EQUIVARIANCE BREAKS THE LEARNING RATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Equivariant networks are commonly trained with Adam, yet recent work reports that matrix structured optimizers such as Muon can perform better, with the reasons for these gains only partly understood. We identify one source of this difference inside equivariant layers. An equivariant layer learns one channel mixing matrix W_l per degree l, which we call an irrep block, and shares it across the $2l + 1$ components, giving the expanded map $W_l \otimes I_{2l+1}$. This sharing sums gradient contributions across components and can produce different update scales under SGD. Adam's entrywise normalization reduces sensitivity to gradient scale, but neither optimizer directly controls the effective step size of each block. A single learning rate can therefore produce different effective step sizes across blocks. Muon instead controls the effective step size by approximately equalizing the singular values of each momentum matrix. We normalize each irrep block update by a single scalar, preserving its singular value ratios while letting the learning rate control its size. We implement this with spectral normalization or a simpler root-mean-square normalization. We evaluate spectral normalization in a controlled $\mathrm{SO}(3)$-equivariant model with a matched non-equivariant model. In this setting, the step size mismatch grows with width in the equivariant model but not in the non-equivariant model. We evaluate both variants across molecular force prediction on the rMD17 and MD22 datasets, QM9 molecular property prediction, and charged particle dynamics. Across these applications, block normalization generally improves Adam and closes part of its gap to Muon. These results highlight an overlooked interaction between equivariant architectures and their optimizers. Studying and designing the two together may help explain and address training difficulties often attributed to equivariance itself.

1 INTRODUCTION

Equivariant neural networks encode known geometric symmetries directly into their architecture, providing a strong inductive bias for learning from structured data (Thomas et al., 2018; Satorras et al., 2021; Bronstein et al., 2021). Equivariance is particularly important for physical sciences, where most models' predictions should not depend on the choice of coordinate frame, hence equivariant models have become widely used for molecular property prediction, atomistic simulation, and learned interatomic potentials (Aykent and Xia, 2025; Batzner et al., 2022; Batatia et al., 2022). Yet hard equivariance constraints can also make these models difficult to optimize, producing unfavorable loss geometry and sometimes underperforming less constrained alternatives at scale (Petrache and Trivedi, 2023; Xie and Smidt, 2025; Brehmer et al., 2025). Much recent work addresses this difficulty by modifying the architecture through approximate or relaxed equivariance (Wang et al., 2022; Pertigkiozoglou et al., 2024; Manolache et al., 2025; Elhag et al., 2025), while leaving the optimizer itself largely unchanged, even though the optimizer shapes both convergence and which solutions are reached (Pascanu et al., 2025).

Only recently has the optimizer choice become an explicit part of this discussion. Matrix structured optimizers such as Muon (Jordan et al., 2024) can outperform Adam (Kingma and Ba, 2015) on equivariant and geometric models (Stupariu and Manolache, 2026), although comparisons on interatomic potentials find that the gains depend on the architecture and optimizer (Harari et al., 2026). Why these optimizers behave differently on equivariant networks remains mostly unclear.

In this work, we study why matrix structured optimizers such as Muon can outperform Adam on equivariant networks. Parameter sharing can change SGD's update scales through gradient

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.

accumulation, an effect Adam largely compensates for. A separate issue remains at the matrix level: Adam rescales entries individually without controlling each block's effective step size, so a single learning rate can produce substantially different updates across irrep blocks within a layer. In a controlled setting, this step size mismatch grows with width in an equivariant model but not in a matched non-equivariant model.

Our contributions are as follows:

- We characterize two distinct effects of equivariant layer structure on optimization: the gradient scaling caused by irrep-dependent parameter sharing, which Adam largely compensates for, and a separate step size mismatch across irrep blocks that persists under Adam.
- We study spectral and root-mean-square normalization of optimizer updates, including their interaction with the learning rate and Adam's moment coefficients.
- We evaluate spectral normalization in a controlled $\mathrm{SO}(3)$-equivariant model with a matched non-equivariant model, and both normalizations in two interatomic potential architectures on rMD17 (Christensen and von Lilienfeld, 2020) and MD22 (Chmiela et al., 2023), molecular property prediction on QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014), and charged particle dynamics (Kipf et al., 2018; Satorras et al., 2021).

Block normalization often improves training without relaxing the model's equivariance constraints. These gains challenge the view that equivariant networks are intrinsically difficult to train and call for treating optimization as part of their design.

2 Related Work

Equivariance and training. How inputs and intermediate features transform under a symmetry determines which maps an equivariant layer can learn and where its weights are shared. Petrache and Trivedi (2023) quantify the approximation-generalization trade-off under approximate or partial equivariance, relating the symmetry of the model to that of the data. Xie and Smidt (2025) show that, under particular conditions, the geometry of an equivariant parameter space can prevent learning a global minimum. Nordenfors et al. (2025) compare equivariant architectures with data augmentation. Under suitable conditions, the two approaches share equivariant stationary points, although those points can differ in stability. One response has been to relax exact equivariance while retaining symmetry as an inductive bias. Soft priors and learned relaxations allow a model to accommodate imperfect symmetries (Finzi et al., 2021; Wang et al., 2022; van der Ouderaa et al., 2022). Relaxed equivariant layers can also represent symmetry breaking, including cases where an output should not preserve every symmetry of an individual input (Hofgard et al., 2024; Kaba and Ravanbakhsh, 2023). Other approaches relax the constraint during training or encourage equivariance through the objective (Pertigkiozoglou et al., 2024; Manolache et al., 2025; Elhag et al., 2025).

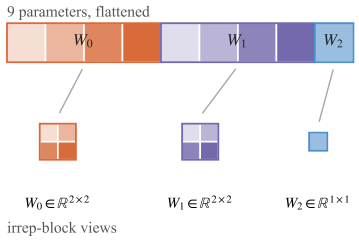
Optimizers for equivariant models. Harari et al. (2026) compare Adam (Kingma and Ba, 2015) with Muon (Jordan et al., 2024) and SOAP (Vyas et al., 2025) on equivariant interatomic potential models, finding strong results for SOAP and Muon. Li et al. (2026) train DPA4 with a hybrid optimizer that applies Muon separately to the channel mixing matrices of each degree and uses Adam updates for other parameters, while Chen et al. (2026) apply Muon to two-dimensional weights and Adam to the remaining parameters of equivariant protein fold classification models, reporting substantially improved convergence with Muon. Zhang (2026) introduces PolarAdamW, which applies Muon's polar transform to AdamW's preconditioned update, showing how a $\mathrm{SO}(3)$ model's updates respond to changes of basis among irrep channels. Stupariu and Manolache (2026) find that Adam and Muon reach different performance, loss geometry, and learned weights across equivariant and geometric models, showing how optimizers can shape the solutions reached (Pascanu et al., 2025). Optimizers such as Muon are becoming a more common choice for training equivariant networks. Much of the evidence comes from performance comparisons and observed differences in learned weights and loss geometry, which document the gains but leave their causes only partly understood.

Matrix update magnitude. Several works control the size of a matrix update during optimization. Yang et al. (2024) show that the spectral norms of weight matrices and their updates must scale appropriately with model width. Large et al. (2024) combine norms for components such as linear

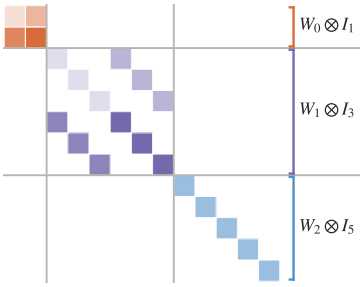
maps and convolutions into a network-wide norm used to rescale optimizer updates. Methods such as LARS (You et al., 2017) and LAMB (You et al., 2020) scale each layer's update by the ratio of its weight norm to the norm of the proposed update. Shazeer and Stern (2018) use the root-mean-square size of an update to limit large steps from old second moment estimates. These works study update size more generally, without examining the parameter structure imposed by equivariance. Accounting for this structure raises a more specific question: how should the effective step size of each irrep block be controlled within a layer? We first describe how equivariant layers organize and share parameters, then examine how this structure affects optimizer updates.

