JADORE: ONLINE DATA MIXING THROUGH THE LENS OF OPTIMIZATION GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Training data mixtures greatly impact the generalization performance of language models. Existing gradient-based mixing methods rely on raw gradients, ignoring the optimizer dynamics. We introduce *JAcobian DOmain REweighting* (JADORE), an online data mixing framework that estimates the local effect of mixtures on optimization geometry. It defines the domain scores based on the derivative of the optimizer mapping from raw gradients to actual updates. We derive the action of the optimizer's Jacobian for multiple update geometries and show its efficient computation. To ensure scalability, we develop a single backward pass implementation for both training and scoring. Extensive experiments demonstrate accuracy improvements across diverse benchmarks in both pretraining and fine-tuning on multiple datasets. These gains are consistent across optimizers and model scales.

1 INTRODUCTION

Data mixture has become a central design choice in large language model (LLM) training. Modern LLMs are trained on heterogeneous corpora from a mixture of various source domains, such as academic papers, books, and code repositories (Brown et al., 2020; Grattafiori et al., 2024). The composition of these mixtures strongly influences model performance, motivating methods that optimize domain sampling weights (Gao et al., 2020; Du et al., 2022). Classical offline methods, e.g., (Xie et al., 2023; Fan et al., 2024b; Liu et al., 2025b), learn fixed domain weights using small proxy models. Yet the weight of a domain need not remain constant: as the model evolves, the data most useful for its next updates can also change (Jiang et al., 2025; Chen et al., 2025). This motivates studying data mixing not only as a dataset-design problem, but also as an online decision linked to how the model is optimized.

Recent advances in optimizer design make this perspective especially important. Beyond Adam's adaptive rescaling (Kingma & Ba, 2014), the recently proposed Muon optimizer uses the spectral norm to define matrix update directions (Jordan et al., 2024b). It has achieved state-of-the-art results in the nanoGPT speedrun (Jordan et al., 2024b) and has been adopted for large-scale LLM training (Liu et al., 2025a; DeepSeek-AI et al., 2026). More broadly, norm-based frameworks derive update rules from different norm choices (Pethick et al., 2025a; 2026a). These rules transform raw gradients into actual parameter updates differently, so the benefit of increasing a domain's weight can depend on the *optimizer*.

However, existing gradient-based data mixing methods (Fan et al., 2024b;a) reweight domains in the space of raw gradients, without explicitly accounting for how the optimizer transforms them into actual parameter updates. For example, increasing a gradient's magnitude without changing its direction can leave a normalized update unchanged. Domain reweighting should therefore account for how changing a domain's weight affects the optimizer update and adapt to the actual training dynamics. This raises a fundamental question: *how can online domain reweighting account for distinct optimizer update geometries in a principled and computationally efficient way?*

We introduce JADORE, a Jacobian-based framework for optimizer-aware online domain reweighting. It estimates domain utility as the sensitivity of validation loss to domain weights by differentiating the optimizer's mapping from gradients to parameter updates. We develop this analysis across adaptive rescaling and norm-constrained update geometries. By reusing optimizer information and training computations, we integrate domain scoring into each training step without separate backward passes for individual domains. Specifically, our contributions are summarized as follows:

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.

- We propose an **optimizer-aware perspective** of online domain reweighting: domain utility should reflect how changing a domain's weight affects the optimizer's parameter update. From this principle, we derive a general Jacobian-based framework that estimates the marginal effect of domain weights on predicted validation-loss reduction, with analysis spanning adaptive rescaling and norm-constrained update geometries.
- We develop an **efficient online algorithm** that computes domain scores at every training step alongside model updates in a single shared forward-backward pass. It shares optimizer-aware computations across domains and reuses optimizer information and training intermediates, avoiding explicit construction of full optimizer Jacobians and repeated backward passes.
- We demonstrate the **effectiveness of JADORE across diverse update geometries** represented by Adam, Muon, RMNP, and Signum. With each optimizer, JADORE achieves the highest mean accuracy among the evaluated mixing methods in both pretraining and supervised fine-tuning. Ablations support the benefits of Jacobian-based domain scoring, while further analysis shows the computational efficiency of JADORE.

Related Works. Early *domain reweighting* methods learn fixed weights using small proxy models (Xie et al., 2023; Fan et al., 2024b), but may not suit larger models or do not evolve training dynamics (Jiang et al., 2025; Ye et al., 2025; Chen et al., 2025). This motivates online methods guided by losses (Albalak et al., 2023; Jiang et al., 2025; Chen et al., 2025) or gradients (Fan et al., 2024a; 2025; Li et al., 2025). Among these, the most closely related methods estimate domain utility through alignment with validation gradients (Fan et al., 2024a). They require separate domain-gradient computations and use raw-gradient alignment that may not reflect the optimizer's actual updates. We address these limitations with optimizer-aware scoring that reuses training computations.

Different optimizers induce *distinct update geometries*: Adam applies adaptive scaling (Kingma & Ba, 2014), while Muon uses Newton-Schulz (NS) iterations (Jordan et al., 2024b). Recent data selection methods use frozen-preconditioner approximations (Wang et al., 2026b;a). Our approach differs by deriving domain scores from the optimizer's gradient-to-update Jacobian in a principled framework across diverse geometries. Pethick et al. (2025a) further connect more norm choices to row-/column-normalized and sign updates, which our framework also covers.

2 METHOD

We recast data mixing as a problem of measuring how the optimizer update responds to changes in the training mixture. This perspective leads to a principled online rule for adapting domain weights based on their influence on the optimizer update. The optimizer transforms training gradients before they affect the target loss: in our perspective, a domain's utility depends on how the optimizer translates an increase in its weight into progress on the target distribution. Our efficient streaming implementation enables a single forward–backward pass to support both domain weighting and model updates.

Data Mixing Optimization. Consider the training of a language model on a corpus of K domains $\{\mathcal{D}_d\}_{d=1}^K$ (e.g., Wikipedia, GitHub, ...). Data mixing optimizes domain sampling probabilities $\alpha \in \Delta^K$, where $\Delta^K := \{\alpha \in \mathbb{R}^K : \alpha_d \geq 0,\ \sum_{d\in[K]} \alpha_d = 1\}$ is the probability simplex (Xie et al., 2023; Fan et al., 2024b). The target is to find domain weights α that minimize the validation loss $L_R(\theta(\alpha))$ on a reference set $\mathcal{D}_R$. Formally, data mixing is formulated as the problem (Fan et al., 2024b; Tan et al., 2026):

$$\min_{\alpha\in\Delta^K} L_R(\theta(\alpha)) \quad \text{subject to} \quad \theta(\alpha) \in \arg\min_{\theta} L(\theta, \alpha) := \sum_{d\in[K]} \alpha_d L_d(\theta), \tag{1}$$

where $L_d(\theta)$ is the expected next-token prediction loss on domain $\mathcal{D}_d$. The inner problem trains the model under the mixture, and the outer problem chooses the mixture to minimize the validation loss.

2.1 OPTIMIZER-AWARE DOMAIN UTILITY

Common approaches to solve (1) use raw gradient alignment as the scoring rule (Fan et al., 2024b;a), which can be interpreted as a one-step greedy expansion where the inner optimization is approximated

by one SGD step. However, such scoring rules omit the transformations applied by adaptive or non-Euclidean optimizers used for training LLMs. We retain the optimizer geometry by explicitly leveraging its transformation: for current parameters θ and a candidate mixture $\alpha \in \Delta^K$, this update is:

$$\theta^+(\alpha) = \theta - \eta\Phi(\nabla_\theta L(\theta, \alpha)), \tag{2}$$

where $\eta > 0$ is the learning rate, and Φ maps the mixed gradient to the optimizer's update direction.

We seek domain weights that minimize validation loss after one optimizer step at θ^+, or equivalently maximize the loss reduction. Linearizing L_R around θ gives

$$V(\alpha) \coloneqq L_R(\theta) - L_R(\theta^+(\alpha)) \approx \langle r, \theta - \theta^+(\alpha)\rangle = \eta\,\langle r, \Phi(\nabla_\theta L(\theta, \alpha))\rangle, \tag{3}$$

where $r = \nabla_\theta L_R(\theta)$. Under SGD, the map Φ in the first-order term is the identity operator, leading to standard gradient-alignment scores (Fan et al., 2024b). For optimizers induced by different geometries, the same term depends on how Φ transforms the gradient as the mixture changes.

We quantify this dependence by measuring the marginal effect of each domain's weight on the predicted gain. Let $g = \nabla_\theta L(\theta, \alpha)$ be the current mixed gradient and assume that Φ is differentiable at g. Since the weighted loss in eq. (1) is linear in α, we have $g = \sum_d \alpha_d g_d$, where $g_d = \nabla_\theta L_d(\theta)$. With the current parameters and incoming optimizer state held fixed, we differentiate the gain approximation in eq. (3) with respect to α. The chain rule gives

$$\mathrm{d}V(\alpha) \approx \eta\langle r, \mathrm{d}\Phi\rangle = \eta\left\langle r, J\sum_d g_d\,\mathrm{d}\alpha_d\right\rangle = \eta\sum_d \langle r, Jg_d\rangle\,\mathrm{d}\alpha_d, \tag{4}$$

where $J \coloneqq \partial\Phi(g)/\partial g$ is the optimizer Jacobian matrix with respect to its gradient input, evaluated at the current mixed gradient. We define domain utility as the coefficient of $\mathrm{d}\alpha_d$, normalized by the shared learning rate η across domains:

$$s_d \coloneqq \langle r, Jg_d\rangle = \langle g_d, J^\top r\rangle. \tag{5}$$

This quantity measures the marginal predicted gain from increasing domain d's weight through the optimization geometry. The adjoint form shares one Jacobian-transformed validation gradient (JVG) $J^\top r$ across all domains, avoiding computing a separate Jg_d for each domain.

Combining eqs. (4) and (5) gives $\nabla_\alpha V(\alpha) \approx \eta s$ with $s = (s_d)_{d=1}^K$. We regularize the local objective by a KL penalty to avoid degenerate mixtures and solve the resulting problem by linearization of V and primal averaging, which is a common technique in generalized conditional gradient methods (Bredies et al., 2009; Yu et al., 2017; Beck, 2017). This yields the mixture update rule:

$$\begin{aligned}\hat{\alpha} &= \arg\max_{\alpha\in\Delta^K}\{\langle \eta s, \alpha\rangle - \tau D_{\mathrm{KL}}(\alpha\|u)\} = \mathrm{softmax}(\eta s/\tau),\\ \alpha^+ &= \beta_\alpha\alpha + (1-\beta_\alpha)\hat{\alpha},\end{aligned} \tag{6}$$

where $u = \mathbf{1}/K$ is the uniform vector, $\tau > 0$ controls the KL penalty, and $\beta_\alpha \in [0, 1)$ controls mixture smoothing. To reduce softmax sensitivity to the score scale, we set $\tau = \eta\,\mathrm{std}(s)/\lambda$, where std denotes the population standard deviation and $\lambda > 0$ is the inverse regularization strength.

2.2 Computing optimizer JVGs

Computing domain utilities in eq. (5) requires only the shared JVG $J^\top r$, without realizing the full Jacobian separately. The JVG depends on how the optimizer transforms the gradient. We derive JVGs for diverse optimization geometries: first we derive JVG for Adam's adaptive diagonal rescaling, then consider norm-constrained update maps. The same derived rules apply with decoupled weight decay since it is independent of the gradient (Loshchilov & Hutter, 2019).

Adaptive Scaling. Adaptive optimizers are commonly used in deep learning (Kingma & Ba, 2014; Dozat, 2016), with the most notable example being Adam. Adam rescales a momentum-based direction using an adaptive denominator derived from second-moment estimates. Its JVGs account for the gradient dependence of both the numerator and denominator. Let $m_{\mathrm{in}}, v_{\mathrm{in}}$ denote the incoming moments and $n \geq 1$ the update count. With $0 \leq \beta_1, \beta_2 < 1$, $\epsilon > 0$, and using element-wise square:

$$\widehat{m} = \frac{\beta_1 m_{\mathrm{in}} + (1-\beta_1)g}{1-\beta_1^n}, \quad \widehat{v} = \frac{\beta_2 v_{\mathrm{in}} + (1-\beta_2)g^2}{1-\beta_2^n}, \quad q = \sqrt{\widehat{v}} + \epsilon, \quad \Phi_{\mathrm{Adam}}(g) = \frac{\widehat{m}}{q}. \tag{7}$$

Table 1: Optimizer geometries, and their corresponding updates and JVGs. Listed optimizers illustrate each update family. Formulas are shown for the optimizers in bold, which we analyze and evaluate experimentally. Stabilizers are omitted for brevity. Signum's JVG uses a smooth surrogate.

Geometry	Examples	Update Φ	JVG $J^\top r$	Ref.
Adaptive Scaling	**Adam**, NAdam	$\frac{\widehat{m}}{\sqrt{\widehat{v}}}$	$\left(\frac{\kappa_1}{\sqrt{\widehat{v}}} - \frac{\kappa_2 \widehat{m} \odot g}{\widehat{v}^{3/2}}\right) \odot r$	(Prop. 2.1)
Spectral	**Muon**, Scion	$\gamma\,\mathrm{NS}(M)$	$\gamma c\, J_{\mathrm{NS}}(M)^\top R$	(Prop. 2.2)
Row/Column Norm	**RMNP**, SCALE	$\gamma \frac{M_{i,:}}{\lVert M_{i,:}\rVert_2}$	$\frac{\gamma c}{\lVert M_{i,:}\rVert_2}\left(R_{i,:} - \frac{M_{i,:}\langle M_{i,:}, R_{i,:}\rangle}{\lVert M_{i,:}\rVert_2^2}\right)$	(Prop. 2.3)
Sign	**Signum**, Lion	$\mathrm{sign}(m)$	$\frac{c}{\tau_s}\,\mathrm{sech}^2(m/\tau_s) \odot r$	(Prop. 2.4)

Proposition 2.1 (Adam JVG). *Suppose every coordinate of $\widehat{v}$ is positive, and set $\kappa_i = (1-\beta_i)/(1-\beta_i^n)$ for $i = 1, 2$. The JVG of eq. (7) is*

$$J_{\mathrm{Adam}}^\top r = \left[\frac{\kappa_1}{q} - \frac{\kappa_2 \widehat{m} \odot g}{\sqrt{\widehat{v}} \odot q^2}\right] \odot r. \qquad \text{(Adam-JVG)}$$

All operations above are elementwise. See Appendix A.2 for the derivation.

