Factored One-sided Adam-Moment for Practical and Scalable SOAP

Jaemyung Yu* **Byeongho Heo** **Sangdoo Yun** **Dongyoon Han***

NAVER AI Lab

Abstract

Second-order optimizers (*e.g.,* Shampoo, SOAP) have emerged as compelling alternatives to Adam/AdamW and promise faster per-step convergence than AdamW in transformer training. However, they still remain costly in practice, and even SOAP still pays substantial optimizer-state memory and eigendecomposition overhead. We ask whether the benefits of SOAP can be retained at nearly the efficiency of AdamW. In this paper, we answer yes with a lightweight SOAP variant, Factored One-sided Adam-Moment (FOAM). The path to practical impact is twofold: first, FOAM employs input-only preconditioning, dropping the output-side Kronecker factor to reduce memory and remove one eigendecomposition. The Kronecker structure of the Hessian shows that this is exact for a diagonal output-side curvature and nearly lossless for a near-diagonal one, as we observe empirically. Second, FOAM replaces SOAP's dense rotated second moment with a factored Adam-moment estimator based on row-by-column statistics. Under the K-FAC Fisher assumption with a diagonal output curvature, this estimator exactly recovers SOAP's dense second moment, and we measure only a 15–20% K-FAC violation, with negligible effect on the update. Our empirical results support the claim that FOAM matches strong SOAP and Muon baselines in validation loss across scales of large language model (LLM) pre-training, uses half of SOAP's optimizer state, runs at Muon-class wall-clock time, and retains robustness across a three-fold learning-rate range. The results hold from 124M to 760M parameters, demonstrating that SOAP's gains need not require SOAP's cost. Code: https://github.com/naver-ai/foam.

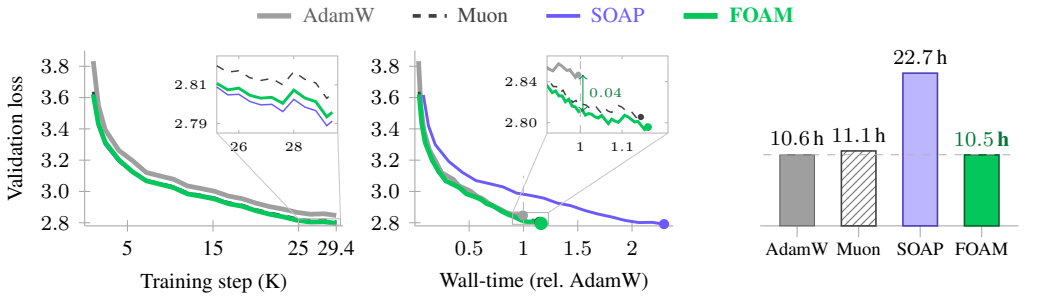
1 Introduction

Adam [23] and AdamW [32] have long been the default for pre-training transformer architectures, including large language models (LLMs). Their successors, including memory-efficient adaptive optimizers [50, 51, 53, 15], show that comparable optimization quality can be achieved with substantially less optimizer memory. In another direction, Shampoo [16], a foundational method in the family of Kronecker-factored second-order optimizers, has shown that the optimization quality of adaptive first-order methods such as Adam can be surpassed. Its most practical variant, SOAP [44], the Adam-in-eigenbasis version of Shampoo, reports 35–40% improvements in wall-clock time and step count over AdamW at large batch sizes, building upon the Distributed Shampoo baselines [3, 40] that won the external-tuning track of the first AlgoPerf competition [6, 21].

Still, practitioners often default to AdamW or Muon [20], because SOAP achieves its gains at the expense of computational efficiency, requiring about $4.5\times$ the optimizer state of AdamW and a significantly slower per-step wall-clock time. More specifically, SOAP's memory is dominated by the first moment, the second moment in the eigenbasis-rotated space, and the two Kronecker factor matrices $L = GG^\top$ (m^2) and $R = G^\top G$ (n^2) of a gradient $G \in \mathbb{R}^{m\times n}$. Its runtime is

*Correspondence to {jaemyung.yu, dongyoon.han}@navercorp.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Val. loss vs. step (b) Val. loss vs. wall-time (c) Time to AdamW's final loss

Figure 1: **FOAM is a Pareto-efficient second-order optimizer.** GPT-2 large (760M), 29.55K steps, each optimizer at its tuned learning rate. Curves are 3-seed means smoothed over 1K steps, and grey boxes mark the regions magnified in the insets. **(a) Per-step.** Validation loss against training step. In the last 4K steps (inset) SOAP ends slightly below FOAM (2.791 vs. 2.796, within about one seed std), both below Muon, while AdamW stays above the inset range. **(b) Wall-clock.** Same runs vs. wall time, normalized so AdamW finishes at $1.0\times$ (Muon, FOAM, and SOAP finish at $1.15\times$, $1.17\times$, and $2.31\times$). Within AdamW's own budget, FOAM is already 0.04 nats below AdamW's final loss (inset), and it reaches SOAP's quality band at 51% of SOAP's wall-clock time. **(c) Time to AdamW's final loss** (2.847; Table 2). FOAM gets there as fast as AdamW itself (10.53 vs. 10.55 h) and slightly ahead of Muon, while bilateral SOAP needs $2.15\times$ as long.

further occupied by the periodic eigendecomposition of these factor matrices, which makes SOAP substantially more expensive than AdamW despite its strong optimization performance.

This paper aims to retain SOAP-level quality while approaching AdamW-level cost. Our motivation is simple. If SOAP's computational overhead can be sufficiently reduced, its optimization advantages could make it a practical successor to AdamW. To this end, we revisit SOAP's major cost components and show that several can be relaxed or replaced without sacrificing performance. The original SOAP paper [44] already evaluated one-sided and factored variants, but as cost-reducing ablations without a criterion for when they preserve quality. Frans et al. [12] then found that input-only rotation and a factored variance each retain most of SOAP's quality, and other works [30, 11] improved SOAP along other axes. Our contribution is thus not one-sided preconditioning or factorization itself, but explaining *which side to retain*, *when factorization is justified*, and *why their combination is effective*.

Concretely, our approach is built on two structural simplifications: input-only preconditioning and a factored Adam-Moment update. The first simplification drops the output-side factor L and retains only the input-side factor R, eliminating its m^2 memory and one of the two eigendecompositions. Its justification follows from the Kronecker structure of the Hessian, in which all output rows of an affine layer share the same input-side outer product $xx^\top$ [25], while the Hessian of transformer layers is empirically near-block-diagonal along output neurons or heads [50], making the input-only form the natural one-sided approximation. The second simplification replaces the dense rotated second moment with a row-and-column factored estimator, $r \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, with trace normalization, reducing the buffer size from mn to $m+n$. Under the K-FAC Fisher assumption [35] with a diagonal output curvature, this estimator exactly recovers SOAP's dense second moment, and the K-FAC part holds to a 15–20% relative error in practice. Together, these simplifications reduce SOAP's optimizer state from $2mn + 2m^2 + 2n^2$ to $mn + 2n^2 + m + n$ and the amortized eigendecomposition count from two to one. We call the resulting method *FOAM* (**F**actored **O**ne-sided **A**dam-**M**oment).

Figure 1 highlights FOAM's main advantages. On pre-training, FOAM matches the strong Muon and SOAP baselines on validation loss with half of SOAP's optimizer state and reaches AdamW's final loss within 7% of AdamW's own wall-clock time, while SOAP needs more than twice as long at 760M, and FOAM matches AdamW's hyperparameter robustness over a three-fold learning-rate range. The advantage holds from `124M` to `760M` parameters with learning rates re-tuned at every size from a μP [46] anchor. In summary, our contributions are: (i) a structural justification of input-only preconditioning, with the Kronecker structure of the Hessian and empirical verifications with LLMs with recent architectural elements; (ii) a factored Adam-Moment update with trace normalization, exact under K-FAC with a diagonal output curvature and empirically validated through K-FAC-residual measurements during training; and (iii) the resulting algorithm FOAM, a scalable and practical SOAP variant that delivers SOAP-class quality with half of SOAP's optimizer state (+7.4% measured peak training memory over AdamW) and Muon-class wall-clock time.

2 Related Work

Adaptive optimizers can be categorized by how aggressively they approximate the empirical Fisher $F = \mathbb{E}[\mathrm{vec}(G)\,\mathrm{vec}(G)^\top]$, which all of Adam [23]/AdamW [32], Shampoo [16], and SOAP [44] implicitly use as a curvature surrogate in place of the true Fisher [24].