3 Background

Equivariance. A function $f : X \to Y$ is equivariant to a group G if $f(\rho_X(g)x) = \rho_Y(g)f(x)$ for all $g \in G$ and $x \in X$, where ρ_X and ρ_Y describe how the group acts on the input and output. Thus, transforming the input before applying f is equivalent to transforming its output afterward. Compositions of equivariant functions remain equivariant, allowing an equivariant network to be built one layer at a time. Convolutional networks are a standard example: sharing a filter across spatial positions makes convolution translation equivariant, so translating the input translates the resulting feature maps. In this work, we focus on $G = O(3)$, the group of rotations and reflections in three dimensions. An element of $O(3)$ transforms point coordinates and polar vector features by the same orthogonal matrix. Even scalar features remain unchanged, whereas pseudoscalars change sign under inversion. For physical predictions, energy is an invariant scalar, while forces are polar vectors that transform with the input coordinates.



(a) Flat storage and irrep-block views.



(b) The corresponding expanded map.

Figure 1: Stored and expanded views of an equivariant linear layer. Equal colors indicate tied weights.

Structure of equivariant layers. Many neural networks equivariant to $O(3)$ organize their features into irreducible representations, or irreps, as in e3nn (Geiger and Smidt, 2022) (e.g., through spherical harmonics (Varshalovich et al., 1988; Thomas et al., 2018)). An irrep of degree l has $2l + 1$ components that transform together, with $l = 0$ and $l = 1$ corresponding to scalars and vectors. The multiplicity m_l of an irrep counts its copies, or channels. For example, $m_1 = 2$ denotes two vector channels. For $O(3)$, an irrep also has a parity, which we suppress in the notation. The index l implicitly includes parity, and equivariant linear layers only mix channels with the same degree and parity. At each degree l, an equivariant linear layer learns channel mixing matrices $W_l \in \mathbb{R}^{m_l^{\text{out}} \times m_l^{\text{in}}}$. Equivariance requires the same matrix to act on all $2l + 1$ components, giving the expanded map $W_l \otimes I_{2l+1}$. We refer to this channel mixing matrix as an irrep block. e3nn additionally multiplies each irrep block by a constant determined by the number of input channels. We measure and normalize updates to W_l without this constant. Figure 1 shows a layer in its stored and expanded forms, omitting this additional scaling. The example has $m_0 = m_1 = 2$ and $m_2 = 1$ at both input and output, so W_0 and W_1 contain four parameters each, whereas W_2 contains only one. In the expanded map, each entry is repeated one, three, and five times, respectively.

4 Block Updates

Optimizer updates. Let $G_l^{(t)}$ denote the gradient of W_l at optimization step t. Stochastic gradient descent applies $\Delta W_l^{(t)} = -\eta_t G_l^{(t)}$, where η_t is the learning rate at step t. Equivariant implementations generally store the irrep blocks W_l and reuse their entries across irrep components, rather than storing the full operators $W_l \otimes I_{2l+1}$. The matrices may be stored separately or packed into a flat parameter vector, as in e3nn. Because the same weight is shared for all $2l + 1$ components, backpropagation sums their contributions to compute the gradient. Assuming that these contributions are identical, both the gradient and the SGD update are $2l + 1$ times larger than those from one compo-

nent alone. Blocks within one layer can therefore receive different effective step sizes simply because their parameters are shared different numbers of times. Normalizing each block update removes this multiplicative factor, as we describe in Section 4.1. Nevertheless, our main focus is Adam, which is commonly used to train equivariant networks. Unlike SGD, Adam can compensate for the increase in gradient magnitude caused by shared parameters. To see this, let g_s denote the contribution from one irrep component to the gradient of the shared weight $(W_l)_{ij}$ at step $s \in \{1, \dots, t\}$. Assume that at every training step s, all $2l+1$ components contribute the same value g_s to the gradient of the shared weight $(W_l)_{ij}$. The accumulated gradient is then $(2l+1)g_s$. Writing $\widehat{m}_t$ and $\widehat{v}_t$ for the bias-corrected moments of the gradient history, Adam's update becomes

$$\begin{aligned}\Delta W_{l,ij}^{(t)} &= -\eta_t \frac{\frac{1-\beta_1}{1-\beta_1^t}\sum_{s=1}^{t}\beta_1^{t-s}(2l+1)g_s}{\sqrt{\frac{1-\beta_2}{1-\beta_2^t}\sum_{s=1}^{t}\beta_2^{t-s}\big((2l+1)g_s\big)^2}+\varepsilon} \\ &= -\eta_t \frac{(2l+1)\widehat{m}_t}{(2l+1)\sqrt{\widehat{v}_t}+\varepsilon} \approx -\eta_t \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}}.\end{aligned}$$

The factor $2l+1$ cancels when the numerical stabilizer ε is negligible. Adam largely compensates for this gradient scaling, but still rescales weights individually rather than controlling each block's effective step size. Therefore, a single learning rate can still produce different effective step sizes across blocks. We examine this remaining mismatch next.

Effective step sizes. For an ordinary linear layer, the learning rate scales a single matrix update. An equivariant layer instead contains one irrep block W_l for each irrep type, all using the same learning rate. Let $U_l^{(t)}$ be the optimizer's proposed update. For any matrix norm, $\|\Delta W_l^{(t)}\|_\star = \eta_t \|U_l^{(t)}\|_\star$. Adam rescales the entries of $U_l^{(t)}$ independently, without using block boundaries or controlling the norm of the complete matrix. The norms can therefore differ across irrep blocks, so the same learning rate gives the blocks different effective step sizes $\|\Delta W_l^{(t)}\|_\star$. We call this the step size mismatch. Choosing η_t to keep the largest update stable may make the other updates unnecessarily small, while increasing it for those blocks may make the largest update unstable. Changing the learning rate cannot resolve this mismatch because it scales every block by the same factor. We test whether making the learning rate directly set each block's effective step size improves training.

4.1 Block Normalization

Block normalization. We normalize Adam's updates separately for each irrep block W_l so that the learning rate directly sets each block's effective step size. Adam proposes

$$U_l^{(t)} = \frac{\widehat{m}_l^{(t)}}{\sqrt{\widehat{v}_l^{(t)}}+\varepsilon},$$

where $\widehat{m}_l^{(t)}$ and $\widehat{v}_l^{(t)}$ contain the bias-corrected first and second moment estimates for the gradients of W_l, and all operations are entrywise. We then set

$$\Delta W_l^{(t)} = \begin{cases} -\eta_t \dfrac{U_l^{(t)}}{\|U_l^{(t)}\|_\star}, & U_l^{(t)} \neq 0, \\ 0, & U_l^{(t)} = 0, \end{cases}$$

where $\|\cdot\|_\star$ is the norm used for normalization. Every nonzero update therefore satisfies $\|\Delta W_l^{(t)}\|_\star = \eta_t$. This sets each nonzero block update's norm to the learning rate, while leaving Adam's moment-update rule unchanged and preserving the direction of each proposed block update.

Spectral normalization. We consider two choices of norm. The first is the spectral norm, which measures a matrix by its largest singular value:

$$\|U\|_2 = \sigma_{\max}(U).$$

Dividing by this value makes the largest singular value of each block update equal to the learning rate, while preserving its singular value ratios. For small irrep blocks, we compute $\sigma_{\max}$ by singular value decomposition in float64. For GotenNet's wider matrices, repeated squaring of the Gram matrix $U^\top U$ gives a high-order Schatten norm, an upper bound on the spectral norm (Horn and Johnson, 2012). Appendix A gives the approximation and its implementation details.

Table 1: Test loss on the synthetic task (mean $\pm$ std over eight seeds; lower is better). *spectral* applies spectral block normalization with Adam's default moment coefficients. *Transferred mom.* applies to unnormalized Adam the coefficients selected with normalization, while *both* combines those coefficients with normalization. Bold marks the best result per width and underline the second best. Spectral normalization outperforms Muon from width 128 onward.