Beyond adaptive diagonal scaling, matrix-based optimizers such as Muon have demonstrated strong performance and improved training efficiency in LLM pretraining (Jordan et al., 2024b; Liu et al., 2025a). This motivates extending our JVG analysis to a broader range of update geometries.

The norm-constrained formulation of Pethick et al. (2025a; 2026a;b) provides a common construction for several such transformations. Given a momentum direction $m(g)$ and an arbitrary norm $\lVert\cdot\rVert$, the corresponding update direction is

$$\Phi_{\lVert\cdot\rVert}(g) \in \arg\max_{u:\lVert u\rVert\le 1} \langle m(g), u\rangle. \tag{8}$$

Its negative is the linear minimization oracle (LMO) (Pethick et al., 2025a). For matrix parameters, different operator-norm constraints lead to spectral, row/column-normalized, and sign updates. We differentiate the specified implementations of these update forms below. Together with Adam's adaptive diagonal rescaling, these transformations define the cases summarized in Table 1.

For updates depending on g only through momentum $m(g)$ with $\partial m/\partial g = cI$, the chain rule gives $J = c\,\partial\Phi/\partial m$; the same applies to matrix inputs. The Nesterov momentum $m = (1+\mu)g + \mu^2 m_{\mathrm{in}}$ gives $c = 1+\mu$, while the EMA momentum $m = \beta m_{\mathrm{in}} + (1-\beta)g$ gives $c = 1-\beta$.

Spectral Norm. A spectral-norm $\lVert\cdot\rVert_{2\to 2}$ constraint on the update yields a polar factor of the momentum matrix as the maximizing direction in eq. (8). For a linear layer with input and output dimensions d_{in} and d_{out}, let $M, R \in \mathbb{R}^{d_{\mathrm{out}}\times d_{\mathrm{in}}}$ denote the momentum and validation gradient matrices. We consider the finite Newton-Schulz (NS) update $\Phi_{\mathrm{Spectral}} = \gamma\,\mathrm{NS}(M)$, where $\gamma > 0$ is a gradient-independent scale (Bernstein & Newhouse, 2024). To compute $\mathrm{NS}(M)$, initialize $X^{(0)} = M/\rho_M$, where $\rho_M = \lVert M\rVert_F + \epsilon$ with stabilizer $\epsilon > 0$. For fixed coefficients (a_0, a_1, a_2), iterate

$$C^{(i)} = X^{(i)} X^{(i)\top}, \qquad L^{(i)} = a_0 I + a_1 C^{(i)} + a_2 (C^{(i)})^2, \qquad X^{(i+1)} = L^{(i)} X^{(i)}, \tag{9}$$

for $i = 0, \ldots, N-1$, with $\mathrm{NS}(M) = X^{(N)}$.

Proposition 2.2 (Spectral JVG for finite NS iterations). *For any finite integer $N \ge 0$, initialize $\overline{X}^{(N)} = R$ and propagate backwards for $i = N-1, \ldots, 0$:*

$$\begin{aligned}
\overline{L}^{(i)} &= \overline{X}^{(i+1)} X^{(i)\top}, \\
\overline{C}^{(i)} &= a_1 \overline{L}^{(i)} + a_2\left(\overline{L}^{(i)} C^{(i)\top} + C^{(i)\top}\overline{L}^{(i)}\right), \\
\overline{X}^{(i)} &= L^{(i)\top}\overline{X}^{(i+1)} + \left(\overline{C}^{(i)} + \overline{C}^{(i)\top}\right) X^{(i)}.
\end{aligned} \tag{10}$$

Then, the JVG of Φ_{Spectral} *is*

$$J^{\top}_{\text{Spectral}} R = \gamma c \left(\frac{\overline{X}^{(0)}}{\rho_M} - M \frac{\langle M, \overline{X}^{(0)} \rangle_F}{\rho_M^2 \|M\|_F} \right), \tag{Spectral-JVG}$$

where the second term is zero at $M = 0$*. The result holds for rectangular and rank-deficient matrices. This JVG is exact for the specified finite NS map; see Appendix A.3 for the derivation.*

Spectral updates are used in many matrix-based optimizers (Carlson et al., 2015b;a;c), including Muon and Scion for LLMs (Jordan et al., 2024b; Pethick et al., 2025a). The Muon optimizer uses Nesterov momentum and shape scaling $\gamma = \max(1, \sqrt{d_{\text{out}}/d_{\text{in}}})$.

Row and Column Norm. Row- and column-normalized updates arise from the operator norms $\|\cdot\|_{2\to\infty}$ and $\|\cdot\|_{1\to 2}$, respectively. We derive the JVG for row normalization, with column normalization as an extension. The update rule is $[\Phi_{\text{RowNorm}}]_{i,:} = \gamma \frac{M_{i,:}}{\rho_i}$ with $\rho_i = \|M_{i,:}\|_2 + \epsilon$.

Proposition 2.3 (Row- and column-normalization JVGs)**.** *The JVG of* Φ_{RowNorm} *is*

$$[J^{\top}_{\text{RowNorm}} R]_{i,:} = \gamma c \left(\frac{R_{i,:}}{\rho_i} - M_{i,:} \frac{\langle M_{i,:}, R_{i,:} \rangle}{\rho_i^2 \|M_{i,:}\|_2} \right). \tag{RowNorm-JVG}$$

The second term is zero at $M_{i,:} = 0$*. Column normalization follows similarly by exchanging rows and columns. See Appendix A.4 for details.*

Row and column normalization are used in RMNP (Deng et al., 2026) and SCALE (Glentis et al., 2026), respectively. RMNP uses the same momentum and shape scaling settings as Muon.

Sign. The $\|\cdot\|_{1\to\infty}$ operator-norm constraint is equivalent to an entrywise ℓ_∞ constraint. It reduces eq. (8) to a coordinatewise maximization problem, giving the map $\Phi_{\text{Sign}} = \text{sign}(m)$. The sign's derivative is zero away from zero and undefined at zero; therefore, we use the following smooth map.

Proposition 2.4 (Surrogate Sign JVG)**.** *For the scoring surrogate* $\widetilde{\Phi}_{\text{Sign}}(g) = \tanh(m(g)/\tau_s)$ *with fixed* $\tau_s > 0$*, the JVG is*

$$\widetilde{J}^{\top}_{\text{Sign}} r = \frac{c}{\tau_s} \operatorname{sech}^2\left(\frac{m}{\tau_s}\right) \odot r. \tag{Sign-JVG}$$

The tilde denotes the surrogate Jacobian; updates use $\text{sign}(m)$*. See Appendix A.5 for the derivation.*

Sign-based optimizers are originally considered in (Carlson et al., 2015b;a;c) and are further developed in Signum (Bernstein et al., 2018) and Lion (Chen et al., 2023) with EMA momentum.

2.3 One-backward streaming implementation

We now turn the optimizer-aware scoring rule into an online mixing procedure that shares a single forward-backward pass with training. Computing the domain utility in eq. (5) with the step-t JVG $J_t^{\top} r_t$ would require the current optimizer transformation during streaming backward. To avoid this dependency, we use $J_{t-1}^{\top} r_t$ instead: only the optimizer Jacobian is taken from the preceding step, while both domain and validation gradients remain current. The expressions in Section 2.2 compute each layer's JVG directly, without explicitly forming the Jacobian.

We realize both domain scoring and the model update from a single forward-backward pass on the concatenation of a reference minibatch and training data sampled under the mixture α_t. Following Wang et al. (2024), we reuse input activations $a_{t,i}^{\ell}$ and output-gradient vectors $b_{t,i}^{\ell}$ from the shared pass at each selected linear layer $\ell \in \mathcal{L}$. Their outer product $b_{t,i}^{\ell} (a_{t,i}^{\ell})^{\top}$ gives the gradient contribution at position i. The domain scores can be written in the following way from those factors:

$$s_{t,d} = \left\langle g_{t,d}, J_{t-1}^{\top} r_t \right\rangle = \sum_{\ell \in \mathcal{L}} \frac{1}{|\mathcal{B}_{t,d}|} \sum_{i \in \mathcal{B}_{t,d}} b_{t,i}^{\ell\,\top} \left(J_{t-1}^{\ell\,\top} r_t^{\ell} \right) a_{t,i}^{\ell}. \tag{11}$$

Here, $g_{t,d}$ and r_t collect gradients over $\mathcal{L}$, and $\mathcal{B}_{t,d}$ indexes positions in domain-d samples. We accumulate scores during backward and release layer-local buffers after use. We remove the validation gradient from the joint gradient before the optimizer update, leaving only the training gradient for model updates. Implementation details are given in Appendix B. To reduce minibatch noise, we accumulate scores over H-step windows and use eq. (6) to determine the next window's mixture.

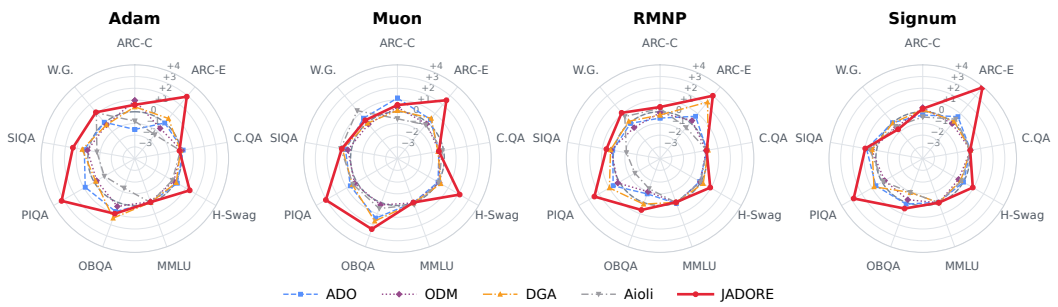

Figure 1: Benchmark breakdown on SlimPajama. Accuracy differences from Uniform of 124M model. JADORE has the highest mean accuracy across optimizers. Full results in Appendix C.2.

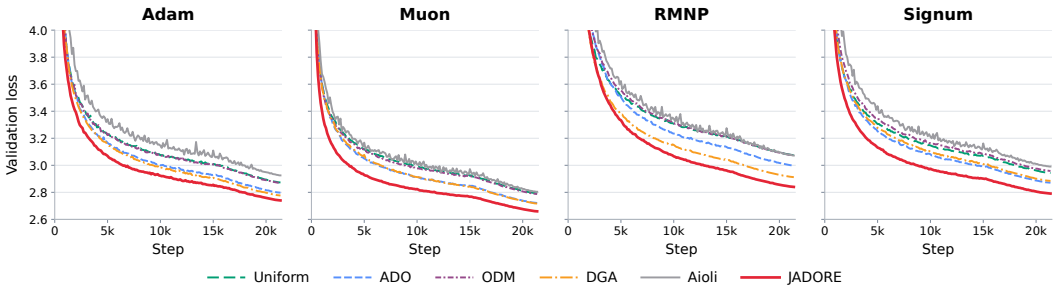

Figure 2: Validation-loss trajectories on the 1.07B model pretrained on SlimPajama across optimizers.

3 Experiments

We evaluate JADORE in pretraining and supervised fine-tuning across optimizers, examining downstream performance and how learned domain allocations vary with the optimizer and training stage. The following experiments are not to claim universal superiority of a single optimizer, which can be problem-dependent, but rather to show the benefits of optimizer-aware data mixing. We then empirically examine JADORE's computational efficiency with our JVG streaming implementation.

3.1 Pretraining

We assess the effectiveness of optimizer-aware data mixing in pretraining across model scales, training corpora, and optimizers. Our experiments are conducted on SlimPajama ($K = 7$) (Soboleva et al., 2023) and The Pile ($K = 17$) (Gao et al., 2020). We evaluate Adam, Muon, RMNP, and Signum to cover the four update geometries introduced in Section 2.2.

We evaluate 0-shot accuracy on nine downstream benchmarks: MMLU, HellaSwag, ARC-Challenge, ARC-Easy, PIQA, SIQA, OBQA, WinoGrande, and CommonsenseQA. The reference set is constructed following Wang et al. (2026b) by retrieving documents from each corpus' validation split based on embedding similarity to benchmark validation queries. All baselines requiring reference sets share this construction. Results are averaged over three training seeds, and average is the unweighted mean across the benchmarks. More training and evaluation details are provided in Appendix C.1.

Baselines. Alongside uniform weights, we compare with four representative online methods: ADO allocates data according to learning potential estimated from per-domain scaling laws (Jiang et al., 2025); ODM adapts sampling through a multi-armed bandit with training-loss rewards (Albalak et al., 2023); DGA reweights domains using raw-gradient alignment with a reference set (Fan et al., 2024a); and Aioli optimizes domain proportions through a dynamic mixing law fitted to validation-loss changes (Chen et al., 2025). On SlimPajama, we additionally compare with fixed mixtures from DoReMi (Xie et al., 2023) and DoGE (Fan et al., 2024b), using weights reported by Xie et al. (2025).

Table 2: Benchmark results on SlimPajama with 1.07B models. Mean accuracy (%) over three seeds. "Avg." denotes the benchmarks average; † marks offline mixing baselines.