2.1 Empirical Fisher and Kronecker-factored Optimizers

AdaGrad [8] introduced per-coordinate scaling by an empirical second-moment statistic, an early diagonal approximation of the inverse empirical Fisher. K-FAC [35] replaces the diagonal with a Kronecker product, $F \approx \mathbb{E}[aa^\top] \otimes \mathbb{E}[ss^\top]$ where a is the layer input and s the gradient with respect to the layer's pre-activation output, the assumption under which our factored Adam-Moment update is exact. Eschenhagen et al. [10] adapt K-FAC to transformer architectures via the *expand* and *reduce* flavors. Shampoo [16] carries this idea to the matrix parameters. It maintains EMAs of $L = GG^\top$ and $R = G^\top G$ and steps in $L^{-1/4}GR^{-1/4}$, i.e., it preconditions by $(R \otimes L)^{-1/4}$; Morwani et al. [36] show that $R \otimes L$ equals one power-iteration step toward the optimal Kronecker-product approximation of the empirical Fisher and is empirically close to it. SOAP [44] eigendecomposes L and R and runs Adam in the resulting rotated coordinates, an EKFAC-style construction [14] that Lu et al. [33] interpret as gradient whitening. Variants modify specific pieces: CASPR [9] combines the per-axis factors through a Kronecker sum instead of a Kronecker product. KL-Shampoo and KL-SOAP [30] recast the family via KL-divergence minimization, and Purifying Shampoo [11] shows that trace scaling of the preconditioner can replace learning-rate grafting, which corroborates our trace-norm choice in §4.2. The one-sided fallback in distributed Shampoo [3] drops a factor on memory grounds, whereas we give a structural argument for keeping the input side (§4.1). Purifying Shampoo's adaptive tolerance criterion for skipping the periodic eigendecomposition is adopted unchanged as FOAM's amortized eigenbasis-refresh policy (§4.2).

2.2 Optimizers with Coarser Approximations

A second line of work adopts coarser approximations to the empirical Fisher in exchange for lower computational cost. Adam [23, 32] keeps only the diagonal of F, sacrificing all matrix structure. Signum [4] keeps only the sign of the momentum. Muon [20] replaces the second-moment buffer with Newton-Schulz orthogonalization $M(M^\top M)^{-1/2}$, which can be read as a square-root natural-gradient step under a structured Fisher $\tilde{R} \otimes I_m$ with an input-side factor $\tilde{R} \in \mathbb{R}^{n\times n}$ shared across output rows [34, 15]. AdaMuon [41] adds an element-wise Adam-style second moment on top of the orthogonalized update. SPlus [13], a related SOAP variant, replaces the rotated second moment with sign normalization in the Shampoo eigenbasis and adds iterate averaging. Recent Muon variants (NorMuon [28], Muon^2 [31], NAMO [48], Polar Express [2]) refine specific components of Muon. A parallel branch targets memory-efficient AdamW rather than matching SOAP-class quality: Adam-mini [50], GaLore [51], APOLLO [53], Alice [15], and SWAN [34]. We exclude these methods from the main comparison since they mainly target a smaller memory footprint than AdamW, whereas we target SOAP-level performance at near-Adam cost.

Positioning our method. FOAM lies at the intersection of these two branches. It stays within the Kronecker-factored Fisher framework but borrows the factored-second-moment idea from Adafactor [39]'s row-by-column estimator, applied here in the rotated coordinates of SOAP. Both ingredients appear in the SOAP paper as cost-reducing ablations, a one-sided variant and an Adafactor-style factored second moment in the eigenbasis, and Frans et al. [12] found empirically that input-only rotation and a factored variance each preserve most of SOAP's quality; the distinction lies in the criteria. SOAP's one-sided variant keeps the side with the smaller dimension, whereas we keep the input side irrespective of size (§4.1), so the two rules choose opposite sides for the MLP down-projection; likewise, we factorize only after the input-side rotation, which makes the rotated second moment separable into row and column factors (Lemma 1, §4.2).

3 Preliminary: Shampoo and SOAP

We consider a transformer with matrix parameters $W \in \mathbb{R}^{m\times n}$, where n and m denote the input and output dimensions, respectively. Let $G_t \in \mathbb{R}^{m\times n}$ denote the mini-batch gradient at step t. Following

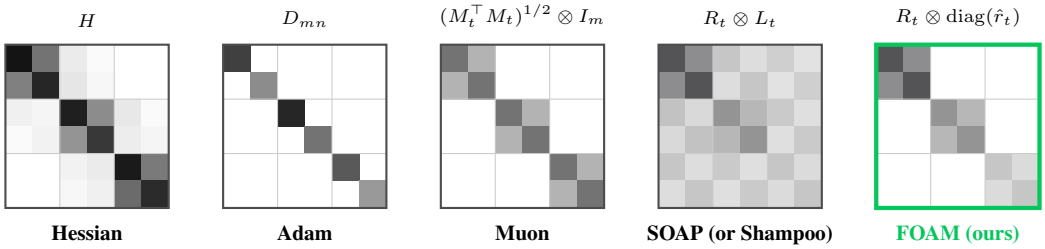


Figure 2: **Hessian and its approximation across optimizers.** For $W \in \mathbb{R}^{m\times n}$ with $m{=}3, n{=}2$, we visualize the $mn \times mn$ Hessian alongside the preconditioner of each optimizer; shaded cells are non-zero and darker means larger; coordinates are grouped by output row ($n \times n$ blocks), while labels follow the column-major convention of Lemma 1. The Hessian is near-block-diagonal. **Adam** (D_{mn}): fully diagonal approximation with no Kronecker structure. **Muon** ($(M_t^\top M_t)^{1/2} \otimes I_m$): m identical outer-diagonal blocks, built statelessly from the current momentum via $M_t(M_t^\top M_t)^{-1/2}$. **SOAP** ($R_t \otimes L_t$): full Kronecker fit with EMAs on both sides, with all m^2 outer blocks populated. **FOAM** ($R_t \otimes \mathrm{diag}(\hat{r}_t)$, ours): replaces Muon's identity output with a per-block diagonal scaling, and its stateless input factor with the EMA of $G_t^\top G_t$.

the convention of SOAP [44], we define the two Kronecker factors L_t and R_t as the second-moment matrices of the gradient:

$$L_t \;=\; \mathbb{E}[G_t G_t^\top] \;\in\; \mathbb{R}^{m\times m}, \qquad R_t \;=\; \mathbb{E}[G_t^\top G_t] \;\in\; \mathbb{R}^{n\times n}, \tag{1}$$

L_t acts on the left side (**output**) of G_t, and R_t acts on the right side (**input**). In practice, both are estimated by exponential moving averages (EMAs), for which we reuse the same symbols, and we reserve the term *estimator* for such EMA buffers. Shampoo [16] takes a fractional inverse of each factor and applies the preconditioned update:

$$W_{t+1} \;=\; W_t \;-\; \eta\, L_t^{-1/4}\, G_t\, R_t^{-1/4}, \tag{2}$$

with learning rate η. SOAP [44] instead eigendecomposes L_t and R_t to obtain orthogonal rotations $Q_L \in \mathbb{R}^{m\times m}$ and $Q_R \in \mathbb{R}^{n\times n}$, and runs Adam in the rotated coordinates:

$$G_{\mathrm{rot}} \;=\; Q_L^\top\, G\, Q_R \;\in\; \mathbb{R}^{m\times n}, \tag{3}$$

following the EKFAC tradition [14] that tracks diagonal second moments in the eigenbasis of a Kronecker-factored preconditioner. The per-step update maintains a dense second moment $v_{\mathrm{rot}} \in \mathbb{R}^{m\times n}$ in the rotated space. The rotations Q_L and Q_R are refreshed every `pf` (precondition frequency) steps via two eigendecompositions of L_t and R_t. L_t and R_t require $m^2 + n^2$ memory, the rotated second moment v_{rot} requires an additional mn memory, and the periodic eigendecompositions incur further computational overhead. These costs constitute the overhead structure that FOAM addresses.