Width	Adam	+ spectral	+ transferred mom.	+ both	Muon
16	0.675 ± 0.021	$\underline{0.654 \pm 0.031}$	0.705 ± 0.028	0.674 ± 0.017	$\mathbf{0.634 \pm 0.022}$
32	0.557 ± 0.022	0.564 ± 0.031	0.554 ± 0.019	$\underline{0.537 \pm 0.022}$	$\mathbf{0.513 \pm 0.017}$
64	0.449 ± 0.014	0.419 ± 0.019	0.479 ± 0.031	$\mathbf{0.376 \pm 0.017}$	$\underline{0.394 \pm 0.022}$
128	0.330 ± 0.017	$\underline{0.279 \pm 0.014}$	0.399 ± 0.021	$\mathbf{0.225 \pm 0.006}$	0.300 ± 0.018
192	0.265 ± 0.019	$\underline{0.220 \pm 0.017}$	0.257 ± 0.015	$\mathbf{0.156 \pm 0.008}$	0.249 ± 0.009
256	0.230 ± 0.008	$\underline{0.178 \pm 0.014}$	0.220 ± 0.008	$\mathbf{0.121 \pm 0.004}$	0.224 ± 0.010

Root-mean-square normalization. A simpler alternative uses the root-mean-square (RMS) norm,

$$\|U\|_{\mathrm{RMS}} = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} U_{ij}^2} = \frac{\|U\|_{\mathrm{F}}}{\sqrt{mn}},$$

for an $m \times n$ block. This makes the RMS of each block update equal to the learning rate without computing singular values. Its spectral norm, however, remains dependent on the update:

$$\|\Delta W\|_2 = \eta_t \sqrt{mn}\, \frac{\|U\|_2}{\|U\|_{\mathrm{F}}}.$$

For example, normalizing $U = I_k$ gives a spectral norm of $\eta_t\sqrt{k}$, whereas normalizing a $k \times k$ matrix of ones gives $\eta_t k$. These examples have the same RMS after normalization but different spectral norms. Spectral normalization controls the largest singular value, whereas RMS normalization controls the scale of the update entries. RMS normalization can improve training without equalizing the spectral norms of the block updates. Zero updates remain zero under either normalization.

5 Empirical Evaluation

We study how equivariant structure affects optimizer updates and whether controlling their effective step sizes improves training. Our experiments address four questions: ***RQ1:*** How do effective step sizes differ across irrep blocks, and how do these differences change with model width? ***RQ2:*** Does block normalization improve optimization and predictive performance across equivariant tasks? ***RQ3:*** How does normalization interact with the learning rate and Adam's moment coefficients? ***RQ4:*** Which parts of an equivariant architecture benefit from different optimizer choices?

Models and datasets. We first evaluate a toy controlled $\mathrm{SO}(3)$-equivariant model against a matched non-equivariant network, followed by three realistic applications. For molecular force prediction, we use a NequIP model (Batzner et al., 2022) implemented in e3nn (Geiger and Smidt, 2022), and GotenNet (Aykent and Xia, 2025) on the rMD17 (Christensen and von Lilienfeld, 2020) and MD22 (Chmiela et al., 2023) molecular datasets. We further evaluate GotenNet on the QM9 molecular properties dataset (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014), and a NequIP model on charged particle dynamics simulations (Kipf et al., 2018; Satorras et al., 2021).

Protocol. We compare Adam, spectral and root-mean-square block normalization, and Muon using the same data splits and number of epochs. We search learning rates and momentum coefficients for each method, selecting hyperparameters and checkpoints using validation performance. Runs use cosine learning rate decay and no weight decay. We report means and standard deviations over multiple seeds. Architectures, hyperparameter grids, seed counts, and implementation details are provided in Appendix sections B and C.

Table 2: Test force MAE (kcal/mol/Å, ↓) for NequIP on rMD17 and MD22. Results show the mean and std over three seeds. For each optimizer, we select learning rates, moment coefficients, and checkpoints using validation error. Bold: best result; underline: second best.

	Dataset	Adam	Spectral	RMS	Muon
rMD17	ethanol	0.366 ± 0.006	0.375 ± 0.002	$\underline{0.319 \pm 0.009}$	$\mathbf{0.262 \pm 0.001}$
	malonaldehyde	0.542 ± 0.029	0.545 ± 0.010	$\underline{0.474 \pm 0.015}$	$\mathbf{0.408 \pm 0.006}$
	benzene	0.159 ± 0.006	0.135 ± 0.004	$\underline{0.118 \pm 0.002}$	$\mathbf{0.094 \pm 0.003}$
	uracil	0.487 ± 0.019	0.473 ± 0.011	$\underline{0.382 \pm 0.015}$	$\mathbf{0.357 \pm 0.002}$
	toluene	0.489 ± 0.037	0.451 ± 0.007	$\underline{0.392 \pm 0.009}$	$\mathbf{0.360 \pm 0.004}$
	salicylic acid	0.642 ± 0.006	0.637 ± 0.008	$\underline{0.564 \pm 0.007}$	$\mathbf{0.478 \pm 0.008}$
	naphthalene	0.477 ± 0.023	0.421 ± 0.007	$\underline{0.356 \pm 0.009}$	$\mathbf{0.321 \pm 0.002}$
	paracetamol	0.681 ± 0.012	0.690 ± 0.008	$\underline{0.597 \pm 0.013}$	$\mathbf{0.542 \pm 0.009}$
	aspirin	0.817 ± 0.034	0.824 ± 0.002	$\underline{0.735 \pm 0.005}$	$\mathbf{0.679 \pm 0.005}$
	azobenzene	0.633 ± 0.028	0.622 ± 0.009	$\underline{0.524 \pm 0.012}$	$\mathbf{0.466 \pm 0.007}$
MD22	Ac-Ala3-NHMe	1.008 ± 0.090	0.962 ± 0.005	$\mathbf{0.867 \pm 0.019}$	$\underline{0.878 \pm 0.003}$
	stachyose	1.782 ± 0.055	1.478 ± 0.009	$\mathbf{1.386 \pm 0.016}$	$\underline{1.419 \pm 0.020}$

5.1 Synthetic dataset

Synthetic dataset. To answer ***RQ1***, we consider $\mathrm{SO}(3)$-invariant regression on 8192 training point clouds, each containing 16 random 3D points, with 512 validation and 2048 test point clouds. The task is to predict the largest eigenvalue of each cloud's second-moment matrix, a scalar unchanged by rotation. We compare a $\mathrm{SO}(3)$-equivariant network against a dense MLP with the same hidden dimension. We train widths 16, 32, 64, 128, 192, and 256 for 1000 steps, searching the learning rate, initialization scale, and moment coefficients over eight seeds. We compare Adam, Adam with spectral block normalization, Adam with tuned moment coefficients, their combination, and Muon.

Figure 2 measures the variation in the spectral norms of the block updates at Adam's first step. We compute

$$R = \frac{\max_{b \in \mathcal{B}_t} \|\Delta W_b^{(t)}\|_2}{\min_{b \in \mathcal{B}_t} \|\Delta W_b^{(t)}\|_2},$$

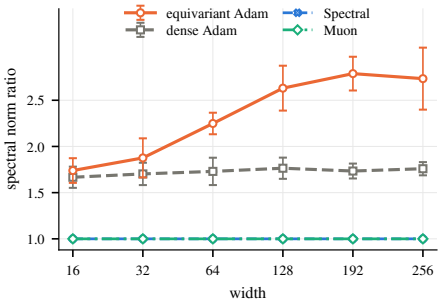


Figure 2: Ratio between the largest and smallest spectral norms of the matrix updates at Adam's first update. The mismatch increases with width in the equivariant model but remains nearly constant in the matched dense MLP. Spectral and Muon have a ratio of one by construction. Mean and standard deviations over eight seeds.

where $\mathcal{B}_t$ contains the nonzero matrix updates in all three hidden layers; input and readout matrices are excluded. We take the ratio over all hidden layers because the non-equivariant model has only one matrix per layer; Figure 7 in the appendix shows the ratio within each layer. Width m gives the equivariant model m scalar and m vector channels, matching the non-equivariant model at hidden dimension $4m$. In these architectures, R increases with width for the equivariant model but remains nearly constant for the non-equivariant model. This may make the learning rate harder to choose, since stabilizing the largest updates can slow the others. Exact spectral normalization and Muon's exact polar update give $R = 1$ by construction.

To examine whether removing the mismatch improves training, Table 1 compares the test losses of the best checkpoints. In the narrower models, spectral normalization performs similarly to Adam. As width increases, normalization provides larger improvements and eventually outperforms Muon. Combining normalization with tuned moment coefficients further improves the wider models, whereas moment tuning alone provides no consistent benefit across widths. Having found that normalization helps in this synthetic setting, we next evaluate it on more realistic models and datasets.