	Method	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	Uniform	24.45	44.63	19.66	37.70	23.11	28.20	66.87	39.38	50.00	37.11
	DoReMi†	25.00	44.95	19.49	39.64	22.98	30.40	68.61	38.74	51.41	37.91
	DoGE†	25.51	**46.72**	19.74	42.45	23.55	30.60	68.39	39.66	51.72	38.71
	ADO	24.36	45.43	19.12	40.17	23.81	30.20	68.23	39.36	51.38	38.01
	ODM	23.38	43.43	19.57	37.78	23.50	29.70	66.81	38.72	50.55	37.05
	DGA	**25.81**	44.91	19.57	41.31	23.14	28.70	68.50	**39.71**	**52.13**	38.20
	Aioli	24.49	42.76	19.57	35.89	23.01	28.10	66.46	37.64	51.50	36.60
	JADORE	25.56	46.70	**19.94**	**42.88**	**24.04**	**30.80**	**70.02**	39.66	51.26	**38.98**
Muon	Uniform	24.66	44.99	19.86	40.90	23.91	31.10	68.34	38.84	**52.76**	38.37
	DoReMi†	**26.28**	44.87	20.23	41.96	**24.59**	30.20	68.72	39.92	51.54	38.70
	DoGE†	25.51	46.80	19.33	45.69	22.60	29.40	69.75	39.51	52.22	38.98
	ADO	25.47	46.72	19.41	44.34	24.13	31.20	69.53	**40.43**	51.70	39.21
	ODM	25.47	44.91	19.90	41.55	23.90	30.70	68.04	39.33	51.14	38.33
	DGA	24.70	45.37	**20.80**	44.38	24.16	31.10	69.80	39.59	51.86	39.08
	Aioli	24.70	44.99	19.66	40.32	23.47	31.00	67.87	38.23	51.70	37.99
	JADORE	26.02	**47.43**	19.41	**47.45**	22.99	**33.50**	**71.08**	39.33	52.13	**39.93**
RMNP	Uniform	22.48	40.30	19.57	32.86	22.90	28.10	64.01	37.21	**51.78**	35.47
	DoReMi†	23.63	41.04	19.57	33.64	22.94	27.80	65.34	38.02	50.67	35.85
	DoGE†	24.32	42.89	19.57	35.82	22.85	28.80	65.83	38.59	51.43	36.67
	ADO	24.32	41.41	19.57	34.43	22.95	30.00	65.86	38.59	51.66	36.53
	ODM	22.74	39.67	19.57	32.51	22.96	27.40	63.47	37.77	50.32	35.16
	DGA	**24.87**	42.99	19.57	36.21	22.87	28.30	66.59	38.87	49.61	36.65
	Aioli	23.29	40.93	19.57	32.55	**22.98**	28.50	64.99	38.28	50.47	35.73
	JADORE	24.45	**43.92**	**19.66**	**38.92**	22.93	**30.20**	**68.25**	**39.97**	50.63	**37.66**
Signum	Uniform	23.04	42.24	**19.57**	35.58	23.08	29.80	66.13	37.18	50.99	36.40
	DoReMi†	**25.26**	41.41	19.45	36.61	22.92	29.60	66.70	38.38	52.09	36.94
	DoGE†	24.74	43.78	19.45	38.55	**23.68**	29.40	66.90	38.79	49.41	37.19
	ADO	**25.26**	43.62	19.33	37.80	23.37	29.10	66.98	37.74	50.75	37.11
	ODM	23.85	41.58	**19.57**	35.64	22.98	29.00	65.97	37.95	51.18	36.41
	DGA	24.66	43.27	**19.57**	37.28	23.52	28.80	66.81	38.08	52.13	37.12
	Aioli	22.95	41.58	**19.57**	34.68	22.98	29.20	65.72	37.77	**52.17**	36.29
	JADORE	24.57	**44.84**	19.45	**40.20**	22.90	**30.20**	**67.68**	**38.82**	50.67	**37.70**

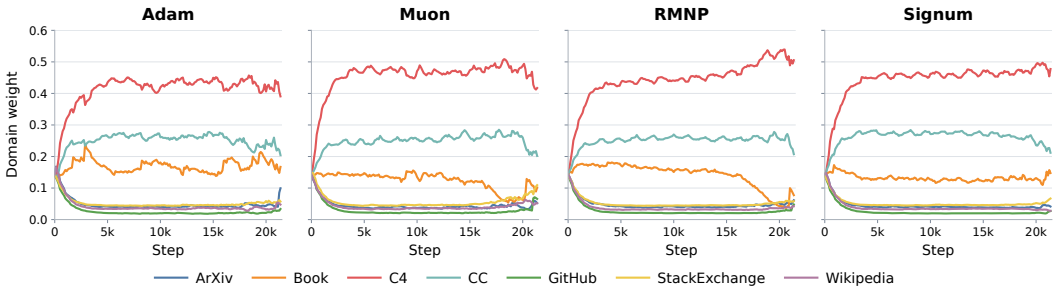

Figure 3: JADORE's domain weights for the 1.07B model trained on SlimPajama ($K = 7$).

SlimPajama at 124M and 1.07B. We train both model sizes with approximately $1\times$ Chinchilla token budgets. At 124M, Figure 1 shows gains over Uniform across multiple benchmarks and all four optimizers, with particularly clear improvements on ARC-E and PIQA. The full results in Appendix C.2 show that JADORE achieves the highest average accuracy among the evaluated online and offline methods for each optimizer. Such gains also extend to out-of-distribution benchmarks (Appendix C.4).

These benefits extend to 1.07B, where JADORE retains the highest unweighted 0-shot average accuracy across all four optimizers as shown in Table 2. Its gains over Uniform reach 1.30–2.19 percentage points, accompanied by lower validation loss through most of training in Figure 2. Moreover, the domain-weight trajectories in Figure 3 show shared domain preferences alongside optimizer-dependent effects. While all four optimizers favor C4 and CC, Adam and Signum generally retain more weight on Book late in training, whereas Muon increasingly emphasizes GitHub and StackExchange. The domain-wise view in Appendix C.2 (Figure 7) highlights these differences.

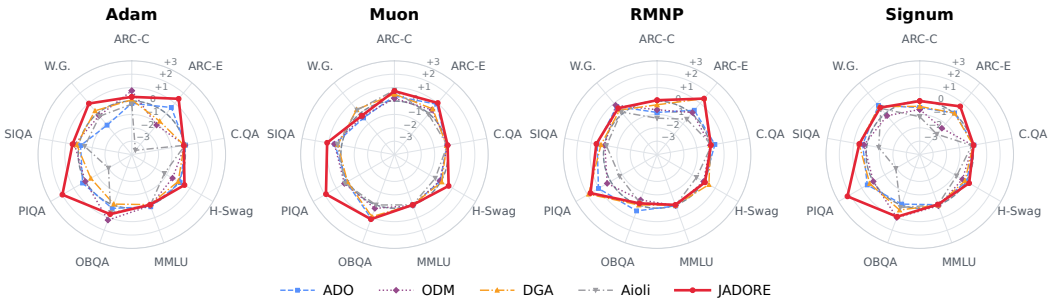

Figure 4: Benchmark breakdown on The Pile. Accuracy differences from Uniform of 124M model. JADORE has the highest mean accuracy across optimizers. Full results in Appendix C.2.

Table 3: MMLU subjects 5-shot accuracy (%) for LLaMA-2-7B fine-tuned on LESS data.

	Method	Soc.	Pol.	Alg.	Anat.	Astr.	Eth.	Clin.	Bio.	Chem.	Avg.
Adam	Uniform	54.89	59.00	34.00	48.64	**43.42**	48.67	48.18	43.75	35.67	46.25
	ADO	54.89	59.00	34.33	48.40	42.11	48.67	**48.43**	43.75	36.33	46.21
	ODM	55.89	59.67	35.67	48.64	42.76	49.00	47.55	43.75	35.33	46.47
	DGA	55.06	**60.33**	33.33	48.64	**43.42**	49.00	48.30	44.21	35.00	46.37
	Aioli	**56.88**	59.33	35.00	48.40	**43.42**	49.33	47.80	44.44	35.33	46.66
	JADORE	56.72	57.67	**36.67**	**49.38**	41.67	**50.00**	47.92	**44.68**	**36.67**	**46.82**
Muon	Uniform	61.19	55.00	34.00	48.89	**40.13**	**45.00**	47.04	45.83	44.00	46.79
	ADO	61.19	55.00	34.00	48.89	39.69	**45.00**	**47.92**	45.37	44.00	46.78
	ODM	**62.35**	**55.67**	35.33	**49.63**	39.91	**45.00**	46.54	45.83	42.33	46.95
	DGA	62.19	**55.67**	34.67	49.14	39.69	**45.00**	47.04	45.60	44.33	47.03
	Aioli	61.03	54.67	**35.67**	49.14	39.69	**45.00**	47.42	45.60	42.67	46.77
	JADORE	62.02	**55.67**	35.00	48.89	39.69	**45.00**	47.55	**46.30**	**45.00**	**47.24**
RMNP	Uniform	54.89	60.33	36.33	47.41	41.45	50.67	47.42	43.98	35.00	46.39
	ADO	55.06	59.33	37.67	46.91	41.01	51.33	47.67	**44.21**	**36.67**	46.65
	ODM	54.73	61.00	36.67	48.15	41.23	51.00	**48.05**	42.82	34.67	46.48
	DGA	55.39	60.33	36.33	48.15	41.67	50.33	47.80	42.82	35.00	46.42
	Aioli	54.73	**61.33**	**38.33**	46.42	41.23	52.33	47.80	41.67	36.33	46.69
	JADORE	**58.21**	59.67	35.00	**49.14**	**42.11**	**54.00**	47.92	**44.21**	35.00	**47.25**
Signum	Uniform	47.26	61.33	33.33	45.68	40.13	51.00	**49.56**	42.36	35.00	45.07
	ADO	47.76	62.33	33.67	45.93	40.13	51.00	49.06	42.13	**35.67**	45.30
	ODM	48.26	61.33	35.00	44.94	38.60	**52.00**	48.68	40.97	32.33	44.68
	DGA	48.26	61.67	33.00	46.17	39.47	51.00	49.31	42.59	35.00	45.16
	Aioli	48.42	60.33	32.33	45.43	40.13	50.67	48.55	41.67	33.00	44.50
	JADORE	**54.39**	**63.33**	**36.33**	**47.65**	**41.23**	49.33	48.93	**43.06**	33.00	**46.36**

The Pile. To extend across datasets, we train a 124M model on the 17-domain Pile dataset with a 2× Chinchilla token budget. Figure 4 shows gains over Uniform across all four optimizers, supporting the applicability of JADORE across both training corpora and a larger number of domains. The full results in Appendix C.2 show that JADORE achieves the highest average accuracy with each optimizer.

3.2 Supervised fine-tuning

We extend our evaluation to supervised fine-tuning (SFT) of LLaMA-2-7B (Touvron et al., 2023) with LoRA (Hu et al., 2022) on the LESS training data (Xia et al., 2024), comprising $K = 4$ domains: FLAN-v2, CoT, Dolly, and OpenAssistant. Following Wang et al. (2024), we target nine MMLU subjects (Hendrycks et al., 2020), pooling their validation splits as reference set. We report 5-shot test accuracy averaged over three seeds, and further training and evaluation details in Appendix C.1.

As shown in Table 3, JADORE achieves the highest average accuracy among the evaluated online methods with every optimizer, extending its benefits from pretraining to SFT. The corresponding weight trajectories in Figure 5 reveal different allocations across optimizers for the same target tasks. Muon shifts weight from FLAN-v2 toward CoT late in training, whereas Adam and Signum finish with CoT below its initial weight. Together with the pretraining observations, these trajectories support adapting data allocation to the optimizer and training stage.

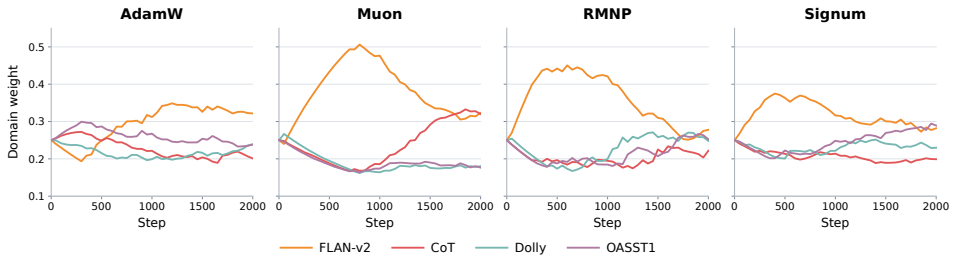

Figure 5: JADORE's domain weights for the LLaMA-2-7B fine-tuned on LESS data ($K = 4$).

Table 4: Scoring comparison. Accuracy of Raw, OPUS-P, and JADORE on SlimPajama pretraining.

	Method	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	Raw	21.84	36.49	**19.82**	28.20	**22.95**	25.87	60.83	36.76	51.88	33.85
	OPUS-P	22.16	36.52	19.66	28.41	22.91	**26.20**	60.70	36.39	51.62	33.84
	JADORE	**22.92**	**37.40**	19.55	**28.81**	22.93	26.13	**62.21**	**36.98**	**51.99**	**34.32**
Muon	Raw	22.61	38.22	**19.60**	30.13	**22.94**	27.20	62.62	36.69	50.78	34.53
	OPUS-P	22.78	**38.34**	19.55	30.26	**22.94**	26.07	62.40	36.51	50.49	34.37
	JADORE	**22.90**	38.33	19.57	**30.34**	22.91	**27.40**	**63.24**	**36.93**	**50.86**	**34.72**
RMNP	Raw	21.50	32.98	**19.60**	27.26	**22.94**	25.27	58.72	35.41	49.83	32.61
	JADORE	**22.44**	**36.39**	**19.60**	**28.22**	22.92	**26.73**	**61.52**	**36.39**	**51.93**	**34.02**
Signum	Raw	21.81	35.51	19.57	27.81	**22.93**	25.60	59.97	**36.63**	**51.88**	33.52
	JADORE	**22.10**	**37.29**	**19.60**	**28.30**	**22.93**	**25.93**	**61.14**	36.39	50.83	**33.83**

3.3 Ablation Study and Computational Analysis

Optimizer-aware scoring. To isolate the contribution of the optimizer Jacobian, we vary only the scoring rule under the same SlimPajama 124M setup. *Raw* uses $\langle g_{t,d}, r_t \rangle$, following existing gradient-based domain reweighting (Fan et al., 2024a). Recently, OPUS introduces optimizer-aware data selection using preconditioned gradients (Wang et al., 2026b). We test its preconditioner in our mixing framework as *OPUS-P*, with score $\langle P_{t-1} g_{t,d}, r_t \rangle$. For Adam, it fixes the adaptive denominator, omitting the second term in Adam-JVG; for Muon, it uses a frozen polynomial preconditioner from the first Newton-Schulz iteration. OPUS evaluates neither RMNP nor Signum, so we compare these only with Raw. In Table 4, JADORE achieves the highest average accuracy for all four optimizers and the best or tied-best accuracy on most benchmarks, supporting the benefit of our optimizer-aware score. The SFT ablation in the appendix (Table 10) shows consistent conclusions.

Computational efficiency. Table 5 reports the training time of JADORE. Compared to sequential-domain scoring, JADORE substantially reduces the computational overhead. Moreover, computing the JVG in JADORE introduces negligible cost beyond raw-gradient scoring through the same joint pass. We also evaluate scoring only the last transformer block, which reduces training time while maintaining similar average accuracy, as shown in Appendix C.3 alongside timing and memory analyses.

Additional experiments. We compare $J_t^\top r_t$ and $J_{t-1}^\top r_t$ elaborated in Appendix C.5, finding comparable average accuracy at lower cost with the latter. We also examine sensitivity to H, λ, and β_α in Appendix C.7. Moreover, we refine DoReMi and DoGE mixtures using them as KL anchors, improving average accuracy over the corresponding fixed mixtures (Appendix C.6).

Table 5: Training time (min) on SlimPajama 124M.

