TRAINING-FREE DATA MIXTURE OPTIMIZATION VIA NEURAL TANGENT KERNELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Determining the optimal data mixture is an important problem in continual pre-training and fine-tuning of large language models (LLMs), as it often governs the trade-off between preserving general capabilities and improving domain-specific expertise. Existing approaches usually estimate the optimal mixture through proxy models, scaling laws, or model merging, while directly modeling how the target LLM evolves under different data mixtures remains under-explored. In this paper, we propose a theoretical framework for data mixture optimization that, to our knowledge, the first time directly models the target LLM's own training dynamics. Specifically, we characterize training in the output space and leverage the neural tangent kernel (NTK) to transform discrete gradient updates into continuous logit dynamics. To make the resulting dynamics tractable, we theoretically justify kernel freezing when the NTK direction remains approximately stable throughout training, allowing the time-varying NTK to be approximated by its initial value and reducing the dynamics to a fixed-kernel ordinary differential equation. We then solve the resulting system using an adaptive numerical ODE solver to generate predicted logit trajectories for ranking candidate mixtures without training the target model. Extensive experiments show the effectiveness of our framework.

1 INTRODUCTION

Determining the *data mixture*, i.e., the proportions of training samples drawn from different domains, is an important problem in training large language models (LLMs) (Que et al., 2024; Gu et al., 2024; Ye et al., 2025; Zheng et al., 2024; Shao et al., 2024; Team et al., 2025; Guo et al., 2025). As shown in Fig. 1, when adapting a base LLM that already possesses strong general capabilities to several target domains such as mathematics and code, the data mixture decides how the training budget is allocated across these domains, which plays a vital role in the resulting model's performance (Li et al., 2025; Cobbe et al., 2021; Chen et al., 2021a; Wei et al., 2022). A well-chosen data mixture can well preserve the general language ability acquired during pre-training while strengthening the model's expertise on the target domains. Consequently, finding the optimal data mixture that well balances general and domain-specific abilities has attracted much attention in continual pre-training (Que et al., 2024; Gu et al., 2024) and LLM fine-tuning recently (Li et al., 2025).

Yet this problem is very challenging, as finding the data mixture that maximizes the resulting model's downstream performance involves searching over a large space of candidate data mixtures. To achieve this, a naive approach is brute-force search, i.e., training a separate LLM for each candidate mixture and keeping the one that finally performs best on downstream tasks. However, this incurs an extremely high, and often even unaffordable, computational and training cost, making it infeasible in practice. To avoid such training costs, existing solutions replace the target model's actual training process with small proxy models (Liu et al., 2025; Diao et al., 2026), fitted scaling laws (Ye et al., 2025; Ge et al., 2024), or parameter-space combinations of domain experts (Li et al., 2026; Berasi et al., 2026). Although cheaper, these alternatives overlook that a data mixture acts on the target model through its actual training process, the data mixture shapes how the model is updated during training, and these updates determine the model downstream performance. Thus, directly modeling the target model's training process can better simulate how different data mixtures affect its downstream performance, and thereby help identify the optimal data mixture.

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.

This gap motivates our central question: *can we directly model the target model's training dynamics under different data mixtures, rather than relying on small proxy models or actually training the target model itself, to predict its training trajectory?* If so, we could estimate the optimal data mixture by modeling the target model's training dynamics. However, this goal is highly challenging, because it requires characterizing how an LLM would evolve under a given data mixture during training, a process that is long, high-dimensional, and non-linear and involves billions of parameters. To achieve it, we need to answer two questions: (i) how can an LLM's training process be modeled; and (ii) how can the discrete gradient updates be turned into an analyzable mathematical form.

In this paper, to address the above two questions, we build on the neural tangent kernel (NTK) (Jacot et al., 2018) to characterize how the target LLM itself evolves during training. The NTK is a classical analysis tool that enables discrete gradient updates to be expressed as an analyzable differential equation governed by the kernel itself. This makes the NTK a natural choice for modeling the target model's training dynamics. However, naively leveraging the NTK to obtain the optimal data mixture raises three difficulties. First, while the NTK models an LLM's training process, it does not involve the data mixture, we therefore need to model the mixture as part of the dynamics as well. Second, the NTK coefficient is time-varying as training proceeds, so obtaining the coefficient still requires the training we aim to avoid. Third, a differential equation needs to be solved to obtain the optimal data mixture for the target model.

To handle these three difficulties, we propose a novel framework with tailored designs. For the first, we describe the dynamics over the model's logits, where a data mixture is exactly a set of weights on the per-domain gradients and therefore enters the equation directly as its coefficients. For the second, leveraging the near-stability of the NTK direction during training (Wang et al., 2026), we theoretically justify replacing the time-varying NTK with its initial value, making the equation solvable without training. For the third, we integrate the resulting equation numerically with an adaptive ODE solver. To our knowledge, we are the first to take advantage of the NTK for data mixture optimization, directly modeling the target model's training dynamics without training. We next describe how these designs are incorporated in our framework.

As shown in Fig. 2, we proceed as follows. **(1) Output-space modeling.** To model the training process of an LLM under a given data mixture, we need to first choose the space in which we can describe the dynamics. Instead of modeling the enormous parameter space, we model the evolution of the *output* space, namely the logits, which is directly tied to the training objective and where a data mixture enters directly as the weights on the per-domain gradients. **(2) Continuous dynamics.** To turn the discrete gradient updates into an analyzable mathematical form, we use the NTK to rewrite them as a continuous-time ordinary differential equation (ODE) with a time-varying NTK coefficient. **(3) ODE simplification.** Since previous work has shown that the NTK's direction stays nearly fixed during training (Wang et al., 2026), we replace the time-varying NTK coefficient with the initial NTK and we prove that this operation incurs only bounded error, reducing the equation to a constant-NTK ODE. **(4) Numerical solving.** To solve this constant-NTK ODE, which still has no closed form under the cross-entropy loss, we integrate it numerically with an adaptive ODE solver. As a result, our method is able to predict the training trajectory of candidate mixtures without training the target model, serving as an effective indicator for selecting the optimal data mixture.

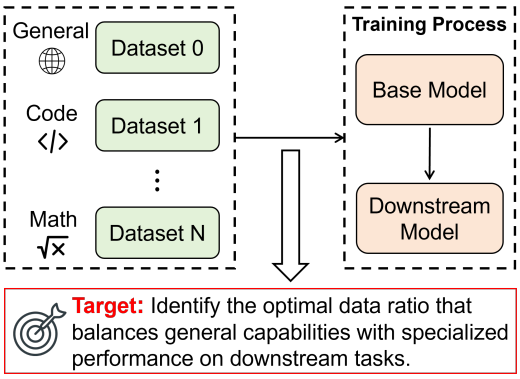

Figure 1: Illustration of the data mixture optimization task. When a base LLM is adapted to several target domains such as mathematics and code, the goal is to find the data mixture that best balances general and domain-specific ability.

In summary, our contributions are: 1) We propose a *training-free* framework for estimating the optimal data mixture for the LLM training process, which, to the best of our knowledge, is the first to do so by directly modeling the target model's own training dynamics; 2) We model the training dynamics by converting the discrete process into continuous logit dynamics, and then solve the resulting system with a tailored numerical ODE solver; 3) Extensive experiments on multiple evaluation benchmarks show the efficacy of our method.

2 Related Work

Data Mixture Optimization. Determining the optimal data mixture in large language model (LLM) training has attracted sustained research attention, since the domain proportions used during training largely affect the model capability (Que et al., 2024; Gu et al., 2024; Ye et al., 2025; Zheng et al., 2024; Shao et al., 2024; Team et al., 2025; Guo et al., 2025; Xie et al., 2023; Fan et al., 2023; Chen et al., 2025; Albalak et al., 2023; Shukor et al., 2026; Belenki et al., 2025; Shin et al., 2026). To avoid the prohibitive cost of brute-force search, existing work estimates the optimal mixture along three lines, which typically rely on a cheaper substitute rather than on the target model's own training. *Proxy-model* methods learn domain proportions on a small proxy model and then transfer them to the target LLM (Liu et al., 2025; Diao et al., 2026; Xie et al., 2023; Fan et al., 2023; Tan et al., 2026). For example, RegMix (Liu et al., 2025) trains many small models on diverse mixtures, fits a regression that predicts downstream performance from the mixture, and applies the top-predicted mixture to train the large target model. FastMix (Tan et al., 2026) instead learns mixture weights through validation-guided gradient updates while training a single proxy model. *Scaling-law* methods fit a parametric mixture–performance relation from small-scale runs and extrapolate it to the target scale (Ye et al., 2025; Shukor et al., 2026; Que et al., 2024; Gu et al., 2024; Li et al., 2025). For example, the D-CPT Law (Que et al., 2024) fits such a parametric law from small-scale runs to predict performance across arbitrary mixture ratios, model sizes, and token budgets, thereby locating the optimal general-to-domain ratio for continual pre-training. *Model-merging* methods approximate a data mixture by combining separately trained domain experts in parameter space (Li et al., 2026; Berasi et al., 2026). For example, Li et al. (2026) scale data mixing via model merging by training one expert per domain and merging them under different weights to simulate candidate mixtures, thereby decoupling the mixture search from actual training. Different from them, our method models the target model's training dynamics to reveal how each candidate mixture shapes its final performance, and thereby identifies the optimal mixture without any training.

Neural Tangent Kernel Methods. The neural tangent kernel (NTK) (Jacot et al., 2018), which makes otherwise intractable training dynamics analytically tractable and thus serves as a theoretical tool to characterize the optimization process, has attracted much research attention (Murray et al., 2023; Yang et al., 2021; Hanin & Nica, 2020; Zeng et al., 2025; Wang et al., 2022), and has been applied to model selection (Hu et al., 2024), training-free architecture search and pruning (Chen et al., 2021b; Wang et al., 2023), and data selection and dataset distillation (Wang et al., 2026; Nguyen et al., 2021). Unlike the works above, to our knowledge, we are the first to directly model the target LLM's training dynamics for data mixture optimization. Yet, modeling these dynamics is non-trival. It requires a suitable representation for modeling the training process and an analyzable formulation of discrete updates. The NTK offers a natural solution by expressing training as a differential equation over model's outputs. However, applying the NTK still faces three questions, namely how to incorporate mixture proportions into the dynamics, how to obtain the evolving kernel without training, and how to solve the resulting equation. We thereby propose a novel framework with tailored designs. To deal with the first question, we embed mixture weights in logit dynamics. To address the second, we theoretically justify freezing the kernel at initialization, and to address the third, we solve the equation numerically. Via these designs, our framework substantially outperforms existing methods.

3 Method

In this work, to handle the data mixture optimization task, we aim to predict, without brute-force search, the training trajectory of the target model under any candidate mixture α, and thereby identify the data mixture that minimizes the validation loss. Achieving this is challenging, since it requires describing how an LLM evolves during training without running that training. We therefore build on the NTK, turning training into a differential equation on the model's outputs. However, using it is still non-trivial. The NTK takes no account of the data mixture, computing it along the trajectory would require the training we aim to avoid, and the equation it yields still has to be solved to obtain the optimal mixture. To deal with these difficulties, we propose a novel framework, which (1) describes the training dynamics in output space rather than parameter space; (2) rewrites the discrete gradient updates as a continuous-time ODE for analysis; (3) to make this ODE solvable, further freezes the time-varying NTK into a constant; and (4) solves the question numerically to

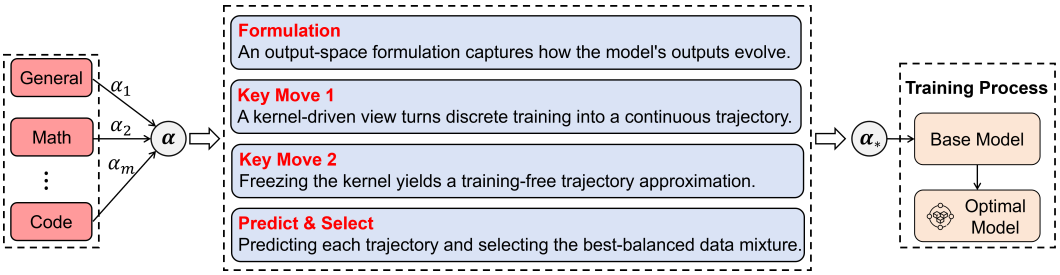

Figure 2: Overview of our framework. Our framework predicts the optimal data mixture and then uses the predicted data mixture to train the target model to obtain the best-performing model on downstream tasks.

yield a concrete prediction. The result is a closed ODE, determined entirely by target model quantities, whose solution is the predicted training trajectory of any candidate mixture. Below, Section 3.1 sets up the preliminaries, Section 3.2 develops the above steps in detail, and Section 3.3 describes the overall testing procedure.

