8B. Even when feasibility is not reached, DYNAMICS yields lower constraint violations than the best baseline, showing that constraint-aware weighting limits forgetting even when full satisfaction is out of reach (Appendix J and K).

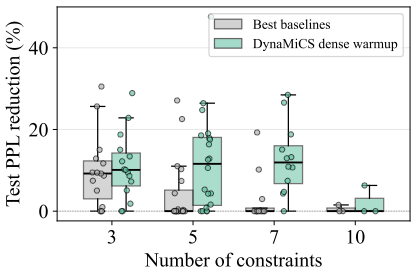


Figure 4: Test PPL reduction by number of constraints (Qwen3-8B, LoRA-32). DYNAMICS maintains gains as difficulty increases, while baselines degrade.

Ablating dynamicity. The mixtures DYNAMICS selects evolve differently across scenarios and models. A frequent pattern emphasizes specialized data early and shifts toward generic corpora later, where they act as regularizers (Appendix P). To isolate what dynamic weight allocation contributes, we ablate it: we solve the constrained problem once at step 0 and freeze the resulting mixture, against the full schedule with dynamic updates (Qwen2.5-3B, r=32, all 50 scenarios, single seed). Feasibility is identical in both setups, while mean test-perplexity reduction rises from $2.2 \pm 3.6\%$ to $10.1 \pm 9.7\%$. The constrained formulation therefore delivers feasibility, and repeated re-optimization converts it into target improvement: a mixture chosen at step 0 preserves the protected capabilities but goes stale, whereas re-solving at each update keeps the path feasible while continuing to reduce the target loss.

6 CONCLUSION

DYNAMICS introduces an efficient framework for multi-domain fine-tuning by casting data mixture selection as a constrained optimization problem. By leveraging short domain-specific probing runs to estimate a slope matrix of cross-domain effects, it dynamically derives mixture weights that balance target improvement with explicit knowledge preservation constraints, without auxiliary models or manual tuning. Because constraints enter through finite differences of evaluation metrics, any metric computable on a held-out batch can be used without a differentiable surrogate, a flexibility prior methods cannot offer by construction. Across a wide range of scenarios, model families, and scales, this formulation yields consistent gains over baselines, achieving stronger target-domain performance while maintaining constraint satisfaction at a fraction of the cost of grid search.

Limitations. The method's reliance on local linear approximations can limit performance when update intervals are too long or when feasible solutions become scarce in highly constrained settings. Additionally, the slope matrix assumes that cross-domain effects combine additively (a simplification shared with prior work). Moreover, the approach depends on the quality of evaluations, and may fail to preserve capabilities when these are small or unrepresentative. Finally, because mixture updates are chosen greedily from local estimates, early over-specialization may bias the subsequent training trajectory. Overcoming this locality typically requires back-tracking or evolutionary methods: an engineering challenge that we defer to future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of selected methods, including the drafting and refinement of software code, and to create and modify scientific figures. We also used generative AI tools to improve the clarity and readability of the manuscript, draft the initial structure of some sections, and help identify potentially relevant literature to complement works identified independently by the authors. All AI-assisted outputs were subsequently reviewed and validated by the authors before being incorporated into the work. In particular, AI-assisted code was inspected and tested for correctness, scientific figures were checked against the underlying data and results, suggested references were manually verified against the original sources, and manuscript edits were reviewed and approved by the authors. No AI-generated output was incorporated without author review. The authors take full responsibility for the final content of this work, including all text, claims, results, code, figures, and other artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

All experiments in this paper use publicly available models and datasets. The 50 fine-tuning scenarios, with the target and constrained domains of each, are defined in Appendix C, and the complete list of datasets is given in Appendix B. Models, training budget, optimizer and LoRA settings are reported in Section 4.2, with their hyper-parameter specification, including the penalty and tolerance grid over which the constrained problem is solved at each update, in Appendix D. The DYNAMICS configurations we report, namely the update schedules and probe lengths, are described in Section 4.3. Baselines are specified in Section 4.4, and each additional baseline is given its own formulation and hyper-parameter sweep in Appendix M. Our evaluation protocol, feasibility criterion and checkpoint-selection rule are defined in Section 4.6, and the compute cost is spelled out in Section 4.5 and Appendix E.

REFERENCES

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. doi: 10.1109/TAC.1974.1100705.

AutoIF. Autoif-instruct-61k. https://huggingface.co/datasets/Post-training-Data-Flywheel/AutoIF-instruct-61k, 2024.

Samuel J. Bell and Neil D. Lawrence. The effect of task ordering in continual learning, 2022. URL https://arxiv.org/abs/2205.13323.

Louis Bethune, David Grangier, Dan Busbridge, Eleonora Gualdoni, Marco Cuturi, and Pierre Ablin. Scaling laws for forgetting during finetuning with pretraining data injection, 2025. URL https://arxiv.org/abs/2502.06042.

Mayee Chen, Nicholas Roberts, Kush Bhatia, Jue WANG, Ce Zhang, Frederic Sala, and Christopher Ré. Skill-it! a data-driven skills framework for understanding and training language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 36000–36040. Curran Associates, Inc., 2023. doi: 10.52202/075280-1562. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70b8505ac79e3e131756f793cd80eb8d-Paper-Conference.pdf.

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing, 2024. URL https://arxiv.org/abs/2411.05735.

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.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected

by supervised fine-tuning data composition, 2023. URL https://arxiv.org/abs/2310.05492.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *ArXiv preprint*, abs/2507.16812, 2025a. URL https://arxiv.org/abs/2507.16812.

Simin Fan, David Grangier, and Pierre Ablin. Dynamic gradient alignment for online data mixing, 2024a. URL https://arxiv.org/abs/2410.02498.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. DOGE: domain reweighting with generalization estimation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024b. URL https://openreview.net/forum?id=7rfZ6bMZq4.

Simin Fan, Maria Ios Glarou, and Martin Jaggi. Grape: Optimize data mixture for group robust multi-target adaptive pretraining. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Glaive AI. Glaive function calling v2 dataset. https://huggingface.co/datasets/glaiveai/glaive-function-calling-v2, 2023.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, Wanjia Zhao, John Yang, Shreyas Pimpalgaonkar, Kartik Sharma, Charlie Cheng-Jie Ji, Yichuan Deng, Sarah Pratt, Vivek Ramanujan, Jon Saad-Falcon, Jeffrey Li, Achal Dave, Alon Albalak, Kushal Arora, Blake Wulfe, Chinmay Hegde, Greg Durrett, Sewoong Oh, Mohit Bansal, Saadia Gabriel, Aditya Grover, Kai-Wei Chang, Vaishaal Shankar, Aaron Gokaslan, Mike A. Merrill, Tatsunori Hashimoto, Yejin Choi, Jenia Jitsev, Reinhard Heckel, Maheswaran Sathiamoorthy, Alexandros G. Dimakis, and Ludwig Schmidt. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

Sonam Gupta, Yatin Nandwani, Asaf Yehudai, Dinesh Khandelwal, Dinesh Raghu, and Sachindra Joshi. Selective self-to-supervised fine-tuning for generalization in large language models, 2025. URL https://arxiv.org/abs/2502.08130.

Yifan Hao, Xingyuan Pan, Hanning Zhang, Chenlu Ye, Rui Pan, and Tong Zhang. Understanding overadaptation in supervised fine-tuning: The role of ensemble methods, 2025. URL https://arxiv.org/abs/2506.01901.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. Analyzing the forgetting problem in the pretrain-finetuning of dialogue response models, 2019. URL https://arxiv.org/abs/1910.07117.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Clifford M Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989. doi: 10.1093/biomet/76.2.297.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=DimPeeCxKO.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7482–7491, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Kendall_Multi-Task_Learning_Using_CVPR_2018_paper.html.

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, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Anastassia Kornilova and Vladimir Eidelman. BillSum: A corpus for automatic summarization of US legislation. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (eds.), *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 48–56, 2019. URL https://aclanthology.org/D19-5406.

Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. Understanding catastrophic forgetting in language models via implicit inference. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024. URL https://openreview.net/forum?id=VrHiF2hsrm.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F. Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M. Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Raghavi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alex Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/19e4ea30dded58259665db375885e412-Abstract-Datasets_and_Benchmarks_Track.html.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. (https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

Yuan Li, Zhengzhong Liu, and Eric P. Xing. Data mixing optimization for supervised fine-tuning of large language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, 2025. URL https://openreview.net/forum?id=19kqoNoc2N.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://huggingface.co/datasets/Open-Orca/OpenOrca, 2023.

Xize Liang, Lin Yang, Jie Wang, Yiyang Lu, Runyu Wu, Hanzhu Chen, and Jianye Hao. Boosting multi-domain fine-tuning of large language models through evolving interactions between samples. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37427–37441, 2025. URL https://proceedings.mlr.press/v267/liang25q.html.

Keer Lu, Keshi Zhao, Zhuoran Zhang, Zheng Liang, Bin Cui, Tengjiao Wang, and Wentao Zhang. Versatune: An efficient data composition framework for training multi-capability llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6634–6658, 2025.

Michael McCloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:109–165, 1989.

MedAlpaca. Medical meadow medical flashcards. https://huggingface.co/datasets/medalpaca/medical_meadow_medical_flashcards, 2023.

MelioAI. Melioai/safety-qa-sample. https://huggingface.co/datasets/MelioAI/safety-qa-sample, 2023.

João Monteiro, Pierre-André Noël, Étienne Marcotte, Sai Rajeswar Mudumba, Valentina Zantedeschi, David Vázquez, Nicolas Chapados, Chris Pal, and Perouz Taslakian. Repliqa: A question-answering dataset for benchmarking llms on unseen reference content. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/2b23626015b6311369e95a70735cbb72-Abstract-Datasets_and_Benchmarks_Track.html.

David Mueller, Mark Dredze, and Nicholas Andrews. Multi-task transfer matters during instruction-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14880–14891, 2024. URL https://aclanthology.org/2024.findings-acl.883/.

NVIDIA. nvidia/nemotron-sft-safety-v1. https://huggingface.co/datasets/nvidia/Nemotron-SFT-Safety-v1, 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel Bowman. BBQ: A hand-built bias benchmark for question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2086–2105, 2022. URL https://aclanthology.org/2022.findings-acl.165.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024. URL https://openreview.net/forum?id=qAml3FpfhG.

Qwen Team. Qwen2.5: A party of foundation models, 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=LhY8QdUGSuw.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4334–4343, 2018. URL http://proceedings.mlr.press/v80/ren18a.html.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey, 2024a. URL https://arxiv.org/abs/2404.16789.

Zhengyan Shi, Adam X. Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. Instruction tuning with loss over instructions. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/7ffb43adf37b3eeaba559098bc084cc6-Abstract-Conference.html.

Haebin Shin, Lei Ji, Xiao Liu, Zhiwei Yu, Qi Chen, and Yeyun Gong. Dynamixsft: Dynamic mixture optimization of instruction tuning collections. *ArXiv preprint*, abs/2508.12116, 2025. URL https://arxiv.org/abs/2508.12116.

Mustafa Shukor, Louis Béthune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *ArXiv preprint*, abs/2507.09404, 2025. URL https://arxiv.org/abs/2507.09404.

Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Minh Chien, Sebastian Ruder, Surya Guthikonda, Emad A. Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. Aya dataset: An open-access collection for multilingual instruction tuning, 2024.

Guiyao Tie, Zeli Zhao, Dingjie Song, Fuyang Wei, Rong Zhou, Yurou Dai, Wen Yin, Zhejian Yang, Jiangyue Yan, Yao Su, Zhenhan Dai, Yifeng Xie, Yihan Cao, Lichao Sun, Pan Zhou, Lifang He, Hechang Chen, Yu Zhang, Qingsong Wen, Tianming Liu, Neil Zhenqiang Gong, Jiliang Tang, Caiming Xiong, Heng Ji, Philip S. Yu, and Jianfeng Gao. A survey on post-training of large language models, 2025. URL https://arxiv.org/abs/2503.06072.

Jiaxing Wang, Deping Xiang, Jin Xu, Mingyang Yi, Guoqiang Gong, Zicheng Zhang, Haoran Li, Pengzhang Liu, Zhen Chen, Ke Zhang, Ju Fan, and Qixia Jiang. Tandem: Bi-level data mixture optimization with twin networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. Balancing training for multilingual neural machine translation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8526–8537, 2020. URL https://aclanthology.org/2020.acl-main.754.

Wikimedia. wikimedia/wikipedia. https://huggingface.co/datasets/wikimedia/wikipedia, 2024.