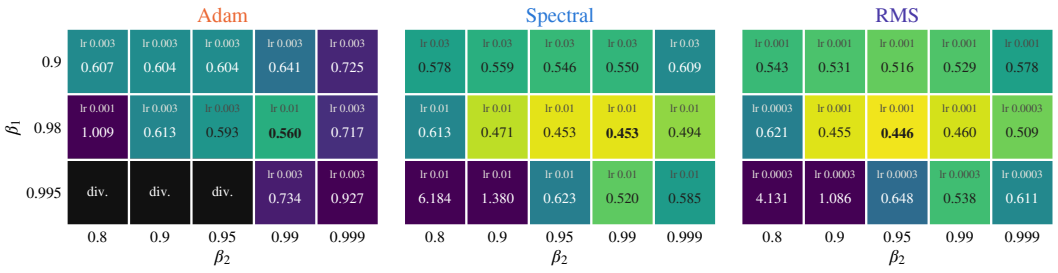
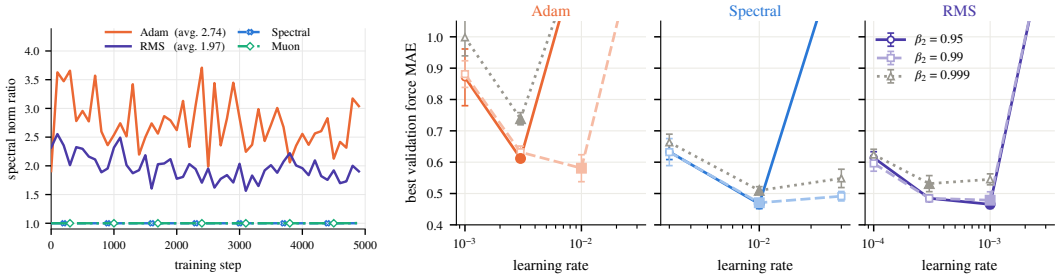


Figure 3: Test force MAE (kcal/mol/Å, ↓) for GotenNet on rMD17 aspirin across moment coefficients (β_1, β_2). Results are averaged over three seeds, with 3000 steps per run. Learning rates are selected using validation error and shown above each result. Bold marks the best configuration per method; black indicates divergence at all tested learning rates.



(a) Ratio of spectral norms.

(b) Validation force MAE (↓) against the learning rate at $\beta_1 = 0.98$.

Figure 4: Results on rMD17 aspirin. (a) Ratio between the largest and smallest spectral norms of irrep block updates; legend averages are over the displayed training steps. (b) GotenNet validation force MAE versus learning rate, with $\beta_1 = 0.98$ and varying β_2, averaged over three seeds. Curves leaving the plot indicate divergence at the next tested rate.

5.2 Interatomic Potentials

Effective step sizes during training. We now turn to molecular data to test whether the mismatch observed in the synthetic setting persists during training in interatomic potential models (***RQ1***). Computing the spectral norm by SVD for every block at every step becomes costly as the number and size of the layers increase. For GotenNet, we use the spectral norm approximation described in Section 4.1. We also evaluate RMS normalization as a simpler alternative that does not require computing singular values. Figure 4a plots the ratio between the largest and smallest spectral norms of the block updates during NequIP training on rMD17 aspirin. For Adam, this ratio remains high throughout training. Spectral normalization keeps it at one, while RMS normalization reduces the differences between blocks without fully equalizing their magnitudes.

Molecular force prediction. To answer ***RQ2***, we examine whether normalization improves prediction accuracy. For NequIP, Table 2 shows that RMS normalization achieves lower mean test force error than both Adam and spectral normalization on every dataset. It also gives the lowest errors on MD22, while Muon performs best on rMD17. The GotenNet results in Appendix Table 9 show a different pattern. Both normalizations consistently improve Adam, but Muon achieves the lowest error on every dataset, and spectral normalization is generally the stronger of the two. The preferred normalization therefore depends on the interatomic potential model, and we investigate potential reasons for Muon's remaining advantage later in this section.

Learning rate and moment coefficients. To answer ***RQ3***, we vary the learning rate and Adam's moment coefficients for GotenNet on rMD17 aspirin, training each model for 3000 steps. Figure 3 compares the resulting test errors, with the learning rate selected separately for each pair of moment coefficients. Both normalizations improve Adam at every tested pair. The models also remain trainable at several moment settings where Adam diverges, although some of these settings still

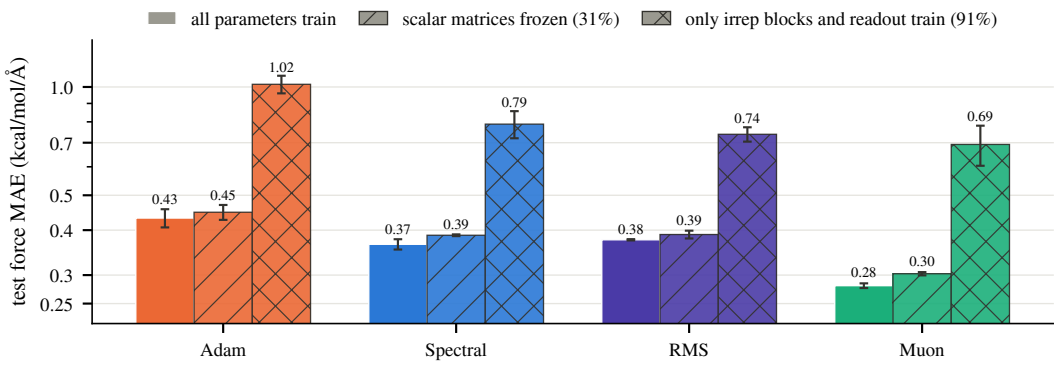


Figure 5: Test force MAE (kcal/mol/Å, ↓) for GotenNet on rMD17 aspirin with all parameters trained, scalar matrices frozen, or only the irrep blocks and readout trained. The number in parentheses is the percentage of parameters frozen at their initial values. Mean and std over three seeds.

produce high errors. The learning rate search in Figure 4b shows a similar difficulty for Adam: for some moment settings, fewer of the tested rates lead to stable training. Normalization therefore broadens the range of useful hyperparameter settings and makes the search less prone to divergent runs, making it easier to find well-performing models with a limited tuning budget.

Other base optimizers. Block normalization rescales the update proposed by the base optimizer, so it applies to SGD as well as to Adam. SGD with momentum accumulates gradients without Adam's entrywise rescaling, leaving its updates sensitive to the gradient scaling discussed in Section 4. Table 3 compares the two base optimizers on rMD17 aspirin with NequIP. Without normalization, SGD has substantially higher test error than Adam. Both spectral and RMS normalization nearly halve its error. Block normalization removes the gradient scaling from parameter sharing, which Adam removes implicitly through its second moment estimate, and with it the two base optimizers reach similar test errors.

Table 3: Test force MAE (kcal/mol/Å, ↓) on rMD17 aspirin with NequIP for Adam and SGD, with and without normalization. Subscripts give the standard deviation over three seeds.

Base	None	Spectral	RMS
Adam	$0.817_{\pm 0.034}$	$0.824_{\pm 0.002}$	$0.735_{\pm 0.005}$
SGD	$1.534_{\pm 0.019}$	$0.800_{\pm 0.008}$	$0.761_{\pm 0.008}$

Ablation study. Muon's larger advantage on GotenNet raises the possibility that it benefits ordinary scalar matrices more than irrep blocks (***RQ4***). We investigate this on rMD17 aspirin by freezing subsets of parameters at initialization. Figure 5 shows that freezing scalar matrices, including attention projections and the readout, leaves the optimizer differences largely unchanged. We next train only the irrep blocks and readout, freezing everything else. This includes the scalar networks that GotenNet uses to scale its steerable features, along with the atom embeddings and feature normalization parameters. Although all methods become less accurate, both normalizations retain their improvement over Adam. RMS normalization reaches a similar mean error to Muon, whose results vary more across seeds. Muon's larger advantage in the full model may depend on how the irrep blocks are optimized together with other components. We leave a closer study of these interactions to future work.

5.3 Molecular Properties and Particle Dynamics

Finally, we test whether the benefits of block normalization extend beyond molecular force prediction (***RQ2***). We consider invariant molecular property prediction on QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) and equivariant prediction of future positions in charged particle dynamics (Kipf et al., 2018; Satorras et al., 2021).

