BEYOND DATA SCHEDULE: FAST LLM PRETRAINING ON DATA MIXTURES BY LOSS-REWEIGHTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often acquire specialized capabilities through successive pretraining stages, but these gains can come at the cost of forgetting earlier knowledge. In this work, we investigate whether such trade-offs can be improved by changing how strongly each domain contributes to learning while keeping its data exposure fixed. We find that, in controlled knowledge–mathematics experiments, loss reweighting produces a single checkpoint that surpasses the best knowledge and reasoning scores attained along either stage trajectory. This reveals an independent control that data-mixture ratios alone do not capture. Building on this finding, we propose LARE (Loss-Adaptive Reweighting under fixed Exposure), which automatically adjusts domain weights from running loss statistics. Motivated by a theoretical analysis of gradient noise, its normalized inverse-square-root rule balances relative loss scales without changing the sampled mixture or requiring additional runs to search for weights. Across knowledge, mathematics, and code, LARE improves all five evaluated benchmarks for a 1.5B model trained on 240B tokens, averaging +7.84 pp over a matched direct mixture, including gains of +17.06 pp on GSM8K and +9.00 pp on MATH. These findings demonstrate that adjusting domain loss scales can improve joint capabilities without additional domain exposure.

1 INTRODUCTION

Large language models are increasingly expected to master multiple capabilities simultaneously—from broad world knowledge to mathematical reasoning, code generation, and instruction following. Meeting these demands has pushed pretraining beyond a single stationary data distribution: practical recipes now begin with broad-coverage corpora and subsequently introduce midtraining or continued-pretraining stages enriched with curated, domain-specialized data (Gururangan et al., 2020; OLMo et al., 2025; Liu et al., 2026a; Lu et al., 2024). Yet whether successive stages are the best way to combine multiple domains remains an open question. Emphasizing a new domain can rapidly improve the corresponding capability, but these gains often come at the cost of forgetting what was learned earlier (Ke et al., 2023; Çağatay Yıldız et al., 2025; Marek et al., 2026). In effect, multi-stage training selects a point on an acquisition–retention trade-off rather than producing a model that is simultaneously strong across domains.

A natural alternative is to train all domains jointly from a stationary mixture, replacing stage ordering with a data-mixture ratio (Liu et al., 2026b; Chen et al., 2026c). Prior work has devoted substantial effort to selecting or adapting these sampling proportions (Xie et al., 2023; Fan et al., 2024; Liu et al., 2025b; Ye et al., 2025). By keeping every domain active throughout training, joint mixtures avoid the abrupt distribution shifts introduced by stage boundaries—yet they do not automatically recover the strongest outcome of each specialized stage. Different ratios favor different domains, so a direct mixture often settles at an interior compromise (see Figure 1). Recent work further shows that domain learning can exhibit threshold behavior as mixture shares vary, while repeated exposure yields scale-dependent returns (Gu et al., 2025; Li et al., 2026b;a). Optimizing the mixture ratio therefore searches for a better point on the trade-off rather than eliminating it.

Our work starts with the identification of a central limitation of ratio tuning: a single control variable is asked to serve two fundamentally different roles. Increasing a domain's sampling share changes both its *exposure*—how much data the model sees—and, under the standard unweighted objec-

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.

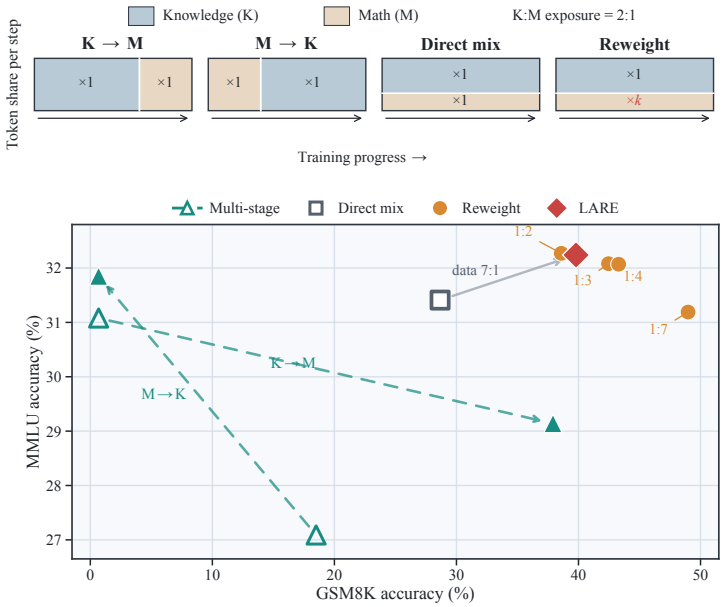


Figure 1: **Loss reweighting, rather than additional domain exposure, yields a joint improvement in knowledge and mathematical reasoning. Top:** colored areas show cumulative Knowledge:Math exposure; the 2:1 schematic keeps both domains visible. Labels $\times 1$ and $\times k$ denote relative loss multipliers: direct mixing weights both domains equally, whereas reweighting increases the mathematics loss. **Bottom:** under matched 7:1 exposure, open triangles mark first-domain peaks and filled triangles mark selected post-switch checkpoints, while the square, circles, and diamond denote direct mixing, fixed coefficients, and LARE. Fixed 1:2 weighting and LARE exceed the coordinate-wise stage maxima; LARE reaches this region online without changing exposure.

tive, its *optimization scale*—how strongly prediction errors in that domain shape parameter updates. These two quantities are conceptually distinct (Salmani et al., 2025), and their preferred allocations need not coincide: the amount of data needed for sufficient coverage may differ from the relative loss weight that yields the strongest joint model. This observation motivates a different intervention: hold the sampled-data mixture fixed and adjust domain loss coefficients directly, decoupling *what the model sees* from *how strongly each domain drives learning*. Figure 2 visualizes this distinction: remixing and reweighting trace different paths through the capability plane.

Figure 1 illustrates the opportunity created by this decoupling. With the sampled-data mixture and token budget held fixed, a suitable mathematics loss coefficient improves both MMLU and GSM8K over direct mixing—and even surpasses the best value attained on each benchmark along the multi-stage trajectories. A fixed coefficient, however, merely trades one search problem for another: its optimal value is unknown a priori, expensive to sweep at pretraining scale, and may not transfer across data compositions or training configurations.

We therefore introduce LARE (*Loss-Adaptive Reweighting under fixed Exposure*), an online domain-level controller that derives relative loss weights from running domain-loss statistics. After learning-rate warmup, it applies a normalized inverse-square-root rule that adapts relative optimization scale without globally rescaling the training objective. Under this rule, the 7:1 adaptive run reaches a competitive region of the fixed-weight trade-off in a single training run, eliminating the need for a post-hoc oracle grid. The core contributions of this work are threefold:

- We disentangle domain exposure from optimization scale in multi-domain pretraining. Under matched 30B-token exposure, stage scheduling, direct mixing, and loss reweighting yield markedly different endpoints; a suitable fixed loss coefficient simultaneously exceeds the best knowledge and mathematical-reasoning scores attained along either stage trajectory (Figure 1 and Section 2).
- We introduce LARE, a lightweight online controller that derives domain coefficients from running loss statistics. Its normalized inverse-square-root rule adapts relative optimization scale after

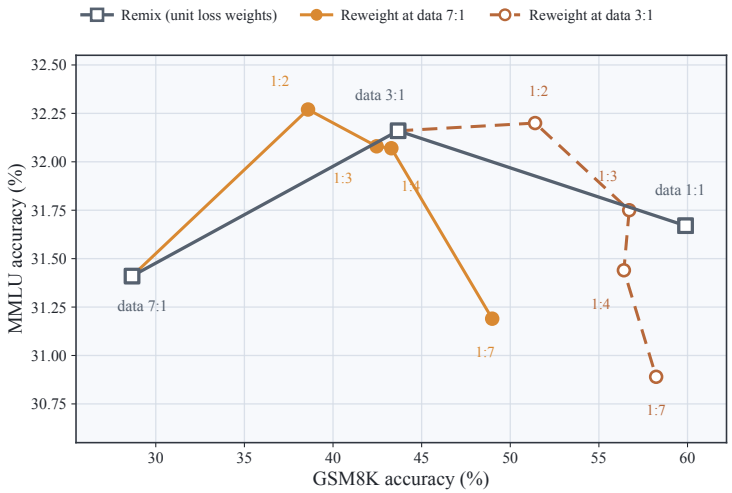


Figure 2: **At fixed exposure, reweighting expands the observed remix frontier.** Gray squares trace direct mixtures with unit loss coefficients at K:M ratios 7:1, 3:1, and 1:1. Warm curves instead fix exposure at 7:1 or 3:1 and vary only the post-warmup loss ratio from 1:1 to 1:7. At both fixed mixtures, 1:2 reweighting improves MMLU and GSM8K simultaneously; the reasoning-heavy 1:1 remix remains the boundary of the measured comparison.

warmup while keeping sampled exposure fixed and the mixture-averaged coefficient at one. The controller reaches a competitive region of the fixed-weight trade-off in a single run, without auxiliary search runs (Section 3).

- We validate the method across optimizers, sampled-data ratios, domain compositions, and training budgets (Section 4). At $4\times$ Chinchilla with K:M = 7:1, LARE improves MMLU by 0.39 points, GSM8K by 13.57 points, and MATH by 8.40 points. Extending training with mathematics and code to $8\times$ Chinchilla yields gains of 0.83, 17.06, 9.00, 6.71, and 5.60 points on MMLU, GSM8K, MATH, HumanEval, and MBPP, respectively, over direct mixing (Table 2).

2 MIXTURE RATIOS COUPLE EXPOSURE AND SCALE

Data-mixture ratios determine both which domains the model observes and how strongly their losses enter a standard unweighted update. We first show empirically that changing loss weights at fixed exposure reaches a broader capability frontier than changing the mixture itself. We then formalize why these interventions are not interchangeable and return to matched-exposure training trajectories to isolate their consequences.

2.1 REWEIGHTING EXPANDS THE REMIX FRONTIER

At a common parameter θ_t, let m_i be the sampling proportion of domain i in a mixed batch of size B, and let $\hat{g}_{i,t}^{(n)}$ be its mean unweighted gradient from n samples. With a shared scalar learning rate $\bar{\eta}$, the two SGD updates are

$$\begin{aligned} \Delta\theta_{t,\text{remix}} &= -\bar{\eta}\sum_i m_i' \hat{g}_{i,t}^{(Bm_i')}, \\ \Delta\theta_{t,\text{reweight}} &= -\bar{\eta}\sum_i m_i w_i \hat{g}_{i,t}^{(Bm_i)} = -\sum_i m_i \eta_i \hat{g}_{i,t}^{(Bm_i)}, \qquad \eta_i = \bar{\eta} w_i. \end{aligned} \tag{1}$$

Remixing changes both m_i and the number of samples used to estimate its gradient; reweighting keeps the sample allocation fixed and scales the estimate by w_i. At fixed θ_t, the expectation of $\hat{g}_{i,t}^{(n)}$ is the same domain gradient for any n, but its variance depends on n. Both interventions can change later domain gradients by producing different parameter trajectories. The effective-rate identity is exact for SGD; with nonlinear gradient transformations, it is an abstraction rather than an optimizer-level equality.

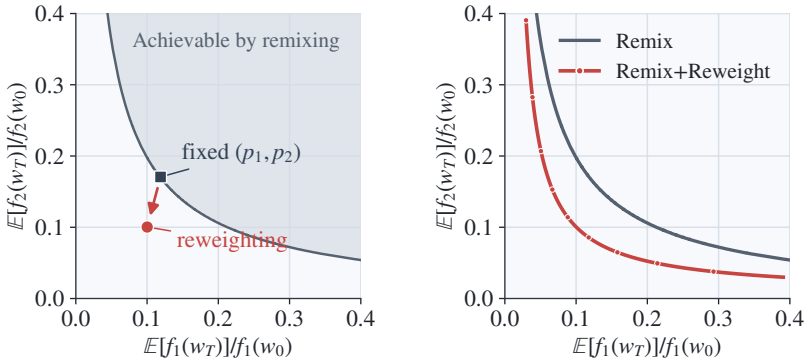


Figure 3: **Remixing cannot subsume reweighting.** Expected losses for the two-domain linear regression model, with $B = 128$, $T = 20$, block dimensions $d_1 = d_2 = 512$, and expected curvatures of 2.5 and 1 for the two domains, respectively. **Left:** At fixed $p_1 = p_2 = 1/2$, reweighting improves both domain losses and is not Pareto-dominated by any choice of mixture and shared learning rate. **Right:** Reweighting improves the Pareto frontier achievable by remixing alone.

Figure 2 reports final-checkpoint MMLU and GSM8K after 30B sampled tokens. The plotted K:M ratios specify full-token exposure before loss masking. Runs share the 1.5B model, learning-rate schedule, and optimizer assignment (Muon for hidden matrices; AdamW for embeddings and output weights). The gray direct-mixture path uses K:M ratios of 7:1, 3:1, and 1:1 at $w_{\mathrm{K}} = w_{\mathrm{M}} = 1$. The reweighting paths fix K:M at 7:1 or 3:1 and set $w_{\mathrm{M}}/w_{\mathrm{K}} \in \{1, 2, 3, 4, 7\}$ after the shared 20,000-step warmup. Within each ratio, runs use the same frozen training stream and loss masks.

The reweighting curves do not merely add nearby operating points to the remix trade-off. Together they form a broader Pareto envelope across the central operating region: the 7:1 and 3:1 direct anchors are each Pareto-dominated by their corresponding 1:2 reweighted run, and larger coefficients continue to increase mathematical reasoning before eventually trading away knowledge accuracy. The central result is therefore a strict outward expansion of the observed frontier. This separation shows that the sampled-data ratio does not uniquely determine the attainable trade-off and motivates treating exposure and optimization scale as distinct controls.

2.2 Theory: Why Reweighting Can Outperform Remixing

Equation 1 shows the additional degree of freedom available at fixed exposure. We now ask whether changing the mixture under a shared rate can match the loss trade-offs reached by separate domain rates.

We formalize this distinction in a two-domain linear regression model. Let $A_1, A_2 \in \mathbb{R}^{d\times d}$ be nonzero symmetric positive semidefinite matrices with orthogonal supports, i.e., $A_1 A_2 = 0$. In domain $j \in \{1, 2\}$, samples follow $x_j \sim \mathcal{N}(0, A_j)$ and $y_j = x_j^\top w^\star$. Using a common $w^\star$ is without loss of generality: if the two domains have different ground-truth parameters, their projections onto the respective orthogonal supports can be summed to obtain a shared parameter vector. We consider the MSE loss, and the population loss in domain j is

$$f_j(w) = \tfrac{1}{2}\,\mathbb{E}_{x_j}\big[(x_j^\top w - y_j)^2\big] = \tfrac{1}{2}\,(w - w^\star)^\top A_j (w - w^\star).$$

Starting from w_0 with $f_1(w_0), f_2(w_0) > 0$, we run $T \geq 1$ SGD steps with batches of $B \geq 2$ samples. Domain j supplies $Bp_j \in \mathbb{N}^+$ samples per batch, where $p_1 + p_2 = 1$ and $p_j \in (0, 1)$. Write $L_j(p_j, \eta_j) = \mathbb{E}[f_j(w_T)]$ for its expected terminal loss under domain rate η_j. Orthogonal supports make this loss depend only on p_j and η_j; the full update and sampling assumptions are in Appendix B.2.