3.1 Preliminaries

Problem definition. We consider adapting a base LLM with parameters θ_0, which already possesses strong general ability, to m candidate domains through further training. A *data mixture* is a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ on the probability simplex, i.e., $\alpha_i \geq 0$ and $\sum_{i=1}^m \alpha_i = 1$, where α_i is the *sampling proportion* of domain i, i.e., the probability that a training sample is drawn from domain i. Training on the mixture α optimizes the α-weighted objective $\mathcal{L}(\alpha; \theta) = \sum_{i=1}^m \alpha_i \bar{\mathcal{L}}_i(\theta)$ over the model parameters θ initialized at θ_0, where $\bar{\mathcal{L}}_i(\theta)$ is the per-token average cross-entropy (CE) loss on domain i. We denote the resulting trained parameters by $\theta(\alpha)$. Let V denote the vocabulary size. A sample a is a token sequence of length L (taken common across samples for notational simplicity, as variable lengths only change the counting), at each position of which the model outputs a V-dimensional logit vector. Stacking these over the L positions gives its logits $f_a \in \mathbb{R}^Q$ of dimension $Q = L \times V$. We write the predicted distribution as $p_a = \mathrm{softmax}(f_a)$, applied per position, and the label as $y_a \in \{0,1\}^Q$, which stacks the L position-wise one-hot labels. The per-sample CE loss $\ell(f_a, y_a) = -y_a^\top \log p_a$ accordingly sums the CE losses of the L predicted positions. The *residual* $r_a = \partial \ell(f_a, y_a)/\partial f_a = p_a - y_a$ is the gradient of this loss with respect to the logits, and $J_a = \partial f_a / \partial \theta$ is the per-sample Jacobian. To measure how a mixture shapes the resulting model, we hold out a *probe set* val of N_{val} samples and evaluate its validation loss $\mathcal{L}_{\text{val}}(\alpha)$, normalized per predicted position so that it coincides with the standard per-token validation CE:

$$\mathcal{L}_{\text{val}}(\alpha) = \frac{1}{M_{\text{val}}} \sum_{a=1}^{N_{\text{val}}} \ell\big(f_a(\theta(\alpha)),\, y_a\big), \qquad M_{\text{val}} = N_{\text{val}}\, L, \tag{1}$$

where M_{val} counts the predicted token positions of the probe set. Throughout, N denotes sample (sequence) counts and M the corresponding counts of predicted positions. Our goal is to find the optimal data mixture $\alpha^\star$ *without* training the target model for each candidate α:

$$\alpha^\star = \arg\min_{\alpha \in \Delta_{m-1}} \mathcal{L}_{\text{val}}(\alpha), \tag{2}$$

where $\Delta_{m-1} = \{\alpha \in \mathbb{R}^m : \alpha_i \geq 0,\ \sum_{i=1}^m \alpha_i = 1\}$ is the probability simplex.

Neural tangent kernel. We next introduce the neural tangent kernel (NTK) (Jacot et al., 2018), theoretical tool at the core of our method. The NTK is a classical tool for making training dynamics analytically tractable. For two samples u and v, the empirical NTK

$$\Theta_{u,v}(\theta_t) = J_u J_v^\top, \qquad J_u = \frac{\partial f_u}{\partial \theta}, \tag{3}$$

is the inner product of the two samples' output gradients. It measures how a gradient step taken on v changes the output of u, propagating the loss gradient from the sample being trained to the sample being observed. As network width grows, the NTK converges to a limit that stays nearly constant throughout training, which is what makes the dynamics analytically tractable. We will use the NTK in following subsections to characterize the model's output dynamics.

3.2 Modeling Training Dynamics in Logit Space

In this section, we introduce our training-dynamics modeling framework. Our goal is to obtain the optimal data mixture for the target model by modeling its training dynamics, without actually training the model. We achieve this in the steps outlined above, and detail each of them below.

(1) Output-space modeling. To characterize the target model's training dynamics in a training-free setting (i.e., without ever running the training), we need to track how some quantity of the model evolves under gradient updates. The quantity must both capture the model's behavior and keep the resulting dynamics tractable. We therefore first need to determine the space in which to describe the dynamics. The two natural candidates are the parameter space and the output space. However, the parameters θ are extremely high-dimensional and hard to characterize directly, whereas what we ultimately care about is the model's output on each sample. We therefore track the logits f rather than tracking the parameters, as logits are directly tied to the training objective and a data mixture enters their dynamics directly as the weights on the per-domain gradients.

To do so, we need to understand how each training update moves the logits. During training the logits evolve as gradient descent updates the parameters, so their trajectory is shaped by the individual gradient steps. To characterize this evolution, we therefore first consider a single gradient step on the mixture-weighted loss $\mathcal{L}(\alpha;\theta) = \sum_{i=1}^{m} \alpha_i \bar{\mathcal{L}}_i(\theta)$ (Section 3.1), which updates the parameters by $\Delta\theta = -\eta\, \partial\mathcal{L}(\alpha;\theta)/\partial\theta$ with learning rate η. Expanding each per-domain CE loss as a per-token average over the N_i samples in domain i, $\bar{\mathcal{L}}_i(\theta) = \frac{1}{M_i}\sum_{b\in i} \ell(f_b, y_b)$ with $M_i = N_i\, L$ the number of predicted positions in domain i, which matches the per-token objective optimized in actual training, its gradient becomes

$$\frac{\partial\mathcal{L}(\alpha;\theta)}{\partial\theta} = \sum_{i=1}^{m} \alpha_i \frac{\partial\bar{\mathcal{L}}_i(\theta)}{\partial\theta} = \sum_{i=1}^{m} \frac{\alpha_i}{M_i} \sum_{b\in i} \frac{\partial\ell(f_b, y_b)}{\partial\theta}. \tag{4}$$

To evaluate Eq. (4), we still need the per-sample gradient $\partial\ell(f_b, y_b)/\partial\theta$. By the chain rule, it splits into two factors from Section 3.1, namely, the residual $r_b = \partial\ell(f_b, y_b)/\partial f_b = \mathrm{softmax}(f_b) - y_b = p_b - y_b$, which measures how the loss changes with the logits, and the Jacobian $J_b = \partial f_b/\partial\theta$, which measures how the logits change with the parameters,

$$\frac{\partial\ell(f_b, y_b)}{\partial\theta} = \Big(\frac{\partial f_b}{\partial\theta}\Big)^{\top} \frac{\partial\ell(f_b, y_b)}{\partial f_b} = J_b^{\top} r_b. \tag{5}$$

Substituting Eq. (5) into Eq. (4) and grouping each domain's samples, a single gradient-descent step updates the parameters by

$$\begin{aligned} \Delta\theta = -\eta \frac{\partial\mathcal{L}(\alpha;\theta)}{\partial\theta} &= -\eta \sum_{i=1}^{m} \frac{\alpha_i}{M_i} \sum_{b\in i} \frac{\partial\ell(f_b, y_b)}{\partial\theta} \\ &= -\eta \sum_{i=1}^{m} \frac{\alpha_i}{M_i} \sum_{b\in i} J_b^{\top} r_b = -\eta \sum_{i=1}^{m} w_i\, J_i^{\top} r_i, \end{aligned} \tag{6}$$

where J_i and r_i stack the per-sample Jacobians and residuals of domain i. The weight $w_i = \alpha_i/M_i$ combines the sampling proportion α_i with the number of predicted positions M_i in domain i, so that α keeps its sampling-proportion semantics regardless of how many probe samples represent each domain. So far, we have obtained the parameter change $\Delta\theta$ of a single step in Eq. (6). To model the dynamics in the logit space, and since each logit f_c is a function of θ, we apply a first-order Taylor expansion to convert $\Delta\theta$ into the logit change of any observed sample c,

$$\begin{aligned} \Delta f_c = f_c(\theta + \Delta\theta) - f_c(\theta) &\approx \frac{\partial f_c}{\partial\theta}\, \Delta\theta = J_c\, \Delta\theta \\ &= -\eta \sum_{i=1}^{m} w_i\, J_c J_i^{\top}\, r_i = -\eta \sum_{i=1}^{m} w_i\, \Theta_{c,i}\, r_i, \end{aligned} \tag{7}$$

where the neural tangent kernel $\Theta_{c,i} = J_c J_i^{\top}$ measures how the update taken on domain i propagates to the logits of sample c. In this step, we have modeled how the target LLM's output space changes under a single discrete update. We next turn this discrete form into a continuous one.

(2) Continuous dynamics. The previous step yields only the discrete, single-step change of the logits in Eq. (7). This discrete form cannot be used directly, because recovering the full logit evolution

from it would require iterating the update over the many optimization steps of actual training, leaving no efficient way to capture the trajectory without paying the training cost we seek to avoid. In this step, our goal is thus to bypass this iterative computation and derive a dynamical equation that is efficient to analyze. The idea is to apply a limit approximation that converts the discrete, step-by-step updates into a differential equation, whose solution traces the full logit trajectory without running the training. Concretely, the discrete update in Eq. (6) defines the iteration $\theta_{t+1} = \theta_t - \eta \sum_{i=1}^{m} w_i \, J_i^\top r_i$, which is exactly the forward-Euler discretization, with unit step size, of the gradient-flow ODE

$$\dot{\theta} = -\eta \sum_{i=1}^{m} w_i \, J_i^\top r_i. \tag{8}$$

We therefore adopt this ODE as the continuous-time model of training, replacing the discrete iterates $\{\theta_t\}$ by a smooth trajectory $\theta(t)$. A formal statement of this discrete-to-continuous transformation is given in Appendix. By the chain rule, the logits of any sample c evolve as $\dot{f}_c = J_c \dot{\theta}$. Substituting Eq. (8) into $\dot{f}_c = J_c \dot{\theta}$, the NTK $\Theta_{c,i} = J_c J_i^\top$ arises:

$$\dot{f}_c = -\eta \sum_{i=1}^{m} w_i \, \Theta_{c,i}(\theta_t) \, r_i, \tag{9}$$

where $c \in \{1, \ldots, m, \mathrm{val}\}$ ranges over the m source domains and the probe set (val), and $\Theta_{c,i}(\theta_t) = J_c J_i^\top$ is the NTK evaluated at the current parameters θ_t, which propagates the loss gradient from the trained samples to the observed sample c and varies over time as θ moves. We emphasize that the mixture α enters, through w_i, the dynamics equation itself rather than as a post-hoc weighting of a fixed trajectory: changing α changes the motion of every sample's logits and the mutual interaction among source domains. In this step, we have converted the discrete update into a continuous-time ODE that governs the coupled logit dynamics through a time-varying NTK coefficient $\Theta(\theta_t)$. Below, we further simplify this coefficient to turn the ODE into a solvable form.

(3) ODE simplification. The previous step yields the continuous-time ODE in Eq. (9), in which the evolution of the logits is driven by the coefficients on its right-hand side, whose core is the NTK coefficient $\Theta(\theta_t)$. This coefficient is time-varying, changing as training proceeds, so solving the equation would still require training the model to obtain $\Theta(\theta_t)$, which contradicts our original goal of estimating the optimal data mixture in a training-free manner. In this step, our goal is thus to simplify this coefficient and reduce the ODE to a solvable form. A natural remedy is to appeal to classic NTK theory, under which an infinitely wide network keeps its kernel exactly constant throughout training, so that $\Theta(\theta_t)$ could be replaced by a single constant kernel and the ODE would be converted into a directly solvable constant-coefficient form. That constancy, however, relies on infinite width and Gaussian initialization, which do not hold for an LLM during training, leaving the constant-kernel condition invalid in our setting. Previous work (Wang et al., 2026) has shown that although the NTK is not strictly constant when fine-tuning from a strong base model, its *directional* structure remains stable throughout training. Since the base model is already well trained, its representations at θ_0 provide a foundation for adaptation, so continued training builds on these existing representations rather than learning from scratch. The NTK's magnitude can change while its directional structure remains stable (Wang et al., 2026). This picture is also corroborated by kernel- and tangent-space analyses showing that language-model fine-tuning stays close to the model's linearization at initialization (Malladi et al., 2023; Ortiz-Jimenez et al., 2023),and by transfer-learning studies reporting that fine-tuning largely reuses pretrained features rather than forming new ones (Neyshabur et al., 2020; Kumar et al., 2022).Concretely, throughout fine-tuning, the running kernel stays almost aligned with the base NTK at initialization, and their Frobenius cosine similarity stays within a small tolerance δ of one (more analysis in Appendix J). We thus turn this into explicit stability conditions on the kernel. Writing $\Theta^0 := \Theta(\theta_0)$ for the base NTK at initialization and $[0, T]$ for the training horizon, we can then take the direction of the running kernel to remain roughly unchanged from that of Θ^0 along the whole trajectory, drifting only within a small tolerance:

$$\cos_F\big(\Theta(\theta_t),\, \Theta^0\big) := \frac{\langle \Theta(\theta_t),\, \Theta^0 \rangle_F}{\|\Theta(\theta_t)\|_F \, \|\Theta^0\|_F} \;\geq\; 1 - \delta, \qquad \forall\, t \in [0, T], \tag{10}$$

where $\langle A, B \rangle_F = \mathrm{tr}(A^\top B)$ denotes the Frobenius inner product, and $\|\cdot\|_F$ denotes the norm. With the kernel direction nearly unchanged, we can approximate the dynamics using the initial kernel Θ^0. The following proposition bounds the error in the dynamics equation for all source domains and the probe set.