4 Factored One-sided Adam-Moment (FOAM)

We propose **FOAM**, **F**actored **O**ne-sided **A**dam-**M**oment, which combines two structural simplifications of SOAP into a single optimizer. FOAM applies an input-side K-FAC factor combined with a factored output diagonal. It maintains only the input-side factor R_t, runs Adam in the eigenbasis $Q_R = \mathrm{eigh}(R_t)$, and replaces SOAP's dense rotated second moment $v_{\mathrm{rot}} \in \mathbb{R}^{m\times n}$ with a row-column factored estimator $\hat{v}_{\mathrm{rot}} = \tau\, \hat{r}\, \hat{c}^\top$. Here $\hat{r} \in \mathbb{R}^m$ and $\hat{c} \in \mathbb{R}^n$ are bias-corrected EMAs of the row and column means of the squared rotated gradient, and τ is a trace-matching scalar. The two simplifications together give FOAM the implicit Fisher $R_t \otimes \mathrm{diag}(\hat{r}_t)$, a full input-side factor with a diagonal output side, derived in Algorithm 1 and the following sections. Figure 2 compares this preconditioner structure with those of Adam, Muon, and SOAP against the near-block-diagonal Hessian of a transformer layer, where FOAM keeps SOAP's input factor with a diagonal output side.

Derivation overview. The steps below form a single chain from $\mathbb{E}[gg^\top]$, $g = \mathrm{vec}(G)$, to the two stored vectors. (i) Under K-FAC, $\mathbb{E}[gg^\top] = R \otimes L$, and L is near-diagonal (§4.1), so the off-diagonal structure lives on the input side, which is the side to keep. (ii) Rotating by Q_R turns the covariance into $\Lambda_R \otimes L$ (Lemma 1). Like SOAP running Adam in its eigenbasis, FOAM keeps only the diagonal of this covariance, a modeling choice that the rotation makes accurate. (iii) This diagonal, reshaped to $m \times n$, is the rank-one matrix $\mathrm{diag}(L)\, \lambda_R^\top$, so storing $\hat{r}$ and $\hat{c}$ keeps every coordinate's own variance and loses nothing relative to the dense mn buffer, as formalized in §4.2.

Algorithm 1 FOAM update at step t for $W \in \mathbb{R}^{m\times n}$. $\mathbf{1}_d$ is the all-ones vector in $\mathbb{R}^d$, and $G^{\circ 2}$, $\oslash$, and $\sqrt{\cdot}$ act element-wise. $\mathrm{off}(\cdot)$ is the off-diagonal ratio of a matrix (§4.2). Q_R is initialized to the identity, and decoupled weight decay is applied as in AdamW and omitted here.

Require: gradient G; hyperparameters $(\eta, \beta_1, \beta_2, \varepsilon, \texttt{pf}, \mathrm{tol})$; state M, r, c, R, Q_R

$R \leftarrow \beta_2 R + (1-\beta_2)\, G^\top G$ *(input-side factor)*

if $t \bmod \texttt{pf} = 0$ **and** $\mathrm{off}(Q_R^\top R\, Q_R) > \mathrm{tol}$ **then**

  $Q_R \leftarrow \mathrm{eigh}(R)$ *(refresh the eigenbasis)*

end if

$M \leftarrow \beta_1 M + (1-\beta_1)\, G$ *(first moment)*

$G_{\mathrm{rot}} \leftarrow G\, Q_R,\ M_{\mathrm{rot}} \leftarrow M\, Q_R$ *(rotate into the eigenbasis)*

$r \leftarrow \beta_2 r + (1-\beta_2)\, G_{\mathrm{rot}}^{\circ 2}\, \mathbf{1}_n/n,\ c \leftarrow \beta_2 c + (1-\beta_2)\, \mathbf{1}_m^\top G_{\mathrm{rot}}^{\circ 2}/m$ *(factored second moment)*

$\hat{M}_{\mathrm{rot}} \leftarrow M_{\mathrm{rot}}/(1-\beta_1^t),\ \hat{r} \leftarrow r/(1-\beta_2^t),\ \hat{c} \leftarrow c/(1-\beta_2^t)$ *(bias correction)*

$\hat{v}_{\mathrm{rot}} \leftarrow \tau\, \hat{r}\, \hat{c}^\top$ with $\tau = \mathrm{tr}(R)/(\|\hat{r}\|_1 \|\hat{c}\|_1)$ *(rank-one second moment at R's scale)*

$U_{\mathrm{rot}} \leftarrow \hat{M}_{\mathrm{rot}} \oslash (\sqrt{\hat{v}_{\mathrm{rot}}} + \varepsilon)$ *(Adam step in the eigenbasis)*

$W \leftarrow W - \eta\, U_{\mathrm{rot}}\, Q_R^\top$ *(rotate back and update)*

4.1 Input-only Preconditioning

Our idea is to remove the output-side factor L_t and maintain only $R_t \leftarrow \beta_2 R_{t-1} + (1-\beta_2)\, G_t^\top G_t$ with its eigenbasis $Q_R = \mathrm{eigh}(R_t)$. The gradient rotation is applied only to the input axis, i.e., $G_{\mathrm{rot}} = G\, Q_R \in \mathbb{R}^{m\times n}$. This reduces the factor-state memory from $\mathcal{O}(m^2 + n^2)$ to $\mathcal{O}(n^2)$ and the number of eigendecompositions from two to one, halving this cost for square matrices.

A conceptual justification comes from the curvature of an affine layer with cross-entropy loss, the standard loss for language modeling and classification. For $y = Wx$ and $p = \mathrm{softmax}(y)$, the Hessian of the loss $\ell = -\log p_{k^\star}$ for a target class $k^\star$ with respect to W decomposes exactly as

$$H \;=\; xx^\top \otimes \big(\mathrm{diag}(p) - pp^\top\big). \tag{4}$$

The output-side factor $\mathrm{diag}(p) - pp^\top$ has diagonal entries $p_i(1-p_i)$ and off-diagonal entries $-p_i p_k$. Each off-diagonal entry $p_i p_k$ is smaller than the diagonal entry $p_i(1-p_i)$ by a factor of $p_k/(1-p_i)$, which is small for most classes of a large vocabulary even far from a one-hot prediction, in line with the dominant per-class diagonal blocks of softmax linear models [25]. Larger entries arise only where a class takes much of the mass, and each pair (i,k) receives them only from examples in which i or k is such a class, whereas the diagonal accumulates over all examples, so with a large vocabulary the expected output-side curvature is dominated by its diagonal, and under the K-FAC independence approximation the batch expectation becomes

$$\mathbb{E}[H] \;\approx\; \mathbb{E}[xx^\top] \otimes \mathrm{diag}(\kappa), \qquad \kappa_i \;=\; \mathbb{E}\big[p_i(1-p_i)\big]. \tag{5}$$

This has the structure of FOAM's implicit Fisher, with the input-side curvature $\mathbb{E}[xx^\top]$ tracked by R_t and the per-output curvature scale κ_i absorbed into $\hat{r}_t$. The output side need not be homogeneous across rows. FOAM only requires that it is dominated by its diagonal, which this mechanism suggests and the Gauss-Newton measurements of §5.4 support empirically.

Beyond the affine setting. For a hidden affine layer $h = Wx$, the per-example Hessian keeps the same form $xx^\top \otimes \nabla_h^2 \ell$, since h is linear in W, and only the output-side factor changes, from $\mathrm{diag}(p) - pp^\top$ to the curvature with respect to h. The vanishing-off-diagonal mechanism is specific to the output layer, but near-block-diagonal Hessians have been observed in MLPs throughout training [5, 50] and, with blocks along output neurons or heads, in a small transformer [50]. We provide additional indirect evidence specific to our setting through Gauss-Newton alignment measurements in §5.4.

4.2 Factored Eigenvalue Under Input-only Preconditioning

Even with the input-only simplification, SOAP still has overheads due to the dense rotated second-moment buffer $v_{\mathrm{rot}} \in \mathbb{R}^{m\times n}$. Our next step is to replace it with two factored estimators $r \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, EMAs of the row- and column-means of $G_{\mathrm{rot}}^{\circ 2} = G_{\mathrm{rot}} \circ G_{\mathrm{rot}}$, and reconstruct the rotated

Table 1: **Per-parameter complexity.** State, per-step compute, and amortized eigendecomposition cost for $W \in \mathbb{R}^{m\times n}$ with output-side $L \in \mathbb{R}^{m\times m}$ and input-side $R \in \mathbb{R}^{n\times n}$. FOAM keeps Muon's per-step cost.

