MUONIO: PRINCIPLED NORM-AWARE DESCENT FOR EMBEDDING TABLES AND LANGUAGE MODEL HEADS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Muon optimizer derives its update rule for hidden linear layers by solving a local linearization of the loss penalized by the spectral norm, motivated by an RMS-stability argument for dense linear layers. Standard Muon implementations, however, exclude the input (embedding table) and output (language model head) layers from this principled treatment, for which they use AdamW instead. We present MuonIO, a single Muon-style update for both of these layers. For the language model head $\mathbf{L} \in \mathbb{R}^{V \times d}$, we motivate the use of the $2 \to \infty$ operator norm, due to the Lipschitz continuity of the softmax output geometry, while for the embedding table $\mathbf{E} \in \mathbb{R}^{d \times V}$, we draw on the $1 \to 2$ operator norm, based on the one-hot input geometry identified by Bernstein & Newhouse (2025). The identity $\|\mathbf{L}\|_{2\to\infty} = \|\mathbf{L}^\top\|_{1\to 2}$ then puts both matrices in the same vocabulary-oriented geometry: MuonIO applies a single normalized-vector rule, which appears as column normalization for $\mathbf{E}$ and row normalization for $\mathbf{L}$. Empirical evaluations demonstrate the effectiveness of our approach, with MuonIO reducing I/O optimizer state memory by 50% and I/O update FLOPs by $\sim$46% compared to Muon for 1B LLaMA pretraining on C4, while also improving validation perplexity.

1 INTRODUCTION

Gradient-based optimization of neural networks has long been dominated by Adam (Kingma & Ba, 2014) and its descendants. The Muon optimizer (Jordan et al., 2024) revisited a foundational question: what is the right geometry for determining the direction of a weight update? Building on ideas from modular duality (Bernstein & Newhouse, 2024; 2025), Muon chooses its update by minimizing a local first-order approximation of the loss while penalizing the update in the *spectral norm geometry*. This geometry is appropriate for hidden linear layers (Yang et al., 2023) and yields the polar factor update, which is computed in practice by a Newton–Schulz iteration rather than by an explicit SVD (Jordan et al., 2024; Bernstein & Newhouse, 2025). Muon's practical impact has been rapid: it achieved a $1.35\times$ speedup over the previous NanoGPT training record (Jordan et al., 2024), has been used in large-scale models such as Moonlight (Liu et al., 2025) and Kimi K2 (Team et al., 2025), and has inspired variants such as HTMuon (Pang et al., 2026), NorMuon (Li et al., 2026), Polar Express (Amsel et al., 2026), AdaMuon (Si et al., 2025), Scion (Pethick et al., 2025), Dion (Ahn et al., 2025b), and Dion2 (Ahn et al., 2025a).

Despite this progress, standard Muon implementations still exclude two important layer types from the spectral norm update: the input **embedding layer** (`embed`) and the output **language model head** (`lm_head`). Instead, both are typically routed to AdamW, and as Jordan et al. (2024) explicitly note, this exception is not fully explained by the existing theory:

> *"That such dynamics are also different for the output layer does not seem to follow from the theory, and is instead driven by empirics."*

Similarly, Bernstein's derivation note (Bernstein, 2025) observes that the dense layer stability argument "*separates the Linear layer from, say, an Embedding layer that takes one-hot inputs.*" While prior work has identified the norm induced by one-hot embedding lookups (Bernstein & Newhouse, 2024; 2025), and column/row normalization updates have been proposed on other grounds (Pethick et al., 2025; Glentis et al., 2026; Lau & Su, 2026), there has yet to be established a principled norm-aware approach that handles both vocabulary-facing layers with a single update.

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.

1.1 Our Contribution

In this work, we take the embedding layer geometry (Bernstein & Newhouse, 2024; 2025) as a starting point and present **MuonIO**, a single Muon-style update for the two vocabulary-facing matrices. Prior work identifies the $1 \to \mathrm{RMS}$ norm and its column-normalized direction for `embed`, which, under our ℓ_2 convention, is the $1 \to 2$ norm up to a fixed dimension factor. For `lm_head`, we instead study the change in the probability distribution after softmax. A tight total-variation stability bound identifies the $2 \to \infty$ norm of the weight perturbation. These operator norms are related by transposition: $\|\mathbf{L}\|_{2\to\infty} = \|\mathbf{L}^\top\|_{1\to 2}$. Once both parameters are stored in vocabulary-major form, Corollary 2.3 gives the same normalized-vector update for both. In weight-tied models where $\mathbf{L} = \mathbf{E}^\top$ (Press & Wolf, 2017; Inan et al., 2016), it acts directly on the single shared parameter.

Optimizing `embed` and `lm_head` with AdamW also carries concrete computation and memory cost. These two layers are among the largest parameter matrices in a language model: each has Vd entries, with V being the vocabulary size and d the hidden dimension. AdamW maintains two optimizer state moments per parameter, whereas MuonIO keeps only a single momentum buffer. In terms of computation, AdamW takes $13Vd$ FLOPS per matrix while MuonIO takes $7Vd + 3V$. Routing these two layers to AdamW therefore roughly doubles the optimizer state memory and arithmetic cost on the model's biggest tensors. Further details are included in Appendix C.

We evaluate MuonIO by pretraining LLaMA-family models at the 60M, 130M and 1B scales. Replacing the AdamW treatment of the two I/O layers with one geometry-matched update halves their persistent optimizer state and, at the same time, yields improved validation perplexity over a strong Muon baseline that already tunes separate learning rates for the input, output, hidden, and gain/bias parameters.

1.2 Related Work

The work on modular duality (Bernstein & Newhouse, 2025) gives a column normalization update for `embed`, but does not provide a corresponding treatment of `lm_head`. Other optimizer frameworks have nevertheless proposed row-normalized output layer updates, and recent work has applied vocabulary-wise normalization specifically to language model heads, motivated by linear-minimization oracles, empirical gradient variance, and equivariance (Pethick et al., 2025; Glentis et al., 2026; Lau & Su, 2026), while Ember (Shah, 2026) treats both layers specially using factorized second moments. Another related line of work observes, both empirically and theoretically, that `embed` benefits from a learning rate distinct from the hidden layers (Hayou & Liu, 2025; He et al., 2026; Kalra & Barkeshli, 2026). Additionally, Nair (2025) establishes tight Lipschitz bounds for the softmax function under matched ℓ_p norms. For a more detailed discussion, see Appendix A.

2 Background and Notation

2.1 Notation

Scalars are denoted by lowercase letters (e.g., x); vectors by bold lowercase letters (e.g., $\mathbf{x}$); matrices by bold uppercase letters (e.g., $\mathbf{W}$). For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ we write $\|\mathbf{A}\|_F$ for the Frobenius norm, $\|\mathbf{A}\|_2$ for the spectral norm, $\|\mathbf{A}\|_*$ for the nuclear norm, and $\langle \mathbf{A}, \mathbf{B}\rangle = \mathrm{tr}\left(\mathbf{A}^\top\mathbf{B}\right)$ for the Frobenius inner product. We write $\mathbf{A}_{:,t}$ for the t-th column of $\mathbf{A}$ and $\mathbf{A}_{t,:}$ for its t-th row. We write $\mathbf{I}_k$ for the $k\times k$ identity matrix (dropping the subscript when the dimension is clear), and $[V]$ for $\{1, 2, \ldots, V\}$. We write $\Delta^V := \left\{p \in \mathbb{R}^V_{\geq 0} : \sum_{t=1}^V p_t = 1\right\}$ for the probability simplex over V outcomes.

2.2 Steepest Descent Under a Norm

Let $\mathcal{L}(\mathbf{W})$ be a loss function and let $\mathbf{G}_k = \nabla\mathcal{L}(\mathbf{W}_k)$ denote the gradient at iterate $\mathbf{W}_k$. For a given norm $\|\cdot\|$ on the weight matrices, the **steepest descent step** of size $\lambda > 0$ minimizes a penalized local linearization. The following generic result is Bernstein & Newhouse (2024, Proposition 1), stated for matrix spaces under the Frobenius inner product. We use the reciprocal of their sharpness parameter so that λ is a step size.