Proposition 1 (NTK freezing under directional stability)**.** *Stack the logits of the m source domains and the probe set as $f = (f_1; \ldots; f_m; f_{\mathrm{val}})$, so that the coupled dynamics of Eq. (9) read $\dot{f} = -\eta\,\Theta(\theta_t)\,\tilde{\gamma}(f)$, where $\Theta(\theta_t)$ is the full stacked kernel and $\tilde{\gamma}(f) := \nabla_f \mathcal{L}(\alpha; f) = \big(w_1 r_1;\ \ldots;\ w_m r_m;\ \mathbf{0}\big)$ collects the weighted residuals, with zero rows for the probe block since* val *does not enter the training loss. Considering the direction-stability in Eq. (10) holds on $[0, T]$, with $0 \le \delta < 1$, the reparameterized time $s(t) = \int_0^t \rho(\xi)\, d\xi$, with projection coefficient $\rho(t) = \langle \Theta(\theta_t), \Theta^0 \rangle_F / \|\Theta^0\|_F^2 > 0$, is strictly increasing, and the dynamics expressed in s consist of the initial-kernel term and an equation error $\zeta(s)$:*

$$\begin{aligned} \frac{df}{ds} &= -\eta\,\Theta^0\,\tilde{\gamma}\big(f(t(s))\big) + \zeta(s), \\ \|\zeta(s)\|_2 \;&\le\; \eta\,\|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta}\,\big\|\tilde{\gamma}\big(f(t(s))\big)\big\|_2. \end{aligned} \tag{11}$$

Thus, under directional stability, we replace the changing kernel with the initial kernel Θ^0 and describe the logit dynamics in terms of training progress s. The bound above controls the error introduced into the dynamics equation by this replacement. The experiments also show that the predicted logits track the actual logits well, with further details in Appendix L. Motivated by this, we use the frozen-kernel ODE as a predictor for each candidate mixture, computing the kernel only once before training.

The proof and analysis of the equation error term and its effect on the predicted trajectories are provided in Appendix D and Appendix E. Based on Proposition 1, substituting Θ^0 into Eq. (9) reduces the coupled dynamics to a constant-coefficient ODE:

$$\frac{df_c}{ds} = -\eta \sum_{i=1}^{m} w_i\,\Theta^0_{c,i}\,r_i, \tag{12}$$

where $\Theta^0_{c,i}$ denotes the base model's initial NTK, evaluated at θ_0. The NTK is therefore computed before any training, and it generates the logit trajectory of every candidate mixture, letting us compare mixtures in a training-free manner.

Solving Eq. (12) relies on the base kernel Θ^0. However, computing this kernel is prohibitively expensive, as it requires each sample's Jacobian J_a, whose size scales with both the billions of model parameters and the high-dimensional logit outputs, making its construction and storage intractable at LLM scale. We thus compress these two dimensions in two steps. More details are in Appendix K.

First, we replace the full Jacobian kernel with the normalized summed-gradient estimate

$$\widetilde{\Theta}_{u,v}(\theta_t) = \frac{1}{Q}\Big\langle \nabla_\theta \sum_{k=1}^{Q} f_{u,k}(\theta_t),\ \nabla_\theta \sum_{k=1}^{Q} f_{v,k}(\theta_t) \Big\rangle, \tag{13}$$

which requires one backward for gradient calculation (but without needing parameter update) pass per sample instead of Q, while the $1/Q$ factor normalizes the inner product of the summed-output gradients. More details are provided in Sec. F.

Second, let $g_u := \frac{1}{\sqrt{Q}} \nabla_\theta \sum_{k=1}^{Q} f_{u,k}(\theta_0) \in \mathbb{R}^P$, so that $\widetilde{\Theta}_{u,v} = \langle g_u, g_v \rangle$. We use the subsampled randomized Hadamard transform (SRHT) (Ailon & Chazelle, 2006; Tropp, 2011) to compress g_u into $\hat{g}_u \in \mathbb{R}^p$ and compute the kernel. More details are provided in Sec. G.

Overall, the two steps require only one backward pass and one SRHT projection per sample, enabling the frozen kernel computable at the full-parameter scale.

(4) Numerical solving. The previous step reduces the coupled dynamics to the constant-coefficient ODE of Eq. (12), with the required kernel computations now feasible at the full-parameter scale. However, because the cross-entropy residual $r_i = \mathrm{softmax}(f_i) - y_i$ is nonlinear, this ODE has no closed-form solution. To obtain the corresponding logit trajectory, in this step we aim to integrate the ODE efficiently using an adaptive numerical solver. The source-domain logits form a closed subsystem, while the probe logits are driven by this subsystem. Using an effective time variable τ, we solve the resulting dynamics

$$\frac{df_c}{d\tau} = -\sum_{i=1}^{m} w_i\,\Theta^0_{c,i}\,r_i. \tag{14}$$

We treat the integration horizon as a hyperparameter shared by all candidate mixtures. The probe loss obtained from the frozen-kernel ODE serves as the ranking score for candidate mixtures. Appendix E discusses how this shared horizon relates to the mixture-dependent effective time in Proposition 1, and Appendix O evaluates horizon sensitivity and ranking consistency.

To solve Eq. (14) numerically, a naive choice is to use a fixed-step solver. However, it is inefficient for these fast-then-slow dynamics because it must use small steps to resolve the rapid initial changes, while such steps become unnecessary as the trajectory flattens. To tackle these issues, we use adaptive RK45 (Dormand & Prince, 1980), which estimates the local integration error and automatically takes small steps during rapid changes and larger steps near convergence.

Overall, we generate predicted logit trajectories for ranking candidate mixtures by numerically solving this ODE. This requires computing the base logits and frozen NTK once, without training the target model.

3.3 Overall Procedure for Mixture Selection

We compute the mixture-independent base logits and frozen kernel once and reuse them for all candidates. Full pseudocode is provided in Appendix.

Input. A base model θ_0, m source domains, a held-out probe set val, candidate mixtures $\mathcal{A}$, and an integration horizon τ_{end}.

1. **Mixture-independent precomputation.** Compute the initial source and probe logits and the frozen kernel blocks at θ_0.
2. **Per-mixture trajectory prediction.** For each $\alpha \in \mathcal{A}$, form $w_i = \alpha_i / M_i$ and solve Eq. (14) up to τ_{end} using RK45.
3. **Predicted validation loss.** Evaluate $\mathcal{L}_{\text{val}}(\alpha)$ from the probe logits at τ_{end}.
4. **Mixture search.** Select the candidate with the lowest predicted validation loss, as in Eq. (2).

Output. The estimated optimal mixture $\alpha^\star$.

4 Experiments

Datasets. To evaluate the efficacy of our proposed method, we consider two data-mixing settings following prior work: (i) continual pre-training over seven general, mathematics, and code categories, comprising one general set, three mathematics subsets, and three code subsets drawn from the corresponding released corpora (Li et al., 2026), and (ii) supervised fine-tuning over six Tulu3 skill domains, namely general, knowledge recall, math, code, precise instruction following, and safety (Li et al., 2025). More details are provided in Appendix B.

Evaluation metrics and baselines. Following Li et al. (2026; 2025), we evaluate each method from three perspectives: (1) *ranking consistency*, measured by Spearman's ρ over all candidate mixtures and over the top 25% with the lowest true validation losses; (2) *selection quality*, measured by the downstream performance obtained after training the target model with the selected mixture; and (3) *search cost*, measured in equivalent training tokens (following Li et al. (2026), reported in units of B). For pre-training, we compare with Uniform (Li et al., 2026), Heuristic (Li et al., 2026), Random, Inverse-performance, RegMix (Liu et al., 2025), CLIMB (Diao et al., 2026), DeMix (Li et al., 2026), and FastMix (Tan et al., 2026). For SFT, we compare with Original (Li et al., 2025), Equal_T (Li et al., 2025), Equal_I (Li et al., 2025), and SFT-DMO (Li et al., 2025). Ranking-consistency results and details of rank normalization and baseline configurations are provided in Appendix.

Implementation details. Following Li et al. (2026) and Li et al. (2025), we use Qwen3-1.7B for continual pretraining and Llama3.1-8B and Qwen2.5-32B for SFT. To construct the frozen kernel, we sample 128 examples per domain. The probe set contains 5% of the samples from each domain. We set the SRHT projection dimension to 1,024. More implementation details are in Appendix.

Table 1: Comparison of the optimal data mixture performance from different data mixing methods on the pre-training scenario. The best average rank score across all methods is marked in bold.

Method	Search cost (B) ↓	Benchmarks (%) ↑												Rank ↓			
		General						Code			Math						
		ARC-E	HellaSwag	PIQA	SIQA	WinoGrande	Avg.	MBPP	HumanEval	Avg.	GSM8K	MATH	Avg.	General	Code	Math	**Macro Avg.**
Uniform	0	71.34	54.33	73.18	40.52	55.67	59.01	18.50	18.19	18.34	12.50	6.74	9.62	9	44	57	36.67
Heuristic	0	69.57	53.72	71.75	40.32	55.27	58.13	24.27	24.90	24.58	18.20	10.16	14.18	94	5	28	42.33
Random	0	70.37	53.85	72.58	40.48	55.41	58.54	21.80	24.39	23.10	19.86	10.64	15.25	66	9	18	31.00
Inverse-performance	0	70.33	53.96	72.36	40.43	55.41	58.50	21.80	22.56	22.18	16.15	8.88	12.51	66	18	39	41.00
RegMix (Liu et al., 2025)	448	71.43	54.55	73.54	40.76	55.63	59.18	18.33	21.85	20.09	15.30	7.96	11.63	2	32	50	28.00
CLIMB (Diao et al., 2026)	448	70.31	54.19	73.22	40.40	55.57	58.74	19.53	22.66	21.10	20.53	11.61	16.07	47	25	11	27.67
DeMix (Li et al., 2026)	212	71.08	53.78	72.94	40.63	55.43	58.77	20.70	24.29	22.49	20.98	10.55	15.76	46	12	14	24.00
FastMix (Tan et al., 2026)	1.1	70.02	54.17	72.91	41.81	53.28	58.44	24.90	18.29	21.60	11.37	5.54	8.46	67	22	60	49.67
Ours	0.9	69.84	52.75	72.63	42.37	55.33	58.58	40.86	23.78	32.32	28.89	10.56	19.72	64	1	1	**22.00**

Table 2: Downstream performance of the mixtures selected by different data mixing methods on the SFT mixing scenario, which re-weights the Tulu3 collection.

Model	Method	AGIEval	IFEval	MMLU	GSM8K	HumanEval	Safety	Average
Llama3.1-8B	Original	33.9	54.5	56.4	65.3	47.0	43.9	50.1
	Equal_T	32.1	51.9	55.8	65.2	48.2	41.4	49.1
	Equal_I	32.8	42.7	54.9	61.9	50.0	43.9	47.7
	SFT-DMO (Li et al., 2025)	36.1	48.2	56.6	62.3	53.7	45.3	50.4
	Ours	34.7	57.3	57.3	75.3	60.4	40.8	**54.3**
Qwen2.5-32B	Original	49.8	57.7	80.2	89.5	55.5	53.5	64.4
	Equal_T	55.8	61.2	81.0	89.8	61.6	56.4	67.6
	Equal_I	57.4	64.3	80.9	91.4	62.8	54.3	68.5
	SFT-DMO (Li et al., 2025)	58.3	61.3	81.0	90.8	60.5	56.1	68.0
	Ours	61.4	64.9	81.4	93.3	67.7	63.7	**72.0**

4.1 Main Results

In our experiments, we compare our method with existing data mixing methods in pre-training and SFT. The results are reported in Tables 1 and 2, respectively. Our method achieves the best macro-averaged rank in pre-training. For SFT, our method achieves the best average performance on both models, showing its effectiveness.