QM9. We train GotenNet for 200k steps to predict all twelve molecular properties jointly, using a shared representation and a separate prediction head for each target. Table 4 reports normalized

Table 4: Test MAE (↓) on QM9 with GotenNet, scaled by each target's training set standard deviation and multiplied by 100 (mean ± std over three seeds). Block normalization gives the lowest error on every target. Bold: best per row; underline: second best.

	Adam	Spectral	RMS
μ	2.095 ± 0.221	1.970 ± 0.050	$\mathbf{1.884 \pm 0.022}$
α	1.710 ± 0.040	1.649 ± 0.021	$\mathbf{1.632 \pm 0.048}$
$\varepsilon_{\mathrm{HOMO}}$	5.614 ± 0.136	5.394 ± 0.073	$\mathbf{5.347 \pm 0.041}$
$\varepsilon_{\mathrm{LUMO}}$	2.975 ± 0.058	$\mathbf{2.816 \pm 0.038}$	2.827 ± 0.036
$\Delta\varepsilon$	3.855 ± 0.059	$\mathbf{3.666 \pm 0.046}$	3.678 ± 0.052
$\langle R^2 \rangle$	0.522 ± 0.006	$\mathbf{0.462 \pm 0.014}$	0.563 ± 0.009
ZPVE	0.377 ± 0.014	0.381 ± 0.002	$\mathbf{0.363 \pm 0.005}$
U_0	0.469 ± 0.009	0.462 ± 0.003	$\mathbf{0.441 \pm 0.013}$
U	0.464 ± 0.002	0.457 ± 0.007	$\mathbf{0.441 \pm 0.013}$
H	0.459 ± 0.006	0.460 ± 0.003	$\mathbf{0.445 \pm 0.016}$
G	0.500 ± 0.007	0.496 ± 0.004	$\mathbf{0.476 \pm 0.016}$
c_v	1.411 ± 0.033	1.379 ± 0.032	$\mathbf{1.337 \pm 0.016}$
Average	1.704 ± 0.044	1.633 ± 0.021	$\mathbf{1.619 \pm 0.023}$

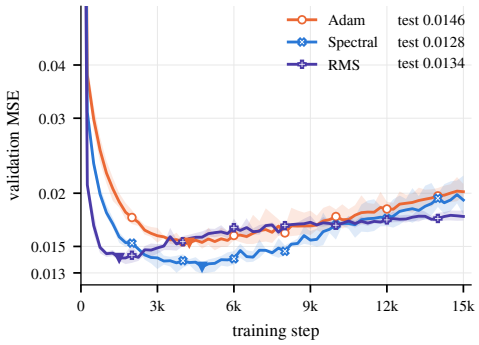


Figure 6: Mean and standard deviation of validation MSE (↓) for NequIP on charged particle dynamics. Triangles mark the minima of the mean curves. The legend reports mean test MSE at checkpoints selected per run using validation error. All models overfit before the first 5000 steps, with RMS converging faster but Spectral offering the best checkpoints.

test MAE ($\times 100$), allowing errors to be averaged across properties with different units. Both normalizations improve on Adam on average. RMS achieves the lowest mean error on most targets, while spectral normalization performs better on a smaller set. The clearest difference is for $\langle R^2 \rangle$, where spectral normalization reduces the error compared with Adam, whereas RMS increases it. Despite these differences across targets, the two methods reach similar average errors.

Particle dynamics. We use a NequIP model with velocity inputs to predict the future positions of five interacting charged particles, training each model for 30k steps. Both normalizations reduce test MSE compared with Adam. Figure 6 shows how the training dynamics differ: RMS reduces validation error more quickly at the start, while spectral normalization reaches lower validation error and test MSE after more updates. In this setting, faster initial progress does not translate into the best predictive performance. For all three methods, validation error eventually increases with continued training, so we only plot the first 15k steps.

6 Conclusion, Limitations and Future Work

We identified and analyzed a step size mismatch in equivariant layers. Parameter sharing can change SGD's step sizes through gradient accumulation. Adam reduces sensitivity to gradient scale, but its entrywise adaptation does not control the effective step size of each irrep block. Spectral and RMS normalization address this mismatch without introducing an additional hyperparameter. In the toy model, spectral normalization became more beneficial as width increased. Across molecular force prediction, QM9, and charged particle dynamics, block normalization generally improved Adam, although the preferred normalization depended on the model and task.

These findings connect the structure of equivariant layers to their optimization and show that this connection has practical consequences. Normalizing block updates improved training without changing the architecture or relaxing its equivariance constraints. This gives a reason to examine how optimizers handle equivariant structure before attributing training difficulties to equivariance itself. Muon remains the stronger optimizer on most datasets, so beyond the step size mismatch analyzed here, some other property of Muon must help equivariant networks.

Our evaluation focused on $O(3)$-equivariant models and two choices of normalization. Future work should examine whether different irrep blocks benefit from different target step sizes and how normalization can be combined with other optimizer mechanisms. The polar transformation studied in PolarAdamW (Zhang, 2026) provides one direction, while our freezing ablations point to the importance of studying irrep blocks together with attention, gating, and other components. Understanding these interactions could guide optimizers designed for equivariant architectures and extend this work to other symmetry groups and other forms of parameter sharing.

LLM USAGE DISCLOSURE

Large language models assisted with writing and debugging code, discussing experimental designs and results, searching related literature, drafting, reviewing, and polishing the manuscript, and checking implementations and mathematical derivations for possible errors. The authors reviewed LLM-generated code and ran numerical tests of the core implementations, including checks of update norms, preservation of update directions, optimizer moment calculations, and model equivariance. The mathematical identities were also checked through derivations and numerical tests. The research questions, central ideas, overall direction, and experimental protocol were determined by the authors, who made all final experimental decisions and decided which suggestions and interpretations to adopt. All AI-assisted work was reviewed by the authors, who take responsibility for the paper's content.

REPRODUCIBILITY STATEMENT

Section 4.1 defines block normalization, Algorithm 1 in Appendix B gives its implementation for e3nn, and Appendix A describes the spectral norm approximation used for GotenNet. Appendix C describes the model architectures, data splits and training budgets, and Tables 6 and 7 list the searched and selected hyperparameters of every experiment. Section 5 and Appendix C.1 fully specify the synthetic task. All other datasets are public: rMD17, MD22 and QM9 are used with the splits given in Appendix C, and the charged particle dynamics data come from the public generator of Satorras et al. (2021). We will release code, including the optimizer implementations, training scripts and the configuration files of all experiments, together with the final version of the paper.

REFERENCES

Sarp Aykent and Tian Xia. Gotennet: Rethinking efficient 3d equivariant graph neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5wxCQDtbMo.

Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gabor Csanyi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11423–11436. Curran Associates, Inc., 2022. doi: 10.52202/068431-0830. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4a36c3c51af11ed9f34615b81edb5bbc-Paper-Conference.pdf.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13, 05 2022. doi: 10.1038/s41467-022-29939-5.

Johann Brehmer, Sönke Behrends, Pim de Haan, and Taco Cohen. Does equivariance matter at scale?, 2025. URL https://arxiv.org/abs/2410.23179.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

Dexiong Chen, Andrei Manolache, Mathias Niepert, and Karsten Borgwardt. Protein fold classification at scale: Benchmarking and pretraining. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=jPKqiaPTEd.

Stefan Chmiela, Valentin Vassilev-Galindo, Oliver T. Unke, Adil Kabylda, Huziel E. Sauceda, Alexandre Tkatchenko, and Klaus-Robert Müller. Accurate global machine learning force fields for molecules with hundreds of atoms. *Sci. Adv.*, 9(2), January 2023. ISSN 2375-2548. doi: 10.1126/sciadv.adf0873.

Anders S Christensen and O Anatole von Lilienfeld. On the role of gradients for machine learning of molecular energies and forces. *Machine Learning: Science and Technology*, 1(4):045018, oct 2020. doi: 10.1088/2632-2153/abba6f. URL https://doi.org/10.1088/2632-2153/abba6f.

Ahmed A. A. Elhag, T. Konstantin Rusch, Francesco Di Giovanni, and Michael M. Bronstein. Relaxed equivariance via multitask learning. In *ICLR 2025 Workshop on Machine Learning for Genomics Explorations*, 2025. URL https://openreview.net/forum?id=8kZSO4WbTh.