We compare the following two strategies in this simplified setup:

- **Remixing** varies mixture proportions $p_1, p_2 \in (0, 1)$ while applying a shared learning rate η to both domains, which may be tuned for each choice of mixture proportions.

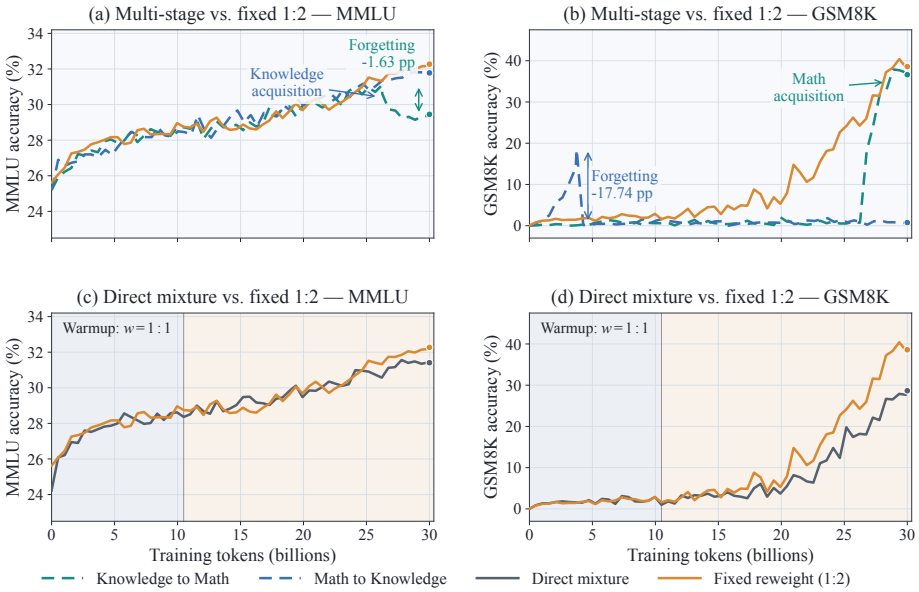


Figure 4: **At matched exposure, stage order determines retention, whereas reweighting improves both capabilities.** Every run receives 26.25B knowledge and 3.75B mathematics tokens. The top row compares stage schedules with fixed reweighting; the bottom row isolates direct mixing versus fixed reweighting. Reversing stage order reverses forgetting. Under the stationary 7:1 mixture, changing the loss ratio from 1:1 to 1:2 moves the final MMLU/GSM8K pair from 31.40/27.67 to 32.27/38.59, above the coordinate-wise maxima of both stage trajectories.

- **Reweighting** instead keeps the mixture proportions fixed and tunes learning rates η_1 and η_2 separately for the two domains.

Proposition 1 (Remixing cannot dominate reweighting)**.** *Fix any feasible mixture p_1, p_2. Let $\eta_1^\star \in \arg\min_{\eta_1} L_1(p_1, \eta_1), \eta_2^\star \in \arg\min_{\eta_2} L_2(p_2, \eta_2)$. Then the optimal domain losses achieved by reweighting $\big(L_1(p_1, \eta_1^\star), L_2(p_2, \eta_2^\star)\big)$ cannot be Pareto-dominated by remixing. Specifically, for every feasible mixture $(p_1', p_2') \neq (p_1, p_2)$ and every shared step size η',*

$$L_1(p_1', \eta') > L_1(p_1, \eta_1^\star) \quad \textit{or} \quad L_2(p_2', \eta') > L_2(p_2, \eta_2^\star). \tag{2}$$

The proof is provided in Appendix B.2.

Figure 3 illustrates Proposition 1 with a concrete instance of the model. The eigenvalues of A_1 and A_2 are drawn i.i.d. and uniformly over $[1.25, 3.75]$ and $[0.5, 1.5]$, respectively. At fixed mixture $p_1 = p_2 = 1/2$, reweighting improves both domain losses, and no choice of mixture and shared learning rate can simultaneously match both losses. Applying reweighting improves the Pareto frontier achievable by remixing. Details are provided in Appendix B.3.

2.3 Same Domain Exposure, Different Outcomes

To test whether cumulative domain exposure alone determines the knowledge–reasoning trade-off, we compare stage scheduling, direct mixing, and loss reweighting under matched token budgets. All runs share the same model configuration, 30B-token budget, learning-rate schedule, and optimizer assignment: Muon for matrix-valued hidden parameters and AdamW for embedding and output parameters. Each run receives 26.25B knowledge tokens and 3.75B mathematics tokens. The two stage baselines present this allocation in opposite orders; the stationary baselines sample both domains throughout training, with direct mixing and fixed reweighting differing only in their relative loss coefficients. Figure 4 shows the resulting unsmoothed MMLU and GSM8K checkpoint trajectories; the exact token schedules, final scores, and first-domain drops are reported in Table 7 of Appendix D.2. Scalability under different optimizers and training budgets is examined in Section 4.

Reversing stage order reverses the trade-off. Knowledge → Math ends at 29.45 MMLU and 36.62 GSM8K after a 1.63-point MMLU drop; Math → Knowledge ends at 31.78 and 0.76 after a 17.74-point GSM8K drop. In each case, the second stage erodes performance on the first domain.

Direct mixing moderates rather than resolves the trade-off. The stationary 7:1 mixture ends at 31.40 MMLU and 27.67 GSM8K, avoiding the severe GSM8K collapse of Math → Knowledge. Still, both scores remain below the respective maxima reached along the two multi-stage trajectories.

Loss reweighting improves both capabilities at fixed exposure. Among the fixed coefficients in Figure 1, $w_{\mathrm{K}} : w_{\mathrm{M}} = 1{:}2$ attains the highest MMLU and is used here. At the same 7:1 exposure, it reaches 32.27 MMLU and 38.59 GSM8K, improving on direct mixing by 0.87 and 10.92 points, respectively, and exceeding the best scores attained anywhere along the two stage trajectories (31.85 and 37.91).

Together, these matched-exposure runs show that domain token counts alone do not determine the knowledge–reasoning outcome: stage order changes which capability is retained, while direct mixing yields a compromise. Reweighting the same stationary mixture improves both metrics, identifying domain loss scale as a separate control over the trade-off.

3 LARE: LOSS-ADAPTIVE REWEIGHTING UNDER FIXED EXPOSURE

Section 2 establishes that loss scale matters, but a fixed weight does not provide a practical rule for choosing the coefficients. Grid search requires a separate pretraining run for each candidate, and the best setting may not transfer across data mixtures, compute budgets, or optimizers. We therefore seek an online alternative that infers and adapts relative domain weights from signals already available during training, eliminating auxiliary search runs.

3.1 FROM DOMAIN LOSS TO OPTIMIZATION SCALE

We derive the inverse-square-root exponent from stochastic-gradient noise near a domain optimum. Consider binary logistic regression with $x \sim \mathcal{N}(0, I_d)$, true label $y_0 = \mathrm{sign}(u^\top x)$ for a unit vector u, and observed label y obtained by independently flipping y_0 with probability $\rho \in (0, 1/2)$. Let $r_\rho^\star$ minimize the population loss along $w = ru$ over $r \geq 0$, write $w_\rho^\star = r_\rho^\star u$ and $\ell_\rho^\star = \mathbb{E}\log(1 + e^{-y(w_\rho^\star)^\top x})$, and let Σ_ρ be the per-example gradient covariance at $w_\rho^\star$.

Theorem 1 (Loss–gradient-noise scaling)**.** *For the Gaussian model above with $d \geq 2$, as $\rho \downarrow 0$,*

$$\ell_\rho^\star = \Theta(\sqrt{\rho}), \qquad u^\top \Sigma_\rho u = \Theta\big((\ell_\rho^\star)^2\big), \qquad v^\top \Sigma_\rho v = \Theta(\ell_\rho^\star) \tag{3}$$

for every unit vector $v \perp u$. Consequently, $\mathrm{tr}(\Sigma_\rho) = \Theta((d-1)\ell_\rho^\star + (\ell_\rho^\star)^2)$, so the root-mean-square stochastic-gradient norm scales as $\Theta(\sqrt{d\ell_\rho^\star})$. Appendix B.1 gives the complete derivation and constants.

Because a loss coefficient w_i scales the root-mean-square gradient linearly, Theorem 1 suggests, for comparable effective dimensions and domain constants,

$$\mathrm{RMS}(w_i g_i) = \Theta\Big(w_i \sqrt{d_i \ell_i^\star}\Big), \qquad w_i \propto (\ell_i^\star)^{-1/2}, \tag{4}$$

which offsets the larger noise scale of a high-loss domain. This stylized model motivates the exponent rather than an exact language-model scaling law. Since $\ell_i^\star$ is unavailable and domain difficulty changes during training, LARE uses smoothed current losses and normalizes the resulting coefficients as described next.

3.2 THE ADAPTIVE REWEIGHTING RULE

At step t, LARE tracks exponential moving averages $\hat{\ell}_{i,t}$ of the mean unweighted cross-entropy on unmasked domain-i targets and $\hat{c}_{i,t}$ of their counts. The estimated loss-token share is $\hat{\pi}_{i,t} = \hat{c}_{i,t} / \sum_j \hat{c}_{j,t}$. All coefficients remain one during learning-rate warmup; afterward, the next-step weights are

$$r_{i,t} = (\hat{\ell}_{i,t} + \epsilon)^{-1/2}, \qquad w_{i,t+1} = \frac{r_{i,t}}{\sum_j \hat{\pi}_{j,t} r_{j,t}}. \tag{5}$$

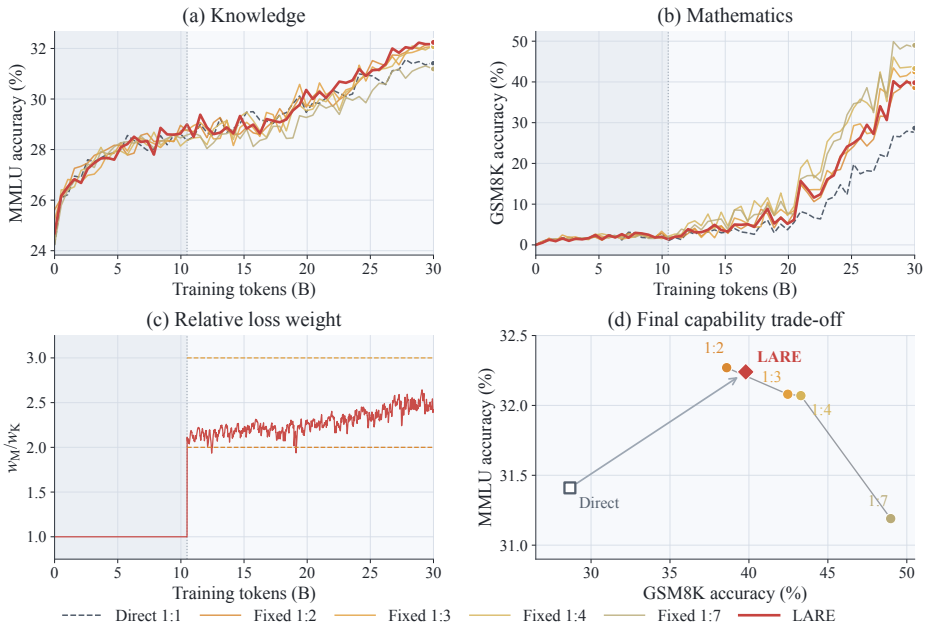


Figure 5: **LARE selects a stable, competitive loss scale in one run.** The top row shows unsmoothed MMLU and GSM8K trajectories; the bottom row shows the selected relative weight and final endpoints. After the shared 20,000-step warmup at $w_M/w_K = 1$, the ratio averages 2.30 from step 25,000 onward and ends at 2.41, between the fixed 1:2 and 1:3 references. LARE moves the direct endpoint from 31.40/27.67 to 32.24/39.80 without an auxiliary coefficient sweep.

Here $\epsilon > 0$ ensures numerical stability. The normalization keeps $\sum_i \hat{\pi}_{i,t} w_{i,t+1} = 1$, changing the relative domain loss scales without changing their mixture-averaged coefficient. Algorithm 2 gives the full update, including absent-domain handling; Appendix D.1 reports the settings.

3.3 Validating the Reweighting Dynamics

We validate LARE in the controlled 30B-token, 7:1 knowledge–mathematics setting used for the fixed-weight sweep. The model, sampling distribution, optimizer, learning-rate schedule, and token budget are all held constant; the loss-weighting rule is the only experimental variable. Figure 5 compares MMLU and GSM8K trajectories of the full fixed-weight grid against LARE, alongside the relative weight $w_{M,t}/w_{K,t}$ selected online. Together, the three panels address two questions: whether the controller is stable after warmup, and whether the scale it selects occupies a useful region of the fixed-weight trade-off.

After warmup, the selected weight ratio remains stable near the fixed 1:2 and 1:3 settings (Figure 5). At the final checkpoint, LARE improves direct 1:1 weighting by 0.84 MMLU and 12.13 GSM8K points. Its endpoint is competitive with fixed 1:2, showing that the controller can select a useful loss scale in one run without an auxiliary sweep.

4 Experiments: Ablations and Scaling Up

4.1 Experimental Setup

All experiments use a common 1.5B GPT-2-style decoder-only Transformer (Radford et al., 2019) and organize the training corpus into knowledge (K), mathematics (M), and code (C). Mixture ratios denote full tokens sampled before loss masking, while $1\times$, $4\times$, and $8\times$ Chinchilla (Hoffmann et al., 2022) correspond to 30B, 120B, and 240B sampled tokens. Following the controlled optimizer comparison in Section 4.2, all subsequent main experiments use ActGD. Unless stated otherwise, each reported cell corresponds to a single run evaluated at its final checkpoint. Appendix C specifies

Algorithm 1 Row-wise $(2, \infty)$ ActGD update for a linear weight matrix.