4.2 Ablation Studies and Further Analyses

In this section, we study the effect of the SRHT projection dimension p. We evaluate three variants with $p \in \{512, 1024, 2048\}$. Selecting by the lowest validation loss yields the same mixture for $p = 1024$ and $p = 2048$, but a different one for $p = 512$. Table 3 reports the downstream performance of the selected mixtures. We therefore use $p = 1024$, the lower dimension with the same selected mixture as $p = 2048$. **More ablations and and additional analyses are provided in Appendix.**

Table 3: Ablation on the SRHT projection with dimension p.

SRHT p	General Avg. (↑)	Code Avg. (↑)	Math Avg. (↑)
512	58.28	32.31	19.13
1024	58.58	32.32	19.72
2048	58.58	32.32	19.72

5 Conclusion

In this work, we proposed a novel method that directly models the target model's training dynamics, characterizing how the target model evolves during training. Building on this, our method estimates the optimal data mixture in a fully training-free manner, without training small proxy models or the target model itself. Extensive experiments demonstrate the efficacy of our method.

AI USE STATEMENT

In this work, we used generative AI tools only to polish the writing, i.e., grammar checking and wording refinements on text written by the authors. We did not use generative AI tools for research ideation, method design, or experiment design. All AI-assisted text was checked by the authors.

ETHICS STATEMENT

This work studies training-free data mixture optimization for large language models. Our purpose is to reduce the computational cost of selecting data mixtures for continual pre-training and supervised fine-tuning. Our experiments use existing models and publicly available datasets and do not involve human-subject studies or the collection of new personal data.

REPRODUCIBILITY STATEMENT

Our method is described in Sec. 3, with full pseudocode and computational cost analysis in Appendix Sec. H. The proof of Proposition 1 is provided in Appendix Sec. D, and supporting derivations and analyses of the continuous-time dynamics and kernel construction are given in Appendix Secs. C, F, and G. The target models and experimental settings are described in Sec. 4, with dataset and probe-set details in Appendix Sec. B. Training hyperparameters, hardware, kernel computation, and numerical solver settings are reported in Appendix Sec. A, and the evaluation metrics are detailed in Appendix Sec. I.

REFERENCES

Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 557–563, 2006.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training. *arXiv preprint arXiv:2312.02406*, 2023.

Lior Belenki, Alekh Agarwal, Tianze Shi, and Kristina Toutanova. Optimizing pre-training data mixtures with mixtures of data expert models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32570–32587, 2025.

Davide Berasi, Matteo Farina, Massimiliano Mancini, and Elisa Ricci. Linear model merging unlocks simple and scalable multimodal data mixture optimization. *arXiv preprint arXiv:2602.04937*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Mayee Chen, Michael Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. In *International Conference on Learning Representations*, volume 2025, pp. 46089–46132, 2025.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four {gpu} hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=Cnon5ezMHtu.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, et al. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. *Advances in Neural Information Processing Systems*, 38, 2026.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. *arXiv preprint arXiv:2310.15393*, 2023.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Bimix: A bivariate data mixing law for language model pretraining. *arXiv preprint arXiv:2405.14908*, 2024.

Jiawei Gu, Zacc Yang, Chuanghao Ding, Rui Zhao, and Fei Tan. Cmr scaling law: Predicting critical mixture ratios for continual pre-training of language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16143–16162, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgndT4KwB.

Zixuan Hu, Xiaotong Li, Shixiang Tang, Jun Liu, Yichun Hu, and Ling-Yu Duan. Lead: Exploring logit space evolution for model selection. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28664–28673. IEEE, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

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*, 2026. URL https://openreview.net/forum?id=uyRIOjFgOn.

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

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 *International Conference on Learning Representations*, volume 2025, pp. 38305–38339, 2025.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23610–23641. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/malladi23a.html.

Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. A fast, well-founded approximation to the empirical neural tangent kernel. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25061–25081. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/mohamadi23a.html.

Michael Murray, Hui Jin, Benjamin Bowman, and Guido Montufar. Characterizing the spectrum of the NTK via a power series expansion. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Tvms8xrZHyR.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=l-PrrQrK0QR.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=0A9f2jZDGW.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/park23c.html.

Haoran Que, Jiaheng Liu, Ge Zhang, Chenchen Zhang, Xingwei Qu, Yinghao Ma, Feiyu Duan, Zhiqi Bai, Jiakai Wang, Yuanxing Zhang, et al. D-cpt law: Domain-specific continual pre-training scaling law for large language models. *Advances in Neural Information Processing Systems*, 37: 90318–90354, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haebin Shin, Lei Ji, Xiao Liu, Zhiwei Yu, Hyunwoo Yoo, Qi Chen, and Yeyun Gong. Dynamixsft: Dynamic mixture optimization of instruction tuning collections. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39590–39603, 2026.

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *Advances in Neural Information Processing Systems*, 38:129554–129579, 2026.

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.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

Haonan Wang, Wei Huang, Ziwei Wu, Hanghang Tong, Andrew J Margenot, and Jingrui He. Deep active learning by leveraging training dynamics. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=aJ5xc1QB7EX.

Pingjie Wang, Hongcheng Liu, Yusheng Liao, Ziqing Fan, Yaxin Du, Shuo Tang, Yanfeng Wang, and Yu Wang. Mining useful general data for low-resource domain adaptation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=y1AV6cX8dX.

Yite Wang, Dawei Li, and Ruoyu Sun. NTK-SAP: Improving neural network pruning by aligning training dynamics. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-5EWhW_4qWP.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PG5fV50maR.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Bx6qKuBM2AD.

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 *International Conference on Learning Representations*, volume 2025, pp. 82263–82287, 2025.

Xinyue Zeng, Haohui Wang, Junhong Lin, Jun Wu, Tyler Cody, and Dawei Zhou. LensLLM: Unveiling fine-tuning dynamics for LLM selection. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=om0CcjvEQh.

Wenzhen Zheng, Wenbo Pan, Xu Xu, Libo Qin, Li Yue, and Ming Zhou. Breaking language barriers: Cross-lingual continual pre-training at scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7725–7738, 2024.

APPENDIX

A ADDITIONAL IMPLEMENTATION DETAILS

Target models. For the pre-training mixing scenario, we follow Li et al. (2026) and use Qwen3-1.7B as the target model. For the SFT mixing scenario, we follow Li et al. (2025) and use Llama3.1-8B and Qwen2.5-32B as target models.

Training hyper-parameters for the ground-truth runs. We follow the configuration of each setting. For pre-training mixing we use batch size 512, sequence length 8192, and an initial learning rate of 3×10^{-4}. For SFT mixing we use batch size 256, sequence length 4096, and 6000 steps, with a learning rate of 1×10^{-5} for Llama3.1-8B and 2×10^{-6} for Qwen2.5-32B. Both settings decay the learning rate on a cosine schedule down to 20% of its initial value. All experiments were done on NVIDIA H200 GPUs.

Kernel construction. All quantities entering the frozen ODE are evaluated at the base parameters θ_0. For both the pre-training and SFT mixing scenarios, we draw 128 samples per source domain and construct the probe set using 5% of the samples from each domain. In both scenarios each sample is truncated to a maximum length of 1024 tokens. For every sampled example we run one forward pass to obtain the base logits $f(\theta_0)$ and the corresponding backward pass (with gradient calculation only, without parameter update) to obtain the summed-output gradient $g_u = \frac{1}{\sqrt{Q}} \nabla_\theta \sum_{k=1}^{Q} f_{u,k}(\theta_0)$ of Eq. (13). Both the base logits $f(\theta_0)$ and the gradients g_u are obtained by forward and backward passes alone, without training the model.

Random projection. The summed-output gradients are compressed by the SRHT of Sec. 3.2 (Ailon & Chazelle, 2006; Tropp, 2011). We pad each gradient to the next power of two and apply the transform one shard at a time, so the projection matrix is never formed. We use $p = 1024$ in all main experiments, justified in Sec. G and ablated in Tab. 3. We fix the random seed, and all candidate mixtures are scored under the same kernel.

ODE solver. We solve Eq. (14) using adaptive RK45 (Dormand & Prince, 1980), with relative and absolute tolerances of 10^{-6} and 10^{-8}. For the cosine learning-rate schedule, we define the reference horizon as $\tau_{\mathrm{ref}} = \int_0^T \eta(t)\, dt = 0.6\, \eta_0 T$. The solver's integration horizon is $\tau_{\mathrm{end}} = c_\tau \tau_{\mathrm{ref}}$, where c_τ is a shared horizon hyperparameter. We use $c_\tau = 1$ in all main experiments. Within each setting, every candidate mixture is evaluated at the same horizon. Using a common horizon places all candidate mixtures on the same effective-progress scale, so that their predicted validation losses can be compared directly; its stability is discussed in Remark 3 and analyzed in Appendix O. Ground-truth training outcomes are used only to evaluate the method. Dense output returns the full trajectory in a single solve.

Mixture search. We draw 100 candidate mixtures uniformly at random from the simplex Δ_{m-1}, and score each by one ODE solve as described in Sec. 3.3. The ground-truth correspondence between mixtures and performance, used to compute the ranking metrics of Sec. I, is obtained by training the target model per candidate mixture.

Baselines. We run the baselines of the main paper under the settings above. We sweep the proxy count and the per-proxy budget for each baseline and report its best-performing setting in Tab. 1: 56 proxies of 8B tokens each for RegMix and CLIMB, and 224 merged experts (7 domain experts pre-trained for 30B tokens each) for DeMix. We adapt FastMix to our experimental setting. We take the Heuristic proportions from Li et al. (2026) and the Original, Equal_T, and Equal_I proportions from Li et al. (2025).

We add two baselines that use no search. Random draws a mixture from a Dirichlet distribution centred on the Heuristic proportions ($\kappa = 50$, smoothed with $\varepsilon = 0.02$ of the uniform mixture). Inverse-performance evaluates the base model on every domain and allocates more data to the domains where it performs worse.

B Additional Details about the Datasets and the Probe Set

Pre-training mixing. Following Li et al. (2026), we search the mixture over $m = 7$ categories, namely one general set together with three mathematics subsets and three code subsets, drawn from the corresponding subsets of the released corpora. The probe set holds out 5% of each category and is never used during training.

SFT mixing. Following Li et al. (2025), we re-weight the Tulu3 collection over six skill domains, namely general, knowledge recall, math, code, precise instruction following, and safety. The probe set contains 5% of the samples from each domain and is never used during training.

C Additional Details about the Discrete-to-Continuous Step in the Main Paper

In step (2) of Sec. 3.2, we adopt the gradient-flow ODE of Eq. (8) as the continuous-time model of the discrete iteration $\theta_{t+1} = \theta_t - \eta \sum_{i=1}^{m} w_i J_i^\top r_i$ defined by Eq. (6). This appendix states formally in which sense the two agree. The discrete iteration is exactly the forward-Euler discretization of the flow with unit step size, so the standard first-order accuracy of that scheme applies, with the effective horizon $\tau = \eta k$ playing the role of physical time.

Proposition 2 (Gradient flow tracks gradient descent to first order)**.** *Assume $\nabla_\theta \mathcal{L}$ is Λ-Lipschitz and $\|\nabla_\theta \mathcal{L}\| \le G$ on a region containing the two trajectories below. Let $\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k)$ be the gradient-descent iterates, and let $\theta(\cdot)$ solve the gradient flow $\dot\theta = -\eta \nabla_\theta \mathcal{L}(\theta)$ of Eq. (8) with $\theta(0) = \theta_0$, time being counted in optimizer steps. Then, for every step count k,*

$$\|\theta_k - \theta(k)\| \;\le\; \frac{\eta G}{2}\left(e^{\eta \Lambda k} - 1\right) \;=\; \frac{\eta G}{2}\left(e^{\Lambda \tau} - 1\right), \qquad \tau := \eta k.$$

At any fixed effective horizon τ the deviation is thus $O(\eta)$, and the logits inherit the same rate through their Lipschitz dependence on θ.