Proposition 2.1 (Penalized steepest descent)**.** *Let $\mathbf{G} \in \mathbb{R}^{m\times n}$ and let $\|\cdot\|$ be a norm on $\mathbb{R}^{m\times n}$, with dual norm $\|\mathbf{G}\|^\dagger := \max_{\|\mathbf{T}\|\le 1}\langle \mathbf{G}, \mathbf{T}\rangle$. For $\lambda > 0$,*

$$\operatorname*{argmin}_{\Delta\mathbf{W}} \left\{ \langle \mathbf{G}, \Delta\mathbf{W}\rangle + \frac{1}{2\lambda}\|\Delta\mathbf{W}\|^2 \right\} = \left\{ -\lambda \|\mathbf{G}\|^\dagger \mathbf{T} : \mathbf{T} \in \operatorname*{argmax}_{\|\mathbf{T}\|=1} \langle \mathbf{G}, \mathbf{T}\rangle \right\}. \tag{1}$$

Gradient descent (Frobenius geometry). Applying Proposition 2.1 with $\|\cdot\| = \|\cdot\|_F$ yields $\Delta\mathbf{W}^\star = -\lambda\mathbf{G}$, the ordinary gradient descent step.

Muon (spectral norm geometry). Taking $\|\cdot\| = \|\cdot\|_2$ and letting $\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the thin SVD of the gradient, Proposition 2.1 gives the following choice of solution:

$$\Delta\mathbf{W}^\star = -\lambda\, \|\mathbf{G}\|_* \mathrm{polar}(\mathbf{G}), \qquad \mathrm{polar}(\mathbf{G}) := \mathbf{U}\mathbf{V}^\top. \tag{2}$$

2.3 Operator Norms

For $p, q \in [1,\infty]$ and $\mathbf{A} \in \mathbb{R}^{m\times n}$, the $p \to q$ **operator norm** is $\|\mathbf{A}\|_{p\to q} := \max_{\|\mathbf{x}\|_p=1} \|\mathbf{A}\mathbf{x}\|_q$. For example, the spectral norm satisfies $\|\mathbf{A}\|_2 = \|\mathbf{A}\|_{2\to 2}$. Bernstein & Newhouse (2024, Proposition 8) establish the following induced-norm identities.

Proposition 2.2. *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$, let $p \in [1,\infty]$, and let p^* be its Hölder conjugate, so that $1/p + 1/p^* = 1$. Then*

$$\|\mathbf{A}\|_{1\to p} = \max_{j\in[n]} \|\mathbf{A}_{:,j}\|_p, \qquad \|\mathbf{A}\|_{p\to\infty} = \max_{i\in[m]} \|\mathbf{A}_{i,:}\|_{p^*}. \tag{3}$$

In particular,

$$\|\mathbf{A}\|_{1\to 2} = \max_{j\in[n]} \|\mathbf{A}_{:,j}\|_2, \quad \|\mathbf{A}\|_{2\to\infty} = \max_{i\in[m]} \|\mathbf{A}_{i,:}\|_2. \tag{4}$$

Combining Propositions 2.1 and 2.2 immediately gives the following identities.

Corollary 2.3. *Let $\mathbf{G} \in \mathbb{R}^{m\times n}$. The dual norms of $\|\cdot\|_{1\to 2}$ and $\|\cdot\|_{2\to\infty}$ are, respectively,*

$$\|\mathbf{G}\|_{1\to 2}^\dagger = \sum_{j=1}^{n} \|\mathbf{G}_{:,j}\|_2, \quad \|\mathbf{G}\|_{2\to\infty}^\dagger = \sum_{i=1}^{m} \|\mathbf{G}_{i,:}\|_2. \tag{5}$$

Define row and column normalization by

$$\big(\mathrm{RowNorm}(\mathbf{G})\big)_{i,:} := \begin{cases} \mathbf{G}_{i,:}/\|\mathbf{G}_{i,:}\|_2, & \mathbf{G}_{i,:} \neq \mathbf{0}, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \qquad \mathrm{ColNorm}(\mathbf{G}) := \mathrm{RowNorm}(\mathbf{G}^\top)^\top. \tag{6}$$

Then the $2 \to \infty$-penalized problem has the row-normalized solution

$$\Delta\mathbf{W}^\star = -\lambda \|\mathbf{G}\|_{2\to\infty}^\dagger \,\mathrm{RowNorm}(\mathbf{G}), \tag{7}$$

whereas the $1 \to 2$-penalized problem has the column-normalized solution

$$\Delta\mathbf{W}^\star = -\lambda \|\mathbf{G}\|_{1\to 2}^\dagger \,\mathrm{ColNorm}(\mathbf{G}). \tag{8}$$

2.4 Layer Types and Their Geometries

A standard transformer language model contains three qualitatively distinct layer types, each with a different input/output geometry. We denote V as the vocabulary size and d as the hidden dimension.

Hidden linear layers. A weight matrix $\mathbf{W} \in \mathbb{R}^{n_{\text{out}}\times n_{\text{in}}}$ maps a *dense* activation $\mathbf{x}$ into $\mathbf{y}$, both of which are expected to be dense vectors. Given a perturbation of the weight $\Delta\mathbf{W}$, the bound $\|\Delta\mathbf{y}\|_2 \le \|\Delta\mathbf{W}\|_2\|\mathbf{x}\|_2$ identifies the spectral norm as the natural penalty (Bernstein, 2025), motivating the Muon polar factor update.

Embedding layers. `embed` ($\mathbf{E} \in \mathbb{R}^{d\times V}$) maps token indices to dense vectors via the lookup $\mathbf{y} = \mathbf{E}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^V$ is *one-hot*. Subsection 3.1 reviews the resulting norm from prior work.

Language model head. `lm_head` ($\mathbf{L} \in \mathbb{R}^{V\times d}$) maps a hidden state $\mathbf{y} \in \mathbb{R}^d$ to logits $\mathbf{z} = \mathbf{L}\mathbf{y} \in \mathbb{R}^V$ and then to probabilities $\mathbf{p} = \mathrm{softmax}(\mathbf{z}) \in \Delta^V$, where Δ^V is the probability simplex. Given a perturbation $\Delta\mathbf{L}$, a good measure of the perturbation is not simply $\|\Delta\mathbf{z}\|_2$, since its effect must be propagated to the probability distribution. Instead, we measure that change by total variation, with the analysis in Subsection 3.2.

3 MUONIO FOR VOCABULARY-FACING LAYERS

The input embedding and output language model head have different storage orientations, but each has one vector associated with every vocabulary item. We first identify the appropriate norm for each layer, then orient both matrices along their vocabulary axis and give a single MuonIO update. Appendix B contains the proof of Propositions 3.3, 3.5, 3.7 and Remark 3.8.

3.1 EMBEDDING GEOMETRY FROM PRIOR WORK

The embedding table $\mathbf{E} \in \mathbb{R}^{d\times V}$ acts as a lookup: for a one-hot token indicator $\mathbf{x} \in \{0,1\}^V$, the forward pass returns $\mathbf{Ex}$, one column of $\mathbf{E}$. Consequently,

$$\max_{\mathbf{x}\in\{0,1\}^V,\, \sum_i x_i = 1} \|\mathbf{Ex}\|_2 = \max_{t\in[V]} \|\mathbf{E}_{:,t}\|_2 = \|\mathbf{E}\|_{1\to 2}. \tag{9}$$

The final equality is the $p = 2$ case of Proposition 2.2. Thus the $1 \to 2$ operator norm, rather than the spectral norm used for dense hidden layers, measures the largest change to any token embedding.

Bernstein & Newhouse (2024) identify the $1 \to \mathrm{RMS}$ operator norm for embedding layers, which is the same as $1 \to 2$ operator norm up to a scaling factor of $\sqrt{d}$. We use that established geometry as one half of MuonIO, and our new ingredient is the total variation justification for the matching output layer geometry developed next.

3.2 LANGUAGE-MODEL-HEAD GEOMETRY

The language model head $\mathbf{L} \in \mathbb{R}^{V\times d}$ maps a hidden state $\mathbf{y} \in \mathbb{R}^d$ to a probability distribution (as opposed to a dense activation vector) over the vocabulary via $\mathbf{p} = \mathrm{softmax}(\mathbf{Ly}) \in \Delta^V$. We thus identify a matched penalty norm by asking: which norm on $\Delta\mathbf{L}$ controls the worst case change in the *output distribution* induced by the weight perturbation?