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, and Gholamreza Haffari. Mixture-of-skills: Learning to optimize data usage for fine-tuning large language models, 2024. URL https://arxiv.org/abs/2406.08811.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning, 2025. URL https://arxiv.org/abs/2505.24844.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *The Thirteenth International Conference on Learning Representations*, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

Wanru Zhao, Yihong Chen, Yuzhi Tang, Wentao Ma, Shengchao Hu, Shell Xu Hu, Alex Iacob, Abhinav Mehrotra, and Nicholas D Lane. Rethinking data curation in llm training: Online reweighting offers better generalization than offline methods. In *The Fourteenth International Conference on Learning Representations*, 2026.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement, 2024. URL https://arxiv.org/abs/2402.14658.

A Extended Related Work

A.1 Forgetting and overfitting in LLM fine-tuning

A fundamental challenge when fine-tuning a pre-trained language model on a specialized domain is *catastrophic forgetting*: the model may lose capabilities acquired during pre-training as it adapts to the new distribution (McCloskey & Cohen, 1989; Shi et al., 2024a; Kirkpatrick et al., 2017). Its severity depends on factors including source–target similarity, gradient compatibility, and task ordering (Kotha et al., 2024; Ramasesh et al., 2021; Bell & Lawrence, 2022).

This issue is especially relevant in SFT, where relatively small specialization datasets are often reused across epochs. Forgetting can then interact with *overfitting* (Shi et al., 2024b): the model may improve rapidly on the target domain while regressing on broader capabilities such as general knowledge, instruction following, or safety behavior. This motivates methods that improve target performance while controlling degradation on capabilities that should remain stable.

A.2 Static data mixture selection

A common way to mitigate forgetting and overfitting is to fine-tune on *data mixtures*, interleaving specialization data with auxiliary or general-domain data (Bethune et al., 2025; He et al., 2019; Ibrahim et al., 2024). The practical question is how to choose the mixture weights. Standard choices include uniform, dataset-size-proportional, and temperature-based sampling (Wang et al., 2020; Mueller et al., 2024). These methods are simple and inexpensive, but their weights are fixed before training and cannot react to evolving transfer and interference. More principled static methods formulate mixture selection as an optimization problem, for example using scaling laws to predict useful mixture weights (Li et al., 2025). Such approaches improve over hand-designed heuristics, but generally optimize a target or aggregate objective rather than enforcing explicit performance bounds on capabilities to be preserved.

A.3 Dynamic data mixture selection

Domain relationships do not remain fixed throughout training: data that is useful early in fine-tuning may become less useful later, while auxiliary data may become increasingly important as the model specializes. Indeed, interactions between training samples can evolve over training (Liang et al., 2025), motivating dynamic mixture adaptation.

Recent methods use several mechanisms. Dong et al. (2023) propose a dual-stage SFT recipe that shifts from specialized toward more general data. Wu et al. (2024) adapt sampling through reinforcement learning, while Shin et al. (2025) formulate selection as a non-stationary multi-armed bandit. Lu et al. (2025) update domain weights using loss-derived notions of learnability and forgetting. At finer granularity, EVIC (Liang et al., 2025) estimates evolving gradient-based *influence* between samples and uses it to construct an adaptive curriculum. Our method instead operates over mixture components or domains, reducing the cost of the control problem while still adapting over training.

A complementary family explicitly estimates cross-domain effects and uses them to update mixture weights (Chen et al., 2023; 2024; Wang et al., 2025). Because these are the closest methodological precursors of DynaMiCS, we discuss them separately in Appendix A.4. The key distinction is that DynaMiCS adapts the mixture under explicit preservation constraints. Existing dynamic methods use staged schedules, rewards, influence scores, loss-derived signals, or estimated cross-domain effects; some explicitly account for forgetting, but none lets a practitioner specify both a capability to preserve and an allowable tolerance for its degradation.

A.4 Explicit cross-domain influence estimation

The closest precursors of DynaMiCS explicitly estimate how training on one data group affects performance on others, then use these estimates to re-weight the mixture. Their mechanisms differ, but they share the general pipeline of influence estimation followed by mixture adaptation. We describe the three closest methods below and the baseline instantiations of the first two in Appendix M.

Skill-It. Skill-It (Chen et al., 2023) is a structural precursor of our slope matrix. It learns a skill-influence graph from auxiliary runs that train on individual skills and measure changes in validation loss on evaluation skills, then uses this graph together with current losses to update sampling proportions online. Two differences are important. First, the graph represents beneficial prerequisite relationships rather than transfer and interference; so harmful interactions are not encoded as negative edges. Second, it is estimated once and then held fixed. Skill-It can target an individual skill, but does not impose preservation bounds on other evaluation skills.

Aioli. Aioli (Chen et al., 2024) casts dynamic data mixing as optimization under a parametric *mixing law* relating mixture weights to per-domain losses. In its linear form, a matrix of cross-domain coefficients captures how training proportions affect validation losses. Our predictor has the same linear structure, but the coefficients are estimated differently. Aioli interleaves near-one-hot mixtures during training and infers cross-domain effects from the resulting loss changes along a single trajectory. DYNAMICS instead branches controlled domain-specific probes from a common checkpoint, isolating each dataset's local effect from the same model state. Aioli optimizes an unconstrained validation-loss objective (either averaged across evaluation groups or, in its OOD extension, on a single target group) whereas DYNAMICS imposes explicit performance constraints on capabilities to be preserved.

TANDEM. TANDEM (Wang et al., 2025) formulates mixture selection as a bi-level problem and approximates it with a penalized single-level objective. It uses twin networks, i.e., a proxy and a reference model, to estimate domain utility before updating the mixture, and evaluates the framework in both pre-training and SFT. Its penalty enforces approximate optimality of the inner training problem, rather than preservation of a specific capability, while the outer objective remains aggregate validation loss. Moreover, the learned mixture is used in a separate final training run, introducing auxiliary optimization overhead.

Positioning of DYNAMICS. DYNAMICS likewise estimates cross-domain effects, but does so with controlled probes from the current checkpoint and represents both beneficial transfer and degradation. Because the effects are measured by finite differences in the evaluation metric itself, constraints can be expressed in loss, perplexity, or non-differentiable metrics such as benchmark accuracy. These estimates are then used in an explicitly constrained optimization problem that trades target improvement against practitioner-specified preservation bounds.

A.5 PRE-TRAINING MIXTURE OPTIMIZATION

Data mixture optimization has also been widely studied in *pre-training* (Shukor et al., 2025). The setting differs from SFT: pre-training typically involves long runs over massive datasets, whereas SFT uses smaller datasets that may be repeated across epochs and starts from a model whose existing capabilities may need to be preserved.

DoReMi (Xie et al., 2023) learns pre-training domain proportions using a reference model and a smaller proxy model. The proxy weights are updated by exponentiated gradient on excess loss under a Group-DRO objective, and their time-average defines a static mixture for the final model. DoGE (Fan et al., 2024b) similarly learns weights with a proxy model, using a bi-level generalization objective based on gradient alignment. TANDEM (Wang et al., 2025), discussed above, extends this line with twin networks and also evaluates SFT.

Other approaches model the relationship between composition and performance. Data Mixing Laws (Ye et al., 2025) fit functions linking mixture proportions to per-domain losses and combine them with scaling laws to predict useful mixtures; the same framework also studies mixtures intended to mitigate catastrophic forgetting in continual training. DataComp-LM (Li et al., 2024) studies benchmark-driven data filtering and mixture construction, while ADAPT (Zhao et al., 2026) performs online sample-level reweighting in both pre-training and instruction tuning.

Gradient-based approaches include Dynamic Gradient Alignment (DGA) (Fan et al., 2024a), which upweights domains whose gradients align with a target set. GRAPE (Fan et al., 2025b) instead performs multi-target reweighting through a minimax objective, while Chameleon (Xie et al., 2025) derives domain weights from a learned embedding-space affinity matrix.

These methods provide increasingly flexible mechanisms for choosing data mixtures, but generally optimize target, aggregate, or predictive criteria rather than explicit practitioner-specified preservation bounds. DYNAMICS instead focuses on SFT and periodically estimates local domain effects from the current checkpoint to update mixture weights under explicit constraints on capabilities to be preserved.

B List of Datasets

Our evaluation suite draws on datasets spanning several categories: instruction-tuning corpora used as target or non-target fine-tuning datasets (and, at times, as constraints as well); open-domain text corpora used only as non-target fine-tuning datasets; and lightweight benchmarks used exclusively as constraint domains. Table 1 summarizes each dataset, its role in our experiments, and the abbreviation used throughout the paper.

Abbrev.	Dataset	Description	Source
Instruction-tuning datasets used for fine-tuning or, in some scenarios, as evaluation-only constrained domains			
MM	MetaMathQA	Math QA bootstrapped from formal problems	Yu et al. (2024)
NM	NuminaMath-CoT	Math reasoning dataset with chain-of-thought	LI et al. (2024)
GS	GSM8K	Human-written math word problems	Cobbe et al. (2021)
CF	CodeFeedback	Filtered code instruction–response pairs	Zheng et al. (2024)
AI	AutoIF	Instruction-following data generated via self-instruct pipelines	AutoIF (2024)
OT	OpenThoughts	Mixed-domain reasoning in math, science, code, and logic	Guha et al. (2025)
OO	OpenOrca	Instruction-tuning data distilled from GPT-4 outputs	Lian et al. (2023)
Ay	AyaDataset	Multilingual instruction-following across diverse tasks	Singh et al. (2024)
MF	MedicalFlashcards	Medical QA in flashcard format	MedAlpaca (2023)
GF	GlaiveFunctionCalling	Tool-use conversations with structured function calling	Glaive AI (2023)
MS	MegaScience	Scientific QA spanning multiple disciplines	Fan et al. (2025a)
BS	BillSum	Summarization of U.S. legislative bills	Kornilova & Eidelman (2019)
RQ	RepliQA	QA conditioned on a reference document	Monteiro et al. (2024)
Open-domain text corpora used as non-target fine-tuning domains only, with unlimited samples			
WP	Wikipedia	Encyclopedic corpus for general knowledge	Wikimedia (2024)
FW	FineWeb	Large-scale filtered web corpus	Penedo et al. (2024)
Evaluation benchmarks used as constrained domains only, never included in the training mixture			
tM	tinyMMLU	Multi-domain knowledge benchmark	Polo et al. (2024)
tT	tinyTruthfulQA	Evaluates resistance to false or misleading outputs	Polo et al. (2024)
tH	tinyHellaswag	Commonsense reasoning via sentence completion	Polo et al. (2024)
tA	tinyAI2_ARC	Science reasoning benchmark with exam-style MCQs	Polo et al. (2024)
NS	NemotronSafety	Evaluates safe refusal and handling of harmful queries	NVIDIA (2026)
SQ	SafetyQA	QA benchmark targeting safety and policy compliance	MelioAI (2023)
BQ	BBQ	Bias evaluation in QA across demographic attributes	Parrish et al. (2022)

Table 1: Datasets used in our experiments. In these scenarios, fine-tuning domains are divided into target domains, whose loss is explicitly minimized, and non-target domains, which provide additional data for mixture selection. Instruction-tuning datasets may appear as target or non-target fine-tuning domains, and some are held out as evaluation-only constraint domains depending on the scenario. Open-domain corpora (WP, FW) are used exclusively as non-target fine-tuning domains with unlimited samples. Evaluation benchmarks appear only as constraint domains and are never included in the training mixture. Abbreviations are used in Table 2.

C 50 EVALUATION SCENARIOS

We evaluate across 50 scenarios of increasing difficulty. Each scenario specifies:

- **Fine-tuning datasets**: available training data. These are divided into:
  - **Targets**: datasets whose loss we aim to minimize. When applicable, the fixed sample budget is shown in parentheses.
  - **Non-targets**: auxiliary fine-tuning datasets included in the training mixture, but not treated as optimization targets.
- **Constraints**: evaluation-only domains whose loss must not exceed the pre-training baseline. They are not included in the fine-tuning data.

The number of targets ranges from 1 to 3, while the number of constraints ranges from 3 to 10, yielding scenarios of increasing difficulty.