Proof. Let $e_k := \|\theta_k - \theta(k)\|$, so $e_0 = 0$. Along the flow, $\|\ddot\theta\| = \eta^2 \|\nabla_\theta^2 \mathcal{L}\, \nabla_\theta \mathcal{L}\| \le \eta^2 \Lambda G$, hence the one-step defect of the flow against the Euler update obeys $\|\theta(k+1) - \theta(k) + \eta \nabla_\theta \mathcal{L}(\theta(k))\| = \left\| \int_k^{k+1} (\dot\theta(s) - \dot\theta(k))\, ds \right\| \le \frac{1}{2}\eta^2 \Lambda G$. Subtracting the two updates and using the Lipschitz property, $e_{k+1} \le (1 + \eta\Lambda)\, e_k + \frac{1}{2}\eta^2 \Lambda G$; unrolling the recursion gives $e_k \le \frac{\eta G}{2}\left((1+\eta\Lambda)^k - 1\right) \le \frac{\eta G}{2}\left(e^{\eta \Lambda k} - 1\right)$. □

Two consequences are worth recording. First, the bound is stated at a fixed effective horizon $\tau = \eta k$ rather than at a fixed step count, which is the same variable in which the frozen ODE of Eq. (14) is integrated; halving the learning rate and doubling the number of steps leaves τ unchanged and halves the deviation. Second, at a fixed parameter state, the gradient-flow vector field depends linearly on α, since α enters Eq. (8) only through the weights $w_i = \alpha_i / M_i$ that multiply the per-domain gradients. Under the frozen-kernel approximation, different mixtures are represented by different weights on the same initial-kernel blocks.

Remark 1 (Robustness across optimizers)**.** *Proposition 2 establishes the discrete-to-continuous connection for gradient descent, providing the theoretical foundation for our continuous-time dynamics. In practice, we find that the mixture-ranking signal captured by the same dynamics remains highly consistent across SGD, Adam, and AdamW. Appendix O verifies this robustness empirically under otherwise identical training settings.*

D Proof of Proposition 1 in the Main Paper

We restate the proposition for convenience and then prove it. The argument follows the reparameterization used by Wang et al. (2026) to justify freezing the kernel of a fine-tuning LLM at its initial value, which we adapt to our setting: the kernel here is the *full stacked kernel* of the mixture system, whose cross-domain and probe blocks describe how a step taken on one domain propagates to another, rather than a single-task fine-tuning kernel.

Proposition 1 (NTK freezing under directional stability, restated)**.** *Stack the logits of the m source domains and the probe set as $f = (f_1; \dots; f_m; f_{\mathrm{val}})$, so that the coupled dynamics of Eq. (9) read $\dot{f} = -\eta\,\Theta(\theta_t)\,\tilde{\gamma}(f)$, where $\Theta(\theta_t)$ is the full stacked kernel and $\tilde{\gamma}(f) := \nabla_f \mathcal{L}(\alpha; f) = \big(w_1 r_1;\ \dots;\ w_m r_m;\ \mathbf{0}\big)$ collects the weighted residuals, with zero rows for the probe block since* val *does not enter the training loss. Considering the direction-stability in Eq. (10) holds on $[0, T]$, with $0 \le \delta < 1$, the reparameterized time $s(t) = \int_0^t \rho(\xi)\,d\xi$, with projection coefficient $\rho(t) = \langle \Theta(\theta_t), \Theta^0 \rangle_F / \|\Theta^0\|_F^2 > 0$, is strictly increasing, and the dynamics expressed in s consist of the initial-kernel term and an equation error $\zeta(s)$:*

$$\frac{df}{ds} = -\eta\,\Theta^0\,\tilde{\gamma}\big(f(t(s))\big) + \zeta(s),$$

$$\|\zeta(s)\|_2 \;\le\; \eta\,\|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta}\,\big\|\tilde{\gamma}\big(f(t(s))\big)\big\|_2.$$

Thus, under directional stability, we replace the changing kernel with the initial kernel Θ^0 and describe the logit dynamics in terms of training progress s. The bound above controls the error introduced into the dynamics equation by this replacement. The experiments also show that the predicted logits track the actual logits well, with further details in Appendix L. Motivated by this result, we use the frozen-kernel ODE as a predictor for each candidate mixture, computing the kernel only once before training.

Proof. Decompose the running kernel orthogonally, in the Frobenius geometry, along the base kernel:

$$\Theta(\theta_t) = \rho(t)\,\Theta^0 + E_t, \qquad \rho(t) = \frac{\langle \Theta(\theta_t), \Theta^0 \rangle_F}{\|\Theta^0\|_F^2}, \qquad \langle E_t, \Theta^0 \rangle_F = 0,$$

which is the least-squares projection of $\Theta(\theta_t)$ onto $\mathrm{span}(\Theta^0)$, so that $\rho(t) = \arg\min_{a \in \mathbb{R}} \|\Theta(\theta_t) - a\Theta^0\|_F$ and the residual E_t is orthogonal to Θ^0 by the normal equations. Let ϕ_t denote the Frobenius angle, so that $\cos\phi_t = \cos_F(\Theta(\theta_t), \Theta^0) \ge 1 - \delta > 0$ by Eq. (10). Then $\rho(t) = \frac{\|\Theta(\theta_t)\|_F}{\|\Theta^0\|_F}\cos\phi_t > 0$, so $s(t) = \int_0^t \rho(\xi)\,d\xi$ is strictly increasing, and by the Pythagorean identity $\|E_t\|_F = \|\Theta(\theta_t)\|_F \sin\phi_t$. Substituting the decomposition into $\dot{f} = -\eta\,\Theta(\theta_t)\,\tilde{\gamma}$ and changing time by $ds = \rho(t)\,dt$,

$$\frac{df}{ds} = \frac{1}{\rho}\,\dot{f} = -\eta\,\Theta^0\,\tilde{\gamma} - \eta\,\frac{E_t}{\rho}\,\tilde{\gamma} \;=:\; -\eta\,\Theta^0\,\tilde{\gamma} + \zeta(s).$$

The defect factor is controlled by

$$\Big\|\frac{E_t}{\rho}\Big\|_F = \frac{\|\Theta(\theta_t)\|_F\,\sin\phi_t}{\big(\|\Theta(\theta_t)\|_F / \|\Theta^0\|_F\big)\cos\phi_t} = \|\Theta^0\|_F\,\tan\phi_t \;\le\; \|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta},$$

where the last step uses $\sin\phi_t \le \sqrt{1-(1-\delta)^2} \le \sqrt{2\delta}$ and $\cos\phi_t \ge 1 - \delta$. Since the spectral norm is dominated by the Frobenius norm, $\|\zeta(s)\|_2 \le \eta\,\|E_t/\rho\|_2\,\|\tilde{\gamma}\|_2 \le \eta\,\|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta}\,\|\tilde{\gamma}\|_2$. This proves the bound on the error term $\zeta(s)$ stated in the proposition. □

We apply Proposition 1 to both the source domains and the probe set. The same argument extends directly to a time-varying learning rate $\eta(t)$: changing time by $d\tau = \eta(t)\rho(t)\,dt$ gives $df/d\tau = -\Theta^0\tilde{\gamma} + \zeta$ with $\|\zeta\|_2 \le \|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta}\,\|\tilde{\gamma}\|_2$, so the effective progress at budget T is $\tau^\star(T) = \int_0^T \eta(t)\rho(t)\,dt$, and Corollary 1 holds with τ_s replaced by τ. The resulting bound controls the error in the dynamics equation; Appendix E derives the corresponding bound on the predicted logit trajectories.

Remark 2 (Size of the driving vector)**.** *The defect bound scales with $\|\tilde{\gamma}\|_2 = (\sum_{i=1}^m w_i^2 \|r_i\|^2)^{1/2}$, which admits a bound depending only on the mixture and the cohort sizes. Each predicted position contributes a residual of norm at most $\sqrt{2}$, since $\|p - y\|_2^2 = \|p\|_2^2 - 2p_y + 1 \le 2(1 - p_y) \le 2$ for any probability vector p and one-hot y; a length-L sample therefore has $\|r_b\| \le \sqrt{2L}$, and summing over the N_i samples of domain i gives $\|r_i\|^2 \le 2N_iL = 2M_i$, so that with $w_i = \alpha_i/M_i$,*

$$\|\tilde{\gamma}\|_2 \;\le\; \Big(2\sum_{i=1}^m \alpha_i^2/M_i\Big)^{1/2} =: \bar{\Gamma},$$

with $M_i = N_i L$ the per-domain position counts of Eq. (4). The bound is largest for mixtures concentrated on a single small domain and smallest for the uniform mixture, and it is computable before any solve.

E Additional Details on the Error of the Frozen-Kernel Prediction

Proposition 1 bounds the defect injected into the *equation*; the quantity consumed downstream is the deviation of the *solution* of the frozen system, integrated from the same initial condition with the defect dropped. The gap between the two is controlled by a Grönwall argument whose rate is the Lipschitz constant of the frozen right-hand side, which we bound first.

Lemma 1 (Spectral bound on the frozen source system)**.** *Collect the source blocks of Eq. (12) as $\dot f_S = -\eta\,\Theta^0_{SS} D_w\, r_S(f_S)$, where D_w is diagonal with entry w_i on the rows of block i and Θ^0_{SS} denotes the frozen source kernel, in whichever representation the ODE uses (the full stacked kernel, or the compressed scalar kernel acting identically on the Q coordinates). Then, for every f_S,*

$$\Big\|\frac{\partial \dot f_S}{\partial f_S}\Big\|_2 \;\le\; \eta\, L_\star, \qquad L_\star := \tfrac{1}{2}\,\big(\max_i w_i\big)\,\lambda_{\max}\big(\Theta^0_{SS}\big). \tag{15}$$

Proof. In vectorized form the Jacobian factors as $-\eta\,(\Theta^0_{SS} D_w)\,\widehat S$ (with a Kronecker factor I_Q in the scalar-kernel case, which does not change spectral norms), where $\widehat S$ is the block-diagonal collection of the per-position softmax Jacobians $\mathrm{diag}(p) - pp^\top$. For any probability vector p and any vector x, $x^\top(\mathrm{diag}(p) - pp^\top)x = \mathrm{Var}_p(x) \le \frac{1}{4}(\max_k x_k - \min_k x_k)^2 \le \frac{1}{2}\|x\|^2$, so $\|\widehat S\|_2 \le \frac{1}{2}$. Combining this with $\|\Theta^0_{SS} D_w\|_2 \le \lambda_{\max}(\Theta^0_{SS})\,\max_i w_i$ proves Eq. (15). For the projected source Gram matrix, $\lambda_{\max}(G_S G_S^\top) = \sigma_{\max}(G_S)^2$, where G_S stacks the projected normalized gradients of all source samples. The same spectral argument gives a Lipschitz bound for the projected source system, computable from the largest singular value of G_S before any ODE is solved. □

Corollary 1 (Solution-error bound for full-kernel freezing)**.** *Let $f_S(\cdot)$ denote the source blocks of the exact reparameterized dynamics of Proposition 1, and let $\hat f_S(\cdot)$ solve the full-initial-kernel system with Θ^0_{SS} retained and the defect dropped, from the same initial condition. Then, with $\tau_s = \eta s$ and $\bar\Gamma$ as in Remark 2,*

$$\big\|\hat f_S(s) - f_S(s)\big\|_2 \;\le\; \int_0^s e^{\eta L_\star (s-\sigma)}\,\|\zeta_S(\sigma)\|_2\, d\sigma \;\le\; \|\Theta^0\|_F\,\frac{\sqrt{2\delta}}{1-\delta}\,\bar\Gamma\,\frac{e^{L_\star \tau_s}-1}{L_\star}, \tag{16}$$

where ζ_S collects the source rows of the defect. The probe block is a passive readout: its error accumulates the propagated source error through the frozen cross kernel, the residual map being $\frac{1}{2}$-Lipschitz in the logits, plus the time integral of its own defect rows; both contributions grow at most linearly in τ_s on top of the source bound above.

Proof. Let $e := \hat f_S - f_S$. Subtracting the two systems gives $\dot e = F(\hat f_S) - F(f_S) - \zeta_S$, with F the frozen right-hand side, so $\frac{d}{ds}\|e\| \le \eta L_\star \|e\| + \|\zeta_S\|$ by Lemma 1. Grönwall's inequality with $e(0) = 0$ yields the first bound; inserting the defect bound of Proposition 1 together with $\|\tilde\gamma\| \le \bar\Gamma$ and integrating yields the second. □

Error bound for the implemented kernel. Fix a mixture α and the same finite source and probe sets as in Proposition 1. Let $\widetilde\Theta^0$ collect the scalar kernel entries before projection, and let G stack the projected gradients $\hat g_u^\top$ over all source and probe samples. The implemented kernel acts on the stacked logits through $(GG^\top)\otimes I_Q$.