3.2.1 TOTAL VARIATION STABILITY AND THE $2 \to \infty$ NORM

A canonical measure of the difference between probability distributions is the total variation distance, $\|\mathbf{p}' - \mathbf{p}\|_{\mathrm{TV}} = \frac{1}{2}\|\mathbf{p}' - \mathbf{p}\|_1$. We show that the $2 \to \infty$ operator norm controls this quantity.

Lemma 3.1 (Jacobian of softmax)**.** *Let* $\mathbf{p} = \mathrm{softmax}(\mathbf{z}) \in \Delta^V$*. The Jacobian of* softmax *at* $\mathbf{z}$ *is*

$$J(\mathbf{z}) = \frac{\partial\, \mathrm{softmax}(\mathbf{z})}{\partial \mathbf{z}} = \mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^\top. \tag{10}$$

Proof. Writing $\mathbf{p}_i = e^{\mathbf{z}_i} / \sum_{s=1}^V e^{\mathbf{z}_s}$, direct differentiation gives $\partial \mathbf{p}_i / \partial \mathbf{z}_j = \mathbf{p}_i\delta_{ij} - \mathbf{p}_i\mathbf{p}_j$. □

Lemma 3.2 ($\infty \to 1$ norm of the softmax Jacobian)**.** *For every* $\mathbf{z} \in \mathbb{R}^V$*,*

$$\|J(\mathbf{z})\|_{\infty\to 1} \le 1. \tag{11}$$

Proof. Fix $\mathbf{u} \in \mathbb{R}^V$ with $\|\mathbf{u}\|_\infty \le 1$, and let $\mathbf{p} = \mathrm{softmax}(\mathbf{z})$. The i-th coordinate of $J(\mathbf{z})\mathbf{u}$ is $\mathbf{p}_i(\mathbf{u}_i - m)$, where $m := \mathbf{p}^\top\mathbf{u} = \sum_i \mathbf{p}_i\mathbf{u}_i$. Hence $\|J(\mathbf{z})\mathbf{u}\|_1 = \sum_{i=1}^V \mathbf{p}_i|\mathbf{u}_i - m|$. Introduce a random variable X that takes value $\mathbf{u}_i$ with probability $\mathbf{p}_i$, so $m = \mathbb{E}[X]$ and $\|J(\mathbf{z})\mathbf{u}\|_1 = \mathbb{E}[|X - \mathbb{E}[X]|]$. By Jensen's inequality, $\mathbb{E}[|X - \mathbb{E}[X]|] \le \sqrt{\mathrm{Var}(X)}$. Since $\|\mathbf{u}\|_\infty \le 1$, all values of X lie in $[-1, 1]$. Popoviciu's inequality on variances (Popoviciu, 1935) states that $\mathrm{Var}(X) \le (b-a)^2/4$ for any random variable supported on $[a, b]$, so with $a = -1$ and $b = 1$ this gives $\mathrm{Var}(X) \le 1$. Therefore $\|J(\mathbf{z})\mathbf{u}\|_1 \le 1$, and taking the supremum over all such $\mathbf{u}$ proves the claim. □

Proposition 3.3 (Tightness of Lemma 3.2)**.** *For every* $V \ge 2$*, inequality* (11) *is attained: there exist* $\mathbf{z} \in \mathbb{R}^V$ *and* $\mathbf{u} \in \{\pm 1\}^V$ *such that* $\|J(\mathbf{z})\mathbf{u}\|_1 = 1$*.*

Lemma 3.4 (Global Lipschitz bound for softmax)**.** *For all* $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^V$*,*

$$\|\mathrm{softmax}(\mathbf{z}') - \mathrm{softmax}(\mathbf{z})\|_1 \le \|\mathbf{z}' - \mathbf{z}\|_\infty. \tag{12}$$

Proof. Let $\gamma(t) = \mathbf{z} + t(\mathbf{z}' - \mathbf{z})$ for $t \in [0, 1]$. By the fundamental theorem of calculus,

$$\operatorname{softmax}(\mathbf{z}') - \operatorname{softmax}(\mathbf{z}) = \int_0^1 J(\gamma(t))(\mathbf{z}' - \mathbf{z})\, dt. \tag{13}$$

Taking norms, applying Lemma 3.2 at each t, and integrating gives

$$\|\operatorname{softmax}(\mathbf{z}') - \operatorname{softmax}(\mathbf{z})\|_1 \le \int_0^1 \|J(\gamma(t))\|_{\infty\to 1}\, \|\mathbf{z}' - \mathbf{z}\|_\infty\, dt \le \|\mathbf{z}' - \mathbf{z}\|_\infty. \tag{14}$$

□

Proposition 3.5 (Tightness of Lemma 3.4)**.** *The constant* 1 *in Lemma 3.4 cannot be improved. Namely, for every* $V \ge 2$*,*

$$\sup_{\mathbf{z}' \ne \mathbf{z}} \frac{\|\operatorname{softmax}(\mathbf{z}') - \operatorname{softmax}(\mathbf{z})\|_1}{\|\mathbf{z}' - \mathbf{z}\|_\infty} = 1, \tag{15}$$

although the supremum is not attained by any finite pair $\mathbf{z}' \ne \mathbf{z}$*.*

Lemma 3.6 (TV stability of the lm_head)**.** *For every hidden state* $\mathbf{y} \in \mathbb{R}^d$ *and every perturbation* $\Delta\mathbf{L} \in \mathbb{R}^{V\times d}$*,*

$$\|\operatorname{softmax}((\mathbf{L} + \Delta\mathbf{L})\mathbf{y}) - \operatorname{softmax}(\mathbf{L}\mathbf{y})\|_{\mathrm{TV}} \le \frac{1}{2}\|\Delta\mathbf{L}\|_{2\to\infty}\, \|\mathbf{y}\|_2. \tag{16}$$

Proof. Set $\mathbf{z} = \mathbf{L}\mathbf{y}$ and $\mathbf{z}' = (\mathbf{L} + \Delta\mathbf{L})\mathbf{y}$, so $\mathbf{z}' - \mathbf{z} = (\Delta\mathbf{L})\mathbf{y}$. Applying Lemma 3.4 and then the definition of the $2 \to \infty$ norm,

$$\|\operatorname{softmax}(\mathbf{z}') - \operatorname{softmax}(\mathbf{z})\|_1 \le \|(\Delta\mathbf{L})\mathbf{y}\|_\infty \le \|\Delta\mathbf{L}\|_{2\to\infty}\, \|\mathbf{y}\|_2. \tag{17}$$

Dividing by 2 converts the ℓ_1 distance to total variation. □

Proposition 3.7 (Tightness of Lemma 3.6)**.** *The constant* $\frac{1}{2}$ *in Lemma 3.6 cannot be improved: for every* $V \ge 2$ *and* $d \ge 1$*,*

$$\sup_{\mathbf{L},\, \Delta\mathbf{L}\ne\mathbf{0},\, \mathbf{y}\ne\mathbf{0}} \frac{\|\operatorname{softmax}((\mathbf{L} + \Delta\mathbf{L})\mathbf{y}) - \operatorname{softmax}(\mathbf{L}\mathbf{y})\|_{\mathrm{TV}}}{\frac{1}{2}\|\Delta\mathbf{L}\|_{2\to\infty}\, \|\mathbf{y}\|_2} = 1, \tag{18}$$

although the supremum is not attained.

Lemma 3.6 shows that $\|\Delta\mathbf{L}\|_{2\to\infty}$ gives sharp uniform control of worst-case TV perturbation when $\|\mathbf{y}\|_2$ is held fixed. The $2 \to \infty$ operator norm is therefore the natural penalty for `lm_head`.

3.3 Transposition symmetry

The two layer geometries are the same after orienting the vocabulary axis consistently. This follows from the relation between the norms identified in Sections 3.1 and 3.2. The observation is a direct consequence of the dual norm property, and it makes the common MuonIO update transparent. For completeness, we include the proof in Appendix B.