	AdamW	**Muon**	**SOAP**	**FOAM**
Optimizer state	$2mn$	mn	$2mn + 2m^2 + 2n^2$	$mn + 2n^2 + (m+n)$
Per-step cost	$\mathcal{O}(mn)$	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn(m+n))$	$\mathcal{O}(mn^2)$
Amortized (per `pf`)	–	–	$\mathcal{O}(m^3 + n^3)$	$\mathcal{O}(n^3)$

second moment as an outer product with trace normalization:

$$\hat{v}_{\text{rot}}[i,j] \;=\; \tau \cdot \hat{r}[i] \cdot \hat{c}[j], \qquad \tau \;=\; \frac{\operatorname{tr}(R)}{\|\hat{r}\|_1 \, \|\hat{c}\|_1}. \tag{6}$$

The state shrinks from mn to $m+n$ per matrix parameter. The trace-matching scalar τ makes the entries of $\hat{v}_{\text{rot}}$ sum to $\operatorname{tr}(R)$, fixing the overall scale of the reconstructed second moment to that of the input-side factor. We now justify why this rank-one factorization is sufficient.

Lemma 1 (Covariance of the input-rotated gradient)**.** *Let $g = \operatorname{vec}(G) \in \mathbb{R}^{mn}$ and assume the K-FAC factorization $\mathbb{E}[gg^\top] = R \otimes L$. Let $g' = \operatorname{vec}(G\,Q_R)$ be the input-rotated gradient in column-major form. Then*

$$\mathbb{E}[g'\, g'^\top] \;=\; \Lambda_R \otimes L, \tag{7}$$

where $\Lambda_R = Q_R^\top R\, Q_R$ is the diagonal eigenvalue matrix of R.

Proof. Using the column-major identity $\operatorname{vec}(AGB) = (B^\top \otimes A)\operatorname{vec}(G)$ with $A = I_m$ and $B = Q_R$, we have $g' = (Q_R^\top \otimes I_m)\, g$. Therefore,

$$\mathbb{E}[g'\, g'^\top] \;=\; (Q_R^\top \otimes I_m)\, \mathbb{E}[g\, g^\top]\, (Q_R \otimes I_m) \;=\; (Q_R^\top \otimes I_m)(R \otimes L)(Q_R \otimes I_m). \tag{8}$$

The Kronecker mixed-product identity $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, applied twice, gives

$$(Q_R^\top \otimes I_m)(R \otimes L)(Q_R \otimes I_m) \;=\; (Q_R^\top R\, Q_R) \otimes (I_m\, L\, I_m) \;=\; \Lambda_R \otimes L. \tag{9}$$

□

Lemma 1 tells us two things about the rotated gradient. First, the input-side covariance becomes diagonal, because rotating by Q_R decorrelates the input directions and replaces R with its eigenvalues Λ_R. Second, the output-side factor L is unchanged, so when L is diagonal (§4.1) the rotated covariance $\Lambda_R \otimes L$ is itself diagonal, and its diagonal, the part FOAM keeps, factorizes as

$$\mathbb{E}\big[(G\,Q_R)[i,j]^2\big] \;=\; \lambda_R[j] \cdot L[i,i], \tag{10}$$

with $\lambda_R[j]$ the j-th diagonal entry of Λ_R. The right-hand side is a per-row factor depending only on i times a per-column factor depending only on j, so the entire $m \times n$ second-moment surface is captured by $m+n$ numbers rather than mn. The row- and column-means of $G_{\text{rot}}^{\circ 2}$ are the natural estimators of these two factors, up to a global scalar that the trace normalization τ removes, so $\tau\, \hat{r}\, \hat{c}^\top$ recovers the dense mn buffer exactly under K-FAC, and SOAP's two-sided one when L is also diagonal. In our runs, the K-FAC assumption is violated by a 15–20% relative error, mostly at the key projection, yet this costs only about 0.002 nats in final validation loss (Table 3, Appendix C).

Amortized eigenbasis update. The eigenbasis Q_R is refreshed every `pf` steps via the adaptive skip rule of Purifying Shampoo [11]. We form $B = Q_R^\top R_t Q_R$ and skip the eigendecomposition whenever its off-diagonal ratio $\operatorname{off}(B) = \|B - \operatorname{diag}(B)\|_F / \|B\|_F \le \text{tol}$, keeping the previous Q_R ($\text{tol} = 0$ refreshes at every `pf`-th step). With a small tol, an eigendecomposition is invoked only when R_t has drifted from diagonal in the current basis, which becomes rare late in training.

Complexity. Table 1 compares per-parameter optimizer state and compute. SOAP stores $2mn + 2m^2 + 2n^2$ values for the first and rotated second moments, two factor EMAs, and two eigenbases. FOAM reduces this to $mn + 2n^2 + (m+n)$ for one first moment, the estimators r, c, and the factor R with its eigenbasis Q_R. Per-step cost is $\mathcal{O}(mn^2)$ from the rotation $G\,Q_R$, matching Muon, and the amortized eigendecomposition cost drops from $\mathcal{O}(m^3 + n^3)$ to $\mathcal{O}(n^3)$.

Table 2: **Pre-training results.** For each method and scale at the best LR/WD (details in Appendix A) of its own search, we report validation loss (*val*, mean ± std over 11 seeds and 3 at large), matrix-group optimizer state in GiB (*state*), per-step wall time in ms on one 8×H100 node (*iter*), and the wall-clock t_{A} (minutes at small, hours otherwise) to reach AdamW's per-size final loss, i.e., the steps to reach it times *iter*.

	small (124M)				**medium** (350M)				**large** (760M)			
Optimizer	*val*	*state*	*iter*	t_{A}	*val*	*state*	*iter*	t_{A}	*val*	*state*	*iter*	t_{A}
AdamW	3.370±.009	0.63	311	23.8	3.040±.012	2.25	856	3.04	2.847±.004	5.27	1449	10.55
Muon	3.313±.008	0.32	361	22.9	2.998±.010	1.12	983	2.95	2.806±.005	2.64	1668	11.12
AdaMuon	3.311±.009	0.63	366	23.2	2.992±.012	2.25	991	2.97	2.799±.004	5.27	1684	10.85
SPlus	3.325±.007	2.85	525	35.0	–	–	–	–	–	–	–	–
SOAP	**3.305**±.009	2.85	536	32.1	**2.983**±.012	10.12	1521	4.56	**2.791**±.005	23.73	3343	22.66
FOAM (ours)	3.309±.008	1.46	393	24.9	2.991±.012	5.06	1083	3.25	2.796±.005	11.87	1692	10.53

5 Experiment

This section asks whether FOAM matches strong Muon and SOAP baselines in pre-training across scales (§5.2), and whether its design components are well-founded and each contributes (§5.4).

5.1 Experimental Setup

Setup. We pre-train the models following a modern LLM architectural setup (RoPE [42], RMSNorm [47], untied embeddings, MLP with GELU [17], no bias) at three scales following GPT-2 [38]: small (124M), medium (350M), and large (760M), on FineWeb [37] at a 1× Chinchilla compute budget (≈ 20 tokens per parameter) [19], giving 2.49B, 7.10B, and 15.49B training tokens over 4,750, 13,550, and 29,550 optimizer steps, respectively, at a fixed batch of 0.5M tokens (1k sequence length). All runs use bfloat16 mixed precision on 8×H100 (FSDP [52] at medium and large), with cosine decay to 10% of the peak LR after 2% warmup, gradient clipping at 1.0, and $\varepsilon = 10^{-8}$.

Parameter groups. Following the setup of Jordan et al. [20], Frans et al. [12], only the matrix parameters (Q/K/V/O attention projections and MLP linears) are preconditioned by the optimizer under study, while embeddings, the lm-head, and 1D and norm parameters are routed to a fixed AdamW group with LR 10^{-2}, $\beta = (0.95, 0.95)$, no weight decay on embeddings, and weight decay 10^{-3} on the lm-head. Unless noted, all groups use $\beta = (0.95, 0.95)$, and SOAP variants use precondition frequency $\mathtt{pf} = 10$ with the Purifying-Shampoo skip tolerance $\mathrm{tol} = 0.1$ [11].

Hyperparameter search. At small we run a 5×5 LR/WD grid (factor $10^{1/8} \approx 1.33$ on LR as in Frans et al. [12], and $10^{1/4} \approx 1.78$ on WD) per optimizer at $\beta = (0.95, 0.95)$, re-centering on grid edges, and re-validate the inner 3×3 neighborhood over three seeds. For medium and large, dense LR/WD grids are infeasible, so we transfer the small inner LR via a μP [46] ($\propto 1/\text{width}$) anchor, sweep 5 LR points (factor $10^{1/8}$) around it with WD held at the small inner optimum, refine the best point with a 3×3 LR/WD grid extended at grid edges, and report the selected cell as the per-size point estimate. Validation loss is reported as mean ± std over 11 seeds at small and medium and 3 seeds at large.