Write $f(s)$ for the exact reparameterized trajectory of Proposition 1, and let $f^{\mathrm{proj}}(s)$ solve

$$\frac{df^{\mathrm{proj}}}{ds} = -\eta\big[(GG^\top)\otimes I_Q\big]\,\widetilde\gamma(f^{\mathrm{proj}}),$$

from the same initial condition. Define the three equation-error contributions, evaluated along $f(s)$, by

$$b_{\mathrm{drift}}(s) := \|\zeta(s)\|_2,$$
$$b_{\mathrm{scalar}}(s) := \eta \left\| \left[\Theta^0 - \widetilde{\Theta}^0 \otimes I_Q\right] \widetilde{\gamma}(f(s)) \right\|_2,$$
$$b_{\mathrm{proj}}(s) := \eta \left\| \left[(\widetilde{\Theta}^0 - GG^\top) \otimes I_Q\right] \widetilde{\gamma}(f(s)) \right\|_2.$$

By the same softmax-Jacobian argument as in Lemma 1, the implemented vector field has Lipschitz constant at most $\eta \widehat{L}_\star$, where

$$\widehat{L}_\star := \tfrac{1}{2} \left(\max_i w_i \right) \sigma_{\max}(G)^2.$$

Here G includes the probe rows because the bound concerns the full stacked trajectory. Then

$$\|f^{\mathrm{proj}}(s) - f(s)\|_2 \le \int_0^s e^{\eta \widehat{L}_\star (s-u)} \cdot \left[b_{\mathrm{drift}}(u) + b_{\mathrm{scalar}}(u) + b_{\mathrm{proj}}(u) \right] du.$$

Proof. Let $e = f^{\mathrm{proj}} - f$ and denote the implemented vector field by F_{proj}. Subtracting the equation in Proposition 1 from the projected-kernel system gives

$$\begin{aligned} \dot{e} = {} & F_{\mathrm{proj}}(f^{\mathrm{proj}}) - F_{\mathrm{proj}}(f) - \zeta \\ & + \eta \left[\Theta^0 - \widetilde{\Theta}^0 \otimes I_Q\right] \widetilde{\gamma}(f) \\ & + \eta \left[(\widetilde{\Theta}^0 - GG^\top) \otimes I_Q\right] \widetilde{\gamma}(f). \end{aligned}$$

The Lipschitz bound and the triangle inequality therefore yield, in the upper-right derivative sense,

$$D^+ \|e\|_2 \le \eta \widehat{L}_\star \|e\|_2 + b_{\mathrm{drift}} + b_{\mathrm{scalar}} + b_{\mathrm{proj}}.$$

Grönwall's inequality with $e(0) = 0$ proves the bound. □

Corollary 2 (Ranking preservation under a loss margin)**.** *Let E_α bound the Euclidean error between the predicted and true stacked probe logits for candidate α at the evaluation endpoints, including any effective-time mismatch. Since the per-token validation CE in Eq. (1) is $\sqrt{2/M_{\mathrm{val}}}$-Lipschitz in these logits, the predicted loss $\widehat{\mathcal{L}}_{\mathrm{val}}$ satisfies*

$$\left| \widehat{\mathcal{L}}_{\mathrm{val}}(\alpha) - \mathcal{L}_{\mathrm{val}}(\alpha) \right| \le \sqrt{\frac{2}{M_{\mathrm{val}}}}\, E_\alpha.$$

Therefore, for two candidates α_a, α_b, if

$$\widehat{\mathcal{L}}_{\mathrm{val}}(\alpha_b) - \widehat{\mathcal{L}}_{\mathrm{val}}(\alpha_a) > \sqrt{\frac{2}{M_{\mathrm{val}}}} \left(E_{\alpha_a} + E_{\alpha_b} \right),$$

then their ordering is preserved: $\mathcal{L}_{\mathrm{val}}(\alpha_a) < \mathcal{L}_{\mathrm{val}}(\alpha_b)$.

Proof. Each position has CE gradient $p - y$ of norm at most $\sqrt{2}$, so the per-token average has stacked gradient norm at most $\sqrt{2M_{\mathrm{val}}}/M_{\mathrm{val}} = \sqrt{2/M_{\mathrm{val}}}$. The loss bound follows by integrating along the line segment between the logits, and the ranking claim follows by the triangle inequality. □

To demonstrate how our framework performs in practice, Appendix K evaluates the accuracy of the scalar and SRHT kernels, while Appendix L supports the directional fidelity of the predicted logit trajectories. These results provide empirical support for the accuracy of the kernel approximations in our experimental settings.

Remark 3 (How to read the bound)**.** *Eq. (16) combines the magnitude of the freezing defect with its accumulation over time. The dimensionless exponent $L_\star \tau_{s,\mathrm{end}}$, where $\tau_{s,\mathrm{end}} := \eta s(T)$, governs the exponential amplification. For a fixed initial kernel, mixture, and effective horizon, the bound scales as $O(\sqrt{\delta})$ as $\delta \to 0$. The initial kernel and mixture weights determine $L_\star$ through Lemma 1; the endpoint $\tau_{s,\mathrm{end}}$ additionally depends on the progress map $s(T)$. For a general learning-rate schedule, the effective progress at budget T is $\tau_\alpha^\star(T) = \int_0^T \eta(t) \rho_\alpha(t)\, dt$, which combines the learning-rate schedule with the kernel projection coefficient along the trajectory of mixture α. For each candidate mixture, the projection coefficient modulates its effective rate of progress, while the frozen kernel Θ^0*

is shared across candidates. Because $\rho_\alpha(t)$ is defined along the training trajectory and is not available before training, we compare mixtures on the common effective-time scale $\tau_{\mathrm{ref}} = \int_0^T \eta(t)\,dt$, which corresponds to $\rho_\alpha \equiv 1$. We therefore use $\tau_{\mathrm{end}} = c_\tau \tau_{\mathrm{ref}}$, with a horizon multiplier c_τ, as the shared progress reference for all candidate mixtures. Writing $\hat{f}_\alpha(\tau)$ for the solution of Eq. (14) in the effective-time variable, integrating the frozen dynamics between the two horizons and using $\|\tilde{\gamma}\|_2 \le \bar{\Gamma}$ from Remark 2 gives

$$\left\|\hat{f}_\alpha(\tau^\star_\alpha(T)) - \hat{f}_\alpha(\tau_{\mathrm{end}})\right\|_2 \;\le\; \lambda_{\max}(\Theta^0)\,\bar{\Gamma}\,\left|\tau^\star_\alpha(T) - \tau_{\mathrm{end}}\right|.$$

Hence, whenever the effective progresses of the candidates are concentrated around the common reference, their frozen trajectories at the shared horizon stay close to those at their own effective progress. Appendix O directly evaluates the corresponding stability of mixture ranking across a $4\times$ range of c_τ and reports the effective progress of representative mixtures. We use $c_\tau = 1$ in the main experiments.

Appendix J reports high NTK alignment at the final training checkpoint, while Appendix L shows directional agreement between predicted and actual logits and their changes from initialization. These observations provide empirical support for kernel directional stability and the directional fidelity of our predictions in the evaluated settings.

F Additional Details about the Jacobian-Free Kernel Estimate

The estimate of Eq. (13) replaces every per-sample Jacobian by a single normalized summed-output gradient. Up to the normalization it is the pseudo-NTK of Mohamadi et al. (2023), who introduce it as the scalar kernel of the single-output network obtained by averaging all output coordinates and show that, for wide networks, the exact kernel block becomes proportional to the identity across outputs while the ratio of off-diagonal to diagonal information decays with width; the same construction is used for LLM fine-tuning by Wang et al. (2026). This appendix records the scale bookkeeping that the ODE requires, namely how the summed-gradient kernel is normalized and used in the logit dynamics. Appendix L describes the trajectory validation.

Proposition 3 (Calibration of the summed-gradient kernel)**.** *Fix a sample pair (u, v) and write the exact kernel block as $\Theta_{u,v} = \kappa_{u,v} I_Q + E_{u,v}$, where $\kappa_{u,v} := \mathrm{tr}(\Theta_{u,v})/Q$ is the mean diagonal level and $E_{u,v}$ collects the diagonal fluctuation around this mean together with all cross-output entries. Then the normalized estimate of Eq. (13) satisfies*

$$\widetilde{\Theta}_{u,v} = \frac{1}{Q}\,\mathbf{1}^\top \Theta_{u,v}\,\mathbf{1} = \kappa_{u,v} + \frac{1}{Q}\sum_{k \neq k'} \left(\Theta_{u,v}\right)_{kk'},$$

i.e., it recovers the per-coordinate level $\kappa_{u,v}$ exactly, up to the averaged cross-output term. Without the $1/Q$ normalization, the kernel is Q times larger, so the ODE evolves Q times faster. The normalization therefore keeps the predicted logits and validation losses on the intended ODE time scale, preventing errors in the predicted performance of data mixtures.

Proof. Since $\nabla_\theta \mathbf{1}^\top f_u = J_u^\top \mathbf{1}$, we have $\langle \nabla_\theta \mathbf{1}^\top f_u, \nabla_\theta \mathbf{1}^\top f_v \rangle = \mathbf{1}^\top J_u J_v^\top \mathbf{1} = \mathbf{1}^\top \Theta_{u,v} \mathbf{1}$, and $\mathbf{1}^\top \Theta_{u,v} \mathbf{1} = \mathrm{tr}(\Theta_{u,v}) + \sum_{k \neq k'} (\Theta_{u,v})_{kk'} = Q\,\kappa_{u,v} + \sum_{k \neq k'} (\Theta_{u,v})_{kk'}$, where the diagonal fluctuation cancels because it sums to zero by the definition of $\kappa_{u,v}$; dividing by Q gives the display. For the time rescaling, let g denote the right-hand side of the frozen ODE and let f solve $f' = g(f)$; then $\tilde{f}(\tau) := f(Q\tau)$ solves $\tilde{f}' = Q\,g(\tilde{f})$, i.e., the Q-scaled system, so reading the Q-scaled system at $\tau_{\mathrm{end}} = \eta T$ returns the unscaled solution at $Q\,\eta T$. □

Bookkeeping in the ODE. Under the substitution $\Theta^0_{u,v} \to \widetilde{\Theta}^0_{u,v} I_Q$, the ODE acts on the residuals coordinatewise: with $\widetilde{\Theta}^0_{c,i} \in \mathbb{R}^{N_c \times N_i}$ the matrix of per-pair scalars and $r_i \in \mathbb{R}^{N_i \times Q}$ the stacked residuals of domain i, the right-hand side of Eq. (12) reads

$$\dot{f}_c = -\eta \sum_{i=1}^{m} w_i\,\widetilde{\Theta}^0_{c,i}\,r_i \;\in\; \mathbb{R}^{N_c \times Q}, \tag{17}$$

so the state of the solver is the $N_c \times Q$ array of logits of each block and the kernel enters only as an $N_c \times N_i$ matrix of sample-pair weights. This is what reduces the object stored for the dynamics from a $(N_S Q) \times (N_S Q)$ kernel to an $N_S \times N_S$ one.

G Additional Details about the Projection Dimension of the SRHT Kernel

The subsampled randomized Hadamard transform used in Sec. 3.2 (Ailon & Chazelle, 2006; Tropp, 2011) preserves every pairwise inner product with high probability up to an *additive* error of order $\|g_u\|\|g_v\|/\sqrt{p}$, at a projected dimension p that grows only logarithmically with the number of samples (Johnson et al., 1984). Two effects, however, set the practical lower bound on p.

First, the guarantee is additive rather than relative. The cross-domain kernel entries that carry the transfer signal are small inner products between large-norm gradients, so recovering them to relative accuracy requires $p \gg 1/\varrho^2$, where ϱ denotes the typical normalized cosine between cross-domain gradients (written ρ in the main text, and not to be confused with the projection coefficient $\rho(t)$ of Proposition 1). Concretely, an entry whose true value is $\varrho\,\|g_u\|\|g_v\|$ is recovered with additive error $\|g_u\|\|g_v\|/\sqrt{p}$, so the relative error is of order $1/(\varrho\sqrt{p})$ and stays below a constant only once $p \gtrsim 1/\varrho^2$.