Remark 3.8 (Transposition identity)**.** For any $\mathbf{A} \in \mathbb{R}^{m\times n}$ and any $p, q \in [1, \infty]$ with Hölder conjugates determined by $1/p + 1/p^* = 1$ and $1/q + 1/q^* = 1$,

$$\|\mathbf{A}\|_{p\to q} \;=\; \|\mathbf{A}^\top\|_{q^*\to p^*}. \tag{19}$$

Specializing to $p = 2$, $q = \infty$ (so $p^* = 2$ and $q^* = 1$) gives

$$\|\mathbf{A}\|_{2\to\infty} \;=\; \|\mathbf{A}^\top\|_{1\to 2}, \tag{20}$$

i.e., the `lm_head` norm of $\mathbf{L}$ equals the `embed` norm of $\mathbf{L}^\top$.

Consequence under weight tying. In language models that tie the embedding and output weights ($\mathbf{L} = \mathbf{E}^\top$) (Press & Wolf, 2017; Inan et al., 2016), Remark 3.8 reduces to the tautology $\|\mathbf{E}^\top\|_{2\to\infty} = \|\mathbf{E}\|_{1\to 2}$. By Corollary 2.3, column normalization of $\mathbf{E}$ is exactly row normalization of $\mathbf{L} = \mathbf{E}^\top$, so MuonIO acts once on the shared parameters. The next subsection expresses the same rule for untied weights.

3.4 ONE MUONIO UPDATE

Put either vocabulary-facing matrix into the canonical row-oriented form

$$\mathbf{W} = \begin{cases} \mathbf{E}^\top \in \mathbb{R}^{V\times d}, & \text{for the embedding table,} \\ \mathbf{L} \in \mathbb{R}^{V\times d}, & \text{for the language-model head.} \end{cases} \tag{21}$$

By (20), both layer penalties become the same $2 \to \infty$ norm in this orientation. By Proposition 2.2, this is the maximum 2-norm of a row. Thus both layers use the same local problem

$$\Delta\mathbf{W}^\star \in \arg\min_{\Delta\mathbf{W}} \langle \mathbf{G}_k, \Delta\mathbf{W}\rangle + \frac{1}{2\lambda}\|\Delta\mathbf{W}\|^2_{2\to\infty}, \qquad \mathbf{G}_k = \nabla\mathcal{L}(\mathbf{W}_k). \tag{22}$$

Corollary 2.3 now applies directly, as the unsmoothed solution row-normalizes $\mathbf{G}_k$ and scales it by $\lambda\|\mathbf{G}_k\|^\dagger_{2\to\infty}$. In the original storage orientation, the same rule normalizes columns of $\mathbf{E}$ and rows of $\mathbf{L}$. Algorithm 1 substitutes the momentum estimate for the raw gradient and adds the usual numerical safeguard as well as an additional scaling factor $\sqrt{d}$ so that each vector in $\mathbb{R}^d$ has 2-norm $O(\sqrt{d})$.

Algorithm 1 IO for embedding table or language model head

input row-oriented $\mathbf{W}_0 \in \mathbb{R}^{V\times d}$ from (21), step size λ, momentum β, loss $\mathcal{L}$, $\varepsilon > 0$
initialize $\mathbf{M}_0 \leftarrow \mathbf{0}$, $k \leftarrow 0$

1: **while** $\mathbf{W}_k$ not converged **do**
2: Obtain $\tilde{\mathbf{G}}_k$ such that $\mathbb{E}[\tilde{\mathbf{G}}_k] = \nabla\mathcal{L}(\mathbf{W}_k)$
3: $\mathbf{M}_{k+1} \leftarrow \beta\mathbf{M}_k + (1-\beta)\tilde{\mathbf{G}}_k$
4: $(\mathbf{U}_{k+1})_{j,:} \leftarrow \dfrac{(\mathbf{M}_{k+1})_{j,:}}{\max\{\|(\mathbf{M}_{k+1})_{j,:}\|_2, \varepsilon\}}$ for every $j \in [V]$
5: $s_{k+1} \leftarrow \sum_{j=1}^{V} \|(\mathbf{M}_{k+1})_{j,:}\|_2$
6: $\mathbf{W}_{k+1} \leftarrow \mathbf{W}_k - \lambda s_{k+1}\sqrt{d}\,\mathbf{U}_{k+1}$
7: $k \leftarrow k+1$
8: **end while**
9: **return** $\mathbf{W}_k$

4 EXPERIMENTS

Our Algorithm 1 acts as a modular component that can be naturally attached to any Muon-style optimizer that treats the I/O layers with AdamW. To demonstrate the effectiveness of our approach, we incorporate IO into Muon (Jordan et al., 2024) and Polar Express (Amsel et al., 2026), yielding the corresponding optimizers MuonIO and Polar Express IO. We also include the recent method Ember (Shah, 2026) as an additional baseline, as it also provides alternative treatment for the I/O layers.

We evaluate these algorithms along three axes: validation perplexity, optimizer arithmetic cost and optimizer state memory. We additionally report various norms of the gradient for the I/O layers to offer a complementary view of the training dynamics. The central claim is that our geometry-matched update for the I/O layers improves pretraining perplexity while saving roughly 50% of the computation and persistant memory costs compared to AdamW.

4.1 EXPERIMENTAL SETUP

We evaluate all optimizers on autoregressive language model pretraining using the English C4 corpus (Raffel et al., 2020), a cleaned Common Crawl dataset released under the Open Data Commons Attribution License. We train LLaMA-family models from scratch at the 60M, 130M, and 1B scales (Touvron et al., 2023; Zhao et al., 2024), using untied input embeddings and language model heads. Text is tokenized with the T5-base tokenizer, and individual examples are truncated or padded to 256 tokens. Padding tokens are excluded from the training and evaluation losses. All

experiments use bfloat16 precision and an effective batch size of 512 sequences, with gradient accumulation adjusted according to the per-device batch size and GPU count. We use nominal training budgets (including padding positions) of approximately 1.15 billion tokens for the 60M and 130M models and 6.87 billion tokens for the 1B models. Learning rates increase linearly during the first 10% of training, followed by cosine decay to 10% of their peak values. Weight decay and gradient clipping are disabled. We evaluate on the held-out C4 validation split using approximately 10 million non-padding tokens per evaluation. Our primary metric is final validation perplexity, computed by exponentiating the average validation loss, with lower values indicating better language-modeling performance.

We organize the six optimizer configurations into two groups. The first consists of four configurations that use standard Muon for the hidden weight matrices while varying the treatment of the I/O layers. The second examines whether our I/O update extends to other Muon-style optimizers, using Polar Express as a representative example. Across all configurations, auxiliary parameters, including normalization gains and biases when present, are optimized with AdamW. Table 1 summarizes the six configurations, where η_a, η_h, η_i, and η_o denote learning rates. Within each configuration, repeated learning rate symbols indicate that the corresponding parameter groups share a common learning rate during hyperparameter tuning.

Table 1: Optimizer configurations used in our comparisons. Each entry specifies the optimizer and learning rate for the corresponding parameter group. Repeated learning rate symbols indicate that the same learning rate is shared across parameter groups. For each configuration, all distinct learning rates shown in the corresponding row are jointly tuned.

Configuration	Embedding		Hidden 2D		LM Head		Gains & Biases	
Plain Muon (baseline 1)	AdamW	η_a	Muon	η_m	AdamW	η_a	AdamW	η_a
Muon Tuned (baseline 2)	AdamW	η_i	Muon	η_m	AdamW	η_o	AdamW	η_a
Ember	Ember	η_i	Muon	η_m	Ember	η_o	AdamW	η_a
MuonIO (ours)	**IO**	η_i	Muon	η_m	**IO**	η_o	AdamW	η_a
Polar Express Tuned	AdamW	η_i	Polar Express	η_h	AdamW	η_o	AdamW	η_a
Polar Express IO (ours)	**IO**	η_i	Polar Express	η_h	**IO**	η_o	AdamW	η_a

4.2 Optimizer Performances

Table 2 reports final validation perplexity at all three model scales.

Table 2: Final validation perplexity on C4 (lower is better). The upper block uses standard Muon hidden updates, while the lower block uses Polar Express. Bold values indicate the lowest mean perplexity within each block at each model scale. For 60M and 130M, we report the mean and sample standard deviation over the three of five seeds with the lowest final validation perplexity. For 1B, we report the mean and sample standard deviation over three seeds.