#	n_T	n_C	Fine-tuning datasets		Constraints (eval only)
			Targets	Non-targets	
1	1	3	MM (2k)	WP, RQ (5k), OO (10k)	tA, tH, MS
2	1	3	CF (500)	FW, OT (2k), Ay (5k)	tT, tH, NS
3	1	3	OT (1k)	WP, Ay (300), GS (5k)	tT, tH, BQ
4	1	3	RQ (10k)	FW, MM (300)	tA, tH, tM
5	1	3	MM (800)	WP, Ay (500), CF (300), OO (30k)	tA, SQ, NS
6	1	3	NM (2k)	WP, FW, BS (15k)	tT, tA, BQ
7	1	5	OO (1.5k)	FW, NM (500), Ay (300), BS (10k)	tT, tA, NS, BQ, MS
8	1	5	AI (1k)	WP, MS (500), OT (8k)	tT, tH, tA, SQ, BQ
9	1	5	GF (300)	FW, MF (500), NM (800), OO (5k)	tT, tA, tM, SQ, BQ
10	1	5	MF (500)	WP, NM (300), Ay (1k), BS (10k)	tT, tH, tA, SQ, NS
11	2	3	OO (3k), AI (1.5k)	FW, WP, RQ (800)	tA, tM, SQ
12	2	3	GF (500), CF (500)	WP, RQ (10k), OT (5k)	tA, tT, NS
13	2	3	CF (500), MM (500)	FW, OT (15k), Ay (5k)	tT, SQ, NS
14	2	3	MF (500), OO (1k)	FW, Ay (2k), GS (5k)	tM, BQ, SQ
15	2	3	NM (3k), GS (500)	WP, RQ (2k), OO (10k)	tH, tA, BQ
16	2	5	GS (500), OO (800)	WP, FW, BS (10k), RQ (8k)	4×tiny, MS
17	2	5	MM (500), MF (300)	FW, OT (2k), Ay (800)	4×tiny, SQ
18	2	5	MM (800), OO (1.5k)	WP, Ay (300), RQ (2k), BS (5k)	tT, tH, tA, BQ, NS
19	2	5	RQ (8k), MM (800)	WP, FW, NM (500)	4×tiny, NS
20	2	5	CF (300), AI (500)	BS (3k), OT (2k), GS (5k)	4×tiny, NS
21	2	5	NM (1k), GF (300)	WP, MF (500), Ay (800), OO (10k)	4×tiny, SQ
22	2	5	MM (500), OO (1k)	WP, FW	4×tiny, NS
23	2	7	NM (800), MF (300)	WP, FW, RQ (3k), Ay (500), OT (5k)	4×tiny, BQ, NS, SQ
24	2	7	Ay (500), GF (500)	WP, FW, OT (2k), MS (800)	4×tiny, NS, SQ, BQ
25	2	7	MM (500), OO (1k)	WP, FW, RQ (10k), BS (8k), Ay (5k)	4×tiny, MS, NS, BQ
26	2	7	CF (200), AI (800)	WP, FW, OT (2k), Ay (500)	4×tiny, SQ, BQ, MS
27	2	7	NM (1k), OO (1.5k)	WP, FW, RQ (3k), MF (300), GS (10k)	4×tiny, NS, SQ, BQ
28	2	7	NM (1k), GF (300)	WP, FW	4×tiny, SQ, BQ, MS
29	3	3	NM (5k), GS (500), MM (2k)	WP, FW	tT, NS, BQ
30	3	3	OO (5k), Ay (500), MF (500)	FW, BS (10k)	tM, tA, BQ
31	3	3	MM (500), OO (3k), CF (500)	WP, RQ (10k)	tA, tH, SQ
32	3	3	AI (3k), GF (500), NM (500)	FW, OT (8k), Ay (5k)	tT, NS, MS
33	3	5	MM (800), OO (1.5k), NM (500)	WP, FW	4×tiny, BQ
34	3	5	MM (800), OO (1.5k), GF (300)	WP, BS (20k), Ay (50k)	4×tiny, MS
35	3	5	AI (500), CF (300), NM (1k)	FW, OT (2k)	4×tiny, SQ
36	3	5	OO (1k), MF (500), GS (500)	WP, Ay (3k), RQ (5k)	4×tiny, BQ
37	3	5	MM (500), OO (1k), CF (300)	FW, RQ (10k), Ay (5k), BS (10k)	4×tiny, NS
38	3	5	NM (800), MF (500), Ay (500)	WP, OT (1.5k), GS (5k)	tT, tA, BQ, NS, SQ
39	3	5	GF (300), OO (1.5k), AI (500)	FW, BS (20k), RQ (10k)	4×tiny, MS
40	3	5	OO (1k), MF (500), GS (500)	WP, FW	4×tiny, NS
41	3	6	NM (800), AI (500), CF (300)	WP, FW	4×tiny, SQ, BQ
42	3	7	AI (500), MF (300), GF (500)	WP, FW	4×tiny, NS, SQ, MS
43	3	7	MM (800), OO (1.5k), CF (300)	WP, FW, RQ (2k), Ay (5k)	4×tiny, MS, SQ, BQ
44	3	7	OO (1k), MF (300), AI (500)	WP, FW, RQ (3k), BS (10k)	4×tiny, BQ, NS, SQ
45	3	7	NM (1k), Ay (300), GF (500)	WP, FW, OT (2k), RQ (1.5k)	4×tiny, MS, NS, BQ
46	3	7	MM (500), OO (1.5k), CF (200)	WP, FW, OT (10k), BS (20k), Ay (5k)	4×tiny, NS, SQ, BQ
47	3	7	AI (800), MF (300), GS (500)	WP, FW, Ay (500), RQ (3k), OO (10k)	4×tiny, BQ, NS, SQ
48	3	10	MM (500), OO (1k), CF (300)	WP, FW, OT (10k), Ay (800), GF (5k)	4×tiny, MS, BS, RQ, NS, SQ, BQ
49	3	10	Ay (500), GF (300), NM (1k)	WP, FW, OT (2k), RQ (3k)	4×tiny, MS, OO, BS, NS, SQ, BQ
50	3	10	OO (1.5k), AI (800), CF (200)	WP, FW, MF (500), Ay (300), GS (5k)	4×tiny, MS, BS, RQ, NS, SQ, BQ

Table 2: Our 50 evaluation scenarios. Sample budgets in parentheses. "4×tiny" = tM, tT, tH, tA. WP and FW are regularizers with unlimited samples.

D Details on Hyper-parameters

Learning rate and warmup schedule. We ran an exhaustive ablation over the learning rate and warmup schedule for fine-tuning on Wikipedia (Wikimedia, 2024). For each presented model and LoRA rank we swept six learning rates 1e-5, 5e-5, 1e-4, 2e-4, 5e-4, 1e-3, as shown in Figure 5, crossed with three warmup configurations (0 steps, 1%, and 2% of total training steps). We found that warmup had no appreciable effect on the final validation loss in any setting, while the learning rates had a consistent optimum around 1e-4 for Qwen2.5-3B with ranks 8, 32, and 64, and 5e-5 for Qwen3-8B and Gemma3-12B, both with rank 32. Based on this, all results in the main text use no warmup, with the per-model best learning rate selected from this sweep.

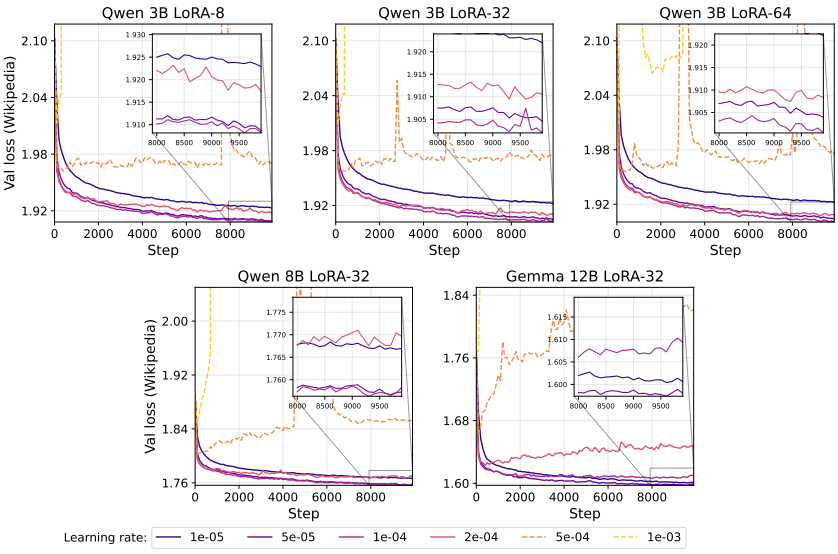


Figure 5: **Learning rate ablation.** Validation loss on Wikipedia across learning rates in Qwen2.5-3B at LoRA $r \in \{8, 32, 64\}$ (top) and Qwen3-8B and Gemma-3-12B at $r = 32$ (bottom). Diverged runs are dashed and allowed to leave the cropped y-axis; insets zoom on the tail.

Training budget. All runs train for 2,048 steps with batch size 8 and sequence length 512, corresponding to approximately $8M$ training tokens per run. We do not use early stopping: training proceeds for the full budget, and the best feasible checkpoint is selected post-hoc based on constraint satisfaction and target loss (Section 4.6).

Dataset sizes and epochs. Fine-tuning datasets vary in size from 300 to effectively unlimited examples (i.e., that will not be exhausted during the training phase), see Appendix C. With mixture sampling, smaller datasets may be repeated multiple times (i.e., trained for more than one epoch), while larger datasets are subsampled.

Hyper-parameters of DynaMiCS optimization. As mentioned in Section 3, at each mixture update, we solve the proxy objective equation 4 over a two-dimensional grid of hyper-parameters (λ, ε). The penalty strength λ is log-spaced over $[1, 5000]$ (15 values) and the constraint tolerance ε is linearly spaced over $[0, 0.1]$ (3 values), yielding 45 candidate solutions per update. Among candidates that satisfy all constraints, we select the one with the lowest target objective. If no candidate is feasible, we select the one with the smallest maximum constraint violation. This grid search is computationally cheap: all candidates are optimized over the same predicted loss surface (the fitted slope matrix), so the grid introduces no additional model evaluations beyond the single slope computation step.

Compute resources. We trained over 8,500 models. Each run was performed on a single GPU node, using either H100 or B200 GPUs. For the bigger model (Gemma3-12B), wall-clock time for one DynaMiCS run with dense warmup schedule, for s50, i.e., one of the largest sets of domains (Section 4), took approximately 10.5 hours. One fixed-weight baseline run for the same scenario has a wall-clock time of 7 hours.

E COMPUTE COSTS ANALYSIS

Both DYNAMICS and the fixed-weight baselines share the same per-step training loop: a forward–backward pass on the current mixture, plus periodic evaluations (every 64 steps, 200 batches per dataset). DYNAMICS additionally runs *slope computation* at each weight update: for every fine-tuning dataset, it trains briefly on that dataset alone (c_t steps) and evaluates the result on all evaluation domains using a reduced budget of 50 batches per dataset.

E.1 LINEAR WEIGHT UPDATE

The linear predictor requires two loss measurements per probe: one before (anchor) and one after c_t training steps. When a weight update aligns with a main evaluation step, the baseline losses are reused at no extra cost (*evaluation recycling*); otherwise, one additional reduced evaluation is needed. The per-update overhead relative to one baseline training interval ρ_t is:

$$\beta_t = \frac{N \times (c_t + C_{\text{eval,red}}) + \mathbf{1}_{\text{no recycle}} \cdot C_{\text{eval,red}}}{\rho_t}, \tag{6}$$

where N is the number of fine-tuning datasets, $C_{\text{eval,red}} = M \times 50 \times C_{\text{fwd}}$ is the cost of one reduced evaluation over M evaluation domains, $\rho_t = H_t \times C_{\text{bp}} + (H_t/64) \times M \times 200 \times C_{\text{fwd}}$, $c_t = \min(H_t,\ c_{\max})$, C_{fwd} the per-batch forward-pass cost, and C_{bp} the cost of one training step on a batch. The total cost of a DYNAMICS run is $\sum_{t \in \mathcal{U}} \rho_t(1 + \beta_t)$. The cost of solving Equation 4 is negligible (a low-dimensional simplex problem on CPU) and excluded from the compute calculations.

Cost of geometric schedules. Table 3 shows the cost of each schedule discussed in Section 5 averaged across all 50 scenarios (which vary in N and M). The overhead is low because the geometric schedules concentrate most training time in long later intervals where $\beta_t \ll 1$, and all updates in the geometric tail align with the evaluation frequency, enabling recycling. Early warmup updates use short probes $c_t = \min(H_t, 128)$, so their absolute cost is small despite higher β_t.

Schedule	Update steps	Updates	Mean cost
No warmup	0, 64, 128, 256, 512, 1024	6	$\sim$1.4$\times$
Light warmup	0, 8, 16, 32, 64, …, 1024	9	$\sim$1.5$\times$
Dense warmup	0, 2, 4, …, 32, 64, …, 1024	11	$\sim$1.7$\times$

Table 3: Compute cost of the schedule variants discussed in Section 4.3 and their mean compute costs ($c_{\max} = 128$, averaged across 50 scenarios).

Cost of fixed update horizons. Results presented in Appendix G use a uniform update schedule with fixed interval $H_t \equiv H$ and $c_t \equiv H$ for all $t \in \mathcal{U}$, so that probing spans the full interval at every update. Higher values of H yield fewer total updates but higher per-update overhead. Table 4 reports the mean cost for the three configurations discussed in the paper.

Interval ($H = c_t$)	Updates	Mean cost
128	16	$\sim$2.3$\times$
256	8	$\sim$2.0$\times$
512	4	$\sim$1.8$\times$

Table 4: Compute cost of schedules with fixed update horizons from Appendix G (averaged across 50 scenarios).