Require: Weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$; batch gradient G_t; uncentered input second moment C_t; EMA states H_{t-1}, S_{t-1} initialized at zero; decays β_g, β_c; learning rate η_t; weight decay λ; numerical constant ϵ

$H_t \leftarrow \beta_g H_{t-1} + (1-\beta_g)G_t$; $S_t \leftarrow \beta_c S_{t-1} + (1-\beta_c)C_t$
$\widehat{H}_t \leftarrow H_t/(1-\beta_g^t)$; $\widehat{S}_t \leftarrow S_t/(1-\beta_c^t)$
$A_t \leftarrow \text{NEWTONSCHULZINVERSE}(\widehat{S}_t, \epsilon)$
$U_t \leftarrow \widehat{H}_t A_t$
for $j = 1, \ldots, d_{\text{out}}$ **do**
    $D_{t,j} \leftarrow U_{t,j}/\big(\sqrt{\max\{0, \widehat{H}_{t,j} U_{t,j}^\top\}} + \epsilon\big)$
end for
$W \leftarrow (1-\eta_t\lambda)W - \eta_t D_t$

Table 1: **ActGD gives the strongest direct-mixture result while using a single optimizer-specific hyperparameter set.** All runs use the same 1.5B model, 1× Chinchilla budget, and K:M = 7:1 mixture. "Blocks" and "Embed./head" identify the update applied to transformer-block matrices and vocabulary-facing weights, respectively. A checkmark indicates that both groups share one optimizer family and hyperparameter set; a cross indicates a hybrid configuration with separately tuned settings. Bold denotes the best benchmark result.

Optimizer	Blocks	Embed./head	Single HP set	MMLU	GSM8K
AdamW	AdamW	AdamW	✓	31.30	21.91
Muon	Muon	AdamW	×	31.40	27.67
ActGD	ActGD	ActGD	✓	**31.82**	**28.58**

data construction and masking; Appendices D and F provide detailed model, training, and optimizer settings, while Appendix D.5 specifies evaluation protocols.

4.2 Ablating over Optimizers

Alongside LARE, we introduce Activation Gradient Descent (ActGD), a base optimizer that constructs row-wise updates in a whitened activation geometry and applies the same update family to every learned matrix. Algorithm 1 summarizes the practical update, while Appendix F provides its derivation and implementation details.

To select the optimizer for the subsequent experiments, we compare AdamW (Loshchilov & Hutter, 2019), Muon (Liu et al., 2025a), and ActGD under the same 1.5B model, 1× Chinchilla budget, and K:M = 7:1 direct-mixture objective. This comparison deliberately excludes LARE so that it isolates the base optimizer rather than its interaction with loss reweighting. Because optimal hyperparameters do not generally transfer across optimizers, unequal tuning can confound the comparison (Wen et al., 2026). We therefore sweep the learning rate or rates independently for each optimizer and select its best configuration at the full training budget instead of transferring a common learning rate; for the hybrid Muon setup, the Muon and auxiliary AdamW learning rates are tuned separately. All remaining model, data, and schedule settings are matched.

Table 1 shows that ActGD obtains the highest MMLU and GSM8K scores, outperforming Muon by 0.42 and 0.91 points and AdamW by 0.52 and 6.67 points, respectively. Beyond this accuracy advantage, ActGD applies the same activation-aware update family to transformer blocks, token embeddings, and the output head, using a single shared hyperparameter set throughout. Muon instead combines a Muon block update with an auxiliary AdamW optimizer for the vocabulary-facing parameters, requiring the two components to be tuned separately. The unified ActGD configuration therefore reduces the optimizer search space and simplifies hyperparameter tuning. Accordingly, all subsequent main-text experiments use ActGD; results with the other optimizers are reported in Appendix D.4.

Table 2: **LARE improves all matched $4\times$ and $8\times$ benchmarks.** Mean gains over direct mixing are 7.45 points for K:M at $4\times$ Chinchilla and 5.90 and 7.84 points for K:M:C at $4\times$ and $8\times$ Chinchilla, respectively. At $8\times$ Chinchilla, our model exceeds Qwen1.5-1.8B, pretrained on 2.2T tokens (over $9\times$ our token budget), by 9.22 points on average across five benchmarks. Direct–LARE pairs match model, sampled mixture, optimizer, and token budget. Avg. Δ averages gains over available benchmarks; bold marks the higher score within each matched pair.

Model	Objective	Size	Tokens	Mixture	MMLU	GSM8K	MATH	HumanEval	MBPP	Avg. Δ (pp)
Ours	Direct mixture	1.5B	120B	K:M = 7:1	33.97	37.00	17.20	–	–	–
Ours	LARE	1.5B	120B	K:M = 7:1	**34.36**	**50.57**	**25.60**	–	–	+7.45
Ours	Direct mixture	1.5B	120B	K:M:C = 7:0.5:0.5	33.52	26.76	13.80	15.85	9.40	–
Ours	LARE	1.5B	120B	K:M:C = 7:0.5:0.5	**33.53**	**41.55**	**20.20**	**21.34**	**12.20**	+5.90
Ours	Direct mixture	1.5B	240B	K:M:C = 7:0.5:0.5	34.13	29.72	15.80	17.68	9.80	–
Ours	LARE	1.5B	240B	K:M:C = 7:0.5:0.5	**34.96**	**46.78**	**24.80**	**24.39**	**15.40**	+7.84
Qwen1.5	Base	1.8B	2.2T	external	33.21	31.46	9.20	23.17	3.20	–
Qwen1.5	Base	7B	2.4T	external	42.31	51.63	14.80	44.51	39.80	–

4.3 SCALING ACROSS TRAINING BUDGETS AND DOMAINS

Reweighting gains persist at $4\times$ Chinchilla. At $4\times$ Chinchilla, LARE improves all three reported benchmarks over the matched K:M = 7:1 direct mixture (Table 2). MMLU rises modestly from 33.97 to 34.36, while GSM8K gains 13.57 points and MATH gains 8.40 points. The reasoning benefit seen at $1\times$ Chinchilla thus persists at the larger budget without a knowledge decline.

LARE also improves multiple domains. With code added at K:M:C = 7:0.5:0.5, LARE improves every measured benchmark at both $4\times$ and $8\times$ Chinchilla (Table 2). The five-benchmark mean gain grows from 5.90 to 7.84 points: the GSM8K gain rises from 14.79 to 17.06 points, the MBPP gain from 2.80 to 5.60 points, and the MMLU gain from 0.01 to 0.83 points. The larger gains at $8\times$ than at $4\times$ on all five benchmarks indicate that LARE's advantage increases over the longer training horizon tested here. Appendix D.6 shows the MMLU, GSM8K, and MBPP training curves. The Qwen1.5 base models (Bai et al., 2023) provide external scale references under the same evaluation protocol (Appendix D.5).

4.4 FURTHER ANALYSES

We also examine data-mixture robustness (Appendix D.3), reweighting across optimizers and fixed-weight baselines (Appendix D.4), and scaling across model sizes at their corresponding Chinchilla budgets (Appendix E).

5 CONCLUSION

Multi-domain pretraining has two controls that standard mixture design conflates: domain exposure and optimization scale. At fixed exposure, loss reweighting produces endpoints beyond the observed stage schedules and expands the observed remix Pareto frontier. Our analysis attributes this separation to domain-specific effective learning rates, which can reach loss trade-offs unavailable to a shared-rate mixture. We therefore introduce LARE, an online controller that converts running domain losses into normalized inverse-square-root weights while preserving the sampled mixture and fixing the mixture-averaged coefficient at one. LARE selects a competitive scale in one run without a coefficient sweep. Experiments cover base optimizers, mixture ratios, domain compositions, and 30B–240B token budgets. At $8\times$ Chinchilla, the 1.5B model improves all five evaluated benchmarks by 7.84 points on average over matched direct mixing, and every gain exceeds its $4\times$ counterpart. These results support treating data allocation and loss scaling as separate pretraining design variables: exposure determines what the model sees, whereas reweighting determines how strongly each domain shapes learning.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, retrieve and discover related work, code development, and assist in proving mathematical claims. All AI-generated or AI-assisted material was subsequently checked by the authors before inclusion. In particular, the authors manually reviewed and validated all code, verified every cited reference against its original source, reviewed every AI-polished passage, and independently checked every mathematical claim and proof. The authors take full responsibility for the content of this paper.

ETHICS STATEMENT

This work studies domain-level loss reweighting for language model pretraining. Models trained with this approach may inherit biases, harmful content, and privacy risks from their training data. Reweighting domain losses may also change the influence of these properties, which is not evaluated in this study. Our reported improvements concern downstream task performance and should not be interpreted as evidence of improved safety or fairness. Models developed using this approach require appropriate safety evaluation before deployment.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide the code, dataset and model in the supplementary materials and anonymous github `https://anonymous.4open.science/r/Fast-LLM-Pretrain-on-Data-Mixture-by-Loss-Reweighting-F7F5`. In the README.md file included with the code, we present a step-by-step guide for reproducing our results.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Lior Belenki, Alekh Agarwal, Tianze Shi, and Kristina Toutanova. Optimizing pre-training data mixtures with mixtures of data expert models, 2025. URL https://arxiv.org/abs/2502.15950.

Huanran Chen, Huaqing Zhang, Xiao Li, Yinpeng Dong, Ke Shen, and Jun Zhu. Nexus: Same pretraining loss, better downstream generalization via common minima, 2026a. URL https://arxiv.org/abs/2604.09258.

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

Mayee F. 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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Mayee F Chen, Tyler Murray, David Heineman, Matt Jordan, Hannaneh Hajishirzi, Christopher Re, Luca Soldaini, and Kyle Lo. Olmix: A framework for data mixing throughout LM development. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=8pOl3azhbL.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018. URL https://openreview.net/forum?id=H1bM1fZCW.

Zhuo Chen, Yuxuan Miao, and Deyi Xiong. Data mixing for large language models pretraining: A survey and outlook. *Data Intelligence*, 8:15–55, 03 2026c. doi: 10.3724/2096-7004.di.2026.0055.

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.

Andrea Cossu, Tinne Tuytelaars, Antonio Carta, Lucia Passaro, Vincenzo Lomonaco, and Davide Bacciu. Continual pre-training mitigates forgetting in language and vision, 2022. URL https://arxiv.org/abs/2205.09357.

Chenhao Dang, Jing Ma, and Mingjie Liao. Holistic data scheduler for llm pre-training via multi-objective reinforcement learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pp. 176–187, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722585. doi: 10.1145/3770854.3780325. URL https://doi.org/10.1145/3770854.3780325.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: domain reweighting with generalization estimation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Simin Fan, David Grangier, and Pierre Ablin. Dynamic gradient alignment for online data mixing, 2025. URL https://openreview.net/forum?id=O3SatrdL97.

Kazuki Fujii, Yukito Tajima, Sakae Mizuki, Masaki Kawamura, Hinari Shimada, Taihei Shiotani, Koshiro Saito, Masanari Oi, Taishi Nakamura, Takumi Okamoto, Shigeki Ishida, Kakeru Hattori, Youmi Ma, Hiroya Takamura, Rio Yokota, Jun Sakuma, and Naoaki Okazaki. Rewriting pre-training data boosts LLM performance in math and code. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=45btPYgSSX.

Xinran Gu, Kaifeng Lyu, Jiazheng Li, and Jingzhao Zhang. Data mixing can induce phase transitions in knowledge acquisition. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tQZK5frjVU.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740/.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

HMMT. Harvard–mit mathematics tournament 2025. February 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, Linzheng Chai, Ruifeng Yuan, Xianzhen Luo, Qiufeng Wang, YuanTao Fan, Qingfu Zhu, Zhaoxiang Zhang, Yang Gao, Jie Fu, Qian Liu, Houyi Li, Ge Zhang, Yuan Qi, Xu Yinghui, Wei Chu, and Zili Wang. OpenCoder: The open cookbook for top-tier code large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33167–33193, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1591. URL https://aclanthology.org/2025.acl-long.1591/.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. Autoscale: Scale-aware data mixing for pre-training LLMs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=rujwIvjooA.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual pre-training of language models, 2023. URL https://arxiv.org/abs/2302.03241.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018. URL https://arxiv.org/abs/1705.07115.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

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.

Jingwei Li, Xinran Gu, Rui Dai, Xintong Hao, Chengyin Xu, Yan Wu, Shuran Zheng, and Jingzhao Zhang. Scaling domain data repetition in llm pretraining, 2026a. URL https://arxiv.org/abs/2608.14071.

Jingwei Li, Xinran Gu, and Jingzhao Zhang. Capacity-aware mixture law enables efficient LLM data optimization. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=DDtL4VOUcT.

Shengrui Li, Fei zhao, Kaiyan Zhao, Jieying Ye, Haifeng Liu, Fangcheng Shi, Zheyong Xie, Yao Hu, and Shaosheng Cao. Decouple searching from training: Scaling data mixing via model merging for large language model pre-training. In *Forty-third International Conference on Machine Learning*, 2026c. URL https://openreview.net/forum?id=uyRIOjFgOn.

Hao Liang, Zhengyang Zhao, Meiyi Qiang, Mingrui Chen, Lu Ma, Rongyi Yu, Hengyi Feng, Shixuan Sun, Zimo Meng, Xiaochen Ma, Xuanlin Yang, Qifeng Cai, Ruichuan An, Bohan Zeng, Zhen Hao Wong, Chengyu Shen, Runming He, Zhaoyang Han, Yaowei Zheng, Fangcheng Fu, Conghui He, Bin Cui, Zhiyu Li, Weinan E, and Wentao Zhang. Dataflex: A unified framework for data-centric dynamic training of large language models, 2026. URL https://arxiv.org/abs/2603.26164.

Emmy Liu, Graham Neubig, and Chenyan Xiong. Midtraining bridges pretraining and posttraining distributions, 2026a. URL https://openreview.net/forum?id=u7L9FOgG7t.

Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew Chi-Chih Yao. Augmenting math word problems via iterative question composing, 2024. URL https://arxiv.org/abs/2401.09003.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025a. URL https://arxiv.org/abs/2502.16982.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=5BjQOUXq7i.

Yajiao Liu, Congliang Chen, Junchi Yang, and Ruoyu Sun. Rethinking data mixture for large language models: A comprehensive survey and new perspectives. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 275–289, Rabat, Morocco, March 2026b. Association for Computational Linguistics. ISBN 979-8-89176-386-9. doi: 10.18653/v1/2026.findings-eacl.15. URL https://aclanthology.org/2026.findings-eacl.15/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathcoder2: Better math reasoning from continued pretraining on model-translated mathematical code, 2024. URL https://arxiv.org/abs/2410.08196.

Jing Ma, Chenhao Dang, and Mingjie Liao. AC-ODM: Actor–critic online data mixing for sample-efficient LLM pretraining, 2026. URL https://openreview.net/forum?id=mRjzWksbGR.

MAA. American mathematics competitions 2023. November 2023.

MAA. American invitational mathematics examination 2024. February 2024.

MAA. American invitational mathematics examination 2025. February 2025.

Martin Marek, Dongkyu Cho, Shikai Qiu, Rumi Chunara, Pavel Izmailov, and Andrew Gordon Wilson. Forgetting in language models: Capacity, optimization, and self-generated replay, 2026. URL https://arxiv.org/abs/2605.26097.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024. URL https://arxiv.org/abs/2402.14830.

Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/b3310bba2be31e673a7ded3386994599-Paper.pdf.

NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, Andrew Tao, Ann Guan, Anna Shors, Anubhav Mandarwal, Arham Mehta, Arun Venkatesan, Ashton Sharabiani, Ashwath Aithal, Ashwin Poojary, Ayush Dattagupta, Balaram Buddharaju, Banghua Zhu, Barnaby Simkin, Bilal Kartal, Bita Darvish Rouhani, Bobby Chen, Boris Ginsburg, Brandon Norick, Brian Yu, Bryan Catanzaro, Charles Wang, Charlie Truong, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christian Munley, Christopher Parisien, Dan Su, Daniel Afrimi, Daniel Korzekwa, Daniel Rohrer, Daria Gitman, David Mosallanezhad, Deepak Narayanan, Dima Rekesh, Dina Yared, Dmytro Pykhtar, Dong Ahn, Duncan Riach, Eileen Long, Elliott Ning, Eric Chung, Erick Galinkin, Evelina Bakhturina, Gargi Prasad, Gerald Shen, Haifeng Qian, Haim Elisha, Harsh Sharma, Hayley Ross, Helen Ngo, Herman Sahota, Hexin Wang, Hoo Chang Shin, Hua Huang, Iain Cunningham, Igor Gitman, Ivan Moshkov, Jaehun Jung, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jian Zhang, Jiaqi Zeng, Jimmy Zhang, Jinze Xue, Jocelyn Huang, Joey Conway, John Kamalu, Jonathan Cohen, Joseph Jennings, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kezhi Kong, Krzysztof Pawelec, Kumar Anik, Kunlun Li, Kushan Ahmadian, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Luis Vega, Maer Rodrigues de Melo, Makesh Narsimhan Sreedhar, Marcin Chochowski, Mark Cai, Markus Kliegl, Marta Stepniewska-Dziubinska, Matvei Novikov, Mehrzad Samadi, Meredith Price, Meriem Boubdir, Michael Boone, Michael Evans, Michal Bien, Michal Zawalski, Miguel Martinez, Mike Chrzanowski, Mohammad Shoeybi, Mostofa Patwary, Namit Dhameja, Nave Assaf, Negar Habibi, Nidhi Bhatia, Nikki Pope, Nima Tajbakhsh, Nirmal Kumar Juluru, Oleg Rybakov, Oleksii Hrinchuk, Oleksii Kuchaiev, Oluwatobi Olabiyi, Pablo Ribalta, Padmavathy Subramanian, Parth Chadha, Pavlo Molchanov, Peter Dykas, Peter Jin, Piotr Bialecki, Piotr Januszewski, Pradeep Thalasta, Prashant Gaikwad, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Rabeeh Karimi Mahabadi, Rajen Patel, Ran El-Yaniv, Ranjit Rajan, Ria Cheruvu, Rima Shahbazyan, Ritika Borkar, Ritu Gala, Roger Waleffe, Ruoxi Zhang, Russell J. Hewett, Ryan Prenger, Sahil Jain, Samuel Kriman, Sanjeev Satheesh, Saori Kaji, Sarah Yurick, Saurav Muralidharan, Sean Narenthiran, Seonmyeong Bak, Sepehr Sameni, Seungju Han, Shanmugam Ramasamy, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shizhe Diao, Shreya Gopal, Shrimai Prabhumoye, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Siddhartha Jain, Somshubra Majumdar, Soumye Singhal, Stefania Alborghetti, Syeda Nahida Akter, Terry Kong, Tim Moon, Tomasz Hliwiak, Tomer Asida, Tony Wang, Tugrul Konuk, Twinkle Vashishth, Tyler Poon, Udi Karpas, Vahid Noroozi, Venkat Srinivasan, Vijay Korthikanti, Vikram Fugro, Vineeth Kalluru, Vitaly Kurin, Vitaly Lavrukhin, Wasi Uddin Ahmad, Wei Du, Wonmin Byeon, Ximing Lu, Xin Dong, Yashaswi Karnati, Yejin Choi, Yian Zhang, Ying Lin, Yonggan Fu, Yoshi Suhara, Zhen Dong, Zhiyu Li, Zhongbo Zhu, and Zijia Chen. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Jiahui Peng, Xinlin Zhuang, Jiantao Qiu, Ren Ma, Jing Yu, He Zhu, and Conghui He. Topic over source: The key to effective data mixing for language models pre-training, 2025. URL https://arxiv.org/abs/2502.16802.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Mahdi Salmani, Pratik Worah, Meisam Razaviyayn, and Vahab Mirrokni. Sampling and loss weights in multi-domain training, 2025. URL SamplingandLossWeightsinMulti-DomainTraining.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 525–536, Red Hook, NY, USA, 2018. Curran Associates Inc.

Daouda Sow, Herbert Woisetschläger, Saikiran Bulusu, Shiqiang Wang, Hans Arno Jacobsen, and Yingbin Liang. Dynamic loss-based sample reweighting for improved large language model pretraining. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=gU4ZgQNsOC.

Haoru Tan, Sitong Wu, Yanfeng Chen, Jun Xia, Ruobing Xie, Bin Xia, Xingwu Sun, and Xiaojuan Qi. Fast data mixture optimization via gradient descent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5gFKVyohGd.

TIGER-Lab. MATH-plus. 2024. URL https://huggingface.co/datasets/TIGER-Lab/MATH-plus.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data, 2024. URL https://arxiv.org/abs/2410.01560.

Jiapeng Wang, Changxin Tian, Kunlong Chen, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. Mergemix: Optimizing mid-training data mixtures via learnable model merging, 2026. URL https://arxiv.org/abs/2601.17858.

Kaiyue Wen, David Leo Wright Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2J51qUZ0iG.

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 *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=lXuByUeHhd.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=mDxarRaTY9.

Jing Xu, Jiazheng Li, and Jingzhao Zhang. Scalable model merging with progressive layer-wise distillation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=xX8NJShgny.

Yuanjian Xu, Tianze Sun, Changwei Xu, XinLong Zhao, Jianing Hao, Ran Chen, Yang Liu, Ruijie Xu, Stephen Chen, and Guang Zhang. Rethinking data mixing from the perspective of large language models, 2026. URL https://arxiv.org/abs/2604.07963.

Kailai Yang, Xiao Liu, Lei Ji, Hao Li, Xiao Liang, Zhiwei Liu, Yeyun Gong, Peng Cheng, and Mao Yang. Data mixing agent: Learning to re-weight domains for continual pre-training. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9447–9473, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.427. URL https://aclanthology.org/2026.acl-long.427/.

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. URL https://openreview.net/forum?id=jjCB27TMK3.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James 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*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning, 2023. URL https://arxiv.org/abs/2309.05653.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web, 2024. URL https://arxiv.org/abs/2405.03548.

Çağatay Yıldız, Nishaanth Kanna Ravichandran, Nitin Sharma, Matthias Bethge, and Beyza Ermis. Investigating continual pretraining in large language models: Insights and implications, 2025. URL https://arxiv.org/abs/2402.17400.

A Related Work

Multi-stage pretraining. Domain-adaptive pretraining and modern midtraining recipes introduce specialized data in later stages to strengthen target capabilities (Gururangan et al., 2020; OLMo et al., 2025; Liu et al., 2026a; Lu et al., 2024). However, this temporal specialization raises a fundamental retention question: studies of continual pretraining report substantial forgetting in some regimes, while showing that self-supervised continuation can preserve prior knowledge more robustly than supervised adaptation in others (Ke et al., 2023; Çağatay Yıldız et al., 2025; Cossu et al., 2022; Marek et al., 2026). These results indicate that forgetting is regime-dependent rather than an inevitable consequence of continued training.

Data-mixture ratio selection. The dominant approach to multi-domain data mixing optimizes the sampling ratio itself (Liu et al., 2026b; Chen et al., 2026c). Offline methods such as DoReMi and DoGE infer domain proportions from proxy losses or generalization estimates, while MDE evaluates candidate mixtures with domain-expert models (Xie et al., 2023; Fan et al., 2024; Belenki et al., 2025). Online methods instead adapt sampling during training: Skill-it! builds skill curricula, DGA uses target-gradient alignment, Data Mixing Agent learns a transferable policy, and AC-ODM and HDS optimize schedules from domain interactions or multiple training signals (Chen et al., 2023; Fan et al., 2025; Yang et al., 2026; Ma et al., 2026; Dang et al., 2026). Further work refines the control space by reconstructing domains from topics, estimating importance in a representation space, modeling domain interactions, or unifying multiple data operations (Peng et al., 2025; Xie et al., 2025; Xu et al., 2026; Liang et al., 2026). All of these works change *what* the model sees. Loss reweighting offers a complementary lever: it changes the contribution of already-sampled data to the gradient, with precedents in task-level balancing and example-level language-model training (Kendall et al., 2018; Chen et al., 2018; Sener & Koltun, 2018; Sow et al., 2025). Importantly, sampling and loss coefficients can interact through gradient variance and generalization, and may become coupled under importance correction or particular bilevel formulations (Salmani et al., 2025; Needell et al., 2014; Tan et al., 2026).

Scaling laws for data mixtures. A ratio selected in one configuration need not remain optimal as model size, training budget, or domain reuse changes. RegMix and Data Mixing Laws fit performance-response models from smaller proxy runs, and recent studies explicitly characterize how useful allocations shift with scale, model capacity, and repeated use of limited domain data (Liu et al., 2025b; Ye et al., 2025; Gu et al., 2025; Kang et al., 2025; Li et al., 2026b;a). Related systems seek lower-cost transfer across configurations: Olmix reuses mixture knowledge as the available domain set evolves, whereas DeMix and MergeMix use separately trained domain models or model merging as proxies for mixture quality (Chen et al., 2026b; Li et al., 2026c; Wang et al., 2026). This body of work underscores why a single sampling ratio is difficult to transfer across scales.

B THEORY

B.1 PROOF OF THE LOSS–GRADIENT-NOISE SCALING

We prove Theorem 1 and make explicit the assumptions behind its use as motivation for LARE. Let $x \sim \mathcal{N}(0, I_d)$, fix a unit vector u that defines the ground-truth decision boundary, and set the ground-truth label to $y_0 = \operatorname{sign}(u^\top x)$. Independently flip y_0 with probability $\rho \in (0, 1/2)$ to obtain the observed label y. We optimize the scale of the logistic classifier $w = ru$, $r \geq 0$, and write $Z = u^\top x \sim \mathcal{N}(0, 1)$ and $T = |Z|$. The population logistic risk is exactly

$$L_\rho(r) = (1-\rho)\mathbb{E}\log(1+e^{-rT}) + \rho\mathbb{E}\log(1+e^{rT}) = \mathbb{E}\log(1+e^{-rT}) + \rho r\mathbb{E}T. \tag{6}$$

The second equality uses $\log(1+e^a) = a + \log(1+e^{-a})$. The risk is strictly convex in r and, for $\rho < 1/2$, its unique minimizer $r^\star_\rho$ satisfies

$$\mathbb{E}\left[\frac{T}{1+e^{r^\star_\rho T}}\right] = \rho\mathbb{E}T. \tag{7}$$

As $\rho \downarrow 0$, this identity forces $r^\star_\rho \to \infty$. With ϕ denoting the standard-normal density, the substitution $s = rT$ and dominated convergence give

$$\mathbb{E}\left[\frac{T}{1+e^{rT}}\right] = \frac{2}{r^2}\int_0^\infty \frac{s\phi(s/r)}{1+e^s}\,ds = \frac{\pi^2}{6\sqrt{2\pi}}\frac{1+o(1)}{r^2}, \tag{8}$$

$$\mathbb{E}\log(1+e^{-rT}) = \frac{2}{r}\int_0^\infty \phi(s/r)\log(1+e^{-s})\,ds = \frac{\pi^2}{6\sqrt{2\pi}}\frac{1+o(1)}{r}. \tag{9}$$

Here we used $\int_0^\infty s/(1+e^s)\,ds = \int_0^\infty \log(1+e^{-s})\,ds = \pi^2/12$ and $\mathbb{E}T = \sqrt{2/\pi}$. Substituting Equation 8 into Equation 7, and then using Equation 9 in Equation 6, yields

$$r^\star_\rho = \frac{\pi}{\sqrt{12\rho}}\big(1+o(1)\big), \qquad \ell^\star_\rho = L_\rho(r^\star_\rho) = \sqrt{\frac{2\pi}{3}}\sqrt{\rho} + o(\sqrt{\rho}). \tag{10}$$

We next compute the covariance of the stochastic gradient with respect to the full parameter vector $w \in \mathbb{R}^d$. For one example,

$$g(w; x, y) = \nabla_w \log(1+e^{-yw^\top x}) = -\frac{yx}{1+e^{yw^\top x}}. \tag{11}$$

Decompose $x = Zu + x_\perp$, where $x_\perp \sim \mathcal{N}(0, I_d - uu^\top)$ is independent of Z and of the label flip. At $w = ru$, the scalar multiplying x in the gradient depends only on Z and the label flip, so the orthogonal component has zero mean. Averaging the parallel component over the label flip gives

$$\mathbb{E}g(ru; x, y) = u\left(\rho\mathbb{E}T - \mathbb{E}\left[\frac{T}{1+e^{rT}}\right]\right). \tag{12}$$

Equation 7 therefore implies $\mathbb{E}g(w^\star_\rho; x, y) = 0$ at $w^\star_\rho = r^\star_\rho u$. By convexity of the population logistic risk in w, this point is also a minimizer over $\mathbb{R}^d$. Its gradient covariance thus equals its second moment. Conditional on T, the label-averaged squared scalar multiplying $xx^\top$ at $w = ru$ is

$$a_{\rho,r}(T) = \frac{1-\rho}{(1+e^{rT})^2} + \frac{\rho}{(1+e^{-rT})^2}. \tag{13}$$

Independence and rotational symmetry then give

$$\Sigma_\rho = \alpha_\parallel uu^\top + \alpha_\perp(I_d - uu^\top), \qquad \alpha_\parallel = \mathbb{E}[T^2 a_{\rho,r^\star_\rho}(T)], \quad \alpha_\perp = \mathbb{E}[a_{\rho,r^\star_\rho}(T)]. \tag{14}$$

For the clean-label term in $\alpha_\perp$, the same substitution $s = rT$ gives

$$\mathbb{E}\frac{1}{(1+e^{rT})^2} = \frac{2\phi(0)}{r}\int_0^\infty \frac{1}{(1+e^s)^2}\,ds + o(r^{-1}) = \Theta(r^{-1}), \tag{15}$$

where the integral equals $\log 2 - 1/2 > 0$. The flipped-label term converges to ρ, because $(1+e^{-rT})^{-2} \to 1$ almost surely and is bounded by one. Hence $\alpha_\perp = \Theta(r^{-1} + \rho) = \Theta(\sqrt{\rho})$. Along the ground-truth direction, the additional factor T^2 changes the unflipped-label term to $\Theta(r^{-3})$, whereas dominated convergence makes the flipped-label term asymptotic to $\rho\mathbb{E}T^2 = \rho$. Therefore

$$\alpha_\perp = \Theta(\sqrt{\rho}) = \Theta(\ell^\star_\rho), \qquad \alpha_\parallel = \Theta(\rho) = \Theta\big((\ell^\star_\rho)^2\big). \tag{16}$$

Equation 3 follows immediately. Moreover, $\|\Sigma_\rho\|_{\mathrm{op}} = \Theta(\ell^\star_\rho)$ and $\operatorname{tr}(\Sigma_\rho) = (d-1)\alpha_\perp + \alpha_\parallel = \Theta((d-1)\ell^\star_\rho + (\ell^\star_\rho)^2)$. Since the population gradient vanishes at the optimum, $\mathbb{E}\|g(w^\star_\rho; x, y)\|^2 = \operatorname{tr}(\Sigma_\rho)$, which establishes the stated root-mean-square scaling. □

Algorithm 2 LARE: online domain-level loss-adaptive reweighting under fixed exposure.