	Adam	Muon	RMNP	Signum
Plain training	37.4	39.8	37.7	37.4
Sequential scoring	46.3	49.6	46.4	46.2
Joint Raw	40.1	42.6	40.5	40.0
JADORE	40.1	43.0	40.6	40.1
JADORE (last block)	38.5	41.0	38.8	38.5

4 Conclusion

We introduce JADORE, a domain reweighting framework built on the principle that a domain's value depends on how the optimizer turns its data into learning. We design an efficient online algorithm applicable across distinct update geometries. Experiments demonstrate consistent improvements in pretraining and supervised fine-tuning across the various optimizers. These findings highlight the importance of considering data-centric problems and optimization together in model training.

AI USE STATEMENT

In this work, we used generative AI tools for <implement methods>. We have not used generative AI tools for <help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat dataset, support qualitative and thematic data analysis, interpret results>, and <Generate synthetic data sets, assist with translation> are not applicable to this work. Additionally, we used generative AI tools for <creating or editing software code, editing a research paper to improve readability>. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses publicly available datasets and models. We have not identified any additional ethical concerns arising from this study.

REPRODUCIBILITY STATEMENT

We provide method derivations, implementation details, and experimental configurations in the appendix. The supplementary materials include code and a README for training 124M models on SlimPajama with Adam, Muon, RMNP, and Signum. We will publicly release the complete experimental code after the camera-ready submission.

REFERENCES

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.

Amir Beck. *First-order methods in optimization*. SIAM, 2017.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*, 2018.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method and its connection to an iterative shrinkage method. *Computational Optimization and applications*, 2009.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015a.

David E. Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015b.

David E. Carlson, Ya-Ping Hsieh, Edo Collins, Lawrence Carin, and Volkan Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 2015c.

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. *International Conference on Learning Representations (ICLR)*, 2025.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Shenyang Deng, Zhuoli Ouyang, Tianyu Pang, Zihang Liu, Ruochen Jin, Shuhua Yu, and Yaoqing Yang. RMNP: Row-momentum normalized preconditioning for scalable matrix-based optimization. *International Conference on Machine Learning (ICML)*, 2026.

Timothy Dozat. Incorporating Nesterov momentum into adam. 2016.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning (ICML)*, 2022.

Simin Fan, David Grangier, and Pierre Ablin. Dynamic gradient alignment for online data mixing. *arXiv preprint arXiv:2410.02498*, 2024a.

Simin Fan, Matteo Pagliardini, and Martin Jaggi. DoGE: Domain reweighting with generalization estimation. In *International Conference on Machine Learning (ICML)*, 2024b.

Simin Fan, Maria Ios Glarou, and Martin Jaggi. GRAPE: Optimize data mixture for group robust multi-target adaptive pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Dmitrii Feoktistov, Timofey Belinsky, Andrey Veprikov, Amir Zainullin, and Aleksandr Beznosikov. Softsign: Smooth sign in your optimizer for better parameter heterogeneity handling. *International Conference on Machine Learning (ICML)*, 2026.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Athanasios Glentis, Jiaxiang Li, Andi Han, and Mingyi Hong. Memory-efficient LLM pretraining via minimalist optimizer design. *International Conference on Machine Learning (ICML)*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *International Conference on Learning Representations (ICLR)*, 2025.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b. URL https://kellerjordan.github.io/posts/muon/.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, 2017.

Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. PiKE: Adaptive data mixing for large-scale multi-task learning under low gradient conflicts. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025a.

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. *International Conference on Learning Representations (ICLR)*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. *International Conference on Machine Learning (ICML)*, 2025a.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Scion. https://github.com/LIONS-EPFL/scion, 2025b.

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Antonio Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (l_0, l_1)-smoothness. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025c.

Thomas Pethick, Kimon Antonakopoulos, Antonio Silveti-Falls, Leena Chennuru Vankadara, and Volkan Cevher. Training neural networks at any scale: An exposition. *IEEE Signal Processing Magazine*, 43(3):21–36, 2026a.

Thomas Pethick, Wanyun Xie, Roman Machacek, and Volkan Cevher. Optimistic dual averaging unifies modern optimizers. *arXiv preprint arXiv:2605.11172*, 2026b.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Haoru Tan, Sitong Wu, Yanfeng Chen, Jun Xia, Ruobing Xie, Bin Xia, Xingwu Sun, and Xiaojuan Qi. Fastmix: Fast data mixture optimization via gradient descent. *arXiv preprint arXiv:2606.14971*, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Fangxin Wang, Peyman Baghershahi, Langzhou He, Henry Peng Zou, Sourav Medya, and Philip S. Yu. Filter-then-weight: Online data selection and reweighting for LLM fine-tuning. *arXiv preprint arXiv:2604.00001*, 2026a.

Jiachen T. Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. GREATS: Online selection of high-quality data for LLM training in every iteration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shaobo Wang, Xuan Ouyang, Tianyi Xu, Yuzheng Hu, Jialin Liu, Guo Chen, Tianyu Zhang, Junhao Zheng, Kexin Yang, Xingzhang Ren, et al. OPUS: Towards efficient and principled data selection in large language model pre-training in every iteration. *arXiv preprint arXiv:2602.05400*, 2026b.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. *International Conference on Machine Learning (ICML)*, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. *International Conference on Machine Learning (ICML)*, 2025.

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 (ICLR)*, 2025.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *arXiv preprint arXiv:2412.04506*, 2024.

Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. Generalized conditional gradient for sparse estimation. *Journal of Machine Learning Research*, 2017.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019.

A Optimizer JVG Derivations

We follow the notation and update maps in Section 2.2. All derivatives are taken with respect to the current gradient, with incoming optimizer state and hyperparameters held fixed. For the norm-based maps, momentum satisfies $\mathrm{d}m = c\,\mathrm{d}g$ or $\mathrm{d}M = c\,\mathrm{d}G$, and γ denotes a gradient-independent output scale when present. We derive the JVGs in terms of these general coefficients and specify the experimental optimizer choices after each derivation.

A.1 SGD JVG derivation

For SGD without momentum, the update is $\theta^+ = \theta - \eta\Phi_{\mathrm{SGD}}(g)$ with $\Phi_{\mathrm{SGD}}(g) = g$, keeping the learning rate η separate as in the main text. An infinitesimal gradient change $\mathrm{d}g$ therefore produces the same change in the update direction:

$$\mathrm{d}\Phi_{\mathrm{SGD}} = \mathrm{d}g, \qquad J_{\mathrm{SGD}} = \frac{\partial \Phi_{\mathrm{SGD}}}{\partial g} = I, \qquad J_{\mathrm{SGD}}^{\top} r = r. \tag{12}$$

Thus the utility in eq. (5) reduces to $s_d = \langle g_d, r\rangle$, which recovers the raw-gradient score used in prior work (Fan et al., 2024b;a; 2025).

A.2 Adaptive diagonal scaling: Adam JVG derivation

We derive the Adam JVG used in eq. (Adam-JVG). We use $\mathrm{d}x$ to denote the first-order differential of a quantity x: for $y = f(x)$, $\mathrm{d}y = f'(x)\,\mathrm{d}x$. Thus, $\mathrm{d}g$ is an infinitesimal change in the gradient and $\mathrm{d}\Phi_{\mathrm{Adam}} = J_{\mathrm{Adam}}\,\mathrm{d}g$; it is not an additional optimizer state. The upright d is distinct from the domain index d. Let m_{in} and v_{in} denote the first and second moments entering the update, and let $n \geq 1$ be the update count after incorporating the gradient g. The Adam direction is

$$\widehat{m} = \frac{\beta_1 m_{\mathrm{in}} + (1-\beta_1)g}{1-\beta_1^n}, \qquad \widehat{v} = \frac{\beta_2 v_{\mathrm{in}} + (1-\beta_2)g^2}{1-\beta_2^n},$$
$$q = \sqrt{\widehat{v}} + \epsilon, \qquad \Phi_{\mathrm{Adam}}(g) = \frac{\widehat{m}}{q}.$$

When bias correction is disabled, each denominator $1-\beta_i^n$ is replaced by 1. As in the main text, define

$$\kappa_1 = \frac{1-\beta_1}{1-\beta_1^n}, \qquad \kappa_2 = \frac{1-\beta_2}{1-\beta_2^n}.$$

During differentiation, m_{in}, v_{in}, and n are held fixed. The moment differentials are therefore

$$\mathrm{d}\widehat{m} = \kappa_1\,\mathrm{d}g, \qquad \mathrm{d}\widehat{v} = 2\kappa_2\, g\,\mathrm{d}g. \tag{13}$$

Assuming every coordinate of $\widehat{v}$ is positive, differentiating the square-root denominator q gives

$$\mathrm{d}q = \frac{\mathrm{d}\widehat{v}}{2\sqrt{\widehat{v}}} = \frac{\kappa_2\, g\,\mathrm{d}g}{\sqrt{\widehat{v}}}. \tag{14}$$

Applying the quotient rule to $\Phi_{\mathrm{Adam}}(g) = \widehat{m}/q$ yields

$$\mathrm{d}\Phi_{\mathrm{Adam}} = \frac{\mathrm{d}\widehat{m}}{q} - \frac{\widehat{m}\,\mathrm{d}q}{q^2} = \left[\frac{\kappa_1}{q} - \frac{\kappa_2 \widehat{m} \odot g}{\sqrt{\widehat{v}} \odot q^2}\right]\mathrm{d}g. \tag{15}$$

Thus the Jacobian is diagonal. Multiplying its transpose by the validation gradient r gives

$$J_{\mathrm{Adam}}^{\top} r = \left[\frac{\kappa_1}{q} - \frac{\kappa_2 \widehat{m} \odot g}{\sqrt{\widehat{v}} \odot q^2}\right] \odot r. \tag{16}$$

All operations above are elementwise. Gradient-independent decoupled weight decay does not contribute to this Jacobian.

A.3 SPECTRAL UPDATES: JVG FOR THE FINITE NEWTON-SCHULZ MAP

For one matrix parameter, let G, $M = M(G)$, and R denote the gradient, momentum, and validation gradient matrices in $\mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. We derive the JVG of $\Phi_{\text{Spectral}}(G) = \gamma\,\text{NS}(M(G))$, where $\mathrm{d}M = c\,\mathrm{d}G$ and NS is the specified finite Newton-Schulz map. We first differentiate NS with respect to M, then account for momentum and output scaling. We present the left-Gram form in Algorithm 1; the right-Gram form follows by transposition.

Algorithm 1 Finite Newton-Schulz map (left-Gram form)

Require: Input M, iterations N, fixed coefficients (a_0, a_1, a_2), stabilizer $\epsilon > 0$
1: $\rho_M \leftarrow \|M\|_F + \epsilon$
2: $X^{(0)} \leftarrow M/\rho_M$
3: **for** $i = 0, \ldots, N-1$ **do**
4: $\quad C^{(i)} \leftarrow X^{(i)} X^{(i)\top}$
5: $\quad L^{(i)} \leftarrow a_0 I + a_1 C^{(i)} + a_2 \big(C^{(i)}\big)^2$
6: $\quad X^{(i+1)} \leftarrow L^{(i)} X^{(i)}$
7: **end for**
8: **return** $\text{NS}(M) = X^{(N)}$

Throughout this derivation, $\mathrm{d}Z$ denotes an infinitesimal perturbation of a matrix Z. Because the NS inputs and outputs are matrices, we pair two matrices using the Frobenius inner product $\langle A, B\rangle_F = \mathrm{tr}(A^\top B) = \sum_{p,q} A_{pq} B_{pq}$, the matrix analogue of a vector dot product. Let R be the matrix direction in which we measure the output change $\mathrm{d}X^{(N)}$; in our score, it is the validation-gradient matrix for this parameter. Our goal is to rewrite $\langle R, \mathrm{d}X^{(N)}\rangle_F$ as an inner product with the input perturbation $\mathrm{d}M$. The matrix paired with $\mathrm{d}M$ is, by definition, the desired VJP $J_{\text{NS}}(M)^\top R$:

$$\mathrm{d}X^{(N)} = J_{\text{NS}}(M)\,\mathrm{d}M, \qquad \left\langle R, \mathrm{d}X^{(N)} \right\rangle_F = \left\langle J_{\text{NS}}(M)^\top R, \mathrm{d}M \right\rangle_F, \tag{17}$$

where the second equality defines the transpose action.

We obtain the transpose action by reversing the N iterations. An overline denotes the coefficient paired with the perturbation of the corresponding forward variable. Starting from the output, initialize

$$\overline{X}^{(N)} = R.$$

For one reverse iteration i, differentiating $X^{(i+1)} = L^{(i)} X^{(i)}$ gives

$$\mathrm{d}X^{(i+1)} = \mathrm{d}L^{(i)} X^{(i)} + L^{(i)} \mathrm{d}X^{(i)}.$$

Pairing both sides with $\overline{X}^{(i+1)}$ and rearranging the trace yields

$$\left\langle \overline{X}^{(i+1)}, \mathrm{d}X^{(i+1)} \right\rangle_F = \left\langle \overline{X}^{(i+1)} X^{(i)\top}, \mathrm{d}L^{(i)} \right\rangle_F + \left\langle L^{(i)\top} \overline{X}^{(i+1)}, \mathrm{d}X^{(i)} \right\rangle_F. \tag{18}$$

The first term defines the quantity propagated to $L^{(i)}$,

$$\overline{L}^{(i)} = \overline{X}^{(i+1)} X^{(i)\top},$$

while the second is the direct contribution to the coefficient of $\mathrm{d}X^{(i)}$.