Marc Finzi, Gregory Benton, and Andrew G Wilson. Residual pathway priors for soft equivariance constraints. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 30037–30049. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/fc394e9935fbd62c8aedc372464e1965-Paper.pdf.

William Fulton and Joe W. Harris. Representation theory: A first course. 1991.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022. URL https://arxiv.org/abs/2207.09453.

Gil Harari, Yoel Zimmermann, Ola Tangen Kulseng, Laura Zichi, Chuin Wei Tan, Marc L. Descoteaux, and Boris Kozinsky. Beyond adam: SOAP and muon for faster, label-efficient training of machine learning interatomic potentials. In *ICML 2026 AI for Science Workshop*, 2026. URL https://openreview.net/forum?id=mYxGJcckHl.

Elyssa Hofgard, Rui Wang, Robin Walters, and Tess Smidt. Relaxed equivariant graph neural networks. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024. URL https://openreview.net/forum?id=eVB1fn37Ay.

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012. ISBN 9781139788885. URL https://books.google.de/books?id=O7sgAwAAQBAJ.

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/.

Sékou-Oumar Kaba and Siamak Ravanbakhsh. Symmetry breaking and equivariant neural networks. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023. URL https://openreview.net/forum?id=d55JaRL9wh.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2693–2702, 2018.

Tim Large, Yang Liu, Minyoung Huh, Phillip Isola, Hyojin Bahng, and Jeremy Bernstein. Scalable optimization in the modular norm. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Tiancheng Li, Wentao Li, Anyang Peng, Jianming Xue, Linfeng Zhang, Duo Zhang, and Han Wang. Dpa4: Pushing the accuracy-cost frontier of interatomic potentials with emfa so(2) convolution, 2026. URL https://arxiv.org/abs/2606.02419.

Andrei Manolache, Luiz F. O. Chamon, and Mathias Niepert. Learning (approximately) equivariant networks via constrained optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=NM4emKloy6.

Oskar Nordenfors, Fredrik Ohlsson, and Axel Flinth. Optimization dynamics of equivariant and augmented neural networks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=PTTa3U29NR.

Razvan Pascanu, Clare Lyle, Ionut-Vlad Modoranu, Naima Elosegui Borras, Dan Alistarh, Petar Velickovic, Sarath Chandar, Soham De, and James Martens. Optimizers qualitatively alter solutions and we should leverage this, 2025. URL https://arxiv.org/abs/2507.12224.

Stefanos Pertigkiozoglou, Evangelos Chatzipantazis, Shubhendu Trivedi, and Kostas Daniilidis. Improving equivariant model training via constraint relaxation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tWkL7k1u5v.

Mircea Petrache and Shubhendu Trivedi. Approximation-generalization trade-offs under (approximate) group equivariance. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=DnO6LTQ77U.

R. Ramakrishnan, O. Dral, P., M. Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 2014. Nature.

L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, pages 2864–75, 2012. ACS.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9323–9332. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/satorras21a.html.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235, 2018. URL https://api.semanticscholar.org/CorpusID:4786918.

Teodor-Mihai Stupariu and Andrei Manolache. How the optimizer shapes learned solutions in equivariant neural networks. In *ICML 2026 Workshop on Weight-Space Symmetries: from Foundations to Practical Applications*, 2026. URL https://openreview.net/forum?id=ymiSLNQlkK.

Nathaniel Thomas, Tess E. Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *CoRR*, abs/1802.08219, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1802.html#abs-1802-08219.

Tycho F.A. van der Ouderaa, David W. Romero, and Mark van der Wilk. Relaxing equivariance constraints with non-stationary continuous filters. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=5oEk8fvJxny.

D A Varshalovich, A N Moskalev, and V K Khersonskii. *Quantum Theory of Angular Momentum*. WORLD SCIENTIFIC, 1988. doi: 10.1142/0270. URL https://www.worldscientific.com/doi/abs/10.1142/0270.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. In *International Conference on Machine Learning*. PMLR, 2022.

YuQing Xie and Tess Smidt. A tale of two symmetries: Exploring the loss landscape of equivariant models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=rH4aGTL4jY.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning, 2024. URL https://arxiv.org/abs/2310.17813.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017. URL https://arxiv.org/abs/1708.03888.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Haozhou Zhang. Polaradamw: Disentangling spectral control and schur gauge-equivariance in matrix optimisation, 2026. URL https://arxiv.org/abs/2605.07067.

A Additional Mathematical Details

Irrep features. An irreducible representation of $O(3)$ has degree l, parity, and $d_l = 2l + 1$ components. We suppress parity in the notation, but treat different parities as separate irrep types. Scalars have $l = 0$, vectors have $l = 1$, and higher degrees describe features with more components. With m_l channels of each type, the feature space is

$$\mathcal{V} = \bigoplus_l \left(\mathbb{R}^{m_l} \otimes \mathbb{R}^{d_l}\right).$$

The first factor indexes channels; the second contains the components transformed by rotations and reflections.

Equivariant layers. An equivariant linear layer mixes channels of the same irrep using the same coefficients for every component. To write this operation, let X_l contain one channel per row and one component per column. The layer computes

$$Y_l = W_l X_l,$$

where $X_l \in \mathbb{R}^{m_l^{\text{in}} \times d_l}$ and $W_l \in \mathbb{R}^{m_l^{\text{out}} \times m_l^{\text{in}}}$. Each coefficient in W_l therefore multiplies an entire channel, not an individual component.

If we arrange the channels consecutively in a vector, the same operation is represented by $W_l \otimes I_{d_l}$. The Kronecker product makes the sharing explicit by replacing each coefficient with a scaled identity matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes I_{d_l} = \begin{bmatrix} aI_{d_l} & bI_{d_l} \\ cI_{d_l} & dI_{d_l} \end{bmatrix}.$$

For example, bI_{d_l} adds the second input channel to the first output channel with coefficient b for every component. Schur's lemma (Fulton and Harris, 1991) requires this structure and prevents a linear layer from mixing different irrep types. The full weight matrix is therefore block diagonal:

$$A = \bigoplus_l \left(W_l \otimes I_{d_l}\right),$$

with one block per irrep type.

Vector example. Consider two vector channels and an irrep block W_1,

$$v_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \qquad v_2 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \qquad W_1 = \begin{bmatrix} 2 & -1 \\ 0.5 & 3 \end{bmatrix}.$$

The output vectors are $u_1 = 2v_1 - v_2$ and $u_2 = 0.5v_1 + 3v_2$, or equivalently

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 2I_3 & -I_3 \\ 0.5I_3 & 3I_3 \end{bmatrix}}_{W_1 \otimes I_3} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

The same four coefficients mix the x, y, and z components. Rotating both inputs by R gives $2Rv_1 - Rv_2 = R(2v_1 - v_2)$, and likewise for u_2. An arbitrary learned transformation of the three components would generally break this property.

Gradients of shared weights. The same matrix W_l acts on all d_l components, so backpropagation adds their contributions when computing its gradient. Let $D_l = \partial\mathcal{L}/\partial Y_l$ be the gradient at the layer's output. For component c, write $x_{l,c}$ for the corresponding column of input features and $\delta_{l,c}$ for the column of output gradients. Then

$$G_l = \frac{\partial \mathcal{L}}{\partial W_l} = D_l X_l^\top = \sum_{c=1}^{d_l} \delta_{l,c} x_{l,c}^\top.$$

Each component contributes one outer product, whose rank is at most one. For one application of the layer, this gives

$$\operatorname{rank}(G_l) \le \min\left(m_l^{\text{in}}, m_l^{\text{out}}, d_l\right).$$

A batch or repeated use of the layer sums several such gradients, so the accumulated gradient can have higher rank.

As discussed in Section 4, if every component contributes the same gradient, the sum is d_l times the contribution from one component. The SGD update grows by the same factor. Adam responds differently: if this scaling holds at every step, its first moment grows by d_l and its second moment by d_l^2. Taking the square root of the second moment leaves a factor d_l in both numerator and denominator, which cancels to $m/\sqrt{v}$. This argument also holds with bias correction and assumes a negligible numerical stabilizer.