Require: Domains $\mathcal{D}$; optimizer $\mathcal{O}$; EMA decay β; numerical constant ϵ; learning-rate warmup length T_{w}

$\hat{\ell}_{i,0} \leftarrow 0$, $\hat{c}_{i,0} \leftarrow 0$, $w_{i,1} \leftarrow 1$, $\mathsf{seen}_i \leftarrow \mathsf{false}$ for all $i \in \mathcal{D}$
for optimizer steps $t = 1, \ldots, T$ **do**
  Draw a mixed batch; let $\mathcal{V}_t$ be its loss-bearing target positions with target-aligned domain IDs
  $q_{k,t} \leftarrow \mathrm{CE}_{\theta_t}(y_k \mid x_{\le k})$ for every $k \in \mathcal{V}_t$
  $\mathcal{L}_t \leftarrow |\mathcal{V}_t|^{-1} \sum_{k \in \mathcal{V}_t} w_{d_k,t} q_{k,t}$
  $\theta_{t+1} \leftarrow \textsc{OptimizerStep}(\mathcal{O}, \theta_t, \nabla_{\theta_t} \mathcal{L}_t)$
  Compute $S_{i,t} = \sum_{k \in \mathcal{V}_t} \mathbf{1}[d_k = i] q_{k,t}$ and $N_{i,t} = \sum_{k \in \mathcal{V}_t} \mathbf{1}[d_k = i]$ for all i
  for each domain $i \in \mathcal{D}$ **do**
    $\hat{c}_{i,t} \leftarrow \beta \hat{c}_{i,t-1} + (1-\beta) N_{i,t}$
    if $N_{i,t} > 0$ **then**
      $\bar{\ell}_{i,t} \leftarrow S_{i,t} / N_{i,t}$
      if $\mathsf{seen}_i = \mathsf{false}$ **then**
        $\hat{\ell}_{i,t} \leftarrow \bar{\ell}_{i,t}$; $\mathsf{seen}_i \leftarrow \mathsf{true}$
      else
        $\hat{\ell}_{i,t} \leftarrow \beta \hat{\ell}_{i,t-1} + (1-\beta) \bar{\ell}_{i,t}$
      end if
    else
      $\hat{\ell}_{i,t} \leftarrow \hat{\ell}_{i,t-1}$
    end if
  end for
  if $t < T_{\mathrm{w}}$ **then**
    $w_{i,t+1} \leftarrow 1$ for all $i \in \mathcal{D}$
  else
    $\hat{\pi}_{i,t} \leftarrow \hat{c}_{i,t} / \sum_{j \in \mathcal{D}} \hat{c}_{j,t}$ for all $i \in \mathcal{D}$
    $r_{i,t} \leftarrow (\hat{\ell}_{i,t} + \epsilon)^{-1/2}$ for each seen domain; $r_{i,t} \leftarrow 0$ otherwise
    $Z_t \leftarrow \sum_{j \in \mathcal{D}} \hat{\pi}_{j,t} r_{j,t}$
    $w_{i,t+1} \leftarrow r_{i,t} / Z_t$ for each seen domain; $w_{i,t+1} \leftarrow 1$ otherwise
  end if
end for

Scope of the argument. The low-loss limit relies on optimizing the classifier scale r. If $\|w\|$ were fixed to one, even perfectly separable Gaussian halfspace data would retain the strictly positive logistic loss $\mathbb{E}\log(1+e^{-|Z|})$, so no $\ell_\rho^\star \to 0$ regime would exist. The theorem is therefore a mechanism result for unconstrained scale optimization, not a claim about unit-normalized classifiers. It also distinguishes the gradient-noise variance, which is linear in loss in the dominant orthogonal subspace, from its standard deviation, which is proportional to the square root of loss. Multiplying a domain loss by w_i scales the latter by w_i, motivating $w_i \propto (\ell_i^\star)^{-1/2}$ when the effective dimensions and hidden constants are comparable. LARE replaces the unavailable optimum $\ell_i^\star$ by an online smoothed loss; this substitution is a practical modeling choice rather than a consequence of the asymptotic theorem.

B.2 Proof of Proposition 1

Model and SGD update. Let $A_1, A_2 \in \mathbb{R}^{d \times d}$ be nonzero symmetric positive semidefinite matrices with orthogonal supports, so $A_1 A_2 = 0$. In domain $j \in \{1, 2\}$, samples follow $x_j \sim \mathcal{N}(0, A_j)$ and $y_j = x_j^\top w^\star$. A shared $w^\star$ is without loss of generality: ground-truth parameters that differ between domains can be projected onto their respective orthogonal supports and summed. The population MSE loss is

$$f_j(w) = \tfrac{1}{2}\,\mathbb{E}_{x_j}\big[(x_j^\top w - y_j)^2\big] = \tfrac{1}{2}\,(w - w^\star)^\top A_j (w - w^\star).$$

Fix an initialization w_0 with $f_1(w_0), f_2(w_0) > 0$, a batch size $B \ge 2$, and a horizon $T \ge 1$. At each step t, the batch contains exactly $Bp_j \in \mathbb{N}^+$ fresh samples from domain j, where $p_j \in (0, 1)$ and

$p_1 + p_2 = 1$. Samples are independent across domains, examples, and iterations. With per-domain learning rates $\eta_1, \eta_2 > 0$, SGD updates

$$w_{t+1} = w_t - \frac{1}{B}\sum_{j=1}^{2}\eta_j \sum_{i=1}^{Bp_j} x_{j,i,t}\big(x_{j,i,t}^\top w_t - y_{j,i,t}\big).$$

We write $L_j(p_j, \eta_j) = \mathbb{E}[f_j(w_T)]$ for the expected terminal loss of domain j.

The ranges of A_1 and A_2 are orthogonal since $A_1 A_2 = 0$. A domain-j sample $x_j \sim \mathcal{N}(0, A_j)$ lies in the range of A_j almost surely, so its stochastic gradient $x_j(x_j^\top w_t - y_j)$ does too. A domain-j update therefore changes only the components of the error $w_t - w^\star$ that lie in the range of A_j. Consequently, the two domains never interact, and the terminal loss L_j depends only on that domain's own mixture proportion p_j and learning rate η_j. We therefore prove the single-domain Lemmas 1 and 2 with the domain index dropped from the notation, and apply them to $j = 1, 2$ in the proof of the proposition.

Single-domain notation. Fix one domain and drop its index. Let $r = \mathrm{rank}(A_j)$ be the dimension of its positive support, and diagonalize the covariance restricted to that support as

$$D = \mathrm{diag}(\lambda_1, \ldots, \lambda_r), \qquad \lambda = (\lambda_1, \ldots, \lambda_r)^\top > 0, \qquad z_{t,k} = \mathbb{E}[e_{t,k}^2], \tag{17}$$

where $\lambda_1, \ldots, \lambda_r > 0$ are the positive eigenvalues, $e_t = w_t - w^\star$ is the error, $e_{t,k}$ (for $k = 1, \ldots, r$) is its coordinate along the k-th eigenvector, and $z_t = (z_{t,1}, \ldots, z_{t,r})^\top$ collects the per-coordinate second moments. Since $\mathbb{E}[f(w_t)] = \frac{1}{2}\mathbb{E}[e_t^\top D e_t] = \frac{1}{2}\lambda^\top z_t$, positive initial domain loss is exactly $\lambda^\top z_0 > 0$.

Lemma 1. *Suppose the domain receives a fraction $p \in (0, 1)$ of each batch (that is, Bp of the B samples) and learning rate $\eta \geq 0$. Then the second-moment vector z_t and the expected terminal loss satisfy*

$$z_{t+1} = K(p, \eta)\, z_t,$$

$$K(p, \eta) = (I - p\eta D)^2 + \frac{p\eta^2}{B}\big(D^2 + \lambda\lambda^\top\big), \tag{18}$$

$$L(p, \eta) = \tfrac{1}{2}\,\lambda^\top [K(p, \eta)]^T z_0. \tag{19}$$

Proof. Write $n = Bp$ for the number of domain samples in a batch. SGD updates

$$e_{t+1} = e_t - \frac{\eta}{B}\sum_{i=1}^{n} x_i\big(x_i^\top w_t - y_i\big) = \Big(I - \frac{\eta}{B}\sum_{i=1}^{n} x_i x_i^\top\Big)e_t, \qquad x_i \sim \mathcal{N}(0, D) \text{ i.i.d.} \tag{20}$$

Condition on $e_t = e$ and set $g_i = x_i(x_i^\top e)$. For a mean-zero Gaussian with diagonal covariance D, Isserlis' theorem gives $\mathbb{E}[x_k^2 x_\ell x_m] = \lambda_k\lambda_\ell\,\delta_{\ell m} + 2\lambda_k^2\,\delta_{k\ell}\delta_{km}$, whence, coordinatewise,

$$\mathbb{E}[g_{i,k}] = \lambda_k e_k, \qquad \mathrm{Var}(g_{i,k}) = \mathbb{E}[g_{i,k}^2] - (\mathbb{E}[g_{i,k}])^2 = \lambda_k^2 e_k^2 + \lambda_k \sum_{\ell=1}^{r}\lambda_\ell e_\ell^2. \tag{21}$$

Coordinate k of equation 20 is $e_{t+1,k} = e_k - \frac{\eta}{B}S_k$ with $S_k = \sum_{i=1}^n g_{i,k}$. Its first two moments are $\mathbb{E}[S_k] = n\lambda_k e_k$ and $\mathbb{E}[S_k^2] = \mathrm{Var}(S_k) + (\mathbb{E}[S_k])^2 = n\mathrm{Var}(g_{1,k}) + n^2\lambda_k^2 e_k^2$, so, expanding the square and using $n/B = p$,

$$\begin{aligned}
\mathbb{E}[e_{t+1,k}^2 \mid e_t = e] &= e_k^2 - \frac{2\eta}{B}\,e_k\,\mathbb{E}[S_k] + \frac{\eta^2}{B^2}\,\mathbb{E}[S_k^2] \\
&= e_k^2 - 2p\eta\lambda_k e_k^2 + p^2\eta^2\lambda_k^2 e_k^2 + \frac{p\eta^2}{B}\mathrm{Var}(g_{1,k}) \\
&= (1 - p\eta\lambda_k)^2 e_k^2 + \frac{p\eta^2}{B}\Big(\lambda_k^2 e_k^2 + \lambda_k\sum_{\ell}\lambda_\ell e_\ell^2\Big).
\end{aligned} \tag{22}$$

Taking the expectation over e_t yields

$$z_{t+1} = K(p, \eta) z_t. \tag{23}$$

Iterating gives $z_T = [K(p, \eta)]^T z_0$, and since $f(w_T) = \frac{1}{2}e_T^\top D e_T = \frac{1}{2}\sum_k \lambda_k e_{T,k}^2$,

$$L(p, \eta) = \mathbb{E}[f(w_T)] = \tfrac{1}{2}\lambda^\top z_T = \tfrac{1}{2}\lambda^\top [K(p, \eta)]^T z_0,$$

which is equation 19. □

Lemma 2. *For fixed B, T, let $V(p) = \min_{\eta \geq 0} L(p, \eta)$. The minimum is attained, every minimizing rate is positive, and $V(p)$ is strictly decreasing in $p \in (0, 1)$.*

Proof. All matrix inequalities in this proof are entrywise. We read $L(p, \eta)$ off the closed form equation 19 for every $p \in (0, 1)$; at a feasible mixture ($Bp \in \mathbb{N}$) it is the SGD loss of Lemma 1. Set the effective mean step $a = p\eta$ and define

$$\widehat{K}_p(a) = (I - aD)^2 + \frac{a^2}{Bp}\big(D^2 + \lambda\lambda^\top\big), \qquad F_p(a) = \tfrac{1}{2}\,\lambda^\top [\widehat{K}_p(a)]^T z_0, \tag{24}$$

so that $F_p(p\eta) = L(p, \eta)$; since $p > 0$, minimizing F_p over $a \geq 0$ is equivalent to minimizing $L(p, \cdot)$ over $\eta \geq 0$. The map F_p is a polynomial, hence continuous, and its derivative at zero is

$$F_p'(0) = -T \sum_{k=1}^{r} \lambda_k^2 z_{0,k} < 0. \tag{25}$$

It is also coercive: $\widehat{K}_p(a) \geq (a^2/(Bp))\,\lambda\lambda^\top$, and

$$F_p(a) \geq \tfrac{1}{2}\lambda^\top z_0 \Big(\frac{a^2 \lambda^\top \lambda}{Bp}\Big)^T \longrightarrow \infty \qquad \text{as } a \to \infty. \tag{26}$$

Continuity and coercivity give a nonempty compact set of minimizers, while equation 25 excludes $a = 0$. So every minimizing rate is positive.