Cost of non-linear weight update. The curve-fitting predictor (Appendix L.2) evaluates at $n_{\text{evals}} = 5$ intermediate checkpoints during each probe, rather than a single endpoint measurement. This increases the evaluation cost per update by a factor of 5 relative to the linear predictor, while the probing training cost (c_t steps) remains unchanged. Table 5 reports the cost for the same fixed-interval configurations with $H_t \equiv H$ and $c_t \equiv H$ for all $t \in \mathcal{U}$ presented in Table 4, when the curve-fitting predictor is used.

Interval ($H = c_t$)	Updates	Mean cost
128	16	$\sim$4.7$\times$
256	8	$\sim$3.2$\times$
512	4	$\sim$2.4$\times$

Table 5: Compute cost of the non-linear (curve-fitting) predictor with $n_{\text{evals}} = 5$ and fixed update horizons from Appendix L.2 (averaged across 50 scenarios).

Projected cost at larger problem sizes. The scenarios we run span $N \in [3, 6]$ fine-tuning sources and up to 13 evaluation domains. To indicate how the overhead behaves beyond that range, Table 6 evaluates the cost model of Equation 5 at larger N and M under the dense warmup schedule. Cost grows with the number of candidate sources, since each adds a probe, and is essentially flat across the columns, confirming that the overhead is driven by the number of sources rather than by the number of evaluation domains. Even the largest configuration remains inexpensive in absolute terms: a 10-target, 20-constraint problem with 12 candidate sources costs $2.15\times$, roughly the price of two fixed-weight runs for a problem with 30 competing objectives, and a budget of two fixed-weight runs yields very low feasibility rates (Figure 2).

Fine-tuning sources N	3T / 10C	5T / 15C	10T / 20C
5	$1.54\times$	$1.52\times$	$1.50\times$
8	$1.85\times$	$1.81\times$	$1.78\times$
12	$2.25\times$	$2.19\times$	$2.15\times$

Table 6: Projected compute cost of DYNAMICS relative to a single baseline run, obtained by evaluating Equation 5 at problem sizes larger than those we run (dense warmup schedule, $c_{\max} = 128$). Columns give the number of target (T) and constrained (C) evaluation domains. Within a row, cost decreases slightly as evaluation domains are added because the $1\times$ reference grows faster than the overhead does: a baseline run evaluates every evaluation domain periodically at 200 batches, while the overhead adds only a single 50-batch reduced evaluation per probe, so the ratio of the two becomes smaller.

F FEASIBILITY RATES

Figure 6 reports feasibility rates, i.e., the fraction of scenarios where a method finds at least one checkpoint *after step 0* that satisfies all constraints and improves target perplexity over the base model. Crucially, for this analysis, we exclude the pre-training checkpoint (step 0).

All DYNAMICS schedules substantially outperform fixed-weight baselines. In Qwen2.5-3B, the no-warmup schedule achieves the highest feasibility despite lower mean PPL reduction than warmup variants (Section 5), but not in the two larger models. Denser early updates may lead the optimizer to find higher-quality solutions when they work, but probing the loss landscape during the initial training phase may occasionally commit to weights that later violate a constraint. The no-warmup schedule avoids this early instability, offering the best feasibility at the lowest compute overhead, which may be a natural choice when reliability matters more than peak gain.

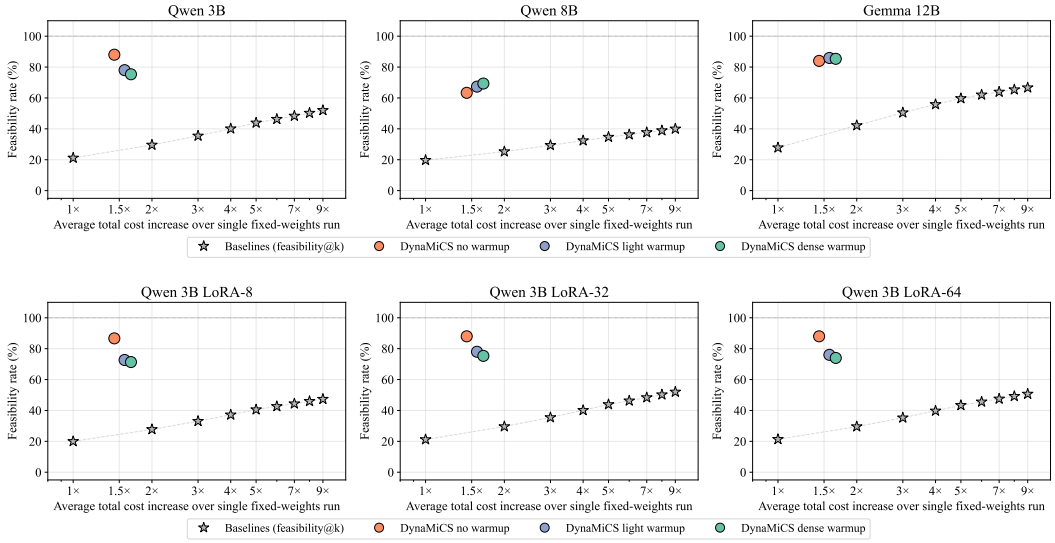


Figure 6: **Feasibility rate vs. compute cost** across 50 scenarios. A run is counted as feasible only if a checkpoint *after step 0* satisfies all constraints while improving target perplexity (the base model checkpoint is excluded, so feasibility requires that fine-tuning actually learned something.) **Top:** Model comparison at LoRA rank 32 (Qwen2.5-3B, Qwen3-8B, Gemma3-12B). **Bottom:** LoRA rank comparison on Qwen2.5-3B (ranks 8, 32, 64). Baseline stars show the "$feasible@k$" rate: the probability that at least one of k independent fixed-weight runs is feasible, averaged across scenarios. Colored circles show the feasibility of a single DYNAMICS run at its schedule-dependent cost. Mean $\pm$ std over three seeds. Higher values are better.

G SENSITIVITY TO UPDATE FREQUENCY AND PROBE LENGTH

The main results use geometric schedules where the update interval doubles over training. To assess DYNAMICS's sensitivity to the frequency of weight updates, we evaluate the method with *fixed* update intervals, where weights are re-estimated every H_t steps throughout training. In this setting, both the interval and the probing horizon are constant and equal: $H_t \equiv H$ and $c_t \equiv H$ for all $t \in \mathcal{U}$. We sweep $H \in \{128, 256, 512\}$, corresponding to 16, 8, and 4 weight updates over the 2,048-step horizon. A larger H means fewer updates.

Figures 7 and 8 show the results. More frequent updates ($H = 128$) yield the highest mean PPL reduction across all model configurations, consistent with the schedule results in Section 5: the optimizer benefits from responding to the evolving loss landscape more often. Less frequent updates ($H = 512$) substantially reduce the advantage and fall below baselines.

Compared to the geometric schedules (Figure 2), fixed intervals are less cost-efficient: the geometric dense warmup achieves comparable or better PPL reduction at $\sim$1.7$\times$ cost by concentrating updates in the early training phase and spacing them out later.

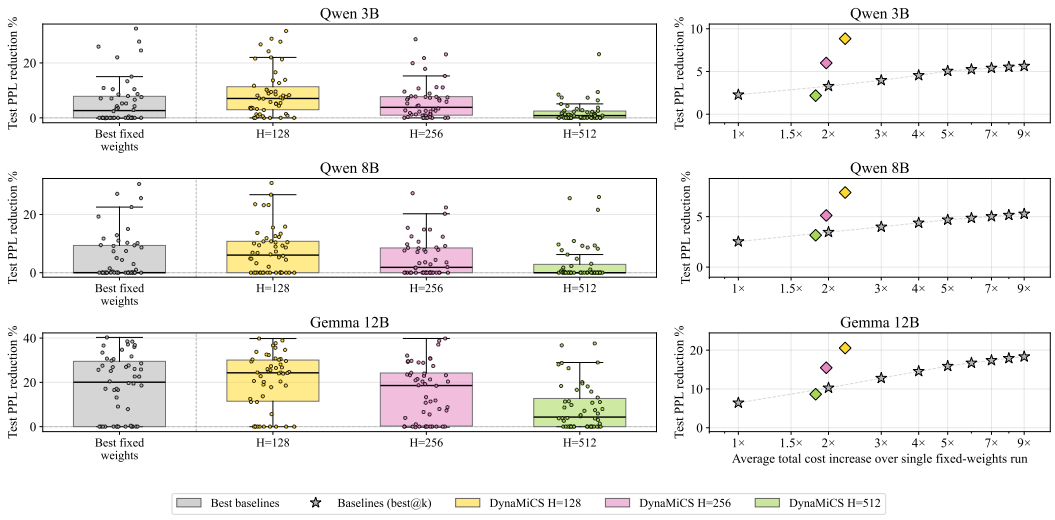


Figure 7: **Fixed-interval update frequency sweep: model comparison at LoRA rank 32.** Boxplots show per-scenario PPL reduction (one dot per scenario); for baselines, we report the oracle best across all fixed-weight configurations per scenario. Scatter plots show mean PPL reduction vs. total compute cost: baseline stars indicate the $best@k$ perplexity reduction, i.e., the best result among k fixed-weight configurations, at the corresponding budget level, while DYNAMICS diamonds show the reduction of a single adaptive run at each update interval's cost. Each DYNAMICS configuration uses a constant reweight interval H throughout training.

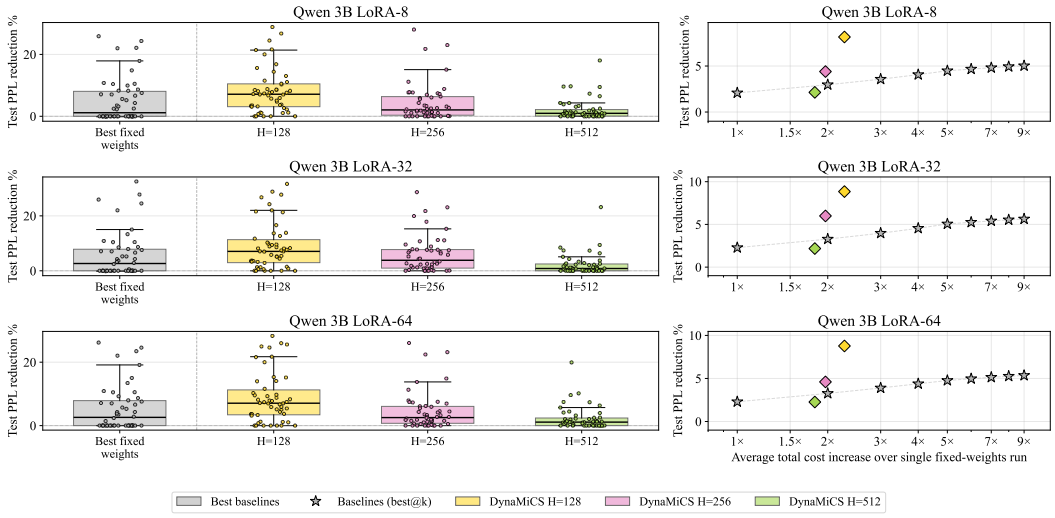


Figure 8: **Fixed-interval update frequency sweep: LoRA rank comparison on Qwen2.5-3B** (ranks 8, 32, 64). Same layout as Figure 7.

Cost–quality trade-off along the probe axis. The sweep above ties the probing budget to the update interval ($c_t \equiv H$), so it moves both at once. To separate them, we fix $H = 128$ and vary the probe length $c \in \{32, 64, 128\}$, that is $\alpha = c/H \in \{0.25, 0.5, 1\}$, on Qwen2.5-3B and Qwen3-8B (Table 7). Very short probes are too noisy to be useful: at $c = 32$ both the perplexity reduction and the number of feasible scenarios drop on both models. Beyond $c = 64$, perplexity reduction no longer improves, while feasibility continues to rise on Qwen2.5-3B (41 to 44 of 50) and is unchanged on Qwen3-8B. Longer probes therefore trade additional compute for feasibility rather than for perplexity, and moderate probes already capture most of the useful transfer and interference signal. This ablation keeps the interval fixed throughout training rather than using a geometric schedule, so its costs are not directly comparable to the $1.4\times$–$1.7\times$ of the geometric schedules in Section 5; the $c = 128$ row coincides with the $H = 128$ configuration of the sweep above.