Configuration	60M	130M	1B
Plain Muon (baseline 1)	36.963 ± 0.054	28.513 ± 0.026	15.116 ± 0.0003
Muon Tuned (baseline 2)	28.502 ± 0.015	24.238 ± 0.008	14.527 ± 0.007
Ember	28.835 ± 0.043	24.074 ± 0.004	14.516 ± 0.032
MuonIO (ours)	$\mathbf{28.366} \pm 0.025$	$\mathbf{23.617} \pm 0.025$	$\mathbf{14.374} \pm 0.132$
Polar Express Tuned	$\mathbf{28.385} \pm 0.009$	24.055 ± 0.010	14.361 ± 0.002
Polar Express IO (ours)	28.430 ± 0.025	$\mathbf{23.646} \pm 0.012$	$\mathbf{14.160} \pm 0.008$

With standard Muon hidden updates, MuonIO achieves the lowest mean validation perplexity among the four configurations at every model scale, including improvements over the strong baseline Muon

Tuned which separately tuned learning rates for all four components of the model. With Polar Express hidden updates, our algorithm lowers mean perplexity at 130M and 1B, while slightly increasing it at 60M. These results show that the improvements at the two larger scales also extend to the Polar Express setting.

Figure 1 complements the final perplexity comparison with validation loss trajectories for the four configurations that used Muon (top 4) for hidden 2D parameters at the 130M scale.

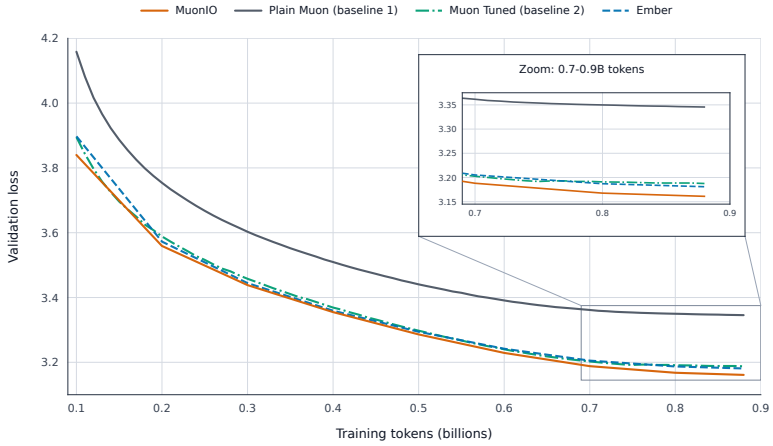

Figure 1: Validation loss trajectories on C4 at the 130M scale for Plain Muon (baseline 1), Muon Tuned (baseline 2), Ember and MuonIO. All four configurations use standard Muon hidden updates. The inset highlights the 0.7–0.9B training token range.

4.3 IO VERSUS ADAMW: EXACT OPTIMIZER COST

Table 3 compares Algorithm 1's exact per-update cost with AdamW on the same $V \times d$ matrix. It needs roughly half the FLOPs and, because it stores only momentum, half the optimizer state. Appendix C gives the counting convention and the operation-by-operation derivation.

Table 3: FLOPs and persistent optimizer state counts for one $V \times d$ I/O matrix and one optimizer update. Derivations are in Appendix C.

I/O-layer optimizer	FLOPs	Optimizer State
AdamW	$13Vd$	$2Vd$ entries
IO	$7Vd + 3V$	Vd entries

4.4 GRADIENT NORM DYNAMICS

To examine how different optimizers affect the training dynamics of the I/O layers, Figure 2 tracks the gradients of these layers during training of the 130M model. We compare three configurations: MuonIO, Muon Tuned, and Ember, all of which use standard Muon updates for the hidden weight matrices and differ only in their treatment of the I/O layers. We consider the gradient both as a matrix and as a flattened vector. From the matrix perspective, we measure its Schatten-2 and Schatten-∞ norms, corresponding to the Frobenius and spectral norms, respectively. From the vector perspective, we consider the ℓ_2 and ℓ_∞ norms. Since the vectorized ℓ_2 norm is identical to the Frobenius norm, this yields three distinct quantities: the Frobenius norm, spectral norm, and maximum absolute entry. These norms capture complementary aspects of the gradient magnitude and serve as diagnostics of training progress on the I/O layers.

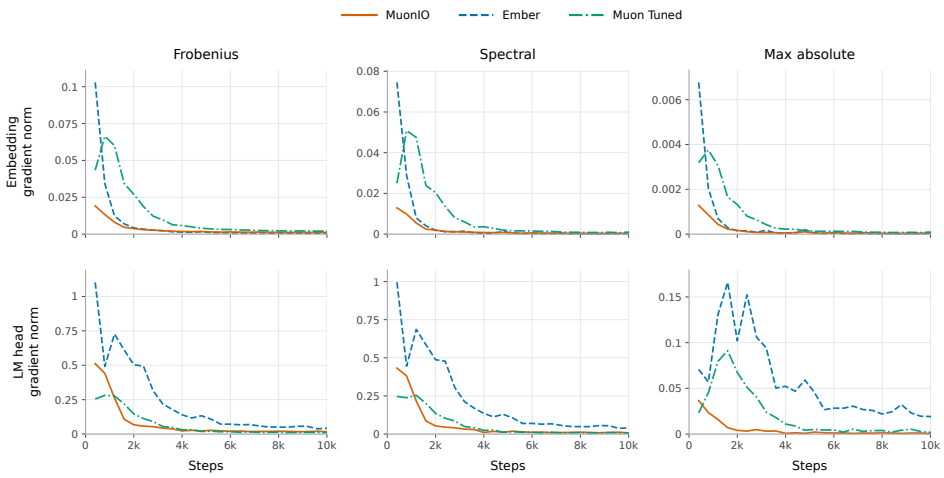

Figure 2: **Gradient magnitudes during 130M model training.** Rows show the input embedding table (top) and language model head (bottom); columns show the Frobenius norm, spectral norm, and maximum absolute gradient entry. Curves correspond to MuonIO (orange, solid), Ember (blue, dashed), and Muon Tuned (green, dash-dotted). The horizontal axis counts microbatch steps, with eight microbatches per optimizer update. Each curve represents one run using the best set of hyperparamters of the corresponding optimizer.

Across all three metrics, MuonIO drives the I/O gradients down more rapidly than the competing configurations. For the embedding table, MuonIO already exhibits substantially smaller gradient norms at the earliest recorded measurements and quickly approaches near-zero values, whereas Muon Tuned shows an initial increase before decaying and Ember starts with considerably larger gradients. The difference is even more pronounced for the language-model head: MuonIO rapidly reduces all three gradient measures within the first few thousand steps, while Muon Tuned decays more gradually and Ember maintains substantially larger gradients throughout much of the displayed interval. The maximum absolute entry metric shows the clearest separation, with Ember exhibiting large and persistent individual gradient coordinates while MuonIO quickly suppresses them. Overall, these trajectories indicate that, under MuonIO, the embedding and language model head gradients contract substantially earlier in training, particularly for the language-model head.

5 Conclusion

We have presented MuonIO, one Muon-style update for the two layer types that the original Muon recipe and many implementations route to AdamW: the input embedding table and the output language model head. For the embedding table $\mathbf{E} \in \mathbb{R}^{d\times V}$, we adopt the established $1 \to \mathrm{RMS}$ geometry induced by one-hot lookup (Bernstein & Newhouse, 2024; 2025), or equivalently $1 \to 2$ up to the fixed $d^{-1/2}$ RMS factor. For the output head $\mathbf{L} \in \mathbb{R}^{V\times d}$, our total variation stability analysis of softmax identifies the $2 \to \infty$ geometry. The resulting algorithm MuonIO is also cheaper than AdamW for vocabulary-facing layers in both optimizer state memory and FLOPs. Our numerical experiments further show that MuonIO improves final loss and perplexity.

6 Limitations and Future Work

Our analysis of `lm_head` measures perturbations of the output distribution using total variation and controls them through the $2 \to \infty$ norm. Although this formulation provides a sharp uniform bound and leads to a computationally cheap row-normalization update, it does not explicitly account for the invariance of softmax to constant shifts of all logits. Future work could investigate shift-invariant formulations as well as other measures of distributional change, such as KL divergence, and evaluate whether their optimization benefits justify any additional computational cost.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing, code development and debugging, and literature search. All technical content, mathematical arguments, experimental design, and conclusions were developed and verified by the authors. The authors take full responsibility for the accuracy and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

We specify the IO update in Algorithm 1 and present its theoretical derivation in Section 3, with proofs of the tightness results in Appendix B. Section 4.1 describes the dataset, preprocessing, training and evaluation protocols, and optimizer configurations. Appendix D provides model architectures, hyperparameter search procedures, selected hyperparameters, and batching settings. The caption of Table 2 specifies the seed-selection procedure and summary statistics used to report perplexity. Appendix C details the FLOP and persistent optimizer-state accounting.