Now take $0 < p < \tilde{p} < 1$ and fix any $a > 0$. With $U = \widehat{K}_p(a)$ and $W = \widehat{K}_{\tilde{p}}(a)$,

$$U - W = \frac{a^2}{B}\Big(\frac{1}{p} - \frac{1}{\tilde{p}}\Big)\big(D^2 + \lambda\lambda^\top\big) > 0, \tag{27}$$

and both U, W have strictly positive entries. The noncommutative telescoping identity

$$U^T - W^T = \sum_{k=0}^{T-1} U^{\,T-1-k}(U - W)\, W^k > 0 \tag{28}$$

then gives $F_{\tilde{p}}(a) < F_p(a)$. Taking any minimizer $a_\star > 0$ of F_p,

$$V(\tilde{p}) \leq F_{\tilde{p}}(a_\star) < F_p(a_\star) = V(p), \tag{29}$$

which proves strict decrease. □

Proof of Proposition 1. By the reduction above, $L_j(p_j, \eta_j)$ is the single-domain loss of Lemma 1 for the domain-j spectrum, and $V_j(p) = \min_{\eta \geq 0} L_j(p, \eta)$ its optimized value. By Lemma 2 each V_j is attained at a positive rate, so the reweighting rates $\eta_j^\star$, which minimize over $\eta_j > 0$, achieve it:

$$L_1(p_1, \eta_1^\star) = V_1(p_1), \qquad L_2(p_2, \eta_2^\star) = V_2(p_2). \tag{30}$$

Consider any remix with mixture $(p_1', p_2') \neq (p_1, p_2)$ and shared rate η'; since $p_1' + p_2' = 1 = p_1 + p_2$, this is equivalent to $p_1' \neq p_1$. Because $V_j(p_j')$ is the minimum of $L_j(p_j', \cdot)$,

$$L_1(p_1', \eta') \geq V_1(p_1'), \qquad L_2(p_2', \eta') \geq V_2(p_2'). \tag{31}$$

If $p_1' < p_1$, strict monotonicity of V_1 (Lemma 2) with equation 31 gives

$$L_1(p_1', \eta') \geq V_1(p_1') > V_1(p_1) = L_1(p_1, \eta_1^\star). \tag{32}$$

If instead $p_1' > p_1$, the same argument applied to domain 2 gives

$$L_2(p_2', \eta') \geq V_2(p_2') > V_2(p_2) = L_2(p_2, \eta_2^\star). \tag{33}$$

In either case one coordinate of the remix strictly exceeds the corresponding coordinate of the reweighting point. This is exactly the separation equation 2, and it shows that no remix Pareto-dominates $\big(L_1(p_1, \eta_1^\star), L_2(p_2, \eta_2^\star)\big)$. □

B.3 NUMERICAL INSTANTIATION OF FIGURE 3

Figure 3 evaluates the two-domain quadratic model of Proposition 1 on a fixed, non-degenerate instance.

Model instance. We use batch size $B = 128$, horizon $T = 20$, and two orthogonal coordinate blocks of dimension $d_1 = d_2 = 512$. The covariance A_j is diagonal on block j, with eigenvalues drawn once, independently, as

$$\lambda_{j,k} = a_j \, u_{j,k}, \qquad u_{j,k} \overset{\text{i.i.d.}}{\sim} \text{Uniform}(0.5, 1.5), \qquad k = 1, \ldots, d_j, \tag{34}$$

with block scales $a_1 = 2.5$ and $a_2 = 1$ and a fixed random seed. Each domain therefore has mean curvature a_j and a genuine spread (within-block condition number 3), rather than the degenerate flat spectrum $\lambda_{j,k} \equiv a_j$; the scale gap between the blocks is what makes the two domains' optimal rates differ. We start from an isotropic error, $z_0 = \mathbf{1}$ (unit expected squared error in every eigencoordinate), so the initial loss $f_j(w_0) = \frac{1}{2}\lambda_j^\top \mathbf{1} > 0$ and the hypothesis of Proposition 1 holds.

Computing the expectation. All losses are computed exactly, with no Monte Carlo. For a domain with spectrum $D = \text{diag}(\lambda)$, batch fraction p, and rate η, we iterate the second-moment recursion equation 18 of Lemma 1 in its coordinate form,

$$z_{t+1,k} = \Big[(1 - p\eta\lambda_k)^2 + \tfrac{p\eta^2}{B}\lambda_k^2\Big] z_{t,k} + \frac{p\eta^2}{B}\,\lambda_k \sum_\ell \lambda_\ell z_{t,\ell}, \tag{35}$$

for $t = 0, \ldots, T-1$ starting from $z_0 = \mathbf{1}$, and read off $L(p, \eta) = \frac{1}{2}\lambda^\top z_T$ via equation 19. The figure plots the normalized loss $L(p,\eta)/L(p,0) = \lambda^\top z_T / \lambda^\top z_0$, which equals $\mathbb{E}[f_j(w_T)]$ up to the fixed positive factor $f_j(w_0)$ and so preserves every Pareto comparison. Because the transition is diagonal-plus-rank-one, each of the T steps costs $O(d_j)$, and the whole rate grid below is evaluated by broadcasting equation 35 over all rates at once.

Sweeping the learning rate. Once the spectrum is non-flat, the minimizing rates have no closed form—Lemma 2 guarantees a positive minimizer but not its value—so we optimize by sweeping. We evaluate $L(p, \eta)$ on a uniform grid of 2000 rates in $[0, \eta_{\max}]$, with $\eta_{\max} = 1.5/\big(\lambda_{\min}\,(p_{\min} + (d+1)/B)\big)$; this inflates the flat-spectrum optimum evaluated at the smallest present eigenvalue $\lambda_{\min}$ and the smallest batch fraction $p_{\min} = 8/B$, and we check that no reported optimum falls on either endpoint. Mixtures are $p_1 = n_1/B$ with $n_1 \in \{8, \ldots, 120\}$, so that $Bp_j \in \mathbb{N}^+$. For each mixture (p_1, p_2):

- the *reweighting* point is $\big(\min_\eta L_1(p_1, \eta),\ \min_\eta L_2(p_2, \eta)\big)$, minimizing each coordinate independently over the grid (the sweep analogue of the $V_j(p_j)$ in the proof);
- the *remixing* attainable set collects $\big(L_1(p_1, \eta), L_2(p_2, \eta)\big)$ over all grid rates η and all mixtures, and its lower-left envelope is the shared-rate Pareto frontier.

The left panel fixes $p_1 = \frac{1}{2}$: the reweighting point uses the two per-domain optimal rates, while the shared baseline is the single rate minimizing the mixture-weighted loss $p_1 L_1 + p_2 L_2$. The right panel overlays the reweighting frontier (the reweighting point traced over all mixtures) on the shared-rate frontier.

C Data Sources and Processing

We use three data domains: general knowledge (K), mathematics (M), and code (C). All corpora are tokenized with the GPT-2 BPE tokenizer, and every stored token carries an aligned loss mask and domain ID (0 for K, 1 for M, and 2 for C). This appendix describes the source pools, filtering rules, QA conversion, and the distinction between sampled-token exposure and loss-bearing tokens.

C.1 Sources and Filtering

Knowledge. We use full-text web documents (Penedo et al., 2024) as the knowledge-domain corpus. The prepared pool contains 289.27B tokens.

Mathematics. We collect mathematical problems and worked solutions from multiple sources (Li et al., 2023; TIGER-Lab, 2024; Yue et al., 2023; Yu et al., 2024; Liu et al., 2024; Li et al., 2024; Toshniwal et al., 2024; Hugging Face, 2025; Mitra et al., 2024; Fujii et al., 2026; Yue et al., 2024). We first remove examples with 13-token n-gram overlap against GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), AIME 2024 (MAA, 2024), AIME 2025 (MAA, 2025), HMMT 2025 (HMMT, 2025), AMC 2023 (MAA, 2023), or the English split of OlympiadBench (He et al., 2024). We then deduplicate normalized $(\text{question}, \text{answer})$ rows in a fixed source order, retaining the first occurrence. This conservative rule preserves distinct solutions to the same question. After filtering and deduplication, the prepared pool contains 17.20B tokens.

Code. We collect code-oriented prompts, responses, and source files from multiple sources (NVIDIA et al., 2025; Huang et al., 2025; Fujii et al., 2026). We remove examples with 10-token n-gram overlap against HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), GSM8K (Cobbe et al., 2021), or MATH (Hendrycks et al., 2021b). We then perform exact cross- and intra-source deduplication on code answers of at least 200 characters, followed by exact $(\text{question}, \text{answer})$ row deduplication at every length. The final training pool contains 245.16B tokens.

Table 3: **Prepared source pools for the 4×/8× experiments.** Counts are pre-mask GPT-2 BPE tokens; code excludes source-level validation files.

ID	Domain	Representation	Tokens
0	Knowledge	Full text	289.27B
1	Mathematics	QA pairs	17.20B
2	Code	QA pairs	245.16B

C.2 Training Records and Loss Masking

Record format. After source-specific processing, every mathematics or code example is serialized as $[q, a]$, where q is the prompt and a is the target response. Any record whose complete serialization exceeds 4,096 GPT-2 BPE tokens is discarded.

Mathematics. The source problem forms q. After a blank line, we append the following fixed instruction:

```
Please put the answer in \boxed{}.
```

The target response a retains the worked solution supplied by the source when one is available and always ends with a normalized `\boxed{...}` line containing the final answer. Rows from which no reliable final answer can be extracted are omitted.

Code. We parse each code source according to its native format and add no shared instruction. For the instruction corpus with explicit reasoning tags, q is the text before `<think>` and a is the complete response after `</think>`; the intervening reasoning trace is removed. For corpora organized

around fenced code blocks, we split at the first fenced code block: the preceding text becomes q, while the fence and everything after it become a. For the corpus of bare source files, each file is wrapped in a Python code fence and stored with an empty q, reducing the record to $[a]$.

Loss masks. For mathematics and code, the stored mask is zero on EOT and prompt tokens and one on response tokens. For full-text knowledge, it is zero on document-separating EOT tokens and one on document text. During training, a second mask suppresses the first 100 prediction targets after every EOT encountered within a loader window. The distance from a preceding EOT is computed only within that window and is not carried across window boundaries. The effective loss mask is the intersection of the stored and training-time masks.

Masks and domain IDs are attached to prediction targets. Thus, for the causal pair (x_t, x_{t+1}), the loss reads the mask and domain ID associated with x_{t+1}. A masked target contributes neither to the language-model loss nor to the per-domain statistics used by LARE. Records are concatenated into a padding-free stream, and the token, mask, and domain-ID arrays remain positionally aligned.

C.3 MIXTURE RATIOS AND REALIZED LOSS-TOKEN SHARES

Every mixture ratio in the paper is defined over sampled tokens before loss masking. For each declared ratio and token budget, we construct one frozen training stream and reuse it for every controlled comparison at that setting. Within every controlled comparison, runs use exactly the same frozen training stream, including sample order and loss masks; only the experimental factor under study changes. This applies to the optimizer comparisons and every direct–LARE pair. Experiments with different ratios or token budgets use their corresponding frozen streams.

Table 4: **Sampled-token exposure and realized loss-token shares.** Shares are means over per-step valid-target counts after the on-disk and 100-target burn-in masks.

		Sampled tokens (B)			Loss-token share (%)		
Budget	Sampled ratio	K	M	C	K	M	C
1×	7:1 K:M	26.25	3.75	–	90.07	9.93	–
1×	3:1 K:M	22.50	7.50	–	79.53	20.47	–
1×	1:1 K:M	15.00	15.00	–	56.42	43.58	–
1×	7:0.5:0.5 K:M:C	26.25	1.875	1.875	90.28	4.97	4.75
1×	3:0.5:0.5 K:M:C	22.50	3.75	3.75	79.69	10.25	10.06
1×	1:0.5:0.5 K:M:C	15.00	7.50	7.50	56.75	21.91	21.34
4×	7:1 K:M	105.00	15.00	–	90.07	9.93	–
4×	7:0.5:0.5 K:M:C	105.00	7.50	7.50	90.12	4.97	4.91
8×	7:0.5:0.5 K:M:C	210.00	15.00	15.00	90.11	4.97	4.92

Table 4 reports both the nominal sampled-token exposure and the mean domain share among valid loss targets after applying the on-disk and 100-target burn-in masks. Masking changes the effective composition: for example, a nominal 1:1 K:M mixture contains only 43.58% mathematics tokens among the targets that contribute to the loss. Sampled-token ratios should therefore be interpreted as data exposure rather than as relative contributions to the unweighted objective. Token allocations use the rounded 1×/4×/8× budgets of 30B/120B/240B.

D Complete Experimental Details and Results

D.1 Detailed Hyperparameters

Model and training. The main experiments use a 1.635B-parameter decoder-only Transformer with 48 layers, width 1600, 25 attention heads, and a 50,304-token vocabulary. All linear maps are bias-free, normalization is parameter-free, and rotary position encoding is used. The 1.5B runs use a 4096-token context, a global batch of 128 sequences, bfloat16 autocasting and linear learning-rate warmup followed by cosine decay over the full training budget. In multi-stage runs, one global learning-rate schedule spans all stages: the warmup and cosine decay do not restart at a stage boundary, and the optimizer state carries over. If the first stage ends before warmup does, warmup continues into the next stage. Each optimizer step consumes 524,288 sampled tokens, yielding 29.9998B, 119.9990B, and 239.9981B tokens after 57,220, 228,880, and 457,760 steps, respectively. Table 5 summarizes the experiment blocks.

Table 5: Model, token budget, data composition, and optimizer for each experiment block. Ratios refer to sampled tokens before loss masking.

Block	Model	Tokens	Steps	Data ratio	Optimizer(s)
1× controlled experiments	1.635B	30B	57,220	three K:M or K:M:C ratios	AdamW/Muon/ActGD
4× K:M	1.635B	120B	228,880	7:1	ActGD
4× K:M:C	1.635B	120B	228,880	7:0.5:0.5	ActGD
8× K:M:C	1.635B	240B	457,760	7:0.5:0.5	ActGD
Model scaling	124M/350M/774M	2.52B/7.08B/15.73B	4,800/13,500/30,000	7:1	ActGD

Optimizers. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.98)$ and $\epsilon = 10^{-8}$. Muon uses momentum 0.9 with Nesterov acceleration and five Newton–Schulz iterations for matrix-valued hidden parameters, together with auxiliary AdamW for the embedding and output matrices. ActGD uses four Newton–Schulz iterations, $(\beta_1, \beta_2) = (0.95, 0.95)$, $\epsilon = 10^{-5}$, and the row-wise $(2, \infty)$ direction for both vocabulary-facing and transformer-block parameters; Appendix F gives the complete update. Learning rates are matched within every direct–adaptive pair, and each reported cell is a single training run.

Table 6: Optimizer and learning-rate schedule settings. Muon uses separate learning rates for Muon matrices and auxiliary AdamW parameters; the model-scaling warmups correspond to 124M, 350M, and 774M.

Block	Optimizer	Peak LR	Weight decay	Warmup
30B AdamW experiments	AdamW	0.001	0.10	20,000
30B Muon experiments	Muon + AdamW	0.005 / 0.001	0.10	20,000
30B ActGD experiments	ActGD	0.010	10^{-5}	20,000
120B/240B main experiments	ActGD	0.010	10^{-5}	20,000
Model scaling	ActGD	0.010	10^{-5}	1,678 / 4,719 / 10,486

Controller. All LARE runs follow Algorithm 2 with $\beta = 0.99$, $\epsilon = 10^{-8}$, and the learning-rate warmup lengths in Table 6. At each optimizer step, the controller aggregates each domain's unweighted cross-entropy sum and valid-target count over the global batch, updates the count EMA for every domain and the loss EMA for each observed domain, and keeps all loss coefficients equal to one during warmup. After warmup, it computes $r_{i,t} = (\hat{\ell}_{i,t} + \epsilon)^{-1/2}$ and applies $w_{i,t+1} = r_{i,t} / \sum_j \hat{\pi}_{j,t} r_{j,t}$, where $\hat{\pi}_{j,t}$ is the EMA count share. Fixed-weight ablations likewise use unit weights during warmup and activate their listed coefficients afterward.