Probe steps c	$\alpha = c/H$	Cost	Qwen2.5-3B	Qwen3-8B
32	0.25	$1.78\times$	7.9 ± 10.4 (30/50)	6.7 ± 10.6 (21/50)
64	0.5	$1.94\times$	9.4 ± 9.8 (41/50)	7.7 ± 9.1 (32/50)
128	1.0	$2.27\times$	8.8 ± 8.4 (44/50)	7.4 ± 7.9 (32/50)

Table 7: Probe-length sweep at a fixed update interval $H = 128$ (16 updates over 2,048 steps). Test perplexity reduction (mean $\pm$ std across the 50 scenarios) and the number of feasible scenarios, at LoRA rank 32. Costs are relative to a single fixed-weight run and are higher than those of the geometric schedules because the interval is held constant throughout training.

H LoRA Rank Sensitivity

Figure 9 shows the effect of varying LoRA rank (8, 32, 64) on Qwen2.5-3B across all three warmup schedules. Mean PPL reduction remains stable at 8.7–9.8% across ranks, and the advantage over fixed-weight baselines is consistent.

To verify that our default rank (32) provides appropriate adapter capacity, we additionally run a single schedule (dense warmup) across a wider range of ranks (2, 8, 32, 64, 128). Ranks 32, 64, and 128 perform comparably in terms of feasibility and PPL reduction, indicating that additional parameters beyond rank 32 do not translate into better fine-tuning quality in our setting (Figure 10). Zooming into feasibility (Figure 11), ranks 2 and 8 achieve lower rates than rank 32, consistent with insufficient capacity, while ranks 32, 64, and 128 perform comparably.

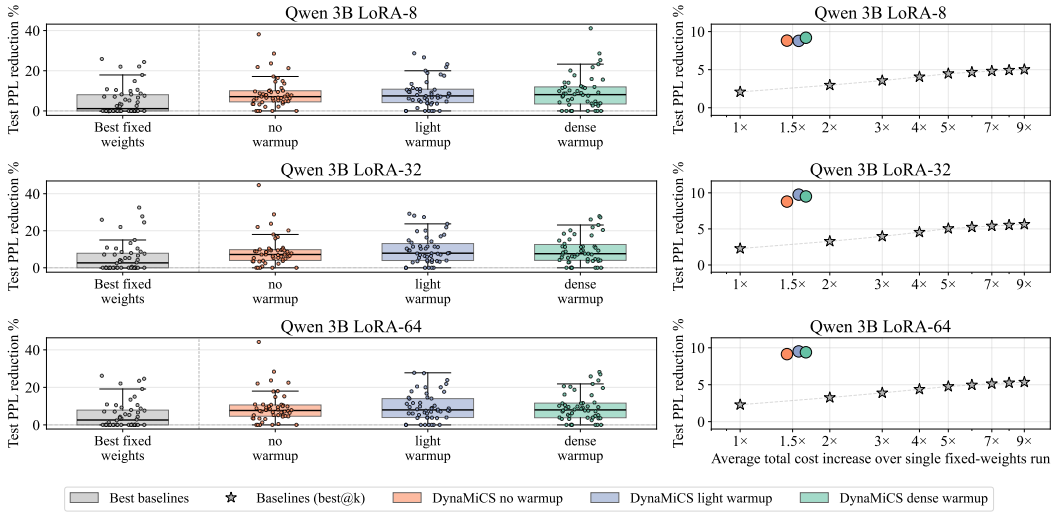


Figure 9: **LoRA rank comparison on Qwen2.5-3B. Left (Boxplots):** Per-scenario PPL reduction for ranks 8, 32, and 64. For baselines, we report the oracle best across all fixed-weight configurations per scenario. **Right (Scatter plots):** Mean PPL reduction vs. total compute cost. Baseline stars indicate the $best@k$ perplexity reduction, i.e., the best result among k fixed-weight configurations, at the corresponding budget level, while DynaMiCS colored circles show the reduction of a single adaptive run at its schedule-dependent cost. Mean $\pm$ std over three seeds.

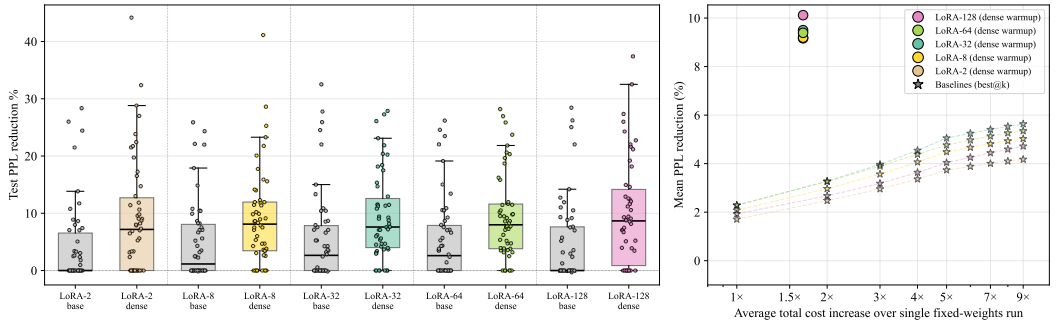


Figure 10: **LoRA rank comparison on Qwen2.5-3B. Left (Boxplots):** Per-scenario PPL reduction for ranks 2, 8, 32, 64 and 12. For baselines, we report the oracle best across all fixed-weight configurations per scenario. **Right (Scatter plots):** Mean PPL reduction vs. total compute. Baseline stars indicate the $best@k$ perplexity reduction, i.e., the best result among k fixed-weight configurations, at the corresponding budget level, while DynaMiCS colored circles show the reduction of a single adaptive run at its schedule-dependent cost.

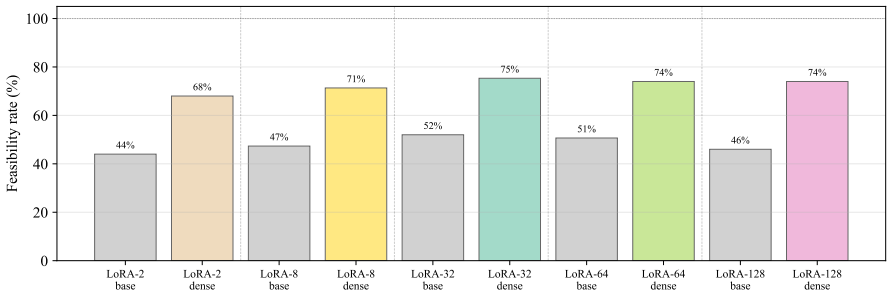


Figure 11: **Feasibility rate across LoRA ranks (dense warmup).** A run is counted as feasible only if a checkpoint after step 0 satisfies all constraints while improving target perplexity (the base model checkpoint is excluded, so feasibility requires that fine-tuning actually learned something). Each pair compares the fixed-weight baseline with the dense warmup schedule at a given rank. Compared to baselines, DYNAMICS improves feasibility at every rank; performance plateaus at rank 32. Higher is better.

I FULL-PARAMETER FINE-TUNING

The main experiments use LoRA, where updates are confined to a low-rank subspace. Here we ask whether DYNAMICS still works when every parameter is free to move, i.e., whether the local modeling choices described in Section 3.3 continue to hold in that regime. We rerun all 50 scenarios with full-parameter fine-tuning of Qwen2.5-3B, using the dense warm-up schedule and the same step budget, batch size and evaluation pipeline as the main experiments ($LR = 10^{-6}$, decided with a sweep).

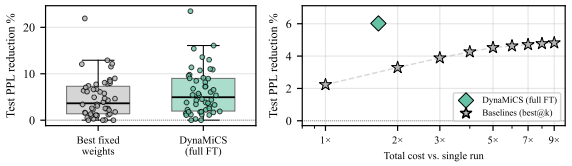


Figure 12: **Full-parameter fine-tuning (Qwen2.5-3B, all 50 scenarios). Left:** per-scenario test perplexity reduction for the best fixed-weight configuration and for a single DYNAMICS run under the dense warm-up schedule; infeasible runs contribute 0%. **Right:** mean reduction against compute cost; stars are best@k over k fixed-weight runs, up to the grid size. Format as in Fig. 2.

In this setup, DYNAMICS reduces target perplexity by $6.0 \pm 4.8\%$, against $4.8 \pm 4.3\%$ for the best fixed-weight configuration selected per scenario, and stays above the best@k curve at every budget (Figure 12). The relative cost of DYNAMICS is unchanged at $\sim 1.7\times$, because full fine-tuning inflates the baseline run and the probing runs by the same factor. The advantage therefore persists outside LoRA. The smaller absolute gains are explained by the smaller learning rate required fore stability by full-parameter fine-tuning: the identical step budget produces less total adaptation than in its LoRA version.

J RESULTS BY NUMBER OF CONSTRAINTS

Figure 13 breaks down the test perplexity reduction by the number of constraints in each scenario, using the dense warmup schedule. DYNAMICS maintains a stable advantage over baselines from 3 to 7 constraints, while baselines degrade steadily and collapse to near-zero reduction by 7 constraints. Only at 10 constraints, the most extreme tier where the model must simultaneously preserve performance on 10 held-out domains, DYNAMICS struggles as well, consistent with the inherent difficulty of satisfying that many preservation requirements at once (note, however, that solutions are still found for Qwen3-8B and Gemma3-12B).

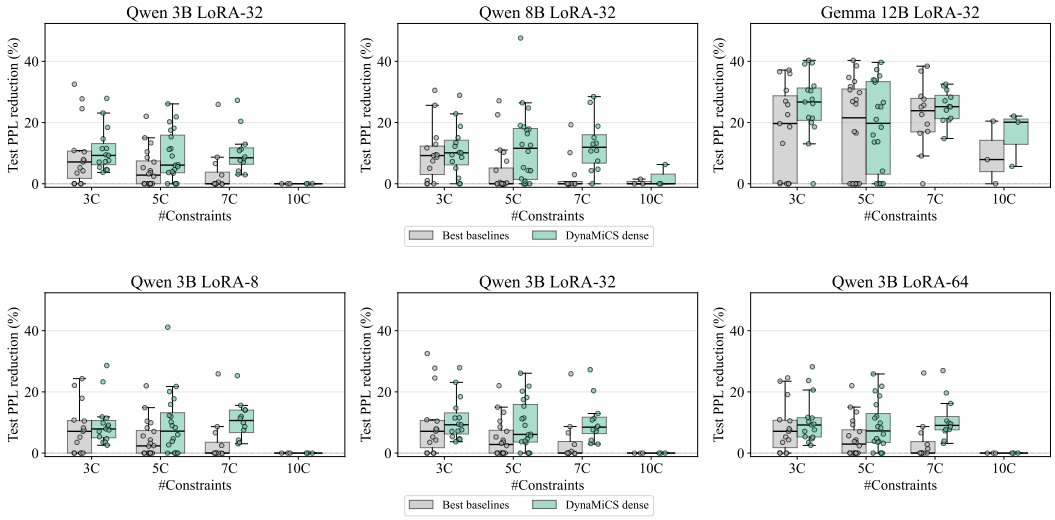


Figure 13: **Test perplexity reduction (%) grouped by number of constraints** (dense warmup vs. oracle baselines). DYNAMICS uses the dense warmup schedule (11 updates); baselines report the oracle best across all fixed-weight configurations. **Top:** Model comparison at LoRA rank 32 (Qwen2.5-3B, Qwen3-8B, Gemma3-12B). **Bottom:** LoRA rank comparison on Qwen2.5-3B (ranks 8, 32, 64). Each dot is one scenario, averaged over three seeds.

K CONSTRAINT VIOLATION IN INFEASIBLE SCENARIOS

Even when neither method finds a fully feasible checkpoint, DYNAMICS fails more gracefully than fixed-weight baselines. We restrict attention to the *jointly infeasible* scenarios, i.e., those where both the best baseline configuration and DYNAMICS (dense warmup) fail to produce any checkpoint satisfying all constraints across all seeds. For each such scenario and method, we report the maximum constraint violation at the checkpoint closest to feasibility (i.e., the checkpoint minimizing the worst-case violation across constrained domains).

Figure 14 shows the results. Across all model configurations, DYNAMICS consistently achieves lower violation than the best baseline, with a more concentrated distribution closer to zero. This indicates that the optimizer's constraint-aware weight selection reduces forgetting even in scenarios where full constraint satisfaction is unattainable, a property that baselines, which have no mechanism to respond to constraint violations, cannot match.