REFERENCES

Kwangjun Ahn, Noah Amsel, and John Langford. Dion2: A simple method to shrink matrix in muon. *arXiv preprint arXiv:2512.16928*, 2025a.

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025b.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yRtgZ1K8hO.

Jeremy Bernstein. Deriving muon, 2025. URL https://jeremybernste.in/writing/deriving-muon.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In *OPT 2024: Optimization for Machine Learning*, 2024. URL https://arxiv.org/abs/2409.20325. arXiv:2409.20325.

Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. In *International Conference on Machine Learning*, pp. 3920–3930. PMLR, 2025.

Athanasios Glentis, Jiaxiang Li, Andi Han, and Mingyi Hong. Memory-efficient LLM pretraining via minimalist optimizer design. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2506.16659. arXiv:2506.16659.

Soufiane Hayou and Liyuan Liu. Optimal embedding learning rate in llms: The effect of vocabulary size. *arXiv preprint arXiv:2506.15025*, 2025.

Di He, Songjun Tu, Keyu Wang, Lu Yin, and Shiwei Liu. One LR doesn't fit all: Heavy-tail guided layerwise learning rates for LLMs. In *ICLR 2026 2nd Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2026. URL https://openreview.net/forum?id=Aj3ZWgxYwt.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

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

Dayal Singh Kalra and Maissam Barkeshli. Quantifying hyperparameter transfer and the importance of embedding layer learning rate. *arXiv preprint arXiv:2605.21486*, 2026.

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

Tim Tsz-Kit Lau and Weijie Su. Symmetry-compatible principle for optimizer design: Embeddings, LM heads, SwiGLU MLPs, and MoE routers. *arXiv preprint arXiv:2605.18106*, 2026. URL https://arxiv.org/abs/2605.18106.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=m1IRWFAMsa.

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*, 2025.

Pravin Nair. Softmax is $1/2$-lipschitz: A tight bound across all ℓ_p norms. *arXiv preprint arXiv:2510.23012*, 2025. URL https://arxiv.org/abs/2510.23012.

Tianyu Pang, Yujie Fang, Zihang Liu, Shenyang Deng, Lei Hsiung, Shuhua Yu, and Yaoqing Yang. Htmuon: Improving muon via heavy-tailed spectral correction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36504–36535, 2026.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. In *International Conference on Machine Learning*, pp. 49069–49104. PMLR, 2025.

Tiberiu Popoviciu. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica*, 9 (129-145):20, 1935.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, 2017.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Kathan Shah. Token geometry. *arXiv preprint arXiv:2607.01455*, 2026. URL https://arxiv.org/abs/2607.01455.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

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*, pp. 61121–61143. PMLR, 2024.

A Related Work

The original Muon (Jordan et al., 2024) optimizer uses approximate polar factor updates for hidden linear layers and assigns embeddings and language model heads to AdamW. Bernstein and Newhouse (Bernstein & Newhouse, 2025) already derive the embedding geometry used by MuonIO: their $1 \to \mathrm{RMS}$ norm formulation, equivalent to $1 \to 2$ up to a dimension-dependent factor, gives an update that normalizes each column, though this choice is not empirically evaluated. Scion's (Pethick et al., 2025) operator norm LMO framework includes column normalization for one-hot inputs and row normalization as an $\mathrm{RMS} \to \infty$ output option, which is motivated by control of the largest output coordinate rather than softmax stability. Separately, Nair (2025) establishes softmax Lipschitz bounds under matched ℓ_p norms. Our analysis instead uses the mixed $\ell_\infty \to \ell_1$ relation to motivate the $2 \to \infty$ norm for a language model head through total variation stability.

Several recent optimizers use related normalization directions or otherwise give special treatment to token-facing matrices. SCALE applies column normalization to every layer and uses momentum only for the final layer (Glentis et al., 2026). After transposition, its final layer normalization resembles one component of MuonIO, but SCALE is otherwise a distinct algorithm motivated by observed gradient variance and token frequency imbalance. By contrast, Lau & Su (2026) use a different principle. They require updates to be equivariant to vocabulary reordering and orthogonal transformations of hidden space. The resulting family includes the token normalization used here. For untied heads, their rule also centers momentum, removes common logit shifts, and chooses scale separately. Ember (Shah, 2026) also treats embeddings and language model heads specially, but follows a different formulation and algorithm using factorized second moments rather than a Muon-style update.

B Proofs

Proof of Proposition 3.3. First suppose V is even. Take $\mathbf{p}_i = \frac{1}{V}$ for all i (i.e., $\mathbf{z} = \mathbf{0}$). Set $\mathbf{u}_i = 1$ when i is even and $\mathbf{u}_i = -1$ when i is odd. Starting from $\|J(\mathbf{z})\mathbf{u}\|_1 = \sum_{i=1}^{V} \mathbf{p}_i |\mathbf{u}_i - m|$, we have $m = 0, |\mathbf{u}_i - m| = 1 \; \forall i$, hence $\|J(\mathbf{z})\mathbf{u}\|_1 = \sum_{i=1}^{V} \mathbf{p}_i |\mathbf{u}_i - m| = \sum_{i=1}^{V} \frac{1}{V} = 1$.

When V is odd, a uniform $\mathbf{p}$ cannot place mass exactly $\frac{1}{2}$ on the coordinates with $\mathbf{u}_i = +1$, so we double the mass of one coordinate instead: take $\mathbf{z} = (0, \ldots, 0, \ln 2)$, so that

$$\mathbf{p} = \left(\tfrac{1}{V+1}, \ldots, \tfrac{1}{V+1}, \tfrac{2}{V+1} \right).$$

Choose $\mathbf{u} \in \{\pm 1\}^V$ with $\mathbf{u}_V = +1$ and exactly $\frac{V-3}{2}$ of the first $V-1$ coordinates equal to $+1$ (an integer since V is odd). The mass on the set $\{i : \mathbf{u}_i = +1\}$ is then

$$\frac{2}{V+1} + \frac{(V-3)/2}{V+1} = \frac{4 + V - 3}{2(V+1)} = \frac{1}{2},$$

so again $m = 0$ and $|\mathbf{u}_i - m| = 1$ for all i, whence $\|J(\mathbf{z})\mathbf{u}\|_1 = \sum_{i=1}^{V} \mathbf{p}_i = 1$. (Conceptually, this is the even case with $V+1$ uniform coordinates, two of which — sharing the sign $+1$ — are merged into a single coordinate of double mass.) □

Proof of Proposition 3.5. *Step 1: the supremum equals* 1. Assume first that V is even and let $\mathbf{u} \in \{\pm 1\}^V$ alternate in sign as in Proposition 3.3. For $\varepsilon > 0$ set $\mathbf{z} = -\frac{\varepsilon}{2}\mathbf{u}$ and $\mathbf{z}' = \frac{\varepsilon}{2}\mathbf{u}$, so that $\|\mathbf{z}' - \mathbf{z}\|_\infty = \varepsilon$. Along the line $\mathbf{z}(t) = t\mathbf{u}$, the softmax takes only two distinct values: $p_i(t) = \frac{2}{V} \cdot \frac{e^t}{e^t + e^{-t}}$ on the $V/2$ coordinates with $u_i = +1$, and $p_i(t) = \frac{2}{V} \cdot \frac{e^{-t}}{e^t + e^{-t}}$ on the rest. Summing the coordinatewise differences between $t = \varepsilon/2$ and $t = -\varepsilon/2$ gives

$$\|\mathrm{softmax}(\mathbf{z}') - \mathrm{softmax}(\mathbf{z})\|_1 = 2\,\frac{e^{\varepsilon/2} - e^{-\varepsilon/2}}{e^{\varepsilon/2} + e^{-\varepsilon/2}} = 2\tanh(\varepsilon/2), \tag{23}$$

so the ratio in (15) equals $2\tanh(\varepsilon/2)/\varepsilon \to 1$ as $\varepsilon \downarrow 0$. For V odd, apply the same construction to the first $V-1$ coordinates and set the last coordinate of both $\mathbf{z}$ and $\mathbf{z}'$ to $-M$; letting $M \to \infty$ recovers the same limit. Combined with the upper bound of Lemma 3.4, the supremum equals 1.