Reproducibility. Within every controlled comparison, the paired runs use the same prepared data streams described in Appendix C. Per-domain loss sums and valid-token counts are aggregated across data-parallel ranks before the controller is updated, so every rank applies the same weights on the next optimizer step. All tables report the unsmoothed final checkpoint; any smoothing in a training-curve figure is used only for display.

D.2 Matched-Exposure Results

Table 7 compares four 30B-token recipes across AdamW, Muon, and ActGD with 26.25B knowledge and 3.75B mathematics tokens. Figure 4 shows the Muon checkpoint trajectories.

Table 7: Matched-exposure comparison at 30B tokens across optimizers. First-domain drop is the final score minus the best pre-switch score; bold marks the best reported final result within each optimizer and benchmark.

Optimizer	Recipe	Token schedule	$w_K : w_M$	MMLU	GSM8K	First-domain drop
AdamW	Knowledge → Math	26.25B K → 3.75B M	1:1	30.57	18.35	MMLU: −1.06
	Math → Knowledge	3.75B M → 26.25B K	1:1	**32.28**	0.83	GSM8K: −7.96
	Direct mixture	stationary 7:1	1:1	31.30	21.91	–
	Fixed reweight	stationary 7:1	1:2	32.03	**31.24**	–
Muon	Knowledge → Math	26.25B K → 3.75B M	1:1	29.45	36.62	MMLU: −1.63
	Math → Knowledge	3.75B M → 26.25B K	1:1	31.78	0.76	GSM8K: −17.74
	Direct mixture	stationary 7:1	1:1	31.40	27.67	–
	Fixed reweight	stationary 7:1	1:2	**32.27**	**38.59**	–
ActGD	Knowledge → Math	26.25B K → 3.75B M	1:1	30.97	11.45	MMLU: −1.70
	Math → Knowledge	3.75B M → 26.25B K	1:1	31.95	0.61	GSM8K: −26.23
	Direct mixture	stationary 7:1	1:1	31.82	28.58	–
	Fixed reweight	stationary 7:1	1:2	**32.65**	**34.19**	–

Across all three optimizers, the two-stage orders retain different capabilities. Fixed 1:2 reweighting improves both MMLU and GSM8K over direct mixing under every optimizer. The Muon reweighted endpoint exceeds the best scores along either stage trajectory; under AdamW, the Math → Knowledge run retains a higher MMLU score.

D.3 Detailed Data-Mixture Results

Figure 6 visualizes the ActGD data-mixture comparisons, and Table 8 reports their final checkpoints.

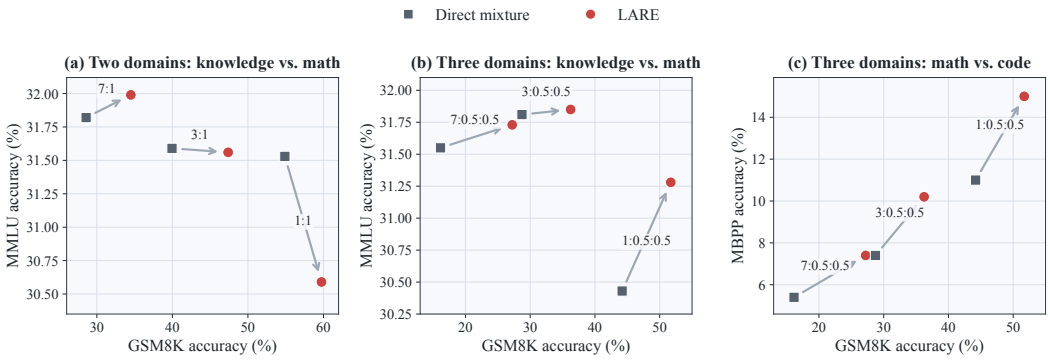


Figure 6: **LARE remains effective across mixture ratios and improves all three capabilities in K:M:C training.** Arrows connect matched direct-mixture and LARE runs with the same 1× Chinchilla budget and ActGD configuration. Panel (a) varies K:M over 7:1, 3:1, and 1:1: GSM8K improves at every ratio, while the largest MMLU cost occurs at the reasoning-heavy 1:1 boundary. Panels (b) and (c) add code with M:C = 1:1; LARE improves MMLU, GSM8K, and MBPP at every tested ratio.

At 1× Chinchilla, LARE improves GSM8K by 4.85–7.43 points across K:M ratios 7:1, 3:1, and 1:1; the corresponding MMLU changes are +0.17, −0.03, and −0.94 points (Figure 6(a)). At the reasoning-heavy 1:1 boundary, mathematics contributes approximately 43% of realized loss tokens but 68% of the weighted objective at the final checkpoint, consistent with the larger MMLU cost. With code added at K:M:C ratios 7:0.5:0.5, 3:0.5:0.5, and 1:0.5:0.5, LARE improves MMLU by

Table 8: Direct and adaptive objectives across 30B-token data mixtures with ActGD. Each pair differs only in online loss reweighting; bold marks the better result within the pair, and "–" indicates that the corresponding domain is absent.

Domains	Data ratio	Objective	MMLU	GSM8K	MBPP
K:M	7:1	Direct mixture	31.82	28.58	–
		LARE	**31.99**	**34.50**	–
	3:1	Direct mixture	**31.59**	39.95	–
		LARE	31.56	**47.38**	–
	1:1	Direct mixture	**31.53**	54.89	–
		LARE	30.59	**59.74**	–
K:M:C	7:0.5:0.5	Direct mixture	31.55	16.15	5.40
		LARE	**31.73**	**27.22**	**7.40**
	3:0.5:0.5	Direct mixture	31.81	28.73	7.40
		LARE	**31.85**	**36.24**	**10.20**
	1:0.5:0.5	Direct mixture	30.43	44.20	11.00
		LARE	**31.28**	**51.71**	**15.00**

0.04–0.85 points, GSM8K by 7.51–11.07 points, and MBPP by 2.00–4.00 points at every ratio (Figure 6(b)–(c)). Additional optimizer replications appear in Appendix D.4.

D.4 Additional Optimizer Results

The ratio study referenced in Section 4.4 and detailed in Appendix D.3 uses ActGD. To test whether its behavior is optimizer-specific, we repeat the same 30B-token K:M sweep with AdamW and Muon. Within each row of Table 9, the direct and LARE runs use the same model, prepared data stream, sampled-data ratio, optimizer settings, learning rate, and training budget; only online loss reweighting differs. The ActGD endpoints remain in Table 8 and are not duplicated here.

Table 9: Additional 30B-token K:M ratio sweeps with AdamW and Muon. Score pairs are MMLU/GSM8K accuracy at the final checkpoint; Δ is LARE minus direct mixture in percentage points.

Optimizer	K:M	Direct mixture	LARE	Δ
AdamW	7:1	31.30 / 21.91	31.40 / 33.66	$+0.10$ / $+11.75$
	3:1	31.98 / 37.07	31.95 / 46.63	-0.03 / $+9.56$
	1:1	31.58 / 54.66	31.02 / 60.50	-0.56 / $+5.84$
Muon	7:1	31.40 / 27.67	32.24 / 39.80	$+0.84$ / $+12.13$
	3:1	32.16 / 43.67	32.05 / 55.19	-0.11 / $+11.52$
	1:1	31.67 / 59.89	31.15 / 65.13	-0.52 / $+5.24$

Across the six additional pairs, LARE improves GSM8K at every ratio by 5.24–12.13 points. Combining these replications with the three ActGD pairs in Table 8, all nine optimizer–ratio settings improve GSM8K. The MMLU shifts follow the same ratio-conditioned pattern under all three optimizers: they are positive at 7:1 ($+0.10$ to $+0.84$), nearly unchanged at 3:1 (-0.03 to -0.11), and more negative at the reasoning-heavy 1:1 boundary (-0.52 to -0.94). This shared pattern indicates that the boundary behavior tracks the sampled-data ratio rather than a particular optimizer.

Table 10 places the online rule against the complete fixed-weight grids at K:M $=$ 7:1 for all three optimizers and at 3:1 for Muon. At 7:1, LARE improves both benchmarks over direct mixing under every optimizer. In the Muon 3:1 setting, it raises GSM8K by 11.52 points with a 0.11-point MMLU decrease. Its endpoint is bracketed by the fixed 1:2 and 1:3 endpoints on both metrics under Muon at either mixture ratio and under ActGD at 7:1. Under AdamW, fixed 1:3 is better by 0.24 MMLU

points and 2.20 GSM8K points. Thus the online rule does not replace a coefficient selected retrospectively from an oracle grid; it reaches a competitive region without the four additional training runs required by each sweep.

Table 10: Fixed post-warmup loss coefficients and LARE across optimizers and K:M mixtures at 30B tokens. Each optimizer uses its learning rate in Table 6; all other conditions are matched within an optimizer–mixture setting. Scores are final-checkpoint accuracies.

Optimizer	K:M	Metric	Direct 1:1	Fixed 1:2	Fixed 1:3	Fixed 1:4	Fixed 1:7	LARE
AdamW	7:1	MMLU	31.30	32.03	31.64	31.83	31.14	31.40
		GSM8K	21.91	31.24	35.86	39.20	41.39	33.66
Muon	7:1	MMLU	31.40	32.27	32.08	32.07	31.19	32.24
		GSM8K	27.67	38.59	42.46	43.29	48.98	39.80
	3:1	MMLU	32.16	32.20	31.75	31.44	30.89	32.05
		GSM8K	43.67	51.40	56.71	56.41	58.23	55.19
ActGD	7:1	MMLU	31.82	32.65	31.85	32.20	31.80	31.99
		GSM8K	28.58	34.19	36.39	40.18	47.31	34.50

D.5 EVALUATION SETTINGS

We evaluate general knowledge with MMLU (Hendrycks et al., 2021a), mathematical reasoning with GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), and code generation with HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). All evaluations use the complete benchmark split, bfloat16 inference, a maximum context length of 4096, and the settings in Table 11.

Table 11: Evaluation protocols. Greedy decoding uses one deterministic completion per problem; Accuracy is the percentage of problems answered correctly or passing all tests.

Benchmark	Examples	Prompting	Prediction rule	Max. new tokens	Metric
MMLU	14,042	0-shot	normalized answer-text likelihood	–	Accuracy
GSM8K	1,319	0-shot	greedy + exact match	256	Accuracy
MATH-500	500	0-shot	greedy + exact match	256	Accuracy
HumanEval	164	0-shot instruct	greedy + tests	400	Accuracy
MBPP	500	3-shot	greedy + tests	400	Accuracy

MMLU. Every MMLU value in the paper uses the 0-shot FineWeb LightEval recipe (Penedo et al., 2024) over all 57 test subjects. The evaluator scores each complete answer text as a candidate continuation, selects by length-normalized log-likelihood, and reports the micro-averaged `acc_norm` over 14,042 questions. This is the MMLU protocol used for both our checkpoints and the Qwen1.5 rows.

GSM8K and MATH. We use the GSM8K main test split and the MATH-500 test split. Each problem receives one 0-shot greedy completion of at most 256 new tokens, followed by exact matching after benchmark-specific answer extraction and normalization.

HumanEval and MBPP. HumanEval uses the 0-shot `humaneval_instruct` prompt template from lm-evaluation-harness, rendered as plain text without a chat template. MBPP uses its canonical 500-problem test set with three fixed demonstrations. For both benchmarks, the evaluator produces exactly one greedy completion of at most 400 new tokens and executes the benchmark tests; Accuracy is the fraction of problems whose completion passes every test.

Greedy decoding always selects the highest-logit token, with no sampling or multiple candidates. The Qwen1.5 base models use the same task implementations, prompt formats, shot counts, generation limits, normalization rules, and scoring as our final checkpoints, with their native tokenizers

and no chat template. All scores are computed once at the final checkpoint and multiplied by 100 for presentation.

D.6 Training Trajectories at Longer Budgets

Figure 7 shows checkpoint trajectories for the matched K:M:C = 7:0.5:0.5 runs at 120B and 240B sampled tokens.

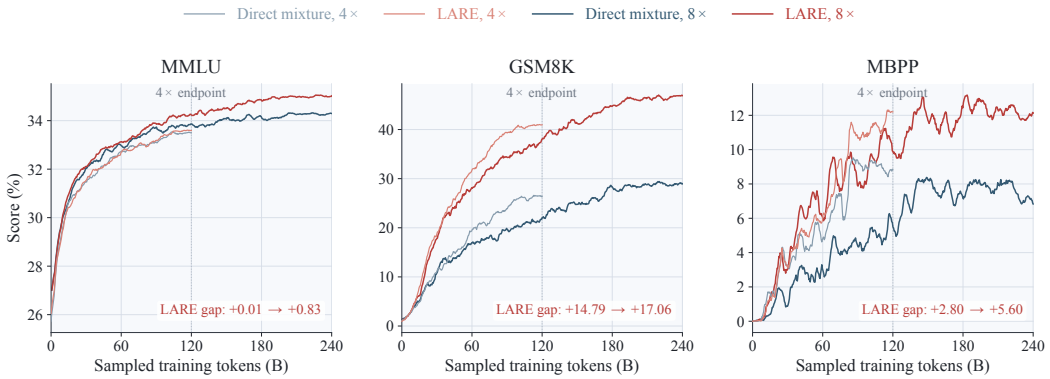


Figure 7: **Checkpoint trajectories for matched 4× and 8× K:M:C runs.** All runs use the K:M:C = 7:0.5:0.5 mixture and ActGD. Blue-gray curves denote direct mixing and red curves LARE; lighter curves end at 120B tokens and darker curves at 240B. Curves are centered 11-checkpoint moving averages.

E SCALING ACROSS MODEL SIZES

E.1 TRANSFERRED SETUP

We test whether the effect of LARE transfers across 124M, 350M, and 774M model configurations, using 12/6/768, 24/16/1024, and 36/20/1280 layers/heads/hidden dimensions, respectively. Each model is trained at its nominal $1\times$ Chinchilla allocation (Hoffmann et al., 2022): 2.52B, 7.08B, and 15.73B sampled tokens. Except for model size, token budget, and the proportionally scaled warmup and LARE activation boundary, all training settings follow those used for the main results in Section 4.3. All runs use K:M = 7:1. Within each model size, the direct-mixture and LARE runs differ only in loss reweighting, while the two stage schedules preserve the same total domain exposure.