Next, differentiating $L^{(i)} = a_0 I + a_1 C^{(i)} + a_2 (C^{(i)})^2$ gives

$$\mathrm{d}L^{(i)} = a_1\,\mathrm{d}C^{(i)} + a_2 \left(\mathrm{d}C^{(i)} C^{(i)} + C^{(i)} \mathrm{d}C^{(i)} \right). \tag{19}$$

Substituting eq. (19) into $\langle \overline{L}^{(i)}, \mathrm{d}L^{(i)}\rangle_F$ and collecting the coefficient of $\mathrm{d}C^{(i)}$ gives

$$\overline{C}^{(i)} = a_1 \overline{L}^{(i)} + a_2 \left(\overline{L}^{(i)} C^{(i)\top} + C^{(i)\top} \overline{L}^{(i)} \right), \qquad \left\langle \overline{L}^{(i)}, \mathrm{d}L^{(i)} \right\rangle_F = \left\langle \overline{C}^{(i)}, \mathrm{d}C^{(i)} \right\rangle_F. \tag{20}$$

Finally, differentiating $C^{(i)} = X^{(i)} X^{(i)\top}$ gives

$$\begin{aligned} \mathrm{d}C^{(i)} &= \mathrm{d}X^{(i)} X^{(i)\top} + X^{(i)} \mathrm{d}X^{(i)\top}, \\ \left\langle \overline{C}^{(i)}, \mathrm{d}C^{(i)} \right\rangle_F &= \left\langle \left(\overline{C}^{(i)} + \overline{C}^{(i)\top} \right) X^{(i)}, \mathrm{d}X^{(i)} \right\rangle_F . \end{aligned} \tag{21}$$

Combining this indirect contribution through $C^{(i)}$ with the direct contribution in eq. (18) gives

$$\overline{X}^{(i)} = L^{(i)\top} \overline{X}^{(i+1)} + \left(\overline{C}^{(i)} + \overline{C}^{(i)\top} \right) X^{(i)}. \tag{22}$$

After reversing all N iterations, $\overline{X}^{(0)}$ satisfies

$$\left\langle R, \mathrm{d}X^{(N)} \right\rangle_F = \left\langle \overline{X}^{(0)}, \mathrm{d}X^{(0)} \right\rangle_F .$$

It remains to reverse the initial normalization $X^{(0)} = M/\rho_M$. For $M \neq 0$,

$$\mathrm{d}\rho_M = \mathrm{d}\|M\|_F = \frac{\langle M, \mathrm{d}M \rangle_F}{\|M\|_F}, \qquad \mathrm{d}X^{(0)} = \frac{\mathrm{d}M}{\rho_M} - \frac{M}{\rho_M^2} \frac{\langle M, \mathrm{d}M \rangle_F}{\|M\|_F}. \tag{23}$$

Consequently,

$$\left\langle \overline{X}^{(0)}, \mathrm{d}X^{(0)} \right\rangle_F = \left\langle \frac{\overline{X}^{(0)}}{\rho_M} - M \frac{\langle M, \overline{X}^{(0)} \rangle_F}{\rho_M^2 \|M\|_F}, \mathrm{d}M \right\rangle_F .$$

Comparing this expression with the VJP definition in eq. (17) yields

$$J_{\mathrm{NS}}(M)^\top R = \frac{\overline{X}^{(0)}}{\rho_M} - M \frac{\langle M, \overline{X}^{(0)} \rangle_F}{\rho_M^2 \|M\|_F}. \tag{24}$$

The first term treats the normalization scale as fixed; the second accounts for the dependence of ρ_M on M. At $M = 0$, the second term is defined as zero, giving the limiting derivative $\overline{X}^{(0)}/\epsilon$.

Finally, applying $\mathrm{d}M = c\,\mathrm{d}G$ and the fixed output scale γ gives the spectral JVG:

$$J_{\mathrm{Spectral}}^\top R = \gamma c\, J_{\mathrm{NS}}(M)^\top R = \gamma c \left(\frac{\overline{X}^{(0)}}{\rho_M} - M \frac{\langle M, \overline{X}^{(0)} \rangle_F}{\rho_M^2 \|M\|_F} \right). \tag{25}$$

The second term is defined as zero at $M = 0$. This is the exact JVG of the specified finite NS map, including for rectangular and rank-deficient inputs. Algorithm 2 summarizes the calculation.

Algorithm 2 Exact spectral JVG for finite NS iterations (left-Gram form)

Require: M, validation gradient R, (a_0, a_1, a_2), N, ϵ, and scalars (c, γ)

Obtain ρ_M, $X^{(0:N)}$, $C^{(0:N-1)}$, and $L^{(0:N-1)}$ as in Algorithm 1
$\overline{X}^{(N)} \leftarrow R$
for $i = N-1, \ldots, 0$ **do**
    $\overline{L}^{(i)} \leftarrow \overline{X}^{(i+1)} X^{(i)\top}$
    $\overline{C}^{(i)} \leftarrow a_1 \overline{L}^{(i)} + a_2 \left(\overline{L}^{(i)} C^{(i)\top} + C^{(i)\top} \overline{L}^{(i)} \right)$
    $\overline{X}^{(i)} \leftarrow L^{(i)\top} \overline{X}^{(i+1)} + \left(\overline{C}^{(i)} + \overline{C}^{(i)\top} \right) X^{(i)}$
end for
$J_{\mathrm{NS}}(M)^\top R \leftarrow \overline{X}^{(0)}/\rho_M - M\langle M, \overline{X}^{(0)} \rangle_F / (\rho_M^2 \|M\|_F)$
    with the second term set to zero when $M = 0$.
$J_{\mathrm{Spectral}}^\top R \leftarrow \gamma c\, J_{\mathrm{NS}}(M)^\top R$
return $J_{\mathrm{Spectral}}^\top R$

Scion (Pethick et al., 2025a;c) uses EMA momentum $M = (1-\beta)G + \beta M_{\mathrm{in}}$, so $c = 1 - \beta$, and shape scaling $\gamma = \sqrt{d_{\mathrm{out}}/d_{\mathrm{in}}}$. In our experiments, we follow Muon (Jordan et al., 2024b) using the Nesterov direction $M = (1+\mu)G + \mu^2 M_{\mathrm{in}}$, so $c = 1 + \mu$, with shape scaling $\gamma = \max(1, \sqrt{d_{\mathrm{out}}/d_{\mathrm{in}}})$.

A.4 Row- and column-normalized updates: JVG derivation

We derive the rowwise JVG in eq. (RowNorm-JVG). Let G, $M = M(G)$, and R denote the gradient, momentum, and validation gradient matrices of the same shape, with $\mathrm{d}M = c\,\mathrm{d}G$. The row-normalized map is

$$\rho_i = \|M_{i,:}\|_2 + \epsilon, \qquad [\Phi_{\mathrm{RowNorm}}(G)]_{i,:} = \gamma \frac{M_{i,:}}{\rho_i},$$

where $\epsilon > 0$ and $\gamma > 0$ are stabilizer and scaling parameters, respectively, and the subscript $i,:$ selects a row. We first differentiate with respect to M, then apply the momentum factor c.

For a nonzero row $M_{i,:}$, differentiating $\|M_{i,:}\|_2^2 = \langle M_{i,:}, M_{i,:}\rangle$ yields $2\|M_{i,:}\|_2\,\mathrm{d}\|M_{i,:}\|_2 = 2\langle M_{i,:}, \mathrm{d}M_{i,:}\rangle$. Hence the normalization scale has differential

$$\mathrm{d}\rho_i = \frac{\langle M_{i,:}, \mathrm{d}M_{i,:}\rangle}{\|M_{i,:}\|_2}. \tag{26}$$

Applying the quotient rule to the normalized row gives

$$\begin{aligned}[\mathrm{d}\Phi_{\mathrm{RowNorm}}]_{i,:} &= \gamma\left(\frac{\mathrm{d}M_{i,:}}{\rho_i} - \frac{M_{i,:}}{\rho_i^2}\,\mathrm{d}\rho_i\right) \\ &= \gamma\left(\frac{\mathrm{d}M_{i,:}}{\rho_i} - M_{i,:}\frac{\langle M_{i,:}, \mathrm{d}M_{i,:}\rangle}{\rho_i^2\|M_{i,:}\|_2}\right).\end{aligned} \tag{27}$$

The first term comes from changing the numerator; the second comes from changing its row norm. Pairing this differential with the validation row $R_{i,:}$ and collecting the coefficient of $\mathrm{d}M_{i,:}$ gives

$$\langle R_{i,:}, [\mathrm{d}\Phi_{\mathrm{RowNorm}}]_{i,:}\rangle = \gamma\left\langle \frac{R_{i,:}}{\rho_i} - M_{i,:}\frac{\langle M_{i,:}, R_{i,:}\rangle}{\rho_i^2\|M_{i,:}\|_2}, \mathrm{d}M_{i,:}\right\rangle.$$

Here we used the symmetry of the row inner product to move $\mathrm{d}M_{i,:}$ into the second argument. Each output row depends only on the corresponding input row, so summing these pairings over i gives the full Frobenius inner product. Substituting $\mathrm{d}M_{i,:} = c\,\mathrm{d}G_{i,:}$ and using $\langle R, \mathrm{d}\Phi_{\mathrm{RowNorm}}\rangle_F = \langle J_{\mathrm{RowNorm}}^\top R, \mathrm{d}G\rangle_F$ identifies each row of the JVG:

$$[J_{\mathrm{RowNorm}}^\top R]_{i,:} = \gamma c\left(\frac{R_{i,:}}{\rho_i} - M_{i,:}\frac{\langle M_{i,:}, R_{i,:}\rangle}{\rho_i^2\|M_{i,:}\|_2}\right). \tag{28}$$

At $M_{i,:} = 0$, the unscaled normalization map $M_{i,:} \mapsto M_{i,:}/\rho_i$ has differential $\mathrm{d}M_{i,:}/\epsilon$, even though the norm itself is not differentiable there. Thus the second term in eq. (28) is defined as zero, giving $[J_{\mathrm{RowNorm}}^\top R]_{i,:} = \gamma c R_{i,:}/\epsilon$.

For column normalization, the derivation applies independently to each column: replace $M_{i,:}, R_{i,:}$ by $M_{:,j}, R_{:,j}$ and set $\rho_j = \|M_{:,j}\|_2 + \epsilon$. The same factor γc applies for the chosen momentum and output scale. When a weight matrix is stored transposed, these column slices correspond to rows in the original orientation.

Scion (Pethick et al., 2025a) suggests scaling the row-normalized update by $\gamma = 1/\sqrt{d_{\mathrm{in}}}$ and column-normalized update by $\gamma = \sqrt{d_{\mathrm{out}}}$. In our experiments, we follow RMNP (Deng et al., 2026) using row normalization with the same Nesterov direction $M = (1+\mu)G + \mu^2 M_{\mathrm{in}}$ and shape scaling as our Muon implementation: $c = 1 + \mu$ and $\gamma = \max(1, \sqrt{d_{\mathrm{out}}/d_{\mathrm{in}}})$.

A.5 Sign updates: surrogate JVG derivation

We derive the surrogate JVG in eq. (Sign-JVG). For a momentum direction $m = m(g)$ with $\mathrm{d}m = c\,\mathrm{d}g$, the hard-sign update map is

$$\Phi_{\mathrm{Sign}}(g) = \mathrm{sign}(m). \tag{29}$$

The sign function is locally constant at every nonzero input, so its derivative is zero there; at zero, it is discontinuous and has no derivative. Consequently, the exact hard-sign Jacobian is zero whenever all momentum coordinates are nonzero. It cannot assign nonzero local utility scores in this region, even though a finite perturbation that crosses a sign boundary can change the update.

For domain scoring, we replace the sign function by a smooth map,

$$\widetilde{\Phi}_{\mathrm{Sign}}(g) = \tanh\left(\frac{m}{\tau_s}\right), \qquad \tau_s > 0,$$

where all operations are elementwise and τ_s controls the width of the transition around zero. We hold τ_s fixed when differentiating, along with incoming optimizer state and hyperparameters. Since $\frac{\mathrm{d}}{\mathrm{d}u}\tanh(u) = 1 - \tanh^2(u) = \mathrm{sech}^2(u)$, with $\mathrm{sech}(u) = 1/\cosh(u)$, the chain rule gives

$$\begin{aligned} \mathrm{d}\widetilde{\Phi}_{\mathrm{Sign}} &= \frac{1}{\tau_s}\,\mathrm{sech}^2\left(\frac{m}{\tau_s}\right) \odot \mathrm{d}m \\ &= \frac{c}{\tau_s}\,\mathrm{sech}^2\left(\frac{m}{\tau_s}\right) \odot \mathrm{d}g. \end{aligned} \tag{30}$$

Thus $\widetilde{J}_{\mathrm{Sign}} = \partial\widetilde{\Phi}_{\mathrm{Sign}}/\partial g$ is diagonal. Pairing the differential with the validation gradient r moves the elementwise multiplier onto r:

$$\left\langle r, \mathrm{d}\widetilde{\Phi}_{\mathrm{Sign}} \right\rangle = \left\langle \frac{c}{\tau_s}\,\mathrm{sech}^2\left(\frac{m}{\tau_s}\right) \odot r, \mathrm{d}g \right\rangle.$$

Identifying the coefficient of $\mathrm{d}g$ yields

$$\widetilde{J}_{\mathrm{Sign}}^{\top} r = \frac{c}{\tau_s}\,\mathrm{sech}^2\left(\frac{m}{\tau_s}\right) \odot r. \tag{31}$$

For fixed τ_s and $c > 0$, this sensitivity is largest at $m = 0$ and decays as $|m|/\tau_s$ grows, emphasizing coordinates near a sign change. The expression is the JVG of the smooth surrogate, not of the hard-sign optimizer. Only scoring uses this surrogate; the model update remains eq. (29). For matrix parameters, the same calculation applies elementwise with G, M, and R in place of g, m, and r.

In our experiments, Signum (Bernstein et al., 2018) uses EMA momentum $m = \beta m_{\mathrm{in}} + (1-\beta)g$, giving $c = 1 - \beta$. There is no additional output scaling in the sign map (equivalently, $\gamma = 1$).

B Online Scoring and Implementation Details

We summarize the whole procedure of JADORE in Algorithm 3. In addition, we explain how the shared backward pass mentioned in Section 2.3 computes domain scores while preserving a training-only model update in this section.

Algorithm 3 Jacobian Domain Reweighting (JADORE)

Initialize: $\alpha_0 \leftarrow \mathbf{1}/K$, $\bar{s} \leftarrow 0$, $s_0 \leftarrow 0$
for $t = 0, \ldots, T-1$ **do**
1: Sample training data with domain probabilities α_t and a fresh reference minibatch
2: Run one joint forward-backward on both training and reference minibatches
3: $s_{t,d} \leftarrow \langle g_{t,d}, J_{t-1}^{\top} r_t \rangle$ for all d, if $t > 0$
4: $\theta_{t+1} \leftarrow \theta_t - \eta_t \Phi_t(g_t)$; $\bar{s} \leftarrow \bar{s} + s_t$; $\alpha_{t+1} \leftarrow \alpha_t$
5: **if** $(t+1) \bmod H = 0$ **then**
6: $\quad \alpha_{t+1} \leftarrow \beta_\alpha \alpha_t + (1-\beta_\alpha)\,\mathrm{softmax}\left(\frac{\lambda \bar{s}}{\mathrm{std}(\bar{s})+\epsilon}\right)$; $\bar{s} \leftarrow 0$

Layerwise scoring. We illustrate the streaming scoring process during backpropagation in Figure 6. Concatenate training examples sampled under α_t and fresh reference examples to share one forward-backward pass. At each selected layer ℓ, we reuse the input activations $a_{t,i}^{\ell}$ and output gradients $b_{t,i}^{\ell}$ at position i. These factors are used for domain scoring as shown in eq. (11). Importantly, these layer-local buffers can be released when finishing backward and scoring; there is no need to maintain all layers' intermediate vectors.