Baselines. The headline comparison spans AdamW [32], Muon [20], AdaMuon [41], SOAP [44] (the bilateral, dense-v variant), and FOAM (ours) at all three scales. At GPT-2 small we additionally evaluate SPlus [13], Shampoo [16], and the SOAP variants of our construction ablation (§5.4).

5.2 Main Results

Table 2 reports the main pre-training results across three model scales. SOAP attains the lowest validation loss at every scale, with FOAM behind by 0.004, 0.008, and 0.005 nats at small, medium, and large, each within about one seed std. FOAM is ahead of Muon, AdaMuon, and AdamW at every scale. The cost gap to SOAP *widens at scale*. FOAM uses half of SOAP's matrix-parameter state at every size, and its measured peak memory at small size is +7.4% over AdamW against +12.9% for SOAP. On wall-clock time, FOAM needs fewer steps than AdamW to reach AdamW's final loss, which offsets its higher per-step cost: its time to that loss, t_{A}, is within 7% of AdamW's own at every scale and equal at large (10.53 vs. 10.55 h), whereas SOAP needs 1.35× at small and 2.15× at large, as its two eigendecompositions grow cubically with width.

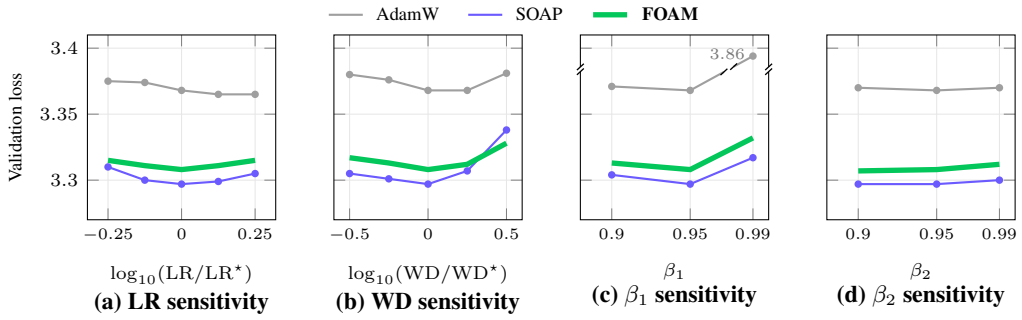


Figure 3: **Hyperparameter robustness around each optimizer's best cell on GPT-2 small.** For AdamW, SOAP, and FOAM we sweep 5 LR points (factor $10^{1/8}$), 5 WD points (factor $10^{1/4}$) at the best LR, and 3 values each of β_1 and β_2 at the best LR/WD. All other HPs are held fixed. Panel (c) breaks the y-axis at the top: AdamW's point at $\beta_1 = 0.99$ is 3.86, far above the other values.

Table 3: **FOAM-construction ablation on GPT-2 small (124M).** Starting from SOAP, input R and output L mark which Kronecker factor (with its rotation) is kept, and factored v replaces the dense second moment by r, c. Each variant uses its own best LR/WD cell; mean $\pm$ std over 3 seeds of the ablation runs' final evaluation. state is the matrix-parameter optimizer state (GiB) and wall is the total training wall-clock on a single 8×H100 node.

input R	output L	factored v	val loss	state (GiB)	wall (h)
✓	✓		$3.295_{\pm.011}$	2.92	0.48
✓	✓	✓	$\mathbf{3.293}_{\pm.008}$	2.59	0.49
✓			$3.302_{\pm.008}$	1.78	**0.44**
	✓		$3.322_{\pm.010}$	1.78	0.46
✓		✓	$3.304_{\pm.007}$	**1.46**	0.46

5.3 Hyperparameter Robustness

A practical optimizer must also tolerate imprecise tuning. Figure 3 sweeps LR, WD, and β_1, β_2 around each optimizer's inner best on GPT-2 small. FOAM's bowl shape and width track AdamW and SOAP across all four axes within 0.05 nats. The one notable asymmetry is AdamW's catastrophic collapse to 3.86 at $\beta_1 = 0.99$, where SOAP and FOAM degrade by only 0.02 nats, so FOAM inherits SOAP's wider safe range on the momentum axis, as panel (c) shows.

5.4 Analysis

We first measure what each FOAM simplification contributes to quality and cost, then ask whether input-only preconditioning is structurally justified on actual GPT-2 layers, and finally whether the factored update stays close to the dense SOAP update in practice.

FOAM-construction ablation. Table 3 ablates, starting from SOAP, the two structural simplifications of FOAM on GPT-2 small, namely input-only preconditioning, which drops the output-side rotation Q_L, and the factored second moment, which replaces the dense v with the row and column estimators r, c. Each variant is evaluated at its own best HP cell with 3 seeds. Input-only preconditioning substantially reduces both optimizer state and wall-clock while keeping validation loss within 0.01 nats of full SOAP, whereas output-only preconditioning, at the same state, loses about 0.02 nats, in line with the input-output asymmetry analyzed below. Factorization further trades a small wall-clock increase for a meaningful reduction in optimizer state with no measurable loss in validation. Stacked, the two simplifications yield FOAM, which retains SOAP's validation while halving the matrix-parameter optimizer state. The same holds at medium and large, where input-only SOAP and FOAM differ by at most 0.01 nats while the factored second moment saves 18% of the state.

Gauss-Newton fit. Figure 4 compares six implicit preconditioners against the per-layer Gauss-Newton matrix H_{GN} on all 72 matrix parameters of the 12-layer GPT-2 small backbone. SOAP's full bilateral preconditioner is the closest Kronecker approximant of H_{GN}, about 6× above Adam-diag and 2× above Muon. The asymmetry between input and output rotations is sharp and persistent. Dropping the output-side rotation costs only a quarter of the alignment, so input-only keeps 75% of

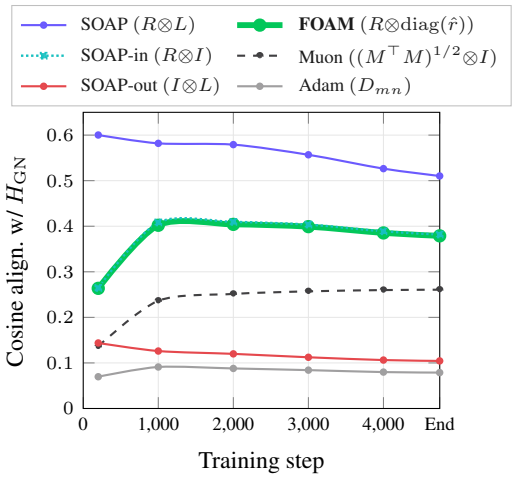


Figure 4: **Layer-wise Gauss-Newton alignment.** For each of 72 matrix parameters, we estimate $H_{\mathrm{GN}} = J_\ell^\top G_y J_\ell$ via 10 to 30 Hutchinson probes with the closed-form softmax-CE output Hessian, and compute the Frobenius cosine $\cos(P, H_{\mathrm{GN}})$ in sqrt-power form for six implicit preconditioners. Curves are means over the 72 matrices along one SOAP run on GPT-2 small.

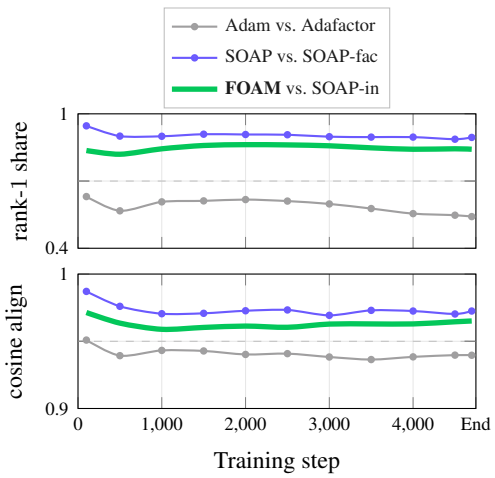


Figure 5: **Validity of the factored second moment.** On the dense run of each regime, we form the rank-1 approximation $V_{\mathrm{fac}} = (V\mathbf{1})(\mathbf{1}^\top V)/(\mathbf{1}^\top V\mathbf{1})$ from the row and column sums of the second-moment estimate V, and report **(a)** the rank-1 share $\sigma_1(V)/\sum_i \sigma_i(V)$ and **(b)** the Frobenius cosine between V_{fac} and V, averaged across all matrix parameters.