Second, the projected source kernel has rank at most p. If $p < N_S$, where $N_S = \sum_{i=1}^{m} N_i$ is the total number of source samples, the frozen ODE inherits an artificially rank-deficient kernel: directions of the residual that fall in the null space of the projected kernel are never driven, and the predicted loss saturates at a floor set by the projection rather than by the data.

We therefore choose $p \gtrsim \max(N_S,\, 1/\varrho^2)$, which remains modest since the sampled cohorts are small; every kernel entry is then recovered to controlled accuracy from a p-dimensional inner product. The effect of varying p around this choice is reported in Tab. 3.

H Overall Algorithm and Cost Analysis

We summarize the full data mixture selection procedure in Algorithm 1.

Algorithm 1 Training-free mixture selection with our framework

Require: base model θ_0; source cohorts $\{S_i\}_{i=1}^m$ with sizes N_i; probe set val with labels; budget T; learning-rate schedule $\eta(t)$; projection dimension p; candidate set $\mathcal{A} \subset \Delta_{m-1}$
Ensure: estimated optimal mixture $\alpha^\star$

// Mixture-independent precomputation (once)
for each sample $u \in S_1 \cup \cdots \cup S_m \cup \mathrm{val}$ **do**
  obtain the base logits $f_u(\theta_0)$ and the summed-output gradient $g_u = \frac{1}{\sqrt{Q}} \nabla_\theta \sum_{k=1}^{Q} f_{u,k}(\theta_0)$
  project it by the SRHT: $\hat{g}_u \leftarrow \mathrm{SRHT}_p(g_u)$
end for
assemble the frozen scalar kernel blocks using the gradients projected by SRHT $\widetilde{\Theta}^0_{c,i} \leftarrow [\langle \hat{g}_u, \hat{g}_v \rangle]_{u \in c,\, v \in i}$ for $c \in \{1, \ldots, m, \mathrm{val}\}$
// Per-mixture trajectory prediction
set the shared horizon before evaluating the candidate mixtures: $\tau_{\mathrm{end}} = c_\tau \int_0^T \eta(t)\, dt$, $c_\tau = 1$
for each candidate $\alpha \in \mathcal{A}$ **do**
  form the weights $w_i \leftarrow \alpha_i / M_i$
  integrate Eq. (14) from $f(\theta_0)$ over $\tau \in [0, \tau_{\mathrm{end}}]$ by RK45 with dense output, using the same horizon for every candidate
  read the probe logits at τ_{end} and evaluate $\mathcal{L}_{\mathrm{val}}(\alpha)$ by Eq. (1)
end for
return $\alpha^\star \leftarrow \arg\min_{\alpha \in \mathcal{A}} \mathcal{L}_{\mathrm{val}}(\alpha)$

Cost analysis. Let $N_{\mathrm{all}} = N_S + N_{\mathrm{val}}$ be the total number of sampled examples, and let K_{RK} be the number of ODE function evaluations. The one-time precomputation requires $O(N_{\mathrm{all}})$ base-model forward and backward passes, $O(N_{\mathrm{all}} P \log P)$ time for gradient projection, and $O(N_{\mathrm{all}} N_S p)$ time for kernel construction. The projected gradients and kernel blocks require $O(N_{\mathrm{all}} p + N_{\mathrm{all}} N_S)$ storage, with $O(P)$ working memory during projection. The logit state additionally requires $O(N_{\mathrm{all}} Q)$ storage, where $Q = LV$, and RK45 maintains multiple state-sized working arrays. Retaining dense output over n_{step} accepted steps requires an additional $O(n_{\mathrm{step}} N_{\mathrm{all}} Q)$ storage. Each candidate

mixture requires $O(K_{\mathrm{RK}} N_{\mathrm{all}} N_S Q)$ arithmetic and no additional model evaluation. Therefore, evaluating all $|\mathcal{A}|$ candidates requires $O(N_{\mathrm{all}})$ base-model passes and $O(N_{\mathrm{all}} P \log P + N_{\mathrm{all}} N_S p + |\mathcal{A}| K_{\mathrm{RK}} N_{\mathrm{all}} N_S Q)$ additional arithmetic.

I Additional Details about Evaluation Metrics

Ranking consistency. This metric measures whether a method ranks candidate mixtures consistently with their true training outcomes. We report Spearman's ρ over all candidates and over the top 25% with the lowest true validation losses. The top-25% correlation measures how well the method ranks the best-performing mixtures.

Selection quality. For continual pre-training, we train the target model for 50B tokens using the mixture selected by each method, then evaluate it on general, math, and code benchmarks. Because scores from different domains are not directly comparable, following Li et al. (2026), we convert each score into a relative rank among 96 reference models and average the ranks across the three domain groups. The general benchmarks are ARC-E, HellaSwag, PIQA, SIQA, and WinoGrande; the code benchmarks are MBPP and HumanEval; and the math benchmarks are GSM8K and MATH. For SFT, we report accuracy on AGIEval, IFEval, MMLU, GSM8K, HumanEval, and safety, together with their average.

Trajectory fidelity. We also directly compare the predicted and actual trajectories. Let $f^{\mathrm{pred}}(\tau)$ and $f^{\mathrm{true}}(\tau)$ be the predicted and actual probe logits for the same mixture and training budget. Their cosine similarity is

$$\mathrm{sim}(\tau) = \frac{\left\langle f^{\mathrm{pred}}(\tau),\, f^{\mathrm{true}}(\tau) \right\rangle}{\left\| f^{\mathrm{pred}}(\tau) \right\| \left\| f^{\mathrm{true}}(\tau) \right\|}, \tag{18}$$

Results are reported in Sec. L.

Search cost. We convert search GPU hours into an equivalent number of training tokens to select its data mixture.

J Validation of the NTK Direction Stability

In this section, since the full NTK is intractable to construct at LLM scale, we evaluate the directional stability of the SRHT kernel used by our framework. For each model and dataset, we compute the kernel at 25%, 50%, 75%, and 100% of the training budget and compare its direction with that at initialization. Across all settings and checkpoints in Tab. 4, the implemented kernel remains closely aligned with its initialization value, supporting the frozen-kernel approximation used in practice.

Table 4: Direction stability of the SRHT kernel used by our framework, reported as its Frobenius cosine similarity to the initialization kernel at 25%, 50%, 75%, and 100% of the training budget.

Scenario	Target model	Direction cosine-similarity to Θ^0 ↑			
		25%	50%	75%	100%
Pre-training mixing	Qwen3-1.7B	0.994	0.987	0.984	0.982
SFT mixing	Llama3.1-8B	0.987	0.979	0.975	0.972
	Qwen2.5-32B	0.958	0.952	0.946	0.941
Average		0.980	0.973	0.968	0.965

K Validation of the Kernel Approximations

In this section, we test whether the scalar and SRHT approximations preserve the logit trajectories produced by the full frozen-NTK ODE.

We follow the main experimental settings, for the trajectory comparison, we report results for the five mixtures used in the trajectory experiment. We compare three ODE systems: the full NTK, the normalized scalar kernel before projection, and the final SRHT kernel. All three systems use the

same pretrained checkpoint, source and probe sets, initial logits, mixture weights, and integration horizon. Their kernels are

$$\begin{aligned} K_{\text{full}} &= \Theta^0, \\ K_{\text{scalar}} &= \widetilde{\Theta}^0 \otimes I_Q, \\ K_{\text{SRHT}} &= (GG^\top) \otimes I_Q. \end{aligned} \tag{19}$$

Each system is integrated independently using its own evolving residuals:

$$\frac{df^A}{d\tau} = -K_A \widetilde{\gamma}_\alpha(f^A), \tag{20}$$

where A denotes the full, scalar, or SRHT kernel. This comparison captures how kernel approximation errors propagate through the nonlinear residual dynamics.

For the full-NTK system, each right-hand-side evaluation uses a VJP followed by a JVP:

$$\begin{aligned} v_\alpha^{\text{full}}(\tau) &= \sum_i w_i (J_i^0)^\top r_i(f_i^{\text{full}}(\tau)), \\ \frac{df_c^{\text{full}}}{d\tau} &= -J_c^0 v_\alpha^{\text{full}}(\tau). \end{aligned} \tag{21}$$

All Jacobians are evaluated at the fixed base parameters θ_0. This computes the full kernel action without forming the Jacobian or kernel explicitly.

We compare the three systems through logit-change errors. Let C subtract the mean across vocabulary coordinates separately at each token, and define

$$\Delta z^A(\tau) = C\big(f^A(\tau) - f(0)\big). \tag{22}$$

For the scalar and SRHT systems, we measure

$$E_A(\tau) = \frac{\|\Delta z^A(\tau) - \Delta z^{\text{full}}(\tau)\|_F}{\|\Delta z^{\text{full}}(\tau)\|_F}. \tag{23}$$

We evaluate this error at 25%, 50%, 75%, and 100% of the integration horizon. Table 5 summarizes the mean of E_A across the five candidate mixtures for each target model.

Table 5: Relative errors $E_A(\tau)$ of centered logit changes against the full frozen-NTK trajectory at four fractions of the integration horizon. Each entry reports the mean across five candidate mixtures, using the stacked source and probe logits. Scalar denotes the normalized kernel before projection; SRHT denotes the final projected kernel. Lower is better.

Target model	Kernel	Logit-change relative error ↓			
		25%	50%	75%	100%
Qwen3-1.7B	Scalar	0.071	0.074	0.076	0.080
	SRHT	0.073	0.075	0.079	0.082
Llama3.1-8B	Scalar	0.088	0.091	0.095	0.097
	SRHT	0.090	0.094	0.096	0.099
Qwen2.5-32B	Scalar	0.093	0.097	0.100	0.103
	SRHT	0.094	0.099	0.101	0.104

These results demonstrate the effectiveness of the scalar and SRHT kernel approximations in preserving the logit changes predicted by the full frozen NTK across target models and integration stages.

For each target model, the full, scalar, and SRHT kernels select the same best-performing mixture from the 100 candidates used in the main experiments.

L Predicted Trajectories against Real Training Trajectories

In this section, to verify the accuracy of the logits predicted by our framework, we compare the logits it predicts during training against the logits of the actual training, measured by the similarity of Eq. (18).

We draw five candidate mixtures spanning the simplex. For each of them we train the target model to the same budget T and record the probe logits at four checkpoints corresponding to 25%, 50%, 75%, and 100% of the training budget, giving the true trajectory $f^{\text{true}}(\tau)$. We then predict the trajectory of the same mixture with a single ODE solve and evaluate the similarity of Eq. (18) at the matching effective times. Table 6 reports these logit cosine similarities. The high similarities demonstrate that a single ODE solve accurately captures the directional structure of the logit trajectories across mixtures and training stages.

Table 6: Fidelity of the predicted trajectory against the real training trajectory, measured by the cosine similarity of Eq. (18) between the predicted and the true probe logits, at four fractions of the training process. Each row is one candidate mixture.

Candidate mixture α	Trajectory cosine-similarity ↑ as training progresses			
	25%	50%	75%	100%
$\alpha^{(1)}$	0.993	0.983	0.974	0.969
$\alpha^{(2)}$	0.989	0.978	0.966	0.958
$\alpha^{(3)}$	0.986	0.975	0.965	0.961
$\alpha^{(4)}$	0.989	0.980	0.971	0.967
$\alpha^{(5)}$	0.990	0.980	0.968	0.959
Average	0.989	0.979	0.969	0.963

To assess whether the predictions capture the changes caused by training, we additionally subtract the initial probe logits from both the predicted and actual logits at each checkpoint. We compare the resulting changes using cosine similarity to assess their directional agreement. Overall, the results suggest that the ODE-based predictions preserve the main direction of the training-induced logit changes across different mixtures and checkpoints.

Table 7: Directional agreement of the predicted and actual logit changes, measured by cosine similarity after subtracting the initial probe logits, at four fractions of the training process. Each row is one candidate mixture.

Candidate mixture α	Logit-change cosine-similarity ↑ as training progresses			
	25%	50%	75%	100%
$\alpha^{(1)}$	0.939	0.937	0.930	0.913
$\alpha^{(2)}$	0.932	0.927	0.918	0.901
$\alpha^{(3)}$	0.940	0.939	0.934	0.921
$\alpha^{(4)}$	0.925	0.921	0.914	0.900
$\alpha^{(5)}$	0.936	0.935	0.928	0.910
Average	0.934	0.932	0.925	0.909

M Data Mixtures Selected by Each Method

In this section, we report the data mixture that each method selects in the pre-training mixing scenario, as listed in Tab. 8. The seven source domains are the candidate datasets of Li et al. (2026): one general, three math and three code. Since we follow their setting, the Uniform, Heuristic, RegMix, CLIMB and DeMix mixtures are taken from that work, each from the same configuration as its row in Tab. 1.