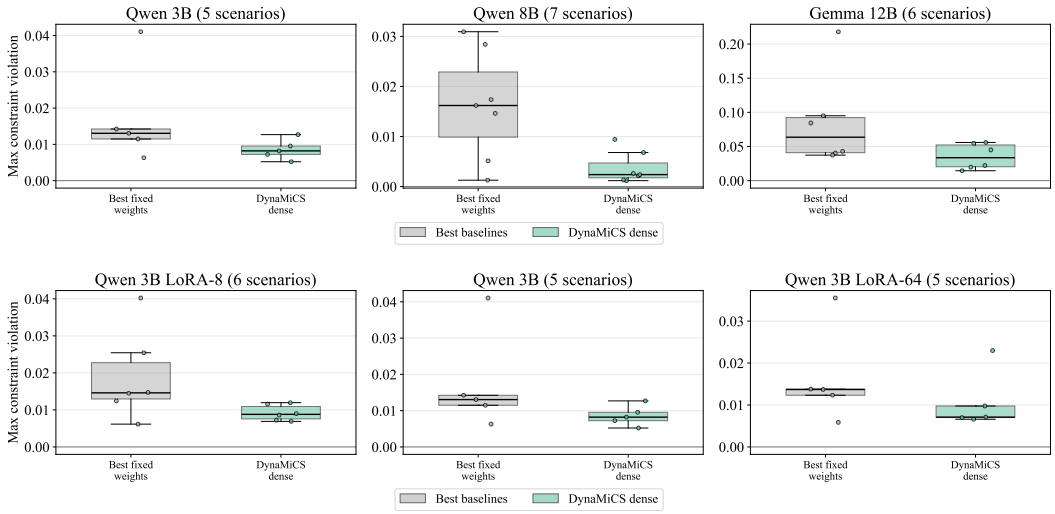


Figure 14: **Constraint violation in jointly infeasible scenarios** (dense warmup schedule). We consider the scenarios where neither DYNAMICS nor any baseline configuration achieves feasibility. Each dot is one scenario (averaged over seeds). The y-axis shows the maximum constraint violation at the checkpoint closest to feasibility, i.e., the checkpoint minimizing the worst-case loss increase across all constrained domains. For baselines, we report the best (lowest violation) across all fixed-weight configurations. Lower is better. **Top:** Model comparison at LoRA rank 32 (Qwen2.5-3B, Qwen3-8B, Gemma3-12B). **Bottom:** LoRA rank comparison on Qwen2.5-3B (ranks 8, 32, 64).

L DOES THE LOCAL LINEAR APPROXIMATION HOLD?

L.1 FIDELITY OF THE LOCAL LINEAR APPROXIMATION

DYNAMICS predicts, at each update, the loss $\hat{L}_i(w^*)$ that training on the selected mixture w^* will produce H_t steps later. After those H_t steps we observe the realized loss $L_i(\theta_{t+H_t})$ on the same evaluation batches, which lets us measure the approximation directly rather than assume it. Per update and evaluation domain we therefore record the prediction error $e_i = L_i(\theta_{t+H_t}) - \hat{L}_i(w^*)$. The realized side is a genuine run on a mixed batch, so any cross-domain interaction that the single-domain probes could not observe is contained in this error, together with time-extrapolation error and evaluation noise. All numbers below are for Qwen2.5-3B under the dense warm-up schedule, across all 50 scenarios.

Error by prediction horizon. The probe length is capped at $c_t = \min(H_t, 128)$, so H_t/c_t measures how far a prediction is used beyond the horizon on which it was fitted. Table 8 separates the two regimes. In seven of the nine scorable intervals the prediction is taken directly from the probe endpoint, and there the mean absolute error is 0.004 nats on losses of approximately 1.76 nats. Predicted and realized changes correlate at $r = 0.93$, and a predictor that assumed no change would incur 4.3 times greater error. Error grows sharply only where the schedule extrapolates past the probe. Similarly, across 2,340 constrained-domain decisions, the predicted verdict on whether a constrained domain stays at or below its reference matches the realized one in 98.0% of cases within the probed horizon, and in 83.5% for extrapolated intervals.

Interval	H_t/c_t	LoRA r=32	Full FT
$H_t \leq 128$ (7 of 9 intervals)	1	0.004 nats	0.003 nats
$H_t \in \{256, 512\}$ (2 of 9 intervals)	2 or 4	0.040 nats	0.022 nats

Table 8: Mean absolute error between predicted and realized losses, by prediction horizon. Qwen2.5-3B, 50 scenarios, dense warm-up schedule. Predictions are logged at the update that follows them, so the prediction issued at the final update ($H_t = 1024$) has no logged counterpart and is not scored; the same applies to the interval preceding the first update, which has no anchor. Nine of the schedule's eleven intervals are therefore scored.

Isolating the contribution of unobserved mixing. Interactions that the probes never see can only arise when the selected mixture spreads weight across several sources. We therefore separate updates whose weights concentrate on a single source from those that distribute across several. The error is 0.002 nats in the concentrated case and 0.007 nats in the distributed case; the 0.005-nat difference is an empirical estimate of the additional error associated with unobserved mixing interactions. This is small relative both to the loss scale of approximately 1.76 nats and to the median distance of 0.015 nats to the constraint boundary.

The result is not specific to LoRA. Under full-parameter fine-tuning the error at $H_t/c_t = 1$ is 0.003 nats, slightly below the LoRA figure, and mixed selections are no less accurate than nearly pure ones. Full-parameter updates therefore do not degrade the local additive model in the regime where the method uses it.

L.2 RELAXING LINEARITY: NON-LINEAR WEIGHT UPDATES

Appendix L.1 shows that the linear predictor is accurate over the horizons at which it is used. It is nonetheless worth asking whether a richer predictor would do better, particularly over the longer intervals where the linear model extrapolates past the probe. We therefore evaluate a variant that fits parametric curves to the loss trajectories observed at multiple intermediate checkpoints during slope estimation, replacing the linear extrapolation with non-linear models. This comes at additional evaluation cost: fitting curves requires multiple checkpoints per probe (we set this number to five in our experiments), substantially increasing the overhead per weight update (see Appendix E.1 for a detailed cost analysis). We use the same models as in our main experiments.

During each probing phase of c_t steps on dataset D_j, we evaluate at five intermediate checkpoints and fit the observed loss trajectory on each evaluation dataset as a function of the normalized probe progress $u \in [0, 1]$ (with $u = 0$ the pre-probe checkpoint and $u = 1$ the end of the c_t step probing phase), using a model selection procedure that chooses the best model among a power law with offset ($a \cdot u^b + d$) and an exponential ($a \cdot e^{-bu} + d$) using corrected Akaike Information Criterion (AICc; Akaike, 1974; Hurvich & Tsai, 1989), falling back to a linear fit when all candidates fail to converge. Curves are fit to absolute losses (anchored at $u = 0$) rather than loss deltas. A candidate mixture $\mathbf{w}$ devotes $w_j H_t$ of the interval's steps to D_j, i.e. $u_j = w_j H_t / c_t$ in normalized units, and the fitted curves combine additively: $\hat{L}_i(\mathbf{w}) = L_i(\theta_t) + \sum_j \left[f_{ij}(u_j) - f_{ij}(0) \right]$. The optimizer solves the constrained problem equation 1 using these predictions at the target horizon. We evaluate this variant with the same fixed-interval schedules as Appendix G, sweeping $H \in \{128, 256, 512\}$ with probing budget $c = H$.

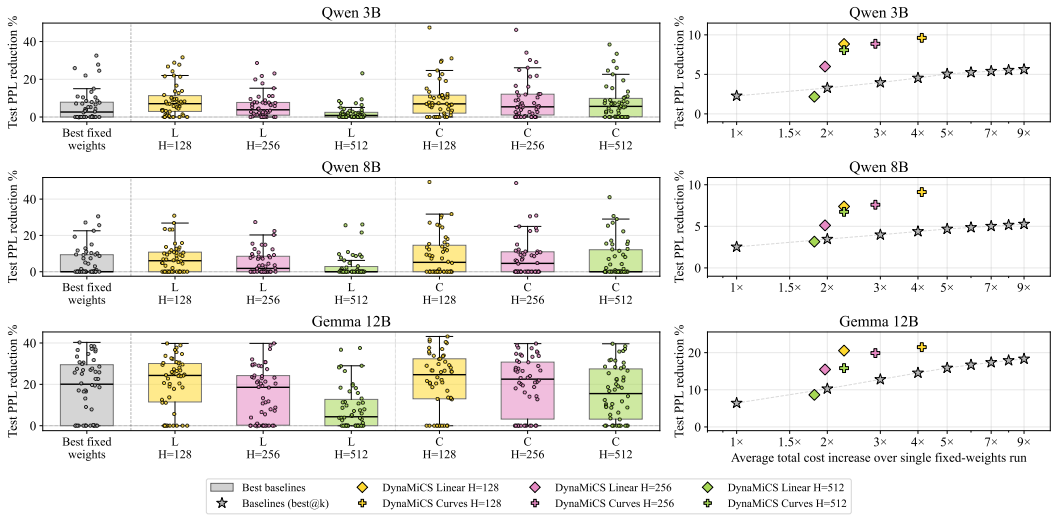


Figure 15: **Curve-fitting predictor: model comparison at LoRA rank 32.** Same layout as Figure 7. Each DynaMiCS configuration uses a constant reweight interval H with probing budget $c = H$ and a non-linear predictor (fitted on five intermediate evaluations steps).

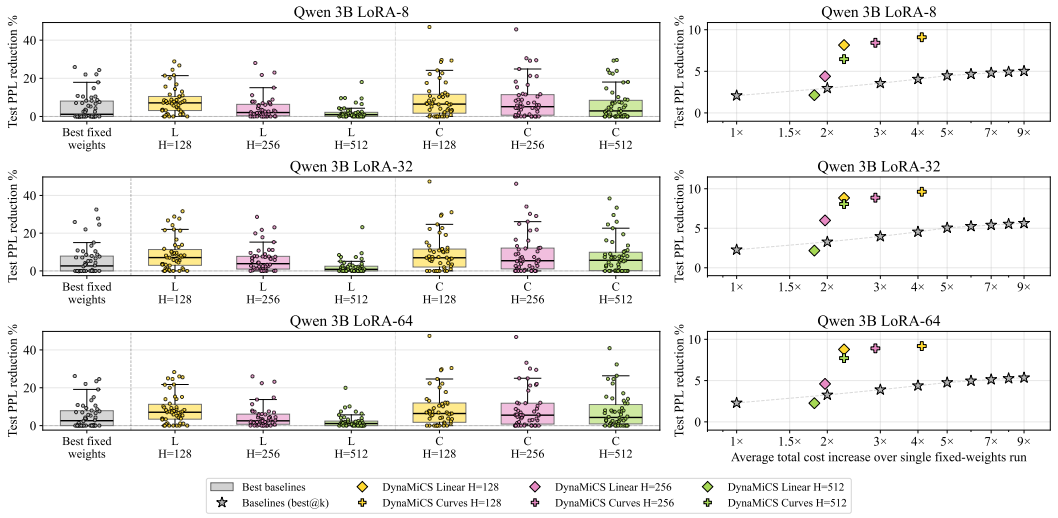


Figure 16: **Curve-fitting predictor: LoRA rank comparison on Qwen2.5-3B** (ranks 8, 32, 64). Same layout as Figure 7.

Figures 15 and 16 show the results. The curve-fitting variant is less sensitive to large update intervals than its linear counterpart: at $H = 512$, the linear predictor degrades sharply, whereas curves retain a larger fraction of the reduction achieved at $H = 128$. This is consistent with the non-linear predictor's ability to describe longer horizons more accurately. However, this robustness comes at a higher compute cost per update (shifted rightward on the cost axis), and does not translate into an overall advantage. These results suggest that the additional compute is better spent on tighter update schedules, such as the geometric schedules in Section 4.3, that concentrate updates early in training, when the loss landscape changes most rapidly, rather than on richer per-probe predictors.

While further exploration of non-linear models remains a promising direction for longer probing horizons or strongly non-linear regimes, for short probing budgets the linear approximation offers the best cost-performance trade-off. This is consistent with Appendix L.1: since the linear predictor's error is already small at the horizons where the schedule places most of its updates, and well inside the margin separating a feasible mixture from an infeasible one, additional predictive accuracy there has little left to correct.

M ADDITIONAL BASELINES

M.1 GRADIENT ALIGNMENT

Formulation. Gradient alignment (Ren et al., 2018; Yu et al., 2020; Fan et al., 2024a) estimates the slope matrix without performing any fine-tuning steps. Instead of probing each dataset for c_t steps and observing loss changes, it computes average gradients over n_b batches and uses their inner products as slope estimates:

$$S_{ij}^{\mathrm{GA}} = -\eta \, \langle g_i, \, d_j \rangle, \tag{7}$$

where g_i is the average gradient of evaluation dataset E_i, d_j is the Adam update direction for fine-tuning dataset D_j (bias-corrected first moment divided by the square root of the second moment), and η is the learning rate. The intuition is that if the Adam step for dataset D_j aligns with the direction that decreases L_i, training on D_j should reduce the loss on E_i.

The matrix is re-estimated at every update step $t \in \mathcal{U}$ and passed to the same constrained optimization (Equation 4) used to derive DYNAMICS's mixture weights, so this baseline is dynamic in the same sense as DYNAMICS and differs from it only in how the slope matrix is obtained. Its cost is driven by the number of batches used to average gradients per dataset (in our experiments, we use 200); since no multi-step probing is required, this is comparable to DYNAMICS configurations with tight update schedules or larger evaluation budgets: the difference in compute is not the deciding factor.