SOAP-full at 0.38, whereas dropping the input-side rotation costs 80% and leaves output-only at 0.10. FOAM, which is input-only and factored, tracks SOAP-input-only within 1–2% at every checkpoint, indicating that factoring the eigenvalue tensor preserves nearly all of input-only's curvature alignment. This asymmetry, in which the input-feature axis carries the dominant curvature structure, is the empirical justification for the input-only recipe used in the main results. It also holds per module, since on the exact layer-wise Gauss-Newton matrix input-only exceeds output-only in all six module types at both 124M and 350M, by 1.5× to 11.5× (Appendix D).

Factored update validity. Figure 5 reports two diagnostics that explain why the factored variants of Table 3 preserve quality. The rank-1 share in panel (a) is the key diagnostic. In Adam's raw g^2 basis, V is only 0.54–0.63 rank-1 separable and drifts down during training, while in SOAP's eigenbasis the rotated V stays near 0.90 for the bilateral rotation and 0.84 for the input-only rotation of FOAM. The cosine in panel (b) follows accordingly, at about 0.97, 0.96, and 0.94 in the bilateral, input-only, and raw bases. Consequently, factoring the SOAP-rotated V loses essentially nothing, costing less than 0.005 nats in validation, whereas Adafactor-style factoring [39] of Adam's raw V is far less accurate. With the H_{GN} alignment of Figure 4, this completes FOAM's rationale, which rotates into the input-side eigenbasis, where V becomes nearly rank-1 separable, and then factors.

5.5 Beyond the Main Setting

Scale. At GPT-2 XL, with 1.56B parameters and 31B tokens, FOAM and SOAP are within 0.001 nats over three seeds and both ahead of Muon and AdamW, which are the other two optimizers run at this scale (Appendix E). *Modality.* On ViT-Tiny trained on Tiny-ImageNet, the matrix-preconditioned optimizers beat AdamW by 6.5 to 8.5 points of top-1 accuracy, with SOAP first and FOAM second, on par with Muon and AdaMuon (Appendix F). *Batch size.* At a fixed token budget, growing the batch from 0.5M to 2M tokens per step raises AdamW's loss from 3.370 to 3.956 but FOAM's only from 3.309 to 3.364. FOAM also stays ahead of Muon in final loss at every batch size, and at 2M tokens per step only FOAM reaches AdamW's 0.5M-batch quality. Larger batches thus stay usable with FOAM, which at 2M tokens per step reaches AdamW's 0.5M-batch quality in about a third of its own 0.5M-batch wall-clock time (Appendix G).

6 Discussion and Conclusion

Limitations and future work. Our experiments stop at GPT-2 scale, up to 1.56B parameters, so whether FOAM's advantages hold for frontier-scale pre-training remains the main open question. Our implementation does not yet shard R and Q_R across devices, which matters at the $\geq$ 7B scale, although the layer-wise distributed optimizer of Khona et al. [22] suggests that this is an engineering rather than a fundamental obstacle. The factored update is least accurate at the key projection, whose gradients pass through RoPE and softmax (Appendix C). Finally, the embeddings and the lm-head stay on AdamW because their vocabulary dimension rules out a dense factor on that side, although a FOAM-style factor on their small hidden dimension would be affordable. Extending FOAM to these layers, and attention-aware factorizations for the key projection, are natural next steps.

When to use FOAM. FOAM fits when SOAP-class quality is needed but SOAP's memory or wall-clock is binding, since it stays within 0.004–0.008 nats of SOAP at half its optimizer state and, at 760M, about half its wall-clock. Against Muon and AdaMuon, quality slightly favors FOAM at every scale and speed is near parity, while AdaMuon keeps less state. Muon and AdaMuon thus remain preferable when optimizer state is the hardest constraint. Against AdamW, FOAM, like other matrix-preconditioned methods, is most valuable at large effective batch, where AdamW degrades (Appendix G) and, at fixed `pf`, the optimizer's compute share shrinks. FOAM is least attractive when memory is dominated by modules with a large input dimension, such as the MLP down-projection, whose n^2 factor carries most of FOAM's residual state over AdamW.

Conclusion. We presented *FOAM*, a scalable and practical SOAP variant built on two structural simplifications: input-only preconditioning, justified by the Kronecker structure of the Hessian and a near-diagonal output curvature, and a factored Adam-Moment update, exact under K-FAC, with Purifying Shampoo's skip rule for the amortized eigenbasis update. Where memory-efficient variants mainly shrink AdamW's memory, FOAM addresses the complementary problem and brings SOAP-level quality close to Adam-level cost. FOAM matches strong Muon and SOAP baselines on LLM pre-training with modern architectures while using half of SOAP's optimizer state and Muon-level wall-clock time, from `124M` to `760M`, with supporting evidence at 1.56B and on ViT.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.13649.

[2] Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The Polar Express: Optimal matrix sign methods and their application to the Muon algorithm. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.16932.

[3] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

[4] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*, pages 560–569, 2018.

[5] Ronan Collobert. *Large Scale Machine Learning*. PhD thesis, Université Paris VI, Paris, France, 2004.

[6] George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking neural network training algorithms. *arXiv preprint arXiv:2306.07179*, 2023.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.11929.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[9] Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S. Dhillon. Combining axes preconditioners through Kronecker approximation for deep learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[10] Runa Eschenhagen, Alexander Immer, Richard E. Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. arXiv:2311.00636.

[11] Runa Eschenhagen, Aaron Defazio, Tsung-Hsien Lee, Richard E. Turner, and Hao-Jun Michael Shi. Purifying Shampoo: Investigating Shampoo's heuristics by decomposing its preconditioner. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.03595.

[12] Kevin Frans, Pieter Abbeel, and Sergey Levine. What really matters in matrix-whitening optimizers? *arXiv preprint arXiv:2510.25000*, 2025.

[13] Kevin Frans, Sergey Levine, and Pieter Abbeel. A stable whitening optimizer for efficient neural network training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. arXiv:2506.07254.

[14] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker-factored eigenbasis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. arXiv:1806.03884.

[15] Wenbo Gong, Meyer Scetbon, Chao Ma, and Edward Meeds. Towards efficient optimizer design for LLM via structured Fisher approximation with a low-rank extension. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2502.07752.

[16] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*, pages 1842–1850, 2018. arXiv:1802.09568.

[17] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[18] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 1, 2021. arXiv:2103.03874.

[19] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. arXiv:2203.15556 (arXiv version titled "Training Compute-Optimal Large Language Models").

[20] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024. Blog post, December 2024. Accessed: 2026-05-07.

[21] Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, Boyuan Feng, Less Wright, Edward Z. Yang, Zachary Nado, Sourabh Medapati, Philipp Hennig, Michael Rabbat, and George E. Dahl. Accelerating neural network training: An analysis of the AlgoPerf competition. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2502.15015.

[22] Mikail Khona, Aditya Vavre, Boxiang Wang, Deyu Fu, Hao Wu, Mike Chrzanowski, Bryan Catanzaro, Dheevatsa Mudigere, Jeff Pool, Michael Lightstone, Mohammad Shoeybi, Mostofa Patwary, Nima Tajbakhsh, and Tijmen Blankevoort. SOAP, Muon, and beyond: Pushing LLM pretraining scales. *arXiv preprint arXiv:2607.20548*, 2026.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.

[24] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. arXiv:1905.12558.

[25] Frederik Kunstner, Alan Milligan, Robin Yadav, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why Adam outperforms gradient descent on language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 30106–30148, 2024. arXiv:2402.19449.

[26] Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. CS231N course project report, Stanford University, 2015. URL http://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

[27] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf, 2024. Hugging Face dataset: https://huggingface.co/datasets/AI-MO/NuminaMath-CoT.

[28] Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. In *International Conference on Machine Learning (ICML)*, pages 68514–68533, 2026. arXiv:2510.05491.

[29] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

[30] Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger B. Grosse. Understanding and improving Shampoo and SOAP via Kullback-Leibler minimization. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.03378.

[31] Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Yequan Zhao, Yupeng Su, Zi Yang, and Zheng Zhang. Muon^2: Boosting Muon via adaptive second-moment preconditioning. *arXiv preprint arXiv:2604.09967*, 2026.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. arXiv:1711.05101.

[33] Yanqing Lu, Letao Wang, and Jinbo Liu. Understanding SOAP from the perspective of gradient whitening. *arXiv preprint arXiv:2509.22938*, 2025.