E.2 REPLICATION RESULTS

Table 12: **The mathematical-reasoning benefit of LARE emerges and strengthens with joint model–token scale.** Scores are percentages at the final checkpoint under matched $1\times$ Chinchilla budgets. At 774M, LARE improves GSM8K by 7.36 points and MMLU by 0.02 points over direct mixing. The 124M result remains at the low-accuracy GSM8K floor, and the multi-stage endpoints remain sensitive to domain order. Bold marks the best result within each model size.

Model	Recipe	Sampled tokens	MMLU	GSM8K
124M	Direct mixture	2.52B	27.64	**2.73**
	K→M		27.82	1.59
	M→K		**27.87**	0.30
	LARE		27.70	1.67
350M	Direct mixture	7.08B	28.69	3.18
	K→M		28.14	2.27
	M→K		**28.73**	1.14
	LARE		27.87	**5.08**
774M	Direct mixture	15.73B	30.28	11.37
	K→M		29.65	3.94
	M→K		29.87	0.30
	LARE		**30.30**	**18.73**

The validation losses provide a more continuous view of the same effect. Relative to direct mixing, LARE reduces mathematics validation loss by 5.70%, 6.73%, and 6.55% at 124M, 350M, and 774M, while increasing knowledge validation loss by only 0.39%, 0.26%, and 0.45%. The downstream GSM8K benefit becomes visible only as the model leaves the low-accuracy regime. Consistently, the final coefficient ratio $w_{\mathrm{M}}/w_{\mathrm{K}}$ selected by LARE rises from 2.00 to 2.20 and 2.50 across the three allocations. Because model size and token budget co-vary in this experiment, these results establish a trend under joint Chinchilla scaling rather than isolating parameter count alone.

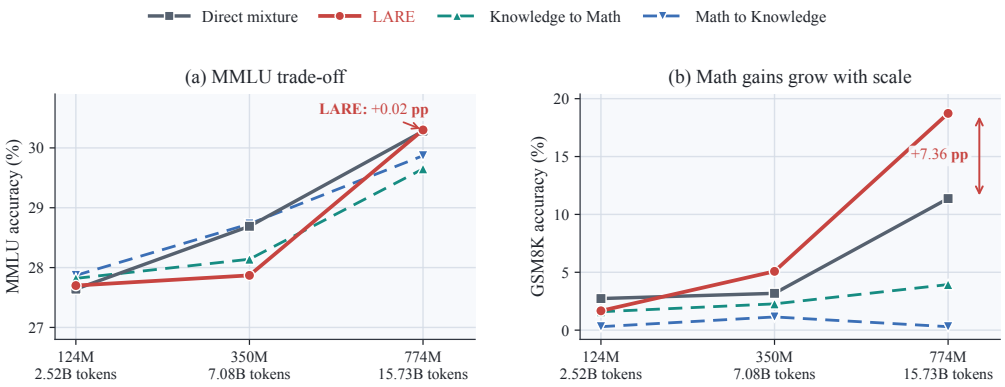


Figure 8: **At the largest scale, LARE's growing mathematical-reasoning advantage no longer incurs an MMLU cost.** Curves connect final checkpoints across the matched 124M/2.52B, 350M/7.08B, and 774M/15.73B model–token allocations. Relative to direct mixing, the GSM8K difference changes from -1.06 points at 124M to $+1.90$ at 350M and $+7.36$ at 774M; the corresponding MMLU differences are $+0.06$, -0.82, and $+0.02$ points. The two ordered stage schedules do not recover the same joint endpoint. Results are single seed and therefore show the observed trend without estimating training-seed uncertainty.

F ACTIVATION GRADIENT DESCENT

ActGD is the base optimizer used in the main experiments, whereas LARE is the domain-level objective controller studied in this paper. The two act at different points in the training pipeline: LARE determines how strongly each domain contributes before the batch gradient is formed, while ActGD maps that gradient to a parameter update. We include the derivation here both to make this separation explicit and to specify the optimizer underlying the results in Section 4.

F.1 FROM ACTIVATION MATCHING TO A WHITENED UPDATE

ActGD is motivated by the view that an optimizer should control the change induced in a layer's activations, rather than only the Euclidean displacement of its parameters. This view parallels layer-wise activation matching in model merging (Xu et al., 2025), but the construction below is local to one optimization step and does not assume an equivalence between model merging and training. More broadly, the observation that different optimization geometries can yield different downstream behavior at similar pretraining loss motivates making this geometry explicit (Chen et al., 2026a).

Consider a linear layer applied to n activation vectors. We collect the fixed inputs as $X = [x_1, \dots, x_n] \in \mathbb{R}^{d_{\text{in}} \times n}$ and the corresponding outputs as $Y = [y_1, \dots, y_n] = WX$, where $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. Suppose an ideal local optimizer would change the outputs by ΔY, while changing the weight matrix by ΔW. Holding X fixed under the local linearization gives

$$(W + \Delta W)X = WX + \Delta WX = Y + \Delta WX. \tag{36}$$

Consequently, exact activation matching requires $\Delta WX = \Delta Y$. More generally, when one matrix update cannot realize every desired token-level displacement exactly, we choose it by minimizing the residual

$$\min_{\Delta W} \frac{1}{2n} \left\| (W + \Delta W)X - (Y + \Delta Y) \right\|_F^2 = \min_{\Delta W} \frac{1}{2n} \left\| \Delta WX - \Delta Y \right\|_F^2 . \tag{37}$$

Let $d_i = \nabla_{y_i} \ell_i$ be the gradient of the per-position loss with respect to y_i, and collect these vectors as $\Gamma = [d_1, \dots, d_n]$. The desired output displacement follows the negative-gradient direction, $\Delta Y = -\alpha\Gamma$, where $\alpha > 0$ sets its magnitude. Substituting this target into Equation 37 gives $\frac{1}{2n}\|\Delta WX + \alpha\Gamma\|_F^2$. Define the batch gradient and uncentered input second moment as

$$G = \frac{1}{n}\Gamma X^\top = \frac{1}{n}\sum_{i=1}^{n} d_i x_i^\top, \qquad C = \frac{1}{n} XX^\top = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top . \tag{38}$$

Expanding the squared Frobenius norm then yields the identity

$$\frac{1}{2n}\left\| \Delta WX + \alpha\Gamma \right\|_F^2 = \frac{1}{2}\operatorname{Tr}(\Delta WC\Delta W^\top) + \alpha\langle G, \Delta W\rangle + \frac{\alpha^2}{2n}\|\Gamma\|_F^2. \tag{39}$$

The last term is independent of ΔW, and the scalar α can be absorbed into the eventual learning rate. Setting it to one therefore recovers the activation-matching surrogate $\Phi(\Delta W) = \langle G, \Delta W\rangle + \frac{1}{2}\operatorname{Tr}(\Delta WC\Delta W^\top)$ used below.

The role of C is important when the activation spectrum is long-tailed. In raw parameter coordinates, a component of ΔW aligned with a high-variance eigenvector of C produces a much larger output displacement than an equally sized component aligned with a low-variance direction. A Euclidean constraint on ΔW therefore allows a few high-variance activation directions to dominate the matching objective. To remove this scale imbalance, we whiten the inputs and express the update in the corresponding parameter coordinates. Assuming C is positive definite for the derivation, with numerical regularization used in practice, define

$$\widetilde{X} = C^{-1/2}X, \qquad \frac{1}{n}\widetilde{X}\widetilde{X}^\top = I, \qquad Z = \Delta WC^{1/2}, \qquad M = GC^{-1/2}. \tag{40}$$

Here Z is precisely the change to the layer weight expressed in the whitened input basis, since $WX = (WC^{1/2})\widetilde{X}$. The change of variables gives

$$\langle G, \Delta W\rangle = \langle M, Z\rangle, \qquad \operatorname{Tr}(\Delta WC\Delta W^\top) = \|Z\|_F^2. \tag{41}$$

Thus whitening converts the activation metric into an isotropic Frobenius metric and prevents the long tail of the covariance spectrum from directly setting the update scale. The unconstrained minimizer is $Z = -M$, which maps back to $\Delta W = -GC^{-1}$. To control the structure of the direction independently of its global magnitude, which is supplied by the learning rate, we instead solve the linear minimization oracle

$$Z^\star \in \arg\min_{\|Z\|_{\mathcal{B}} \le 1} \langle M, Z \rangle, \qquad \Delta W^\star = Z^\star C^{-1/2}. \tag{42}$$

The constraint is therefore imposed in the whitened activation geometry rather than directly on the raw parameter displacement.

F.2 The Row-Wise $(2, \infty)$ Direction

We use $\|Z\|_{2,\infty} = \max_j \|Z_j\|_2$, where Z_j denotes output row j. The corresponding oracle decomposes over rows and yields

$$(\Delta W_j)^\star = -\frac{G_j C^{-1}}{\sqrt{G_j C^{-1} G_j^\top} + \epsilon}, \qquad j = 1, \dots, d_{\text{out}}. \tag{43}$$

Ignoring the numerical constant, each row satisfies $(\Delta W_j)^\star C((\Delta W_j)^\star)^\top = 1$. Hence every output row receives a comparable step in the activation metric even when the raw gradient norms or input variances differ substantially. The normalization is local to each output row, so a small number of large rows cannot set the scale of the entire matrix update.

Table 13 summarizes the closed-form directions induced by the three constraint choices used in our optimizer family. Here $\|\cdot\|_\infty$ is the elementwise maximum norm, $\|\cdot\|_2$ is the spectral norm, and $\mathrm{Orth}(\cdot)$ denotes the polar factor.

Table 13: Closed-form ActGD directions under three constraints in the whitened activation geometry, where $M = GC^{-1/2}$ and $\Delta W^\star = Z^\star C^{-1/2}$. The main experiments use $(2, \infty)$; implementations add ϵ stabilization where needed.

Constraint on Z	Whitened direction $Z^\star$	Parameter-space update $\Delta W^\star$
$\|Z\|_{2,\infty} \le 1$	$Z_j^\star = -M_j / \|M_j\|_2$	$(\Delta W_j)^\star = -G_j C^{-1} / \sqrt{G_j C^{-1} G_j^\top}$
$\|Z\|_\infty \le 1$	$Z^\star = -\operatorname{sign}(M)$	$\Delta W^\star = -\operatorname{sign}(GC^{-1/2}) C^{-1/2}$
$\|Z\|_2 \le 1$	$Z^\star = -\operatorname{Orth}(M)$	$\Delta W^\star = -\operatorname{Orth}(GC^{-1/2}) C^{-1/2}$

We adopt Equation 43: unlike the elementwise sign direction, it preserves within-row gradient structure, and unlike the spectral direction, it normalizes each output row independently rather than coupling all rows through a single matrix-wide polar factor.

F.3 Stateful Optimizer Used in Our Experiments

The practical optimizer smooths both quantities in Equation 38. Let G_t and C_t denote their step-t estimates. For gradient-momentum decay β_g and covariance decay β_c, we maintain

$$\begin{aligned} H_t &= \beta_g H_{t-1} + (1 - \beta_g) G_t, & \widehat{H}_t &= \frac{H_t}{1 - \beta_g^t}, \\ S_t &= \beta_c S_{t-1} + (1 - \beta_c) C_t, & \widehat{S}_t &= \frac{S_t}{1 - \beta_c^t}. \end{aligned} \tag{44}$$

Let A_t be a numerically stabilized Newton–Schulz approximation to $\widehat{S}_t^{-1}$ and set $U_t = \widehat{H}_t A_t$. The implemented descent transform is

$$D_{t,j} = \frac{U_{t,j}}{\sqrt{\max\{0, \widehat{H}_{t,j} U_{t,j}^\top\} + \epsilon}}, \qquad W_{t+1} = (1 - \eta_t \lambda) W_t - \eta_t D_t, \tag{45}$$

Table 14: **ActGD uses one optimizer configuration for both vocabulary-facing and transformer-block parameters in the main experiments.** The settings below determine the update geometry; the common model, batch, and learning-rate schedule are reported in Appendix D.1.

Setting	Value
Embedding/head update	Row-wise $(2, \infty)$
Transformer-block update	Row-wise $(2, \infty)$
Peak learning rate η	0.01 for both groups
Gradient/covariance EMA (β_g, β_c)	$(0.95, 0.95)$
Numerical constant ϵ	10^{-5}
Newton–Schulz iterations	4
Covariance sample cap	$n' = \min(n, d_{\text{in}})$
Decoupled weight decay λ	10^{-5}

where λ is decoupled weight decay. The clamp affects only roundoff-induced negative quadratic forms. Equations 44–45 reduce to the row-wise activation-space direction in Equation 43 when the running estimates and inverse approximation are exact.

Table 14 records the ActGD-specific settings shared by the main experiments. The token embedding, language-model head, and transformer-block matrices use the same row-wise update family and the same hyperparameter values, although they are retained as two parameter groups for bookkeeping. This unified treatment is the single-configuration property highlighted in Table 1.

F.4 Parameter Coverage and Distributed Implementation

For a linear map, forward hooks flatten the leading activation dimensions and estimate $C_t = X_t^\top X_t / n$ in FP32. When the number of activation rows exceeds the input width, the main configuration samples $n' = d_{\text{in}}$ rows with replacement before forming the covariance. Query, key, and value projections in the same attention block consume the same input tensor, so their covariance is computed and communicated once and then reused for all three weights. The covariance EMA and the gradient EMA are stored in FP32; the inverse is evaluated with four fixed Newton–Schulz polynomial iterations.

The embedding layer is the one-hot special case of the same construction. Its input second moment is diagonal, so we store only the empirical token-frequency vector and apply its inverse by row-wise scaling in the stored embedding orientation rather than materializing a vocabulary-sized covariance matrix. The language-model head is treated as an ordinary linear layer using the second moment of its hidden-state inputs. Because the architecture used here has bias-free linear maps and parameter-free normalization, all learned parameters are matrices and no separate one-dimensional fallback optimizer is required.

In distributed training, activation statistics are averaged across data-parallel ranks before the update is constructed. Each parameter's inverse and update are computed on one deterministically assigned owner rank, after which the disjoint update shards are combined with a sum reduction. The communicated update buffer uses bfloat16, while the persistent EMA states remain FP32. The sharded optimizer checkpoints store the step counter and both EMA tensors for each owned parameter; forward-hook statistics, shared-covariance aliases, and temporary inverse buffers are step-local and are cleared after each update.

ActGD stores a gradient-sized first-moment tensor and an input-covariance tensor for each matrix, and its additional computation is dominated by covariance formation and the approximate inverse. Activation subsampling, query/key/value covariance reuse, cached identity matrices, and distributed owner sharding reduce this overhead without changing Equation 45. Since the experiments in this paper match sampled tokens rather than wall-clock time, the comparisons in Tables 1 and 2 should be interpreted as optimization-quality and sample-efficiency comparisons, not as claims of equal throughput or equal peak memory.