Approximation issue. The key limitation of gradient alignment is that Equation 7 provides an *instantaneous* rate of change that is difficult to translate into loss predictions with proper scale over a horizon H_t. One interpretation of S_{ij}^{GA} is that it is a Taylor approximation of the loss drop on E_i after one step of Adam on D_j:

$$S_{ij}^{\mathrm{GA}} \simeq L_i(\theta_t') - L_i(\theta_t), \text{ with } \theta_t' = \theta_t - \eta d_j.$$

Using it to predict the loss after H_t steps leads to accumulating errors due to i) the Taylor expansion higher order terms, ii) the stochastic nature of the algorithm not taken into account.

In contrast, DYNAMICS's multi-step probing observes actual loss trajectories and directly measures the quantity of interest —the loss change after c_t steps— making the linear extrapolation to larger horizon better grounded.

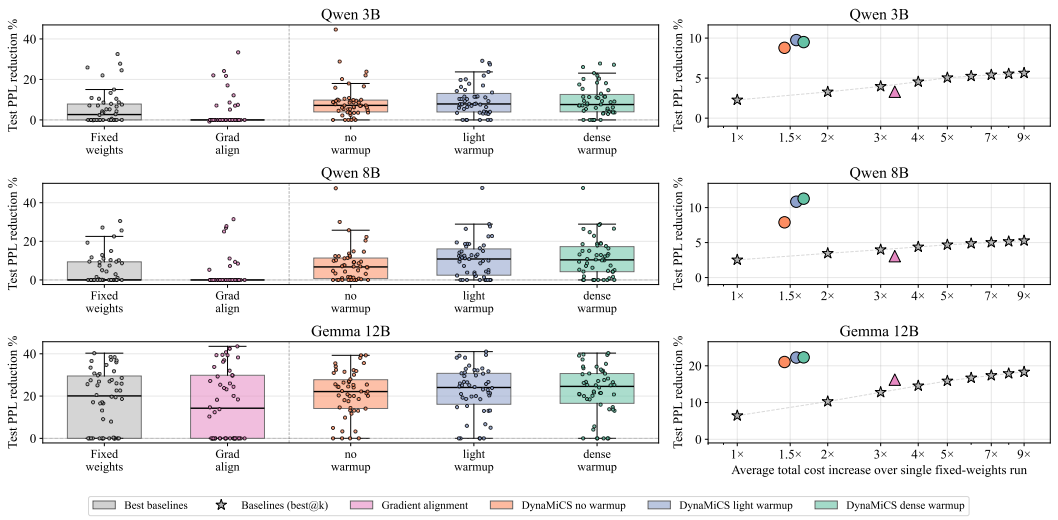


Figure 17: **Gradient alignment vs. DYNAMICS across three models.** (LoRA rank 32, 50 scenarios) Boxplots show per-scenario test PPL reduction (one dot per scenario); for fixed-weight baselines, we report the oracle best across all configurations per scenario. Scatter plots show mean PPL reduction vs. total compute cost: Baseline stars indicate the $best@k$ perplexity reduction, i.e., the best result among k fixed-weight configurations, at the corresponding budget level, while DYNAMICS circles and gradient alignment triangles show the reduction of a single adaptive run at its cost.

Results. Figure 17 compares gradient alignment against DYNAMICS and fixed-weight baselines across three models. Gradient alignment consistently underperforms DYNAMICS. The poor performance is consistent with the scale issue described above: while gradient alignment can identify beneficial *directions* (which datasets help which), it struggles to produce slope magnitudes that are calibrated for the constraint optimization and horizon, leading to either overly conservative or overly aggressive weight allocations.

M.2 EMBEDDING SIMILARITY

Formulation. Like gradient alignment, this baseline keeps DYNAMICS's constrained optimizer and changes only where the slope matrix comes from, testing a natural intuition: that fine-tuning datasets resembling a target should be upweighted. With this goal, we embed every domain with the base model, mean-pooling the last hidden state over n_b batches and centering globally, and take the cosine similarity sim_{ij} between evaluation domain E_i's validation data and fine-tuning dataset D_j's training data.

We establish a threshold to turn this one-signed quantity into a signed influence:

$$S_{ij}^{\mathrm{sim}} = -(\mathrm{sim}_{ij} - \tau)\, s, \tag{8}$$

so that similarity above τ is read as potentially beneficial transfer and similarity below it as possible interference, with s a scale factor. No training or probing takes place, only forward passes to embed the domains at step 0. The matrix is handed to the same weight updater and constrained solve as DYNAMICS, only once at step 0, so the resulting mixture is held fixed for the rest of training. We sweep $\tau \in \{0.1, 0.3, 0.5\}$ with $s = 1$ and $n_b = 8$.

Approximation issue. The key limitation of this baseline is that a similarity cannot be converted into a loss change in any principled way. Equation 8 must map a dimensionless quantity onto the nats in which Equation 4 compares predicted losses against reference levels, and nothing in the signal fixes τ or s: similarity, instead, records how related two domains are, not how much training on one moves the loss on the other. It is also blind to how much data each domain contains and, being computed once from the base model, cannot track interactions that evolve.

Results. Across the three thresholds the mixtures land in a narrow band: mean test perplexity reduction of 2.02% (τ=0.1), 2.29% (τ=0.3) and 1.48% (τ=0.5), feasible in 15, 19 and 17 of the 50 scenarios respectively. Even the best threshold therefore recovers less than a quarter of DYNAMICS's 9.50% and satisfies the constraints in fewer scenarios than the fixed-weight grid (Section M.6). These results are consistent with the embedding similarity lacking the signal the optimizer needs.

M.3 DOREMI

Formulation. DoReMi (Xie et al., 2023) learns domain weights with a small proxy model before the main run, returning a single mixture, which is then held fixed throughout training (see Appendix A for details). We maintain DoReMi's two-phase structure and its published hyper-parameters, adapting it from the pre-training setting it was designed for to fine-tuning. In Phase A a reference model (Qwen2.5-0.5B with LoRA r=32) is fine-tuned for 2,048 steps on the uniform mixture. In Phase B a proxy model of the same size trains for 2,048 steps while its domain weights are updated every 32 steps by exponentiated gradient on the clipped excess loss of proxy against reference (η=1, smoothing c=10^{-3}, excess losses measured on 16 held-out batches per domain), a Group-DRO objective on the worst-case excess loss across the fine-tuning domains. The mixture we transfer to Qwen2.5-3B is the time-average of those weights along the proxy trajectory.

Objective mismatch. DoReMi optimizes a different objective from ours: it minimizes the worst-case excess loss across the datasets it trains on, while we improve some evaluation domains subject to keeping others from degrading. Nothing in that objective distinguishes those two roles, so no reweighting of it can express the requirement that a particular loss stay below a reference level: feasibility is not a quantity DoReMi optimizes. The method also returns a single static mixture, so it cannot react to transfer and interference as they evolve.

Results. The transferred mixture reaches a mean test perplexity reduction of $2.6 \pm 6.3\%$ and is feasible in 10 of the 50 scenarios, against 9.50% for DYNAMICS (Section M.6). It also falls below the fixed-weight grid, which is expected: the grid result is the best of *k* mixtures, picked per scenario as an oracle, based on the results each fixed-weight recipe achieved. In contrast, DoReMi commits to a single mixture derived from the fine-tuning datasets alone, without reference to which evaluation domains the scenario is trying to improve.

M.4 SKILL-IT

Formulation. Skill-It (Chen et al., 2023) is the structural precursor of our slope matrix: it learns a non-negative skill-influence graph once, then reweights the mixture by multiplicative weights. It therefore replaces both parts of DYNAMICS: the estimator of cross-domain effects and the update rule that consumes it. To instantiate it, we build the graph with Skill-It's Algorithm 3, the variant its authors prescribe when training and evaluation skills do not coincide. Algorithm 3 measures exactly what our probe measures, the loss change on each evaluation domain after training on a single fine-tuning dataset in isolation, so we reuse our own probing machinery and read the graph off the slope matrix,

$$A \;=\; \max\big(0,\, -c_0\,\mathbf{S}(0)\big) \;\in\; \mathbb{R}^{M\times N}, \tag{9}$$

where $c_0 = 150$ is the probing budget, so that $c_0\,\mathbf{S}(0)$ is the total loss change over the probe rather than the per-step rate of equation 2, and $A_{ij} > 0$ measures how much training on D_j helps evaluation domain E_i. The clamp is Skill-It's own: its graph learner records an edge only where the probe decreases the loss (Algorithm 3, line 6), so measured interference never enters A. Following the paper, we estimate the graph once, with 150 probing steps, and freeze it for the rest of training. Weights are initialized as in Skill-It's Algorithm 1 (line 3), $w_j^{(1)} \propto \exp\big(\eta \sum_i A_{ij}\big)$, and are updated at each $t \in \mathcal{U}$ by

$$w_j^{(t+1)} \;\propto\; w_j^{(t)}\, \exp\Big(\eta \textstyle\sum_i A_{ij}\, \bar{L}_i\Big), \tag{10}$$

where $\bar{L}_i$ is the running mean of the losses observed on E_i and η is Skill-It's update rate. Training budget, update schedule $\mathcal{U}$ and evaluation cadence are those of DYNAMICS to allow fair comparison.

Skill-It has no natural way to express a constraint, that is, a domain whose performance must not degrade. It is nonetheless interesting to ask whether this distinction can be incorporated into the method. To this aim, we add a weight λ on the constrained-domain rows of A: at $\lambda = 0$ constraints are ignored and only targets drive the update; at $\lambda = 1$ they are treated as additional targets, which is Skill-It as published; $\lambda > 1$ emphasizes them further. We sweep $\lambda \in \{0, 0.1, 1, 10\}$ against Skill-It's update rate $\eta \in \{0.05, 0.1, 0.2, 0.5, 1, 10\}$ and select the best of the resulting configurations independently per scenario, an oracle choice that favors the baseline.

Objective mismatch. Skill-It's update rule treats every evaluation domain as a loss to be minimized, and no reference level informs the objective: what drives the update is the magnitude of each loss, not its position relative to L_i^{ref}. The λ knob does not supply one, since it rescales the constrained rows of A without introducing any level at which a loss counts as acceptable, and the rows it rescales encode positive transfer only. If a constrained domain is actively damaged by a fine-tuning dataset D_j, Skill-It's optimization can therefore react only by up-weighting the datasets that were measured to help that domain. Its cell for D_j is zero, so the damaged domain neither raises nor lowers D_j's weight, and D_j continues to be promoted by the target domains it does help. Raising λ dilutes D_j only indirectly, by promoting the constraint's helpers faster: it never acts on the source of the damage directly.

Results. Across the $\lambda \times \eta$ grid, the per-scenario best configuration reaches a mean test perplexity reduction of $3.5 \pm 7.8\%$ and is feasible in 16 of the 50 scenarios, against 9.50% and 40 of 50 for DYNAMICS (Section M.6). This is the best of 24 configurations selected per scenario after the fact, whereas DYNAMICS is reported at a single setting held fixed across all scenarios. As with the other baselines, this is not a shortcoming of Skill-It on the problem it was designed for: it selects a mixture that accelerates learning across a set of skills, and a graph that cannot represent forgetting offers nothing to optimize against when a capability must be preserved.

M.5 AIOLI

Formulation. Aioli (Chen et al., 2024) is the closest prior method to DYNAMICS. We replace both of its components: the estimator of cross-domain effects and the update rule that consumes it. Its estimator infers the parameters of a linear mixing law from the loss mixture trajectory of the training run itself, interleaving short chunks on near-one-hot mixtures ($1 - \epsilon$ on one domain, ϵ spread over the rest, with $\epsilon = 0.1$); its updater then takes one exponentiated-gradient step on the simplex. We inherit both, keeping Aioli's structure and defaults, and match its measurement budget to ours exactly. Following the paper, updates land at equal intervals rather than on our geometric schedule. We run two versions of this baseline: one loyal to Aioli's published objective, which averages the loss over all evaluation domains, and a target-aimed version that directs the update at the target domains. We sweep the exponentiated-gradient step size, the one hyper-parameter its own sensitivity analysis identifies as influential. Our implementation deviates from the paper in one respect: we restore the model after each measurement stretch, whereas Aioli never rewinds.

Objective mismatch. Aioli minimizes an aggregate loss over the evaluation domains, without constraints. It therefore has no way to express that one particular loss must stay below a reference level, and no reweighting of its objective supplies one: both variants we run are legitimate choices of the group-specific b^t vector of its Lemma 1, but neither can encode a reference level, so aiming the update at the targets pushes harder on them but leaves the remaining domains unprotected, while averaging over all of them dilutes the targets. Its estimator, by contrast, transfers to our setting unchanged: the linear solve is over the measurement mixtures, so each evaluation domain's row is independent and a rectangular matrix over two decoupled index sets—including targets with no training data of their own—costs nothing to support.