Step 2: the supremum is not attained. Suppose equality held in Lemma 3.4 for some finite $\mathbf{z}' \neq \mathbf{z}$. Write $\mathbf{u} = \mathbf{z}' - \mathbf{z}$, let $\gamma(t) = \mathbf{z} + t\mathbf{u}$ for $t \in [0,1]$ be the segment from the proof of Lemma 3.4, and let $\mathbf{p}(t) := \mathrm{softmax}(\gamma(t)) \in \Delta^V$ denote the softmax along this segment. Since $t \mapsto \|J(\gamma(t))\mathbf{u}\|_1$ is continuous and bounded above by $\|\mathbf{u}\|_\infty$, equality of the integral bound in the proof of Lemma 3.4 forces $\|J(\gamma(t))\mathbf{u}\|_1 = \|\mathbf{u}\|_\infty$ for every $t \in [0,1]$. Let X_t take value u_i with probability $p_i(t)$, as in the proof of Lemma 3.2. Equality in Jensen's inequality requires $|X_t - \mathbb{E}[X_t]|$ to be almost surely constant, and equality in Popoviciu's inequality requires X_t to be supported on $\{-\|\mathbf{u}\|_\infty, +\|\mathbf{u}\|_\infty\}$ with mass $\frac{1}{2}$ each. Together these give $m(t) := \mathbf{p}(t)^\top \mathbf{u} = \mathbb{E}[X_t] = 0$ and $\mathrm{Var}(X_t) = \|\mathbf{u}\|_\infty^2$ for every $t \in [0,1]$. Hence $m \equiv 0$ and so $m' \equiv 0$. But differentiating along the segment gives $m'(t) = \mathbf{u}^\top J(\gamma(t))\mathbf{u} = \mathrm{Var}(X_t) = \|\mathbf{u}\|_\infty^2 > 0$, which is a contradiction. Hence the inequality in Lemma 3.4 is strict for every finite pair, consistent with Step 1, where $2\tanh(\varepsilon/2) < \varepsilon$ strictly for every $\varepsilon > 0$. □

Proof of Proposition 3.7. We show that every logit pair in (15) is realized by an lm_head triple with the same ratio. Fix any nonzero $\mathbf{y} \in \mathbb{R}^d$. For $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^V$ with $\mathbf{z}' \neq \mathbf{z}$, define the rank-one matrices

$$\mathbf{L} = \frac{\mathbf{z}\,\mathbf{y}^\top}{\|\mathbf{y}\|_2^2}, \qquad \Delta\mathbf{L} = \frac{(\mathbf{z}' - \mathbf{z})\,\mathbf{y}^\top}{\|\mathbf{y}\|_2^2}, \tag{24}$$

so that $\mathbf{L}\mathbf{y} = \mathbf{z}$ and $(\mathbf{L} + \Delta\mathbf{L})\mathbf{y} = \mathbf{z}'$. The t-th row of $\Delta\mathbf{L}$ is $(\mathbf{z}' - \mathbf{z})_t\,\mathbf{y}/\|\mathbf{y}\|_2^2$, whose ℓ_2 norm is $|(\mathbf{z}' - \mathbf{z})_t|/\|\mathbf{y}\|_2$. Proposition 2.2 then gives

$$\|\Delta\mathbf{L}\|_{2\to\infty}\,\|\mathbf{y}\|_2 = \|\mathbf{z}' - \mathbf{z}\|_\infty. \tag{25}$$

Consequently, the ratio in (18) evaluated at $(\mathbf{L}, \Delta\mathbf{L}, \mathbf{y})$ equals the ratio in (15) evaluated at $(\mathbf{z}, \mathbf{z}')$. Taking the supremum over $\mathbf{z}' \neq \mathbf{z}$ and invoking Proposition 3.5, the supremum in (18) is at least 1; by Lemma 3.6 it is at most 1.

For non-attainment, suppose some triple $(\mathbf{L}, \Delta\mathbf{L}, \mathbf{y})$ with $\Delta\mathbf{L} \neq \mathbf{0}$, $\mathbf{y} \neq \mathbf{0}$ attained equality in (18). If $(\Delta\mathbf{L})\mathbf{y} = \mathbf{0}$, the left hand side of Lemma 3.6 vanishes while the right hand side is positive, so equality fails. Otherwise, equality forces in particular $\|\mathrm{softmax}(\mathbf{z}') - \mathrm{softmax}(\mathbf{z})\|_1 = \|\mathbf{z}' - \mathbf{z}\|_\infty$ for the finite pair $\mathbf{z} = \mathbf{L}\mathbf{y} \neq \mathbf{z}' = (\mathbf{L} + \Delta\mathbf{L})\mathbf{y}$, contradicting Step 2 of the proof of Proposition 3.5. □

Proof of Remark 3.8. Using the dual norm characterization $\|\mathbf{y}\|_q = \sup_{\|\mathbf{v}\|_{q^*} \le 1} \langle \mathbf{v}, \mathbf{y} \rangle$ and bilinearity of the inner product,

$$\begin{aligned}
\|\mathbf{A}\|_{p\to q} &= \sup_{\|\mathbf{x}\|_p \le 1} \|\mathbf{A}\mathbf{x}\|_q \\
&= \sup_{\|\mathbf{x}\|_p \le 1} \sup_{\|\mathbf{v}\|_{q^*} \le 1} \langle \mathbf{v}, \mathbf{A}\mathbf{x} \rangle \\
&= \sup_{\|\mathbf{v}\|_{q^*} \le 1} \sup_{\|\mathbf{x}\|_p \le 1} \langle \mathbf{A}^\top \mathbf{v}, \mathbf{x} \rangle \\
&= \sup_{\|\mathbf{v}\|_{q^*} \le 1} \|\mathbf{A}^\top \mathbf{v}\|_{p^*} \\
&= \|\mathbf{A}^\top\|_{q^* \to p^*}.
\end{aligned}$$

Specializing to $p = 2$, $q = \infty$ (so $p^* = 2$, $q^* = 1$) gives (20). □

C Exact FLOP and Optimizer-State Accounting

This appendix derives the counts reported in Table 3. Let one input or output parameter matrix have shape $V \times d$ (or its transpose) and let $N := Vd$ be its number of entries. We count FLOPs in the following way. One scalar addition, subtraction, multiplication, division, comparison or square root counts as one FLOP; consequently, a fused multiply–add counts as two FLOPs. Operations that have in total $O(1)$ FLOPS throughout one optimizer call are ignored.

AdamW. Denote the learning rate, weight decay and the betas with $\gamma, \lambda, \beta_1, \beta_2$. Let $\mathbf{M}_k$ and $\mathbf{V}_k$ denote the first and second moment buffers. Absorb the learning rate, bias corrections, and decoupled weight decay into scalars that are computed once per tensor. A bias-corrected AdamW update can then be organized into the four stages in Table 4. In the last two rows, $\mathbf{D}_k$ denotes the corrected square root denominator.

Table 4: AdamW tensor-arithmetic count for one matrix and one optimizer update. The elementwise product and division are denoted by $\odot$ and $\oslash$, respectively.

Stage	Tensor operation	FLOPs
First moment	$\mathbf{M}_k \leftarrow \frac{\beta_1}{1-\beta_1^t}\mathbf{M}_{k-1} + \frac{1-\beta_1}{1-\beta_1^t}\mathbf{G}_k$	$3N$
Second moment	$\mathbf{V}_k \leftarrow \frac{\beta_2}{1-\beta_2^t}\mathbf{V}_{k-1} + \frac{1-\beta_2}{1-\beta_2^t}\mathbf{G}_k \odot \mathbf{G}_k$	$4N$
Corrected denominator	$\mathbf{D}_k \leftarrow \sqrt{\mathbf{V}_k} + \varepsilon$	$2N$
Decay and adaptive step	$\mathbf{W}_{k+1} \leftarrow (1-\gamma\lambda)\mathbf{W}_k - \gamma(\mathbf{M}_k \oslash \mathbf{D}_k)$	$4N$
Total		$13N = 13Vd$

The first moment update uses two multiplications and one addition per entry. The second moment update additionally squares the gradient, giving three multiplications and one addition per entry. The corrected denominator uses one square root and one addition per entry. Finally, the parameter update uses two multiplications, one division, and one subtraction per entry. Thus the exact FLOP count is $13N$. AdamW keeps both $\mathbf{M}_k$ and $\mathbf{V}_k$, for $2N$ optimizer state entries.