Table 8: Mixtures selected by each method in the pre-training mixing scenario.

Method	α_{General}	$\alpha_{\text{Math-1}}$	$\alpha_{\text{Math-2}}$	$\alpha_{\text{Math-3}}$	$\alpha_{\text{Code-1}}$	$\alpha_{\text{Code-2}}$	$\alpha_{\text{Code-3}}$
Uniform	0.500	0.111	0.027	0.039	0.020	0.055	0.248
Heuristic	0.200	0.273	0.033	0.000	0.098	0.136	0.259
Random	0.245	0.384	0.022	0.001	0.080	0.123	0.146
Inverse-performance	0.290	0.226	0.055	0.079	0.022	0.059	0.268
RegMix (Liu et al., 2025)	0.503	0.330	0.020	0.001	0.077	0.009	0.059
CLIMB (Diao et al., 2026)	0.417	0.422	0.027	0.011	0.079	0.008	0.035
DeMix (Li et al., 2026)	0.218	0.403	0.002	0.063	0.044	0.176	0.094
FastMix (Tan et al., 2026)	0.152	0.134	0.148	0.132	0.153	0.132	0.148
Ours	0.267	0.117	0.039	0.042	0.439	0.040	0.055

N Additional Ablation Studies and Further Analyses

In this section we conduct more ablation studies and further analyses on our framework. All results are reported on the pre-training mixing scenario with Qwen3-1.7B as the target model. For each hyperparameter setting, candidate mixtures are selected based on the held-out validation loss, and the target model is then trained using the selected mixture. The downstream scores reported below are used only to evaluate the resulting selected mixture.

Impact of the maximum sequence length. To determine the truncation length used for kernel construction, we compare maximum sequence lengths of 512, 1024, and 2048 tokens, and report the downstream performance on the general, code, and math benchmark groups. Based on the held-out validation loss, we use 1024 tokens in all main experiments. Table 9 reports the downstream performance of the mixture selected under each maximum sequence length.

Table 9: Evaluation on the maximum sequence length used to truncate each sample during kernel construction.

Maximum length (tokens)	General Avg.↑	Code Avg.↑	Math Avg.↑
512	57.14	31.76	18.24
1024	58.58	32.32	19.72
2048	58.58	32.32	19.72

Impact of the number of sampled examples per domain. In our framework the kernel is estimated from n samples per source domain. To determine the number of samples n, we vary it from 32 to 256 with all other settings fixed and report the downstream performance on the general, code, and math benchmark groups. Based on the held-out validation loss, we use $n = 128$ in our experiments. Table 10 reports the downstream performance of the mixture selected under each value of n.

Table 10: Evaluation on the number of sampled examples n per source domain.

Samples per domain	General Avg.↑	Code Avg.↑	Math Avg.↑
$n = 32$	58.29	31.92	18.33
$n = 64$	58.32	31.20	18.78
$n = 128$	58.58	32.32	19.72
$n = 256$	58.58	32.32	19.72

Impact of the candidate set size. To examine how search coverage affects mixture selection, we vary the number of candidate mixtures in $\mathcal{A}$ over 50, 100, and 200. Based on the held-out validation loss, we use 100 candidate mixtures in all main experiments. Table 11 reports the downstream performance of the mixture selected under each candidate-set size.

Table 11: Evaluation on the number of candidate mixtures used in mixture search.

Candidate set size	General Avg.↑	Code Avg.↑	Math Avg.↑
50	58.18	32.68	18.22
100	58.58	32.32	19.72
200	58.58	32.32	19.72

Impact of the probe set size. We compare probe sets containing 2%, 5%, and 10% of the samples from each domain. We use a 5% probe set in the main experiments and evaluate its sensitivity by additionally considering 2% and 10%. Table 12 reports the downstream performance of the resulting selected mixtures.

Table 12: Evaluation on the fraction of samples from each domain used to construct the probe set.

Probe set fraction	General Avg.↑	Code Avg.↑	Math Avg.↑
2%	58.27	32.12	18.46
5%	58.58	32.32	19.72
10%	58.25	32.04	19.05

Impact of the probe set composition. We use a balanced probe set in the main experiments and evaluate its sensitivity against probe sets dominated by a single domain group, using General/Code/Math ratios of 1:1:1 for Equal shares, 3:1:1 for Mostly general, 1:1:3 for Mostly math, and 1:3:1 for Mostly code, and keep every other setting unchanged. Table 13 reports the downstream performance of the resulting selected mixtures.

Table 13: Evaluation on the composition of the probe set. Ratios are reported in General/Code/Math order.

Probe set	General/Code/Math ratio	General Avg.↑	Code Avg.↑	Math Avg.↑
Equal shares	1:1:1	58.58	32.32	19.72
Mostly general	3:1:1	58.39	30.60	18.57
Mostly math	1:1:3	58.16	31.21	19.52
Mostly code	1:3:1	58.04	31.99	19.05

Impact of the numerical solver. We compare RK45 with forward Euler using 10, 100, and 1000 steps, with all other settings fixed. Based on the held-out validation loss, we use RK45 in all experiments. Table 14 reports the downstream performance of the mixture selected with each numerical solver.

Table 14: Evaluation on the numerical solver used to integrate the frozen ODE.

Solver	General Avg.↑	Code Avg.↑	Math Avg.↑
Forward Euler, 10 steps	57.65	30.24	17.08
Forward Euler, 100 steps	57.96	30.65	17.10
Forward Euler, 1000 steps	58.21	31.79	18.65
RK45 (ours)	58.58	32.32	19.72

Impact of the solver tolerance. To choose the relative tolerance for RK45, we compare 10^{-4}, 10^{-6}, and 10^{-8} while fixing the absolute tolerance at 10^{-8} and keeping all other settings fixed. Based on the held-out validation loss, we use a relative tolerance of 10^{-6}. Table 15 reports the downstream performance of the mixture selected under each solver tolerance.

Table 15: Evaluation of the RK45 relative tolerance with the absolute tolerance fixed at 10^{-8}.

Relative tolerance	General Avg.↑	Code Avg.↑	Math Avg.↑
10^{-4}	58.01	32.03	18.70
10^{-6}	58.58	32.32	19.72
10^{-8}	58.58	32.32	19.72

O Additional Experiments

This section presents additional experiments on our framework. All of them are conducted in the pre-training mixing scenario with Qwen3-1.7B as the target model.

Sensitivity to the effective horizon. We examine how the evaluation endpoint affects mixture ranking by scoring the same frozen trajectories at $c_\tau \in \{0.5, 0.75, 1.0, 1.5, 2.0\}$. We report Spearman's ρ over all candidate mixtures and over the top 25% according to ground-truth performance. As shown in Tab. 16, $c_\tau = 1$ achieves the highest correlations of 0.85 and 0.61, respectively. We set $c_\tau = 1$ in all experiments, so that $\tau_{\text{end}} = \tau_{\text{ref}}$.

Table 16: Sensitivity of mixture ranking to the effective horizon multiplier c_τ.

c_τ	Spearman's $\rho \uparrow$	Top 25% $\rho \uparrow$
0.5	0.76	0.55
0.75	0.81	0.59
1.0	0.85	0.61
1.5	0.79	0.57
2.0	0.74	0.53

Effective training progress across representative mixtures. We compute the projection coefficient $\rho_\alpha(t)$ for the same five candidate mixtures used in Appendix L at 25%, 50%, 75%, and 100% of the training budget, with $\rho_\alpha(0) = 1$, using the same stacked-kernel construction as in Appendix J. Tab. 17 reports $\rho_\alpha(t)$ for each mixture and checkpoint, together with the normalized effective progress $\bar{\rho}_\alpha = \int_0^T \eta(t)\rho_\alpha(t)\,dt \big/ \int_0^T \eta(t)\,dt$. Across the five mixtures, $\bar{\rho}_\alpha$ ranges from 0.93 to 0.97, with a mean of 0.96. The mean is close to the reference value 1 used by τ_{ref}, while the range of 0.04 shows that the mixture-specific effective progresses remain close to this common scale. Within each mixture $\rho_\alpha(t)$ decays slowly and monotonically over training, from 0.97 on average at 25% of the budget to 0.83 at the end, and the relative ordering of the five mixtures is preserved at every checkpoint. Since $\tau^\star_\alpha(T) = \bar{\rho}_\alpha\,\tau_{\text{ref}}$, scoring all mixtures at the shared horizon $\tau_{\text{end}} = \tau_{\text{ref}}$ therefore integrates each mixture 3% to 7% past its own effective progress, a mild over-integration that is close to common across mixtures rather than one that reorders them. Tab. 16 further shows that the mixture ranking remains stable under a broad common rescaling of the horizon.

Table 17: Projection coefficient $\rho_\alpha(t)$ along the ground-truth trajectories of the five candidate mixtures used in Appendix L, together with the normalized effective progress $\bar{\rho}_\alpha$. The latter is computed by trapezoidal integration of $\eta(t)\rho_\alpha(t)$ over the reported checkpoints with $\rho_\alpha(0) = 1$, normalized by τ_{ref}. Each row is one candidate mixture.

Candidate mixture α	$\rho_\alpha(t)$ as training progresses				$\bar{\rho}_\alpha$
	25%	50%	75%	100%	
$\alpha^{(1)}$	0.96	0.91	0.85	0.79	0.93
$\alpha^{(2)}$	0.97	0.93	0.88	0.81	0.95
$\alpha^{(3)}$	0.98	0.95	0.90	0.82	0.96
$\alpha^{(4)}$	0.98	0.96	0.92	0.84	0.97
$\alpha^{(5)}$	0.98	0.97	0.95	0.87	0.97
Average	0.97	0.94	0.90	0.83	0.96

Consistency with ground-truth mixture rankings. To evaluate how faithfully each method captures the relative performance of candidate data mixtures, we compare the ranking induced by its performance estimates with the ground-truth ranking obtained by training reference models directly on the corresponding mixtures. We report the macro-average Spearman's ρ over all candidate mixtures and over the top 25% according to ground-truth performance. The former measures global ranking agreement, whereas the latter assesses whether a method can reliably distinguish among high-performing mixtures. As shown in Tab. 18, our framework achieves stronger agreement with the ground-truth mixture rankings than RegMix and DeMix, both across all candidate mixtures and among the top-performing 25%.

Table 18: Consistency between the mixture rankings estimated by each method and the ground-truth rankings obtained from reference models.

Method	Spearman's $\rho \uparrow$	Top 25% $\rho \uparrow$
RegMix (Liu et al., 2025)	0.76	0.54
DeMix (Li et al., 2026)	0.74	0.52
Ours	0.85	0.61

Robustness across training optimizers. To verify that our framework is robust to different optimizers, we try three different optimizers during training, SGD, Adam and AdamW, and keep every other setting unchanged. For each optimizer we report Spearman's ρ over all candidate mixtures and over the top 25% of mixtures. Our framework achieves similar rank correlations under all three optimizers (Tab. 19), supporting consistent mixture-ranking performance across the evaluated optimizers.

Table 19: Ranking performance of our framework when the ground-truth runs use SGD, Adam, or AdamW. We report Spearman's ρ over all candidate mixtures and over the top 25% of mixtures.

Optimizer of the ground-truth run	Spearman's $\rho \uparrow$	Top 25% $\rho \uparrow$
SGD	0.85	0.61
Adam	0.86	0.60
AdamW	0.85	0.62

Comparison with other gradient compression methods. To demonstrate the effectiveness of the SRHT, we replace it with two alternatives. Dense Gaussian multiplies each gradient by an explicit $P \times p$ Gaussian matrix (Park et al., 2023; Xia et al., 2024). Random subsampling keeps p coordinates drawn uniformly at random, which is the SRHT without its sign flip and Hadamard transform. For each method we report Spearman's ρ over all candidate mixtures and over the top 25% of

mixtures. Tab. 20 shows that SRHT achieves higher rank correlations than the other evaluated gradient compression methods.

Table 20: Comparison of gradient compression methods at the same projection dimension p.

Compression method	Spearman's $\rho \uparrow$	Top 25% $\rho \uparrow$
Dense Gaussian (Park et al., 2023; Xia et al., 2024)	0.76	0.52
Random subsampling	0.77	0.49
SRHT (ours)	0.85	0.61