Results. Aioli reaches a mean test perplexity reduction of $3.1 \pm 7.8\%$ and is feasible in 9 of the 50 scenarios, against 9.50% for DYNAMICS (Section M.6). Neither raising the step size nor switching between the aggregate and the target-aimed objective moves that feasibility count, and the two step sizes produce visibly different weight trajectories while landing within 0.02 percentage points of each other, so the shortfall is structural rather than a matter of tuning. This is not a shortcoming of Aioli on the problem it was designed for: it optimizes an unconstrained aggregate loss, and a method with no representation of knowledge preservation constraints has no machinery to satisfy one.

M.6 BASELINES SUMMARY

Table 9 collects the baselines on the setting they all share, Qwen2.5-3B at LoRA r=32 across the 50 scenarios. They fall into two groups. Gradient alignment and embedding similarity keep DYNAMICS's constrained optimizer and replace only the estimator of cross-domain effects, so they isolate how much the probed slope matrix contributes; DoReMi, Skill-It and Aioli replace the objective as well, and none of the three can express a preservation constraint in the first place. The fixed-weight and DYNAMICS rows are those reported in the main text and are averaged over three seeds. The additional baselines reported in this appendix use a single seed.

Method	Mean test PPL reduction	Feasible
Best fixed weights	$6.9 \pm 9.2\%$	29/50
Gradient alignment	$3.3 \pm 7.5\%$	12/50
Embedding similarity	$2.3 \pm 5.0\%$	19/50
DoReMi	$2.6 \pm 6.3\%$	10/50
Skill-It	$3.5 \pm 7.8\%$	16/50
Aioli	$3.1 \pm 7.8\%$	9/50
DYNAMICS	$9.5 \pm 7.5\%$	40/50

Table 9: **Baselines on the 50 scenarios** (Qwen2.5-3B, LoRA r=32). Feasibility follows Section 4.6: infeasible runs contribute 0% to the perplexity reduction. Spreads are population standard deviations across the 50 scenarios. Gradient alignment, DoReMi and DYNAMICS each have a single configuration; for the fixed-weight grid and Skill-It we report the best configuration selected per scenario, and for embedding similarity and Aioli the best single configuration applied to all 50 scenarios. The fixed-weight and DYNAMICS rows are averaged over three seeds; additional baselines use a single seed.

N CHOOSING THE ADAPTATION–RETENTION TRADE-OFF

The reference levels L_i^{ref} are set by the experimenter, so how much *learning* to give up in order to protect a capability is a tradeoff that the experimenter can control. To quantify this, we run the two extremes of that choice on all 50 scenarios (Qwen2.5-3B, LoRA r=32, dense warm-up schedule): strict preservation (each L_i^{ref} set to its step-0 value, i.e. zero allowed degradation), and no constraints at all. In practice, in the unconstrained setup the target/constraint partition is removed and the objective is the target loss alone, while the constrained domains are still evaluated only in order to measure their degradation.

	Mean target-PPL reduction	Feasible scenarios
Unconstrained (target loss only)	$16.7 \pm 9.0\%$	11/50
Constrained (zero tolerance)	$9.5 \pm 7.5\%$	40/50

Table 10: The two extremes of the trade-off (Qwen2.5-3B, LoRA r=32, 50 scenarios). Feasibility follows Section 4.6 for both setups, though for the unconstrained arm it is a post-hoc audit, since nothing in that run optimizes for it. The perplexity reduction for the constrained setup is read at its best feasible checkpoint and contributes 0% where none exists, while for the unconstrained setup is read at the best checkpoint its own objective selects.

Strict preservation costs about 7 percentage points of target-perplexity reduction. However, without the constraints, 44 of the 50 scenarios regress a protected capability. Which of the two is preferable depends on the application, and the reference levels let the experimenter place the run anywhere between them.

Overall, data mixing cannot dissolve the underlying conflict between adaptation and retention: if a dataset carries both transfer signal and interference, no allocation of mixture weights can solve the problem, since the conflict lives in the gradients. DYNAMICS does not remove this conflict; it makes it explicit, measurable, and steerable by the practitioner.

O EXPRESSING TARGETS AND CONSTRAINTS IN TERMS OF ACCURACY

O.1 ACCURACY AS A TARGET, PER BENCHMARK

Section 5 reports the held-out accuracy target aggregated over the 50 scenarios. Because the four benchmarks differ in how much of their performance is reachable through transfer from the available fine-tuning data, Figure 18 breaks the same runs down by which benchmark was promoted to target. On average, DYNAMICS improves over the best fixed-weight configuration on all four benchmarks.

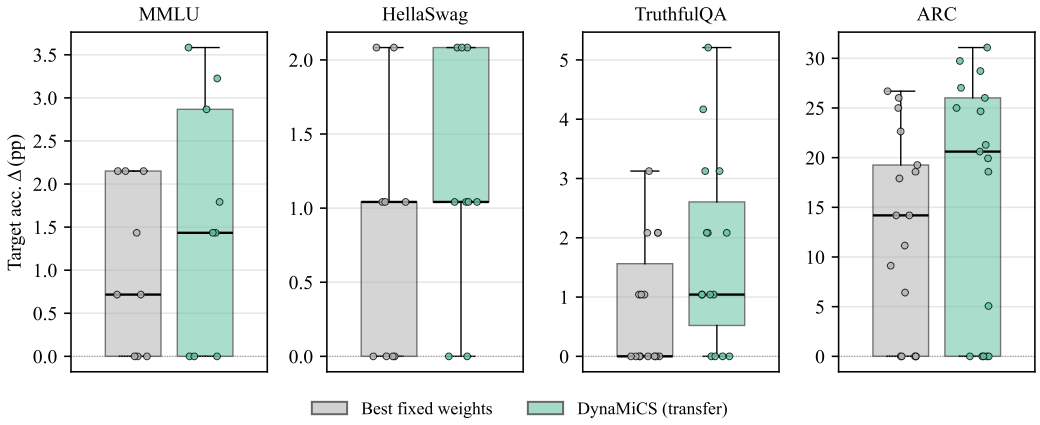


Figure 18: **Held-out accuracy target, by target benchmark (Qwen2.5-3B, r=32).** Per-scenario change in target accuracy (pp) for the best fixed-weight configuration and for a single DYNAMICS run, split by which benchmark was promoted to target. Each panel contains the scenarios in which that benchmark is the target and has its own vertical scale. Higher is better.

O.2 ACCURACY AS A CONSTRAINT

Accuracy can also define a capability to be preserved rather than improved. We rerun all 50 scenarios with the benchmark constraints expressed as accuracy instead of loss, measured as the length-normalized log-likelihood of the choice text: a benchmark is preserved when its accuracy does not fall below its pre-fine-tuning value. Constraints on non-benchmark domains remain loss-based, so scenarios mix the two metrics and DYNAMICS satisfies them jointly. Everything else follows the main experiments. On Qwen2.5-3B, DYNAMICS reduces target perplexity by $12.9 \pm 10.1\%$ against $10.0 \pm 8.8\%$ for the best fixed-weight configuration.

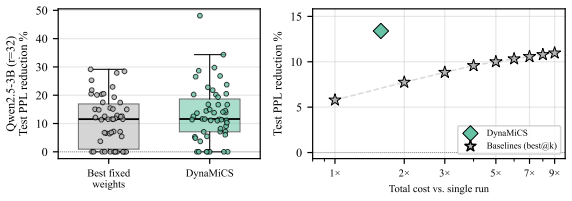


Figure 19: **Accuracy as a constraint (Qwen2.5-3B, r=32). Left:** per-scenario test-perplexity reduction on the target domains, for the best fixed-weight configuration and for a single DYNAMICS run. **Right:** mean reduction against compute cost; stars are best@k over k fixed-weight runs. Format as in Fig. 2.

P EVOLUTION OF WEIGHT ALLOCATION

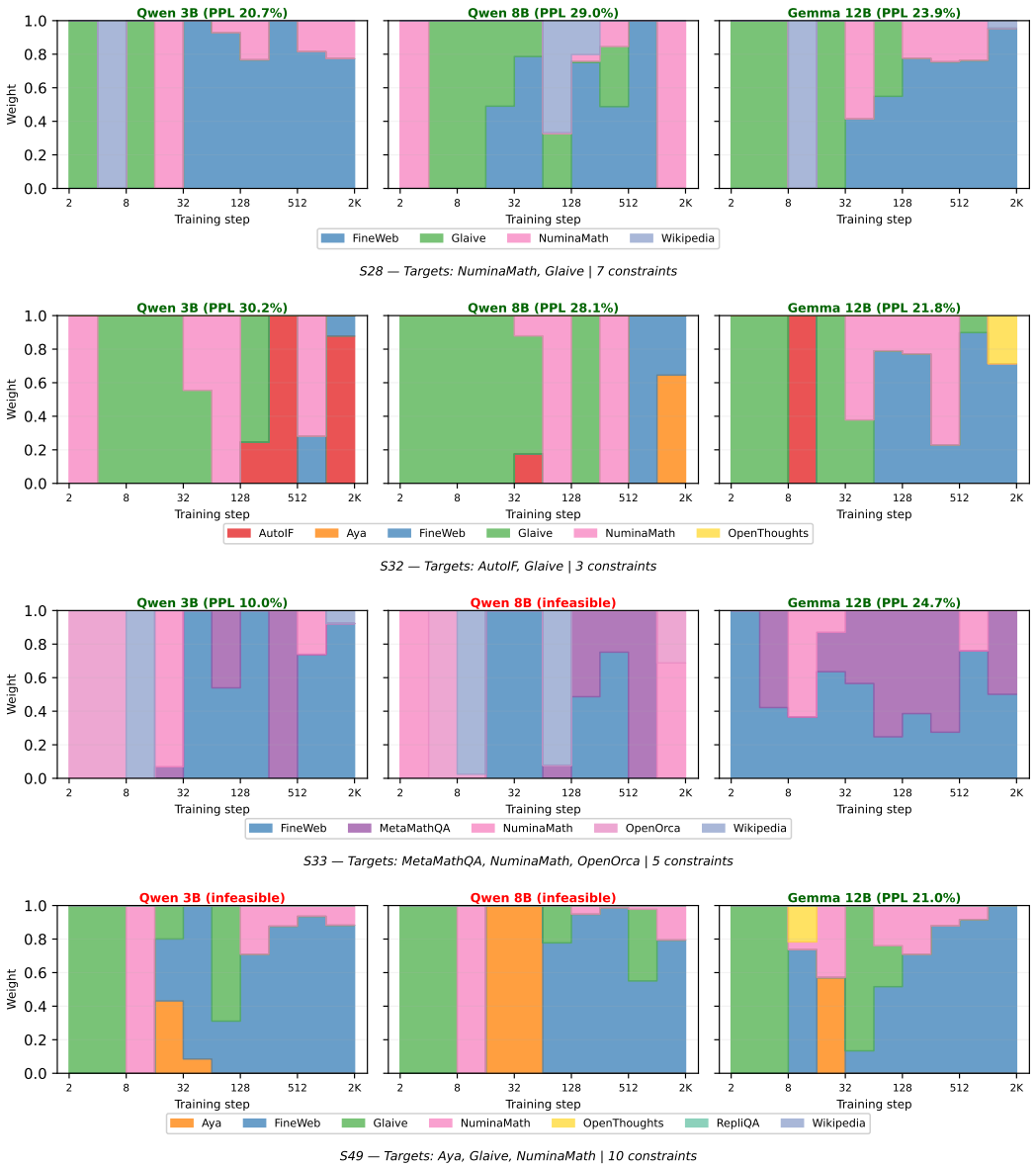


Figure 20: Evolution of dataset weights during training under a schedule of updates with dense warmup, for Qwen 3B, Qwen 8B, and Gemma 12B at LoRA rank 32. Weights are piecewise constant between controller recomputation steps.

Figure 20 illustrates how the dynamic weight allocation varies across model architectures for different scenarios. In S28 (7 constraints), all three models first emphasize the target domains, with Glaive dominating steps 2–16, before shifting weight towards FineWeb, which acts as a regularizer, later in training. In S32 (3 constraints) models converge to similar allocation strategies early on, but differ in a later phase. S33 resembles S28: weight allocations diverge across models, though all place larger weights on FineWeb in later stages. Finally, S49 is a highly constrained scenario (10 constraints) where only Gemma3-12B finds a feasible solution. While the three models begin with the same strategy, Gemma3-12B diverges at step 8, gradually settling into a stable FineWeb-dominated allocation, effectively leveraging a general-purpose data source to satisfy all constraints simultaneously while still reducing perplexity on the target domains by 21%.