Effective step sizes. SGD and Adam multiply each proposed block update U_l by the learning rate η_t. The resulting update $\Delta W_l = -\eta_t U_l$ therefore has norm $\|\Delta W_l\|_\star = \eta_t \|U_l\|_\star$. Blocks with larger proposed updates take larger steps, even though they use the same learning rate. Block normalization divides each proposal by its own norm before multiplying by η_t. Under this exact normalization rule, each nonzero block update has norm η_t.

We consider two choices for an $m \times n$ block:

$$\|U_l\|_2 = \sigma_{\max}(U_l), \qquad \|U_l\|_{\mathrm{RMS}} = \sqrt{\frac{1}{mn}\sum_{i,j}(U_l)_{ij}^2} = \frac{\|U_l\|_{\mathrm{F}}}{\sqrt{mn}}.$$

Here, $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. Spectral normalization sets the largest singular value of the update to η_t. RMS normalization instead sets the root-mean-square size of its entries to η_t, without computing singular values.

The spectral norm is useful because it bounds how much an update can change the block's output. For fixed input features X_l,

$$\|\Delta Y_l\|_{\mathrm{F}} = \|\Delta W_l X_l\|_{\mathrm{F}} \le \|\Delta W_l\|_2 \|X_l\|_{\mathrm{F}}.$$

This connects the update norm to its effect on the layer. It does not imply that equal update norms produce equal changes in the loss, since the inputs and surrounding network also matter. In our experiments, we test whether using a common update norm across blocks improves training.

Block normalization. We rescale each nonzero proposed block update $U_l^{(t)}$ as

$$\Delta W_l^{(t)} = -\eta_t \frac{U_l^{(t)}}{\|U_l^{(t)}\|_\star}, \qquad \star \in \{2, \mathrm{RMS}\}.$$

Zero updates remain zero. The resulting update has norm η_t, while retaining its direction and singular value ratios. The optimizer's moment update rule is unchanged.

Spectral norm approximation. For bigger blocks, we avoid SVD by estimating the largest singular value. First, we form $B = U^\top U$, or $B = UU^\top$ if that gives a smaller matrix. Its eigenvalues are the squared singular values of U. Squaring B twice gives the estimate

$$\widehat{\sigma}(U) = \|B^4\|_{\mathrm{F}}^{1/8} = \left(\sum_i \sigma_i(U)^{16}\right)^{1/16}.$$

The high power makes the largest singular values dominate the sum. This estimate is the Schatten-16 norm. For a nonzero matrix of rank r, it satisfies

$$\|U\|_2 \le \widehat{\sigma}(U) \le r^{1/16}\|U\|_2.$$

We divide the proposed update by this estimate:

$$\Delta W = -\eta_t \frac{U}{\widehat{\sigma}(U)}.$$

Because the estimate is an upper bound, the resulting spectral norm is at most η_t, rather than exactly η_t.

Relation to Muon. Muon combines gradients using momentum and then changes the singular values of the resulting matrix. Writing its reduced SVD as $M_l = P_l \Sigma_l Q_l^\top$, the exact polar update is

$$\Delta W_l^{\text{Muon}} = -\eta_t P_l Q_l^\top .$$

This replaces the singular values by η_t. Spectral and RMS normalization instead multiply the proposed update by one scalar, preserving the ratios between its singular values. Exact spectral normalization and the exact polar update therefore have the same largest singular value, but generally produce different update directions. RMS normalization fixes the size of the entries and doesn't necessarily share the spectral norm. Except on the synthetic task, Muon approximates the polar transformation with five Newton-Schulz iterations, so its singular values are not exactly equal.

B PyTorch Implementation

Algorithm 1 shows the block normalization pseudocode for e3nn linear weights. Adam computes its update and moment estimates as usual. We then split the flat update into irrep blocks, reshape each into a matrix, and rescale each separately. The code for the paper will be made publicly available together with the final version of the paper.

Algorithm 1: Block normalization for e3nn linear weights.

```
def irrep_blocks(model):
    for layer in model.modules():
        if not isinstance(layer, o3.Linear):
            continue

        offset, blocks = 0, []
        for instruction in layer.instructions:
            shape = tuple(instruction.path_shape)
            end = offset + shape[0] * shape[1]
            if end > offset:
                blocks.append((slice(offset, end), shape))
            offset = end

        yield layer.weight, blocks


@torch.no_grad()
def block_normalized_step(model, adam):
    specs = list(irrep_blocks(model))
    old = {id(weight): weight.clone()
           for weight, _ in specs}

    adam.step()

    for weight, blocks in specs:
        eta = current_learning_rate(adam, weight)
        update = weight - old[id(weight)]

        for indices, shape in blocks:
            matrix = update[indices].reshape(shape)
            norm = torch.linalg.matrix_norm(matrix, 2)

            if norm > 0:
                update[indices] = (
                    eta * matrix / norm
                ).reshape(-1)

        weight.copy_(old[id(weight)] + update)
```

Table 5: Time of one optimizer step on rMD17 aspirin (batch size 32, float32, one RTX 5090), in milliseconds.

Update rule	NequIP	GotenNet
Adam	0.2	0.6
RMS	0.9	3.3
Spectral, repeated squaring	2.9	12.8
Muon	2.3	11.8
Spectral, exact SVD	31.4	470.2

Computational cost. Table 5 reports the time of one optimizer step on rMD17 aspirin. The exact SVD costs 31 ms per step on NequIP's small irrep blocks but 470 ms on GotenNet's wider matrices, so we use repeated squaring for GotenNet. Repeated squaring costs about as much as Muon's Newton-Schulz iteration, and RMS normalization is the cheapest variant.

C Experimental Details

C.1 Model Architectures

Toy model. The equivariant network contains m scalar and m vector channels, giving hidden dimension $4m$. The input layer builds scalar features from each point's squared distance to the origin and vector features from its coordinates. Three hidden layers mix scalar and vector channels separately, add learned combinations of squared vector lengths to the scalar features, and apply sigmoid gates to vectors and SiLU to scalars. A final linear layer uses the scalar features to predict the largest eigenvalue. The dense MLP projects flattened coordinates to dimension $4m$ and uses three SiLU layers followed by a final linear layer.

Interatomic potential models. Our NequIP implementation (Batzner et al., 2022; Geiger and Smidt, 2022) uses two gated tensor product interaction layers with hidden irreps $32 \times 0e + 32 \times 1o + 32 \times 2e$, spherical harmonics up to $l_{\max} = 2$, a 4.5 Å cutoff, and eight radial basis functions. A radial network with hidden width 32 produces the tensor product weights. Messages are summed over neighbors and mixed by equivariant linear layers, with SiLU activations for scalars and sigmoid gates for higher degrees. GotenNet (Aykent and Xia, 2025) uses 64 channels, two interaction layers, $l_{\max} = 2$, four attention heads, a 5 Å cutoff, and 16 radial basis functions. We enable edge updates and separate direction and tensor coefficients for each degree. Its energy readout has a SiLU hidden layer of width 32. Both models sum atomic energy predictions and obtain forces as negative energy gradients with respect to atomic positions.

QM9. We use the same GotenNet representation with twelve prediction heads. Dipole moment and electronic spatial extent use the corresponding heads from the GotenNet implementation; the remaining heads predict atomic contributions that are summed over the molecule. We center molecules, include atomic reference contributions where available, and compute normalization statistics from the training set.

Particle dynamics. We adapt NequIP to take charges, speeds, and velocity vectors as inputs, retaining two interaction layers with 32 channels per degree up to $l_{\max} = 2$. All distinct particle pairs are connected. Edge features contain eight Gaussian radial basis functions of the distance and the product of the two particle charges. A readout linear layer predicts a displacement, which is added to the input position.

Hyperparameters. We select learning rates and moment coefficients using mean validation error across seeds, and also search the initialization scale for the synthetic task. In the synthetic ablation, moment coefficients selected with normalization are transferred to unnormalized Adam. Test results are reported only for the selected configurations. All runs use cosine learning rate decay to zero and no weight decay.