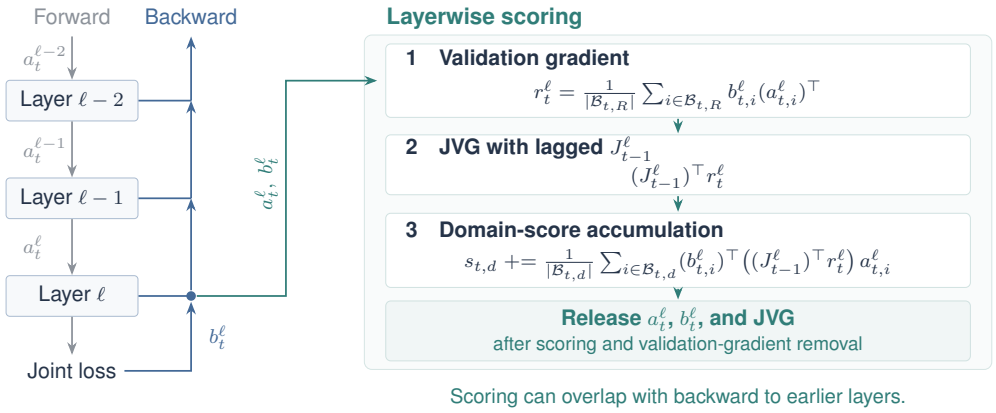

Figure 6: Layerwise scoring in a shared forward-backward pass. Domain scores are accumulated layer by layer using activations, backward signals, and the lagged JVG. Layer-local buffers are released after use.

Scoring computation. Figure 6 illustrates a general implementation that can be readily integrated into standard training pipelines. Standard backpropagation computes the gradients for model updates, while hooks capture layer activations a and output gradients b for a separate scoring computation. In this case, the formula $b_{t,i}^{\ell\top} \left(J_{t-1}^{\ell\top} r_t^{\ell} \right) a_{t,i}^{\ell}$ in eq. (11) for score computation avoids explicit domain gradient computations. Alternatively, a custom backward function can construct domain gradients g_d, evaluate the equivalent score $\langle g_d, J^{\top} r \rangle$ as shown in eq. (11), and reuse these gradients to accumulate the training gradient.

Practical considerations. Our experiments use a joint training-validation forward-backward pass without gradient accumulation, but the scoring rule also supports *separate passes*. With gradient accumulation, one can compute the validation gradient separately to avoid delay and form the lagged JVG once per optimizer step, then reuse it for ghost scoring while accumulating training gradients across microbatches. Overall, these options allow the same scoring rule to accommodate different training workflows, with the most efficient implementation depending on batch configuration, memory constraints, and hardware characteristics.

C Experimental details and additional experiments

C.1 Training setup

The model configurations, training budgets, and optimizer-specific learning rates for pretraining and SFT are summarized in Table 6.

C.1.1 Pretraining

Data and models. We adapt the Scion codebase (Pethick et al., 2025b) and follow its 124M and 1.07B model configurations (Pethick et al., 2025a), both using the GPT-2 tokenizer. We train both model sizes on SlimPajama (Soboleva et al., 2023) and the 124M model on The Pile (Gao et al., 2020). SlimPajama contains seven domains: ArXiv, Book, C4, Common Crawl, GitHub, StackExchange, and Wikipedia. For The Pile, we use the 17-domain `sail/regmix-data` version: Pile-CC, FreeLaw, USPTO, Wikipedia, PubMed Central, PubMed Abstracts, Gutenberg, StackExchange, PhilPapers, ArXiv, NIH, Enron, EuroParl, GitHub, HackerNews, DM, and Ubuntu, following Liu et al. (2025b); Xie et al. (2025).

Training configuration. SlimPajama uses approximately $1\times$ Chinchilla budgets for both model sizes, and The Pile uses approximately $2\times$. Adam and Signum update all trainable parameters. Muon and RMNP update hidden-layer matrices, while auxiliary Adam updates the tied input embedding and output head. The learning-rate schedule consists of optional linear warm-up, a constant-rate

Table 6: Model and training configurations. Batch sizes are global; reference batches serve online scoring. For pretraining, Muon/RMNP optimize hidden-layer matrices and Adam the tied embedding/head. SFT rates apply to LoRA parameters.

Configuration	Pretraining			SFT
	SlimPajama		The Pile	LLaMA-2-7B
	124M	1.07B	124M	LoRA
Model and training budget				
Transformer blocks	12	12	12	Pretrained LLaMA-2-7B
Hidden dimension	768	2560	768	
Attention heads	6	20	6	
Sequence length	1024	1024	1024	1024
Training domains	7	7	17	4
Training batch size	490	980	1,020	16
Reference batch size	22	22	22	4
Training steps	5100	21365	5100	2000
Training tokens	2.56B	21.44B	5.33B	33M
Learning rates and schedules				
Adam LR	1.8×10^{-3}			2×10^{-5}
Muon LR	3.6×10^{-4}			2×10^{-5}
RMNP LR	6.25×10^{-4}			1×10^{-4}
Signum LR	2×10^{-4}			2×10^{-5}
Auxiliary Adam LR	3.6×10^{-3}			–
LR warm-up	4.90% for Adam and Signum			0%
LR decay	Final 28.4%			Entire training

phase, and final linear decay; Table 6 reports the peak rates and schedule fractions. To ensure that every domain is represented during scoring, we include at least two samples from each domain in every training minibatch. Besides, we set $\mu = 0.95$ for Nesterov momentum and $\beta = 0.9$ for EMA momentum, which are common choices in Jordan et al. (2024a); Pethick et al. (2025a).

We use the learning rates of Adam and Muon reported by Pethick et al. (2025a). For RMNP and Signum, respectively, we sweep $\{3.125, 6.25, 12.5, 25, 75\} \times 10^{-4}$ and $\{1, 2, 5, 10\} \times 10^{-4}$ with uniform domain weights on SlimPajama 124M. We select 6.25×10^{-4} and 2×10^{-4} by validation loss. We reuse the selected rates for all pretraining experiments.

Reference set and evaluation. We construct the reference set using the Bench-Proxy procedure of Wang et al. (2026b). A frozen sentence embedding model Arctic-Embed-L v2 (Yu et al., 2024) ranks corpus documents by their maximum cosine similarity to benchmark queries. We retrieve whole documents from each corpus's validation split, targeting approximately 30M GPT-2 tokens. All pretraining methods requiring reference sets share this construction. Note that reference examples are used only for scoring, not for model updates.

We evaluate zero-shot on nine benchmarks: ARC-Challenge, ARC-Easy (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2020), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). We report length-normalized accuracy for ARC-Challenge, ARC-Easy, HellaSwag, OpenBookQA, and PIQA, and accuracy for the remaining tasks. Avg denotes the unweighted mean accuracy across nine benchmarks.

C.1.2 Supervised fine-tuning

Data and model. Following Wang et al. (2024), we fine-tune LLaMA-2-7B (Touvron et al., 2023) on the four instruction-data domains constructed by LESS (Xia et al., 2024): FLAN-v2, CoT, Dolly, and OpenAssistant. We use LoRA (Hu et al., 2022) with rank 128, $\alpha_{\text{LoRA}} = 1$, and zero dropout. The base model remains frozen, and each optimizer updates only the trainable LoRA parameters.

Table 7: 0-shot results for the 124M model trained on SlimPajama (1x Chinchilla). † denotes offline mixing baselines whose domain weights remain fixed during target-model training.

	Method	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Average
Adam	Uniform	22.33	34.50	19.68	27.40	**22.98**	25.13	59.00	35.62	50.83	$33.05_{\pm 0.27}$
	DoReMi†	22.50	35.61	19.57	27.99	**22.98**	24.67	59.63	35.77	51.33	$33.34_{\pm 0.14}$
	DoGE†	21.16	36.53	19.57	28.14	22.93	25.40	60.90	36.88	50.91	$33.60_{\pm 0.08}$
	RegMix†	21.81	**38.58**	19.57	28.39	22.93	25.60	58.22	36.46	50.57	$33.57_{\pm 0.28}$
	Chameleon†	21.96	35.80	19.71	27.85	22.96	25.40	60.43	35.91	51.01	$33.45_{\pm 0.24}$
	ADO	20.82	34.47	**19.87**	27.59	22.96	25.93	59.88	35.99	50.88	$33.15_{\pm 0.25}$
	ODM	**23.29**	33.87	19.74	27.24	22.88	25.47	58.76	35.72	50.59	$33.06_{\pm 0.13}$
	DGA	22.75	34.97	19.63	27.49	22.95	**26.53**	58.85	36.17	50.59	$33.33_{\pm 0.15}$
	Aioli	21.53	33.16	19.63	27.10	22.97	23.80	58.00	34.90	51.93	$32.56_{\pm 0.26}$
	JADORE	22.92	37.40	19.55	**28.81**	22.93	26.13	**62.21**	**36.98**	**51.99**	$\mathbf{34.32}_{\pm 0.19}$
Muon	Uniform	22.33	35.83	**19.98**	28.21	22.93	25.00	60.19	36.11	50.59	$33.46_{\pm 0.18}$
	DoReMi†	22.78	36.05	19.62	28.76	22.89	27.10	60.77	36.28	50.43	$33.85_{\pm 0.22}$
	DoGE†	22.10	37.16	19.57	29.61	22.96	26.07	61.61	36.47	50.36	$33.99_{\pm 0.17}$
	RegMix†	22.35	**39.18**	19.60	29.26	22.95	27.13	58.67	36.90	50.36	$34.04_{\pm 0.18}$
	Chameleon†	22.55	36.71	19.66	28.69	22.96	26.67	61.19	36.47	51.66	$34.06_{\pm 0.18}$
	ADO	**23.49**	36.20	19.57	28.21	22.92	26.40	60.83	36.61	51.09	$33.92_{\pm 0.18}$
	ODM	22.70	35.69	19.60	28.34	22.95	25.20	60.41	36.40	50.43	$33.52_{\pm 0.20}$
	DGA	22.44	36.32	19.49	28.48	22.96	26.67	60.63	36.88	50.54	$33.82_{\pm 0.21}$
	Aioli	21.73	35.59	19.87	28.28	**23.12**	25.53	60.57	**37.02**	**51.93**	$33.74_{\pm 0.32}$
	JADORE	22.90	38.33	19.57	**30.34**	22.91	**27.40**	**63.24**	36.93	50.86	$\mathbf{34.72}_{\pm 0.18}$
RMNP	Uniform	22.04	33.39	19.57	27.28	22.96	26.07	59.05	35.74	50.83	$32.99_{\pm 0.20}$
	DoReMi†	22.21	34.76	19.55	27.81	22.91	25.47	58.96	35.38	51.51	$33.17_{\pm 0.25}$
	DoGE†	21.30	36.13	19.57	27.78	22.94	25.53	59.74	35.63	51.38	$33.33_{\pm 0.24}$
	RegMix†	21.96	**37.25**	19.60	27.57	22.94	25.60	57.74	36.20	50.09	$33.22_{\pm 0.22}$
	Chameleon†	21.33	35.30	19.68	27.52	22.95	25.67	60.14	35.43	51.04	$33.23_{\pm 0.20}$
	ADO	21.50	34.10	19.63	27.18	**23.03**	25.27	59.67	35.84	50.75	$33.00_{\pm 0.21}$
	ODM	21.96	33.63	19.55	27.25	22.96	25.07	59.21	35.98	50.28	$32.88_{\pm 0.23}$
	DGA	21.76	35.69	**19.74**	27.53	22.93	26.20	60.04	36.03	50.99	$33.43_{\pm 0.22}$
	Aioli	22.33	32.84	19.63	27.27	22.96	24.80	57.53	34.65	51.51	$32.61_{\pm 0.17}$
	JADORE	**22.44**	36.39	19.60	**28.22**	22.92	**26.73**	**61.52**	**36.39**	**51.93**	$\mathbf{34.02}_{\pm 0.20}$
Signum	Uniform	21.84	33.36	19.49	27.37	22.93	25.40	58.34	35.41	**51.57**	$32.86_{\pm 0.19}$
	DoReMi†	**22.44**	34.79	19.57	27.60	22.91	24.67	58.81	35.81	50.86	$33.05_{\pm 0.23}$
	DoGE†	20.99	35.89	19.57	28.05	22.92	25.07	59.65	35.86	51.09	$33.23_{\pm 0.15}$
	RegMix†	22.24	36.53	19.57	27.99	22.93	25.13	57.56	36.27	51.20	$33.27_{\pm 0.18}$
	Chameleon†	22.13	35.02	19.55	27.48	**22.98**	**26.13**	59.14	35.62	51.30	$33.26_{\pm 0.12}$
	ADO	21.56	34.03	**19.63**	27.38	22.89	25.53	58.96	36.28	**51.57**	$33.09_{\pm 0.25}$
	ODM	22.18	33.26	**19.63**	26.90	22.94	25.13	58.09	35.84	51.07	$32.78_{\pm 0.22}$
	DGA	22.01	33.53	**19.63**	27.11	22.93	24.47	59.14	35.79	51.54	$32.91_{\pm 0.18}$
	Aioli	21.39	33.75	**19.63**	27.20	22.93	24.47	58.23	36.03	51.14	$32.75_{\pm 0.14}$
	JADORE	22.10	**37.29**	19.60	**28.30**	22.93	25.93	**61.14**	**36.39**	50.83	$\mathbf{33.83}_{\pm 0.15}$

Training configuration. We use a learning rate of 2×10^{-5} for Adam, Muon, and Signum while RMNP uses 10^{-4}, which we select by sweeping $\{1, 2, 5, 10, 20\} \times 10^{-5}$ with uniform domain weights. All SFT runs use linear learning-rate decay to zero without warm-up, with other training details given in Table 6.

Reference set and evaluation. Following Wang et al. (2024), we select nine MMLU subjects (Hendrycks et al., 2020): sociology, US foreign policy, abstract algebra, anatomy, astronomy, business ethics, clinical knowledge, college biology, and college chemistry. We pool their validation splits into a single reference set, sample four examples per step, and train one model jointly for all nine subjects. We evaluate it on test splits and report 5-shot accuracies. Avg denotes the unweighted mean accuracy across nine MMLU subjects.