[34] Chao Ma, Wenbo Gong, Meyer Scetbon, and Edward Meeds. SWAN: SGD with normalization and whitening enables stateless LLM training. In *International Conference on Machine Learning (ICML)*, pages 41907–41942, 2025. arXiv:2412.13148.

[35] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, pages 2408–2417, 2015.

[36] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on Shampoo's preconditioner. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.17748.

[37] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 37, pages 30811–30849, 2024. arXiv:2406.17557.

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[39] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, pages 4596–4604, 2018. arXiv:1804.04235.

[40] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel PyTorch implementation of the distributed Shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023.

[41] Chongjie Si, Debing Zhang, and Wei Shen. AdaMuon: Adaptive Muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

[42] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. arXiv:2104.09864.

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[44] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2409.11321.

[45] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[46] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17084–17097, 2021. arXiv:2203.03466 (extended arXiv version titled "Tensor Programs V: ...").

[47] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. arXiv:1910.07467.

[48] Minxin Zhang, Yuxuan Liu, and Hayden Schaeffer. Adam improves Muon: Adaptive moment estimation with orthogonalized momentum. *arXiv preprint arXiv:2602.17080*, 2026.

[49] Tianxiao Zhang, Wenju Xu, Bo Luo, and Guanghui Wang. Depth-wise convolutions in vision transformers for efficient training on small datasets. *Neurocomputing*, 617:128998, 2025. arXiv:2407.19394.

[50] Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.16793.

[51] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning (ICML)*, pages 61121–61143, 2024.

[52] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023. arXiv:2304.11277.

[53] Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z. Pan, Zhangyang Wang, and Jinwon Lee. APOLLO: SGD-like memory, AdamW-level performance. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025. arXiv:2412.05270.

A Selected Hyperparameters

Table A lists the matrix-group LR and WD that each optimizer selects in the search of §5.1.

Table A: **Selected matrix-group LR / WD** behind Table 2, for each optimizer and scale; all use $\beta = (0.95, 0.95)$ and $\varepsilon = 10^{-8}$, and the tail group is fixed as in §5.1.

Optimizer	small (124M)	medium (350M)	large (760M)
AdamW	1.9e−3 / 0.316	1.06e−3 / 0.316	8.5e−4 / 0.178
Muon	1.92e−2 / 0.0562	1.09e−2 / 0.0316	8.76e−3 / 0.0178
AdaMuon	4.41e−4 / 3.16	3.33e−4 / 1.0	2.0e−4 / 1.0
SPlus	0.333 / 0.00562	–	–
Shampoo	1.4e−2 / 0.1	–	–
SOAP	1.86e−3 / 0.562	3.32e−3 / 0.1	1.99e−3 / 0.1
FOAM (ours)	2.5e−3 / 0.316	1.41e−3 / 0.178	1.51e−3 / 0.1

B Shampoo on GPT-2 Small

We run Shampoo [16] at small only, as the Distributed Shampoo preconditioner with exponent $-1/4$ per factor and no grafting, and otherwise with the recipe and tuning of the other optimizers (Table A). Shampoo lands 0.07 nats behind FOAM and even slightly behind AdamW (Table B), whereas SOAP, which runs Adam in the same Kronecker eigenbasis, closes this gap.

Table B: **Shampoo on GPT-2 small (124M).** Validation loss at each optimizer's selected cell.

Optimizer	val
AdamW	$3.370_{\pm.009}$
Shampoo	$3.379_{\pm.006}$
SOAP	$\mathbf{3.305}_{\pm.009}$
FOAM (ours)	$3.309_{\pm.008}$

C Validity of the K-FAC Assumption

The factored update is exact under the K-FAC factorization (Lemma 1); the diagonal output side matters only for matching SOAP's two-sided rotation (Appendix D). For each instrumented matrix we measure on checkpoints (i) the *fidelity* $\rho(W) = \|\Lambda - rc^\top\|_F / \|\Lambda\|_F$, the relative Frobenius error of the row-column factorization of the rotated second moment Λ ($\rho = 0$ under K-FAC), and (ii) the *sensitivity* $\Delta(W) = \cos(P_{\text{dense}}V, H_{\text{GN}}V) - \cos(P_{\text{fac}}V, H_{\text{GN}}V)$ over random probes V, where H_{GN} is the layer's exact Gauss-Newton matrix (labels sampled from the model) and P_{dense}, P_{fac} are FOAM's preconditioner with the dense and the factored second moment, so that Δ isolates what the violation costs the update. The 18 instrumented matrices are the six module types at three depths, at 124M and 350M. Table C reports every matrix at step 2,000.

Findings. The violation is moderate and scale-stable: ρ averages 0.162 at 124M and 0.147 at 350M, and the row-column marginals nearly reach the best rank-one fit (rank-one energy 0.975). The attention key projection is the outlier (ρ up to 0.79 in middle and late layers; its gradients pass through RoPE and softmax), while W_v and W_o are almost exact. The consequence for the update is small: Δ averages $+0.005$ and $+0.004$, and stays below $+0.05$ even for W_k, consistent with the 0.002-nat end-to-end difference between input-only SOAP and FOAM (Table 3). Across six training stages at 124M (steps 200–4,749), the 18-matrix mean of ρ stays within 0.16–0.19 and that of Δ within $+0.001$–$+0.005$, and at the three stages measured at 350M within 0.15–0.17 and $+0.003$–$+0.004$, so no stage of training makes the factored update meaningfully worse.

Table C: **Per-layer K-FAC violation ρ and its cost Δ** (shown as ρ / Δ) at step 2,000 for every instrumented matrix: blocks 0, 6, 11 of 124M and 0, 12, 23 of 350M.

	124M			350M		
module	early	mid	late	early	mid	late
attention W_q	.209 / +.001	.059 / +.000	.067 / +.001	.350 / −.001	.070 / −.001	.062 / −.001
attention W_k	.372 / +.005	.785 / +.044	.769 / +.046	.352 / +.004	.706 / +.036	.508 / +.030
attention W_v	.017 / +.000	.026 / −.000	.011 / −.000	.027 / +.001	.028 / −.000	.012 / −.000
attention W_o	.013 / −.000	.027 / −.001	.023 / −.001	.014 / −.000	.029 / −.001	.020 / −.001
MLP up	.035 / −.000	.077 / −.001	.122 / +.002	.039 / −.000	.087 / −.000	.060 / +.004
MLP down	.050 / −.000	.145 / −.002	.112 / +.000	.084 / −.000	.154 / −.001	.047 / +.001
mean over 18		.162 / +.005			.147 / +.004	

D Per-module Input/Output Asymmetry

Table D breaks the Gauss-Newton alignment down by module type, measured with the exact layer-wise Gauss-Newton matrix of Appendix C at step 2,000 (Figure 4 uses a Hutchinson estimate over all 72 matrices of a separate run). Removing the output-side rotation from the two-sided preconditioner costs 0.160 of alignment on average, whereas removing the input-side rotation costs 0.375; averaged over depth, input-only exceeds output-only in every module type at both scales (1.5× to 10.9× at 124M and 1.9× to 11.5× at 350M), although a few individual layers reverse the order. Output-only preconditioning (0.132) is barely above diagonal Adam (0.090). The two factors differ in conditioning: after normalizing each by its own diagonal, the ratio of the largest to the 20th eigenvalue is 5.0 on the output side against 67.9 on the input side (7.4 against 213 on the EMA factors SOAP maintains), so on the output side the coordinate axes already capture most of the structure.

Table D: **Per-module input/output asymmetry at 124M.** Gauss-Newton alignment per module (mean over early, mid, and late depth) for output-only, input-only, and two-sided rotations; diagonal Adam scores 0.090 on the same measure, and the 350M values give the same ordering.

module	output only	input only	both	input / output
attention W_q	0.176	0.322	0.499	1.8×
attention W_k	0.118	0.178	0.314	1.5×
attention W_v	0.209	0.453	0.614	2.2×
attention W_o	0.045	0.493	0.629	**10.9×**
MLP up	0.152	0.332	0.522	2.2×
MLP down	0.091	0.300	0.466	3.3×
mean over 18 matrices	0.132	0.347	0.507	2.6×

E GPT-2 XL (1.56B)

We test whether the ordering of the three smaller sizes holds at GPT-2 XL with 1.56B parameters (Table E). SOAP and FOAM are tied (SOAP − FOAM $= -0.0008 \pm 0.0002$, paired over seeds), and both are 0.012–0.013 nats ahead of Muon and 0.04 ahead of AdamW.