Table 6: Hyperparameters ranges. All runs use a cosine learning rate schedule, no warmup and no weight decay. Test error is evaluated at the checkpoint with the lowest validation error.

		rMD17 / MD22			
	Synthetic	NequIP	GotenNet	QM9	Particle dynamics
Models	e3nn, dense MLP	NequIP	GotenNet	GotenNet	NequIP
Train/validation/test	8192/512/2048	950/50/2000	950/50/2000	110,000/10,000/10,831	3000/2000/2000
Training steps	1000	5000	5000	200,000	30,000
Batch size	Full batch	32 / 16	32 / 16	32	100
Evaluation interval	100 steps	250 steps	250 steps	2500 steps	250 steps
Seeds	8	3	3	3	3
Loss	MSE	$\mathrm{MSE}(E) + 10\,\mathrm{MSE}(F)$	$\mathrm{MSE}(E) + 10\,\mathrm{MSE}(F)$	MSE	MSE
Adam (β_1, β_2)	$\beta_1 \in [0, 0.99]$, $\beta_2 \in [0.9, 0.9999]$	$(0.9, 0.999)$, $(0.98, 0.99)$	$(0.9, 0.999)$, $(0.98, 0.99)$	$(0.9, 0.999)$, $(0.9, 0.99)$, $(0.98, 0.99)$, $(0.99, 0.99)$	$(0.9, 0.999)$, $(0.98, 0.99)$
Muon momentum	0.95	0.95, 0.99	0.95, 0.99	-	-
Adam learning rate	0.0001-0.1	0.003-0.3	0.001-0.03	0.0003-0.01	0.0003-0.003
Spectral learning rate	0.0001-0.1	0.03-1	0.003-0.3	0.003-0.1	0.003-0.03
RMS learning rate	-	0.001-0.03	0.0001-0.01	0.0001-0.003	0.0001-0.001
Muon learning rate	0.0001-0.1	0.01-0.3	0.001-0.1	-	-
SGD learning rate	-	0.0001-0.01	-	-	-
SGD + spectral learning rate	-	0.03-1	-	-	-
SGD + RMS learning rate	-	0.001-0.01	-	-	-
SGD momentum	-	0.9, 0.99	-	-	-
SGD + spectral, RMS momentum	-	0.95, 0.99	-	-	-
Initialization scale	0.03, 0.1, 0.3, 1	Default	Default	Default	Default

Table 7: Selected hyperparameters.

Dataset	Method	(β_1, β_2)	Learning rate
Synthetic	Adam	$(0.9, 0.999)$	0.001, 0.00032
	Spectral	$(0.9, 0.999)$	0.01
	+ transferred mom.	$(0.5, 0.99)$	0.001, 0.00032
	+ both	$(0.5, 0.99)$	0.01
	Muon	0.95	0.0032
NequIP, rMD17 / MD22	Adam	$(0.98, 0.99)$	0.03, 0.1
	Spectral	$(0.98, 0.99)$	0.1
	RMS	$(0.98, 0.99)$	0.003
	Muon	0.99	0.03, 0.1
GotenNet, rMD17 / MD22	Adam	$(0.98, 0.99)$	0.003, 0.01
	Spectral	$(0.98, 0.99)$	0.01
	RMS	$(0.98, 0.99)$	0.0003
	Muon	0.99	0.003, 0.01
GotenNet, QM9	Adam	$(0.98, 0.99)$	0.00095
	Spectral	$(0.98, 0.99)$	0.0032
	RMS	$(0.9, 0.99)$	0.0003
NequIP, particle dynamics	Adam	$(0.9, 0.999)$	0.003
	Spectral	$(0.9, 0.999)$	0.03
	RMS	$(0.9, 0.999)$	0.001
NequIP, aspirin, SGD	SGD	0.99	0.001
	Spectral	0.99	0.1
	RMS	0.99	0.003

Table 7 lists the variants used for the main results. All selected synthetic variants use initialization scale 0.3. For molecular force prediction, learning rates and moment coefficients are selected separately for each model and dataset.

Table 8: Test force MAE (kcal/mol/Å, ↓) of Muon with the exact polar update (SVD) and with five Newton-Schulz iterations, for NequIP on rMD17 aspirin and ethanol. Both use momentum 0.99 and learning rate 0.1, the Newton-Schulz selection of Table 2. Test error is read at the checkpoint with the lowest validation error (mean ± standard deviation over three seeds).

Muon update	Aspirin	Ethanol
Exact SVD	0.684 ± 0.012	0.268 ± 0.016
Newton-Schulz	0.679 ± 0.005	0.262 ± 0.001

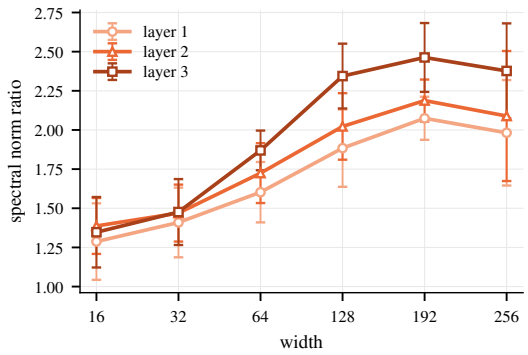


Figure 7: Ratio between the spectral norms of the updates of the scalar and vector irrep blocks, W_0 and W_1, in each hidden layer of the equivariant model on the synthetic task, at Adam's first update. Setting as in Figure 2; mean and standard deviation over eight seeds.

Muon uses Nesterov momentum with the coefficients listed in Table 7. On the synthetic task and in Figure 4a, Muon computes the exact polar update by SVD. In all other experiments, it approximates the polar update with five Newton-Schulz iterations in bfloat16. Table 8 compares the two on rMD17 aspirin and ethanol with NequIP, with the difference between them being negligible.

D Additional Empirical Results

Figure 7 splits the ratio of Figure 2 by hidden layer. In every layer, the ratio between the two irrep blocks grows with width.

Table 9 reports the complete GotenNet results across rMD17 and two MD22 datasets. Both normalizations improve Adam on every dataset, but their ranking reverses compared with NequIP in Table 2. Spectral normalization generally performs better than RMS for GotenNet, whereas RMS consistently performs better for NequIP.

Table 9: Force MAE (kcal/mol/Å; mean ± standard deviation over three seeds; lower is better) for GotenNet on rMD17 and MD22. Each optimizer uses its own validation-selected learning rate and moment coefficients; test error is read at the checkpoint with the lowest validation error. Bold marks the best result per dataset, underline the second best.

	Dataset	Adam	Spectral	RMS	Muon
rMD17	ethanol	0.194 ± 0.001	$\underline{0.152 \pm 0.005}$	0.165 ± 0.004	$\mathbf{0.092 \pm 0.001}$
	malonaldehyde	0.289 ± 0.006	$\underline{0.232 \pm 0.006}$	0.237 ± 0.003	$\mathbf{0.150 \pm 0.004}$
	benzene	0.052 ± 0.004	$\underline{0.038 \pm 0.002}$	$\underline{0.038 \pm 0.003}$	$\mathbf{0.019 \pm 0.001}$
	uracil	0.205 ± 0.014	$\underline{0.172 \pm 0.004}$	0.175 ± 0.007	$\mathbf{0.109 \pm 0.003}$
	toluene	0.167 ± 0.012	$\underline{0.134 \pm 0.005}$	0.135 ± 0.004	$\mathbf{0.084 \pm 0.003}$
	salicylic acid	0.311 ± 0.008	$\underline{0.247 \pm 0.001}$	0.251 ± 0.004	$\mathbf{0.166 \pm 0.002}$
	naphthalene	0.209 ± 0.006	$\underline{0.141 \pm 0.007}$	0.144 ± 0.003	$\mathbf{0.092 \pm 0.005}$
	paracetamol	0.388 ± 0.011	$\underline{0.295 \pm 0.003}$	0.304 ± 0.006	$\mathbf{0.210 \pm 0.002}$
	aspirin	0.432 ± 0.025	$\underline{0.365 \pm 0.012}$	0.376 ± 0.002	$\mathbf{0.280 \pm 0.004}$
	azobenzene	0.326 ± 0.023	$\underline{0.232 \pm 0.005}$	0.234 ± 0.008	$\mathbf{0.154 \pm 0.003}$
MD22	Ac-Ala3-NHMe	0.753 ± 0.018	0.598 ± 0.020	$\underline{0.596 \pm 0.012}$	$\mathbf{0.398 \pm 0.005}$
	stachyose	1.589 ± 0.033	$\underline{1.245 \pm 0.012}$	1.266 ± 0.025	$\mathbf{0.960 \pm 0.013}$