C.1.3 Baseline configurations

We configure ADO (Jiang et al., 2025), ODM (Albalak et al., 2023), DGA (Fan et al., 2024a), and Aioli (Chen et al., 2025) based on their papers and implementations, with method-specific adjustments for our experimental settings and training budgets. We adopt the following configurations: **ADO**: fit scaling laws at 10% training, refit every 200 steps, minimum domain probability 0.01, smoothing coefficients 0.1, and sampling-history exponent 0.5. **DGA**: update domain weights every 100 steps (EMA coefficient 0.1) with learning rates of 0.1 (pretraining) and 0.01 (SFT). **Aioli**:

cumulative updates (LR 1.0) in cycles of 4% of total training; each cycle allocates about 20.6% to mixing law learning via two exploration sweeps (3 steps/domain) assigning 35.7% probability to the selected domain. **ODM**: reward EMA decay 0.9, temperature 1.0, 1% policy warm-up, with rewards aggregated by summing batch losses over seven independent draws before importance weighting.

C.2 Detailed experimental results

Table 7 provides the full results for Figure 1, adding DoReMi (Xie et al., 2023), DoGE (Fan et al., 2024b), RegMix (Liu et al., 2025b) and Chameleon (Xie et al., 2025) with fixed domain weights reported by Xie et al. (2025). JADORE achieves higher average accuracy than all baselines.

Figure 7 replots the trajectories in Figure 3 by domain, making differences across optimizers easier to compare, particularly for Book, C4, and StackExchange. Muon shows an obvious increase in the weights assigned to GitHub and StackExchange toward the end of training.

Table 8 provides the full Pile 124M results for Figure 4.

Table 9 reports three-seed means and standard deviations for the SlimPajama 1.07B and SFT comparisons in Sections 3.1 and 3.2.

Table 10 extends the scoring ablation in Section 3.3 to SFT, comparing JADORE with Raw for four optimizers, and OPUS-P for Adam and Muon.

Table 8: 0-shot results for the 124M model trained on The Pile (2× Chinchilla).

	Method	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	Uniform	21.90	36.17	19.63	27.90	22.95	26.20	57.36	35.16	51.01	$33.14_{\pm 0.36}$
	ADO	21.73	36.76	**19.71**	28.07	**23.09**	26.47	57.60	35.04	49.88	$33.15_{\pm 0.24}$
	ODM	**22.67**	35.05	19.66	27.41	23.01	**27.40**	57.36	35.50	50.80	$33.21_{\pm 0.21}$
	DGA	21.99	35.42	19.60	27.90	22.92	26.13	56.89	35.33	51.28	$33.05_{\pm 0.24}$
	Aioli	21.67	32.59	19.66	26.71	22.98	26.80	55.37	34.73	50.75	$32.36_{\pm 0.26}$
	JADORE	22.21	**37.61**	19.57	**28.44**	22.93	26.93	**59.34**	**35.65**	**51.99**	$\mathbf{33.85}_{\pm 0.21}$
Muon	Uniform	21.81	36.62	19.60	28.48	22.94	26.20	58.67	35.24	51.54	$33.46_{\pm 0.33}$
	ADO	22.18	37.53	19.57	28.42	22.96	27.27	58.80	35.76	51.12	$33.73_{\pm 0.24}$
	ODM	22.01	36.98	**19.63**	28.39	**22.97**	26.47	58.98	35.79	51.25	$33.61_{\pm 0.17}$
	DGA	22.33	37.07	19.55	28.56	22.92	27.13	58.72	35.53	**51.93**	$33.75_{\pm 0.16}$
	Aioli	**22.61**	36.56	19.60	28.00	22.93	26.20	58.94	35.55	51.88	$33.59_{\pm 0.19}$
	JADORE	22.55	**37.65**	**19.63**	**29.15**	22.93	**27.33**	**60.57**	**36.34**	51.33	$\mathbf{34.16}_{\pm 0.19}$
RMNP	Uniform	22.21	34.50	19.49	27.58	22.92	25.20	56.86	34.99	50.28	$32.67_{\pm 0.14}$
	ADO	21.36	34.82	**19.87**	27.66	**22.99**	**25.67**	57.91	35.53	50.96	$32.97_{\pm 0.18}$
	ODM	21.56	34.60	19.52	27.59	22.97	24.80	57.16	34.95	**51.09**	$32.69_{\pm 0.15}$
	DGA	21.93	**35.96**	19.52	**28.05**	22.94	25.20	**58.76**	35.24	50.62	$33.14_{\pm 0.13}$
	Aioli	20.93	33.95	19.63	26.98	22.95	25.00	56.09	34.83	50.43	$32.31_{\pm 0.17}$
	JADORE	**22.27**	**35.96**	19.57	27.71	22.91	25.07	58.60	**35.59**	50.80	$\mathbf{33.16}_{\pm 0.12}$
Signum	Uniform	**22.33**	35.61	19.55	27.52	22.97	25.80	56.93	34.92	50.43	$32.90_{\pm 0.18}$
	ADO	21.84	35.66	**19.66**	27.55	22.95	25.73	57.45	35.24	**51.22**	$33.03_{\pm 0.17}$
	ODM	21.70	34.18	19.57	27.20	**23.04**	**26.80**	56.93	35.14	50.22	$32.75_{\pm 0.22}$
	DGA	21.93	35.65	19.60	27.42	22.98	26.20	57.24	**35.52**	51.12	$33.07_{\pm 0.16}$
	Aioli	21.16	33.60	19.49	26.70	22.95	26.00	54.99	34.03	50.93	$32.21_{\pm 0.23}$
	JADORE	**22.33**	**36.31**	19.63	**27.78**	22.93	26.73	**59.16**	35.50	51.01	$\mathbf{33.49}_{\pm 0.21}$

Table 9: Aggregate accuracy (%) across three seeds for methods evaluated in both settings, corresponding to the main results in Tables 2 and 3. Entries report means with standard deviations as subscripts, computed across 3 seeds after averaging over the nine benchmarks or subjects.

	SlimPajama 1.07B (0-shot)				**LLaMA-2-7B SFT (5-shot)**			
Method	Adam	Muon	RMNP	Signum	Adam	Muon	RMNP	Signum
Uniform	$37.11_{\pm 0.28}$	$38.37_{\pm 0.21}$	$35.47_{\pm 0.18}$	$36.40_{\pm 0.22}$	$46.25_{\pm 0.23}$	$46.79_{\pm 0.24}$	$46.39_{\pm 0.55}$	$45.07_{\pm 0.30}$
ADO	$38.01_{\pm 0.33}$	$39.21_{\pm 0.20}$	$36.53_{\pm 0.32}$	$37.11_{\pm 0.12}$	$46.21_{\pm 0.41}$	$46.78_{\pm 0.18}$	$46.65_{\pm 0.57}$	$45.30_{\pm 0.37}$
ODM	$37.05_{\pm 0.21}$	$38.33_{\pm 0.36}$	$35.16_{\pm 0.17}$	$36.41_{\pm 0.26}$	$46.47_{\pm 0.30}$	$46.95_{\pm 0.24}$	$46.48_{\pm 0.57}$	$44.68_{\pm 0.42}$
DGA	$38.20_{\pm 0.20}$	$39.08_{\pm 0.19}$	$36.65_{\pm 0.12}$	$37.12_{\pm 0.26}$	$46.37_{\pm 0.29}$	$47.03_{\pm 0.29}$	$46.42_{\pm 0.56}$	$45.16_{\pm 0.52}$
Aioli	$36.60_{\pm 0.29}$	$37.99_{\pm 0.12}$	$35.73_{\pm 0.24}$	$36.29_{\pm 0.27}$	$46.66_{\pm 0.24}$	$46.77_{\pm 0.29}$	$46.69_{\pm 0.38}$	$44.50_{\pm 0.49}$
JADORE	$\mathbf{38.98}_{\pm 0.25}$	$\mathbf{39.93}_{\pm 0.30}$	$\mathbf{37.66}_{\pm 0.15}$	$\mathbf{37.70}_{\pm 0.25}$	$\mathbf{46.82}_{\pm 0.37}$	$\mathbf{47.24}_{\pm 0.33}$	$\mathbf{47.25}_{\pm 0.52}$	$\mathbf{46.36}_{\pm 0.45}$

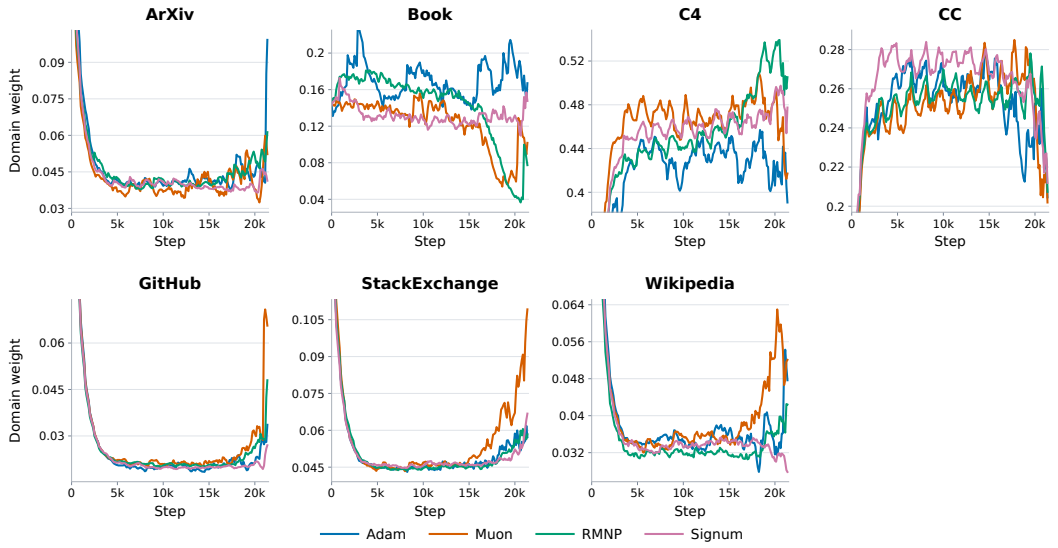

Figure 7: Domain-wise view of JADORE's weight trajectories for the 1.07B model on SlimPajama, replotted from Figure 3. Each panel compares four optimizers on one domain.

Table 10: Scoring-rule comparison of Raw, OPUS-P, and JADORE for LLaMA-2-7B SFT on MMLU. Raw uses raw-gradient alignment; OPUS-P uses the OPUS preconditioner within our framework.

	Method	Soc.	Pol.	Alg.	Anat.	Astr.	Eth.	Clin.	Bio.	Chem.	Avg.
Adam	Raw	**56.72**	**60.67**	35.00	48.15	42.32	49.00	**49.06**	43.98	35.00	46.66
	OPUS-P	56.22	59.33	34.00	48.40	**42.98**	48.67	**49.06**	**44.68**	36.00	46.59
	JADORE	**56.72**	57.67	**36.67**	**49.38**	41.67	**50.00**	47.92	**44.68**	**36.67**	**46.82**
Muon	Raw	61.19	55.00	34.00	**49.63**	39.91	**45.00**	47.80	44.91	**45.00**	46.94
	OPUS-P	61.86	**56.00**	34.67	49.14	**40.57**	**45.00**	**48.55**	45.14	43.67	47.18
	JADORE	**62.02**	55.67	**35.00**	48.89	39.69	**45.00**	47.55	**46.30**	**45.00**	**47.24**
RMNP	Raw	55.72	**61.67**	**39.00**	47.41	41.01	52.00	46.92	43.06	**36.67**	47.05
	JADORE	**58.21**	59.67	35.00	**49.14**	**42.11**	**54.00**	**47.92**	**44.21**	35.00	**47.25**
Signum	Raw	47.93	**63.33**	33.33	46.17	**41.23**	**52.33**	**49.18**	42.59	**36.00**	45.79
	JADORE	**54.39**	**63.33**	**36.33**	**47.65**	**41.23**	49.33	48.93	**43.06**	33.00	**46.36**

C.3 Computational overhead

We test SlimPajama 124M on four RTX 6000 GPUs using distributed data parallelism (DDP), with global training and reference batch sizes of 490 and 22 sequences, respectively. We use a custom backward function that explicitly computes layerwise domain gradients and reuses them for scoring and training-gradient accumulation. As discussed in Appendix B, implementation efficiency depends on the setup and hardware; this strategy achieved the lowest overhead among the implementations we tested. We report the maximum step time across devices after warm-up (ms/step) and peak allocated memory per device (GiB). Overheads are relative to Plain training without scoring.

Joint scoring vs. Sequential scoring. Existing data mixing works using gradient similarity compute validation and domain gradients sequentially, which requires multiple backward passes per domain weight computation (Fan et al., 2024b;a). Its cost grows with the number of domains. On seven-domain SlimPajama, this adds approximately 24% step time as shown in Table 11. Such overhead will increase with a larger number of domains. However, by reusing training intermediates for both score computation (eq. (11)) and model updates, we reduce this overhead to approximately 8% with a joint training-validation forward-backward pass.

Decomposition of overhead. Table 12 compares Joint Raw with JADORE, which additionally computes JVG. Based on the reported percentages, validation-gradient computation and raw-gradient alignment computation account for 87.1–99.6% of the total added step time and approximately 65% or more of the total added memory over Plain. JVG for introducing the optimizer state therefore contributes a smaller share of the overhead.

Further reducing scoring cost. Our main experiments score all hidden-layer linear matrices at every step. To further reduce the scoring cost, we consider two strategies: (1) scoring only once every four steps, and (2) using only the last transformer block for scoring. Table 13 shows their results. The former reduces average step time but retains full-scoring peak memory. Last-block scoring brings reported time and memory overheads below 3% and 1.5%, respectively, with average accuracy within 0.09 percentage points of full scoring across all four optimizers.

Table 11: Time of plain training, sequential-domain scoring, and Joint scoring on SlimPajama 124M.

	Plain	Sequential domains	Joint
Adam	439.93	544.50 (+23.77%)	472.13 (+7.32%)
Muon	468.67	583.84 (+24.57%)	506.21 (+8.01%)
RMNP	443.17	545.84 (+23.17%)	477.34 (+7.71%)
Signum	439.62	543.37 (+23.60%)	471.23 (+7.19%)

Table 12: Full-layer, every-step overhead on SlimPajama 124M. Step time is in ms/step and peak memory is in GiB/rank. Parenthetical percentages are overheads relative to Plain.

	Plain		Joint Raw		JADORE	
	Step (ms)	Peak (GiB)	Step (ms)	Peak (GiB)	Step (ms)	Peak (GiB)
Adam	439.93	57.93	471.21 (+7.11%)	60.51 (+4.46%)	472.13 (+7.32%)	60.67 (+4.73%)
Muon	468.67	57.61	501.38 (+6.98%)	60.19 (+4.47%)	506.21 (+8.01%)	61.58 (+6.89%)
RMNP	443.17	57.61	476.23 (+7.46%)	60.19 (+4.47%)	477.34 (+7.71%)	60.51 (+5.03%)
Signum	439.62	57.47	471.10 (+7.16%)	60.05 (+4.49%)	471.23 (+7.19%)	60.05 (+4.49%)

Table 13: Comparison of full-layer, per-4-step, and last-block scoring on SlimPajama 124M.