The model has 48 layers and width 1600 and otherwise follows §5.1. It is trained for 59,450 steps of 0.5M tokens (31B tokens, 1× Chinchilla) on the FineWeb 100B-token sample with 2% warmup and cosine decay to 10%, and evaluated on a fixed 20M-token validation set. Each optimizer is tuned on 11 to 14 LR and WD cells with seed 42, centered by the rule $\mathrm{LR} \cdot \mathrm{WD} \cdot T \approx$ const fitted on the smaller sizes, and its best cell is re-run on seeds 1339 and 2024. SOAP and FOAM use `pf` $= 10$.

F ViT-Tiny on Tiny-ImageNet

We test whether the comparison transfers across modality, task, and loss by training ViT-Tiny [7, 43] for Tiny-ImageNet [26] classification (Table F). The matrix-preconditioned optimizers beat AdamW

Table E: **GPT-2 XL (1.56B, 31B tokens).** Validation loss on a fixed 20M-token set, mean ± std over seeds $\{42, 1339, 2024\}$ at each optimizer's coarse-grid optimum.

Optimizer	LR / WD	val
AdamW	2.4e−4 / 0.423	$2.6365_{\pm.0002}$
Muon	6.78e−3 / 0.0089	$2.6081_{\pm.0005}$
SOAP	1.17e−3 / 0.0529	$\mathbf{2.5951}_{\pm.0007}$
FOAM (ours)	1.48e−3 / 0.043	$2.5959_{\pm.0003}$

(58.6) by 6.5 to 8.5 points of top-1 accuracy, SOAP leads (67.1), and FOAM is second (65.4), tied with Muon and AdaMuon (both 65.1) within noise.

We follow the plain ViT-Tiny recipe of Zhang et al. [49], with 300 epochs, a cosine schedule with 20-epoch warmup, and its augmentation and regularization, but train at the native image size 64 with patch size 4 and batch 256. Only the attention and MLP weight matrices use the optimizer under study, and all other parameters run AdamW with LR $5{\cdot}10^{-4}$ and WD 0.05. Each optimizer's matrix LR and WD are tuned on a geometric grid with ratios $10^{1/4}$ and $10^{1/2}$ on one seed.

Table F: **ViT-Tiny on Tiny-ImageNet.** Top-1 is the best evaluation over 300 epochs (final-epoch values are 0.1–0.4 points lower, with SOAP first, FOAM second, and AdamW last); single seed. AdamW's matrix group uses $\beta = (0.9, 0.999)$, the others $(0.95, 0.95)$.

Optimizer	matrix LR / WD	top-1 (%)
AdamW	1.0e−3 / 0.158	58.6
Muon	1.0e−2 / 0.0158	65.1
AdaMuon	1.0e−3 / 0.158	65.1
SOAP	5.6e−3 / 0.05	**67.1**
FOAM (ours)	1.8e−3 / 0.158	65.4

G Batch-size Scaling

A fixed token budget can be spent at different batch sizes, and under data-parallel weak scaling each batch doubling roughly halves the wall-clock time, so we test whether that speed is usable (Table G). AdamW degrades sharply (3.370 → 3.465 → 3.956) and never recovers its 0.5M-batch quality at larger batches, whose wall-clock saving thus comes with a worse model. FOAM holds (3.309 → 3.315 → 3.364) and stays ahead of Muon at every batch size, turning the same scaling into about a 3× reduction in the time to AdamW's quality (24.9 → 13.8 → 8.2 minutes).

GPT-2 small follows the recipe of §5.1 with a fixed 2.49B-token budget, i.e., 4,750 steps at 0.5M tokens, and each optimizer's LR is re-tuned at 0.5M, 1M, and 2M tokens per step.

Table G: **Batch-size scaling on GPT-2 small** (under the fixed 2.49B-token budget, LR re-tuned at every batch size). *wall* → 3.370 is the wall-clock time to reach AdamW's own 0.5M-batch quality under data-parallel weak scaling (total wall ∝ steps); N/R (not reached) marks runs that do not reach it, and the 1M and 2M rows are from a single seed each.

	final val loss			wall → 3.370	
tokens/step (steps)	AdamW	Muon	FOAM	AdamW	FOAM
0.5M (4,750)	3.370	3.313	**3.309**	23.8 min	24.9 min
1M (2,375)	3.465	3.323	**3.315**	N/R	13.8 min
2M (1,188)	3.956	3.384	**3.364**	N/R	8.2 min

H LLM Post-training

We test whether FOAM can replace AdamW in two post-training recipes on Qwen3 [45], changing only the optimizer (Table H). FOAM is best in SFT and tied with AdamW in OPD, while Muon is

0.013 below both in OPD. The SFT differences among the three optimizers are within 0.002 NLL, so FOAM serves as a drop-in replacement for AdamW in both recipes.

Only the 2D weight matrices of the transformer blocks use the optimizer under study, and the rest run AdamW. One LR, shared by all groups, is swept per optimizer. FOAM uses $\texttt{pf} = 10$ and $\mathrm{tol} = 0$, and Muon scales its update by $0.2\sqrt{\max(m, n)}$. All runs use no weight decay, gradient clipping at 1.0, a cosine schedule to 10%, bfloat16 with fp32 master weights, and a single seed. *Supervised fine-tuning (SFT).* Qwen3-1.7B-Base is fine-tuned for one epoch on 100K NuminaMath-CoT [27] examples (782 steps of batch 128, up to $4{,}096$ tokens, $\beta = (0.9, 0.95)$) and evaluated by the NLL on $1{,}000$ held-out examples. *On-policy distillation (OPD) [1].* Qwen3-1.7B is distilled from Qwen3-4B-Instruct-2507 in non-thinking mode with a per-token reverse-KL loss on its own rollouts for open reasoning prompts (100 steps of 256 prompts, up to $8{,}192$ tokens, $\beta = (0.9, 0.999)$) and scored by average accuracy on MATH500 [18, 29], AIME24, AIME25, and AMC23.

Table H: **Post-training of Qwen3-1.7B with each optimizer at its best LR** (in parentheses). SFT reports held-out NLL (lower is better) and OPD average math accuracy (higher is better), single seed.

Recipe	base	AdamW	Muon	FOAM
SFT	0.625	0.4677 ($3\mathrm{e}{-5}$)	0.4668 ($3\mathrm{e}{-5}$)	$\mathbf{0.4657}$ ($3\mathrm{e}{-5}$)
OPD	0.338	$\mathbf{0.4719}$ ($1\mathrm{e}{-5}$)	0.4591 ($1\mathrm{e}{-5}$)	$\mathbf{0.4719}$ ($1\mathrm{e}{-5}$)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and Section 1 state the contributions; experiments in Section 5 match the claimed scope (GPT-2-modern up to 760M, with supporting results at 1.56B, on ViT, and in post-training in the appendix).

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the residual $\mathcal{O}(n^2)$ state overhead over AdamW, the missing sharding of R and Q_R, the layers where the K-FAC assumption is weakest, and when FOAM is less attractive than Muon, AdaMuon, or AdamW.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Lemma 1 states its assumptions inline and is proved in Section 4.2.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 and Appendix A fix the recipe, search protocol, per-cell settings, and FOAM's default HPs, and Algorithm 1 with Section 4.2 fully specifies the optimizer.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and sweep configurations are released at `https://github.com/naver-ai/foam`; the datasets are public and the algorithm is fully specified in Algorithm 1, with the selected settings in Appendix A.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 specifies the LR, WD, and β grids, μP transfer, seeds, and reporting convention.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 2 reports mean $\pm$ std over 11 seeds at small and medium, and 3 at large. The ablation and the 1.56B results use 3 seeds, and the ViT, post-training, and 1M and 2M batch-scaling results are single-seed, as flagged in the captions and appendix.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Pre-training runs use $8\times$ H100 (80GB) each and post-training runs 8 GPUs; per-step time, time to AdamW's loss, and state size are in Table 2, and the 1.56B runs of Appendix E take 56–59 hours each.

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: Foundational optimizer research on the public FineWeb [37] corpus; no human subjects.

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: Generic optimizer with no direct path to negative applications beyond the application-agnostic effect of cheaper LLM training.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: No model checkpoints, image generators, or scraped data are released.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All baselines and ingredients are cited at first use; FineWeb [37], Tiny-ImageNet, NuminaMath-CoT, and the Qwen3 checkpoints are used under their public release terms.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code is documented in its repository; no new datasets or models are released.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: No crowdsourcing or human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects research.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are the subject of training, not a component of the method; any LLM use during writing was limited to editing.