MuonIO. We count one iteration of Algorithm 1 on the row-oriented matrix $\mathbf{W}_k \in \mathbb{R}^{V\times d}$ from (21). Write $r_j := \|(\mathbf{M}_{k+1})_{j,:}\|_2$ for the norm of row j. Each r_j is computed once and reused in both the normalization and the sum s_{k+1}. We then form a single per-row scale $c_j = a_{k+1}/\max\{r_j, \varepsilon\}$ with $a_{k+1} = \lambda\sqrt{d}\, s_{k+1}$. This fuses the normalization and scaling into the parameter update, so $\mathbf{U}_{k+1}$ is never materialized. Table 5 gives the resulting count.

Table 5: MuonIO tensor-arithmetic count for one matrix and one iteration of Algorithm 1. The constant $\lambda\sqrt{d}$ is computed once, before training.

Stage	Operation	FLOPs
Momentum	$\mathbf{M}_{k+1} \leftarrow \beta\mathbf{M}_k + (1-\beta)\tilde{\mathbf{G}}_k$	$3N$
Row norms	$r_j \leftarrow \|(\mathbf{M}_{k+1})_{j,:}\|_2,\ j \in [V]$	$V[d + (d-1) + 1] = 2N$
Sum norms	$s_{k+1} \leftarrow \sum_{j=1}^{V} r_j$	$V - 1$
Form scalar scale	$a_{k+1} \leftarrow (\lambda\sqrt{d})\, s_{k+1}$	1
Form row scales	$c_j \leftarrow a_{k+1}/\max\{r_j, \varepsilon\},\ j \in [V]$	$2V$
Fused parameter update	$(\mathbf{W}_{k+1})_{j,:} \leftarrow (\mathbf{W}_k)_{j,:} - c_j(\mathbf{M}_{k+1})_{j,:},\ j \in [V]$	$2N$
Total		$7N + 3V = 7Vd + 3V$

For each row of length d, the norm takes d squarings, $d-1$ additions, and one square root, or exactly $2d$ FLOPs. Summing over the V rows gives $2Vd = 2N$. Forming c_j costs one division and one comparison per row. The fused update, which equals line 6 of Algorithm 1, then costs one multiplication and one subtraction per matrix entry. The only persistent optimizer tensor is the momentum $\mathbf{M}_{k+1}$, so the optimizer state contains N entries.

D Experiment Settings and Hyperparameters

We tune optimizer hyperparameters on the C4 validation set, minimizing validation loss or perplexity. For the four-group and IO variants of Polar Express, we use Bayesian optimization with a maximum of 100 trials per sweep and logarithmic learning rate search ranges of $[0.005, 0.08]$ for

hidden weights, $[0.001, 0.3]$ for embeddings, $[0.0003, 0.2]$ for the output head, and $[0.0003, 0.03]$ for gains and biases. We search the Polar Express iteration count over $\{5, 6, 8\}$. For Muon, we perform a grid search over hidden weight learning rates $\{0.005, 0.01, 0.02\}$ and auxiliary learning rates $\{0.0003, 0.0005, 0.001\}$, since it has the same learning rates for hidden_gains_biases and nonhidden_params. For Muon Tuned, MuonIO and Ember, we independently search discrete embedding and output head learning rate grids spanning $[0.001, 0.3]$ and $[0.0003, 0.5]$, respectively, fixing the hidden weight learning rate to 0.02 and 0.04, respectively.

Table 6: Optimizer hyperparameters for the 18 selected LLaMA pre-training runs on C4. Values use three decimal places; nonzero values below 10^{-3} use scientific notation. μ_h is hidden weight momentum; (β_1^A, β_2^A) are Adam/AdamW coefficients; $\beta_{\mathrm{I/O}}$ is the separate coefficient for non-Adam I/O updates. PE denotes Polar Express.

Method	Size	η_h	η_i	η_o	η_a	μ_h	β_1^A	β_2^A	$\beta_{\mathrm{I/O}}$
Plain Muon	60M	0.010	3×10^{-4}	3×10^{-4}	0.001	0.950	0.900	0.950	–
	130M	0.010	3×10^{-4}	3×10^{-4}	0.001	0.950	0.900	0.950	–
	1B	0.019	3×10^{-4}	3×10^{-4}	0.023	0.950	0.950	0.950	–
Muon Tuned	60M	0.020	0.040	0.020	0.020	0.900	0.900	0.950	–
	130M	0.020	0.040	0.020	0.020	0.900	0.900	0.950	–
	1B	0.016	0.197	0.004	0.015	0.900	0.900	0.950	–
Ember	60M	0.040	0.100	0.020	0.020	0.900	0.900	0.950	0.900
	130M	0.040	0.100	0.010	0.020	0.900	0.900	0.950	0.900
	1B	0.033	0.273	0.002	0.017	0.900	0.900	0.950	0.900
MuonIO	60M	0.040	0.300	0.005	0.020	0.900	0.900	0.950	0.900
	130M	0.040	0.300	0.005	0.020	0.900	0.900	0.950	0.900
	1B	0.031	0.185	0.001	0.020	0.900	0.900	0.950	0.900
PE Tuned	60M	0.030	0.047	0.004	0.012	0.900	0.900	0.950	–
	130M	0.014	0.034	0.013	0.025	0.950	0.900	0.950	–
	1B	0.016	0.104	0.005	0.020	0.900	0.900	0.950	–
PE IO	60M	0.036	0.040	9.086×10^{-4}	0.026	0.900	0.900	0.950	0.900
	130M	0.028	0.161	3.199×10^{-4}	0.014	0.900	0.900	0.950	0.900
	1B	0.026	0.056	0.002	0.017	0.900	0.900	0.950	0.900

Parameter groups. Table 6 lists the selected optimizer hyperparameters. The learning rates η_h, η_i, η_o, and η_a correspond to hidden weights, embeddings, the language model head, and gains/biases, respectively. Muon Tuned denotes the Muon baseline with separately tuned I/O learning rates. PE Tuned and PE IO denote the corresponding Polar Express configurations.

Momentum and Adam coefficients. The coefficients (β_1^A, β_2^A) apply to gains/biases for every method and additionally to embeddings and the language model head for Plain Muon, Muon Tuned, and PE Tuned. For MuonIO and PE IO, $\beta_{\mathrm{I/O}}$ is the I/O momentum coefficient; their I/O updates use no second moment coefficient. For Ember, $\beta_{\mathrm{I/O}}$ is the I/O second moment decay coefficient; its I/O updates use no first moment coefficient. A dash in the table indicates that the I/O layers use the displayed Adam coefficient pair. Hidden weight momentum μ_h is specified separately.

Shared training settings. All runs use C4, a global batch size of 512 sequences, a sequence length of 256, and bfloat16 precision. The learning rate follows a cosine decay schedule to 0.1 of its peak value, with a warmup fraction of 0.1. The embedding and language model head weights are untied.

Batching. We describe each batching configuration by a tuple (microbatch size per GPU, gradient accumulation steps, number of GPUs), whose product is the global batch size of 512. At the 60M scale, runs use $(128, 4, 1)$, except Muon Tuned and MuonIO, which use $(64, 8, 1)$. At the 130M

scale, runs use $(64, 8, 1)$, except Plain Muon, which uses $(128, 4, 1)$. At the 1B scale, runs use $(64, 8, 1)$, except MuonIO, which uses $(64, 2, 4)$.

Model configuration. For the 60M, 130M, and 1B models, respectively, the hidden widths are $(512, 768, 2048)$, the numbers of transformer layers are $(8, 12, 24)$, the numbers of attention heads are $(8, 12, 32)$, and the feed-forward widths are $(1376, 2048, 5461)$. All models use an RMSNorm epsilon of 10^{-6} and a configured maximum position count of 2048, while the training sequence length is 256.