	Full	Per-4-step			Last-block		
	Acc.	Acc.	Step (ms)	Peak (GiB)	Acc.	Step (ms)	Peak (GiB)
Adam	34.32	34.06	452.69 (+2.90%)	60.67 (+4.73%)	34.23	452.60 (+2.88%)	58.61 (+1.17%)
Muon	34.72	34.68	481.65 (+2.77%)	61.58 (+6.89%)	34.75	481.98 (+2.84%)	58.41 (+1.38%)
RMNP	34.02	33.85	455.18 (+2.71%)	60.51 (+5.03%)	33.94	456.42 (+2.99%)	58.30 (+1.19%)
Signum	33.83	33.62	451.49 (+2.70%)	60.05 (+4.49%)	33.77	452.50 (+2.93%)	58.13 (+1.15%)

C.4 Evaluation on out-of-distribution benchmarks

Since our reference set is constructed by selecting samples most similar to the target benchmarks, we further assess the generalization of JADORE on out-of-distribution (OOD) benchmarks. We evaluate SlimPajama 124M checkpoints zero-shot on six out-of-distribution benchmarks: BBH (Suzgun et al., 2023), RACE-Middle, RACE-High (Lai et al., 2017), AX-b, AX-g (Wang et al., 2019), and the English XStoryCloze variant of StoryCloze (Mostafazadeh et al., 2016). We evaluate these benchmarks using OpenCompass-based configurations and report 0-shot accuracy. Table 14 reports per-benchmark results for all methods, with unweighted averages across the six benchmarks. JADORE achieves the highest average OOD accuracy across all four optimizers, suggesting that it can improve generalization to benchmarks not used to construct the reference set.

Table 14: 0-shot OOD accuracy (%) for the 124M model trained on SlimPajama.

	Method	BBH	RACE-M	RACE-H	AX-b	AX-g	StoryCloze	Avg.
Adam	Uniform	4.78	**22.01**	21.44	47.55	50.28	54.25	33.39
	ADO	5.39	21.80	**21.73**	45.02	**50.56**	54.36	33.14
	ODM	**14.30**	21.80	21.50	43.39	49.72	52.54	33.87
	DGA	6.57	**22.01**	21.70	46.38	48.31	53.39	33.06
	Aioli	8.23	21.66	21.44	46.01	50.28	54.09	33.62
	JADORE	10.42	21.87	21.44	**48.22**	49.72	**54.57**	**34.56**
Muon	Uniform	3.66	21.66	21.27	43.93	48.88	54.62	32.34
	ADO	7.77	**22.21**	21.36	44.75	49.72	53.87	33.28
	ODM	11.19	21.80	21.33	43.03	50.00	53.29	33.44
	DGA	11.58	21.87	21.36	42.75	50.00	54.04	33.60
	Aioli	7.16	21.59	**21.47**	41.94	49.44	53.87	32.58
	JADORE	**12.23**	21.66	**21.47**	**46.29**	**50.84**	**55.69**	**34.70**
RMNP	Uniform	13.34	21.66	21.50	41.21	**50.56**	54.36	33.77
	ADO	11.30	21.94	**21.58**	42.21	50.00	54.04	33.51
	ODM	**15.67**	21.66	21.47	41.67	50.00	53.71	34.03
	DGA	14.29	**22.28**	21.53	41.94	50.00	54.04	34.01
	Aioli	12.49	21.80	21.53	**42.66**	**50.56**	53.13	33.69
	JADORE	14.15	21.73	21.56	42.48	50.00	**54.78**	**34.12**
Signum	Uniform	10.98	21.66	21.64	43.39	49.16	54.30	33.52
	ADO	10.11	**22.21**	21.61	44.75	50.56	54.94	34.03
	ODM	9.27	21.80	21.53	41.58	50.00	53.55	32.95
	DGA	12.68	22.14	**21.67**	42.30	**51.69**	**55.53**	34.34
	Aioli	11.84	21.87	21.50	42.03	50.00	53.61	33.47
	JADORE	**13.27**	21.52	21.47	**46.74**	49.72	55.21	**34.66**

C.5 Effect of a one-step-lagged Jacobian

We test the effect of Jacobian lag by comparing $J_t^\top r_t$ with the one-step-lagged direction $J_{t-1}^\top r_t$ used by JADORE. Computing $J_t^\top r_t$ requires updating the optimizer state before scoring each layer; this state can be reused for the model update, but waiting for it delays layerwise scoring. The results reported in Table 15 show that the one-step-lagged Jacobian provides similar performance in terms of average accuracy. However, the time overhead of $J_t^\top r_t$ is 15.23%, 16.43%, 13.88%, and 14.77% for Adam, Muon, RMNP, and Signum, respectively. They are approximately twice the overhead reported for $J_{t-1}^\top r_t$ in Table 12. We therefore still recommend the one-step-lagged Jacobian for its lower computational overhead and comparable average accuracy.

Table 15: Current-step versus one-step-lagged Jacobian scoring for SlimPajama 124M.

		ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	$J_t^\top r_t$	22.54	37.07	19.55	28.66	22.93	26.67	62.33	36.23	51.93	34.21
	$J_{t-1}^\top r_t$	22.92	37.40	19.55	28.81	22.93	26.13	62.21	36.98	51.99	34.32
Muon	$J_t^\top r_t$	23.09	37.84	19.57	30.77	22.97	28.00	62.86	36.85	51.30	34.81
	$J_{t-1}^\top r_t$	22.90	38.33	19.57	30.34	22.91	27.40	63.24	36.93	50.86	34.72
RMNP	$J_t^\top r_t$	22.53	37.09	19.57	28.57	22.90	25.80	60.72	36.61	51.85	33.96
	$J_{t-1}^\top r_t$	22.44	36.39	19.60	28.22	22.92	26.73	61.52	36.39	51.93	34.02
Signum	$J_t^\top r_t$	21.93	37.24	19.63	27.95	22.93	25.33	61.46	36.66	51.83	33.88
	$J_{t-1}^\top r_t$	22.10	37.29	19.60	28.30	22.93	25.93	61.14	36.39	50.83	33.83

C.6 PRIOR-ANCHOR WEIGHTS

When reliable domain weights are available from an offline mixing method, JADORE can refine them online. We replace the uniform anchor u in eq. (6) with a fixed prior $\pi \in \Delta^K$, where $\pi_d > 0$, yielding

$$\hat{\alpha} = \arg\max_{\alpha \in \Delta^K} \{\langle \eta s, \alpha \rangle - \tau D_{\mathrm{KL}}(\alpha \| \pi)\}, \quad \hat{\alpha}_d = \frac{\pi_d \exp(\eta s_d/\tau)}{\sum_{j=1}^K \pi_j \exp(\eta s_j/\tau)}. \tag{32}$$

We use the same temperature $\tau = \eta \,\mathrm{std}(s)/\lambda$ as in Section 2.1, initialize $\alpha_0 = \pi$, and apply the primal-averaging update in eq. (6). The anchor π remains fixed throughout training; setting $\pi = u$ recovers the main method. Table 16 evaluates DoReMi and DoGE weights as priors on SlimPajama 124M. For both priors and all optimizers, JADORE improves average downstream accuracy over the corresponding fixed mixture, showing that online adaptation can further improve offline mixtures.

Table 16: JADORE with prior-anchor weights.

	Method	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	DoReMi	22.50	35.61	19.57	27.99	22.98	24.67	59.63	35.77	51.33	33.34
	DoReMi + JADORE	22.61	37.86	19.57	29.39	22.91	27.60	61.77	36.32	51.59	34.40
	DoGE	21.16	36.53	19.57	28.14	22.93	25.40	60.90	36.88	50.91	33.60
	DoGE + JADORE	22.58	37.42	19.60	29.31	22.95	27.27	61.82	36.32	50.83	34.23
Muon	DoReMi	22.78	36.05	19.62	28.76	22.89	27.10	60.77	36.28	50.43	33.85
	DoReMi + JADORE	22.33	38.54	19.57	30.37	22.94	27.07	62.95	36.54	51.62	34.66
	DoGE	22.10	37.16	19.57	29.61	22.96	26.07	61.61	36.47	50.36	33.99
	DoGE + JADORE	22.47	38.26	19.57	30.54	22.93	27.60	63.15	37.04	50.09	34.62
RMNP	DoReMi	22.21	34.76	19.55	27.81	22.91	25.47	58.96	35.38	51.51	33.17
	DoReMi + JADORE	21.87	36.78	19.57	28.16	22.92	26.43	61.88	36.17	50.75	33.84
	DoGE	21.30	36.13	19.57	27.78	22.94	25.53	59.74	35.63	51.38	33.33
	DoGE + JADORE	21.76	36.90	19.63	28.27	22.92	26.20	60.86	36.54	52.17	33.92
Signum	DoReMi	22.44	34.79	19.57	27.60	22.91	24.67	58.81	35.81	50.86	33.05
	DoReMi + JADORE	22.30	36.99	19.60	28.78	22.95	26.53	61.46	36.75	50.28	33.96
	DoGE	20.99	35.89	19.57	28.05	22.92	25.07	59.65	35.86	51.09	33.23
	DoGE + JADORE	22.50	36.46	19.60	28.83	22.92	26.87	61.52	36.64	51.49	34.09

C.7 HYPERPARAMETER SENSITIVITY

Tables 17–19 examine sensitivity to the accumulation window size H, softmax scale λ, and weight-EMA coefficient β_α of JADORE on SlimPajama 124M, varying one parameter at a time. Average accuracy remains relatively stable over the tested ranges, and every tested configuration exceeds the strongest baseline for the corresponding optimizer in Table 7. Light-blue rows mark the settings used in our main experiments: $H = 100$, $\lambda = 2$, and $\beta_\alpha = 0.9$.

In line with the idea of calibrating smoothing to momentum magnitudes (Feoktistov et al., 2026), we set $\tau_s = 2 \times 10^{-6}$ for pretraining and $\tau_s = 1 \times 10^{-3}$ for SFT in Sign-JVG, matching the order of magnitude of the observed transformer-block momentum RMS.

Table 17: Sensitivity of the window size H for scoring accumulation on SlimPajama 124M.

	H	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	50	21.84	37.54	19.57	29.21	22.92	26.40	62.02	37.26	51.14	34.21
	100	22.92	37.40	19.55	28.81	22.93	26.13	62.21	36.98	51.99	34.32
	150	22.10	38.05	19.57	29.20	22.90	27.20	61.53	37.10	50.12	34.20
Muon	50	23.89	38.55	19.57	30.48	22.95	30.20	63.17	36.80	50.59	35.13
	100	22.90	38.33	19.57	30.34	22.91	27.40	63.24	36.93	50.86	34.72
	150	24.15	38.51	19.49	30.43	23.02	27.80	63.00	36.75	50.36	34.83
RMNP	50	21.76	36.49	19.57	28.31	22.95	26.80	60.88	36.23	51.67	33.85
	100	22.44	36.39	19.60	28.22	22.92	26.73	61.52	36.39	51.93	34.02
	150	21.76	36.48	19.57	28.28	22.91	26.60	61.35	36.38	52.01	33.92
Signum	50	21.59	37.16	19.66	28.47	22.95	26.40	60.94	35.41	52.72	33.92
	100	22.10	37.29	19.60	28.30	22.93	25.93	61.14	36.39	50.83	33.83
	150	22.27	35.77	19.57	28.47	22.94	27.40	61.04	36.69	52.33	34.05

Table 18: Sensitivity of the hyperparameter λ in the softmax on SlimPajama 124M.

	λ	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	1.0	22.47	37.21	19.60	28.44	22.92	26.80	61.59	36.39	51.78	34.13
	1.5	23.04	36.81	19.52	28.64	22.96	26.53	61.72	36.71	52.25	34.24
	2.0	22.92	37.40	19.55	28.81	22.93	26.13	62.21	36.98	51.99	34.32
Muon	1.0	22.87	38.06	19.57	29.81	22.93	26.07	62.89	36.80	50.96	34.44
	1.5	22.38	38.38	19.60	30.15	22.99	27.73	62.89	36.49	51.46	34.68
	2.0	22.90	38.33	19.57	30.34	22.91	27.40	63.24	36.93	50.86	34.72
RMNP	1.0	22.35	36.10	19.57	28.01	22.96	25.87	60.55	36.10	51.93	33.72
	1.5	21.99	36.49	19.60	27.93	22.91	26.20	60.68	36.30	51.54	33.74
	2.0	22.44	36.39	19.60	28.22	22.92	26.73	61.52	36.39	51.93	34.02
Signum	1.0	22.13	35.86	19.57	28.00	22.93	25.93	60.12	35.69	50.86	33.45
	1.5	22.13	37.07	19.60	28.24	22.88	25.27	61.17	36.13	51.14	33.74
	2.0	22.10	37.29	19.60	28.30	22.93	25.93	61.14	36.39	50.83	33.83

Table 19: Sensitivity of the weight-EMA coefficient β_α on SlimPajama 124M.

	β_α	ARC-C	ARC-E	C.QA	H-Swag	MMLU	OBQA	PIQA	SIQA	W.G.	Avg.
Adam	0.85	22.60	37.43	19.68	28.84	22.94	26.07	61.98	36.68	51.67	34.21
	0.90	22.92	37.40	19.55	28.81	22.93	26.13	62.21	36.98	51.99	34.32
	0.95	22.84	37.02	19.68	28.80	22.86	26.03	61.98	36.89	51.57	34.19
Muon	0.85	23.32	38.43	19.57	30.60	22.94	27.00	62.97	36.78	51.25	34.76
	0.90	22.90	38.33	19.57	30.34	22.91	27.40	63.24	36.93	50.86	34.72
	0.95	22.53	37.89	19.57	30.10	22.94	26.93	62.77	36.98	51.07	34.53
RMNP	0.85	22.50	37.01	19.66	28.00	22.90	26.07	61.37	36.49	51.33	33.92
	0.90	22.44	36.39	19.60	28.22	22.92	26.73	61.52	36.39	51.93	34.02
	0.95	22.49	36.31	19.57	28.06	22.90	26.60	61.01	36.46	51.30	33.86
Signum	0.85	22.35	35.72	19.63	27.86	22.93	25.87	61.06	36.98	51.09	33.72
	0.90	22.10	37.29	19.60	28.30	22.93	25.93	61.14	36.39	50.83	33.83
	0.95	21.81	36.69	19.79	27.61	22.89	25.40	61.12	36.42	51.38	33.68