Rotation-Invariant Vector Normalization for Molecular Force Learning

Bum Jun Kim*
The University of Tokyo, Japan
bumjun.kim@weblab.t.u-tokyo.ac.jp

Hyeyun Jeong
Samsung Electronics
hy119.jeong@gmail.com

Abstract

Rotating a molecule should leave its energy unchanged while rotating each force vector by the same rotation, as required in $SO(3)$-equivariant molecular learning. However, conventional normalization layers in neural networks are designed for scalar activations; when applied to vector channels through component-wise centering or scaling, they can break this symmetry. We study normalization modules for equivariant molecular networks under force supervision, where invariant scalar channels and covariant vector channels require different geometric treatment. We formalize an $SO(3)$-equivariance-preserving recipe that computes vector scale statistics from rotation-invariant squared norms and applies the resulting rescaling identically across Cartesian coordinates, which preserves $SO(3)$-equivariance without vector shifts. Building on this recipe, we introduce grouped vector RMS (GroupRMS), which shares vector RMS estimates across channel groups to reduce estimator variance and stabilize vector scaling in small-batch training. The proposed variant is lightweight and compatible with standard scalar normalization branches. We instantiate these ideas in practical scalar-vector normalization blocks for PaiNN-style backbones and evaluate them across several molecular benchmarks. On the paired three-trajectory subset, the strongest GroupRMS-family variant, GroupNorm-GroupRMS, improves average force MAE from 0.1190 to 0.0383 and average energy MAE from 0.0675 to 0.0424 relative to BatchNorm-VectorRMS. With the scalar branch fixed, GroupRMS improves its per-channel counterpart in all 40 paired runs across eight molecular systems. In public implementations, GroupRMS improves the matched ChannelRMS insertion in PaiNN and the native MACE baseline. These results support grouped scale estimation as an effective tool for robust molecular force learning.

1 Introduction

Equivariant architectures are now standard in molecular learning, especially for force supervision, where forces are obtained by differentiating an energy model, $\hat{F} = -\nabla_x \hat{E}$ [8–10]. The broader interatomic-potential literature spans early high-dimensional neural network potentials and message passing models to modern molecular force fields [5, 18, 35, 39–41].

Despite the strong backbones such as PaiNN, NequIP, MACE, GemNet, Equiformer variants, TorchMD-Net, Allegro, and M3GNet [3, 4, 7, 15, 26, 27, 30, 33, 36], normalization layers remain underspecified in equivariant molecular force learning. Although scalar channels are rotation-invariant, vector channels must transform according to the special orthogonal group in three dimensions, $SO(3)$, which should not be shifted arbitrarily. Nevertheless, standard normalization layers such as Batch Normalization (BatchNorm), Group Normalization (GroupNorm), and Layer Normalization [2, 20, 44]

*Corresponding author.

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

can hurt equivariance if applied naively, because fixed vector shifts and coordinate-specific denominators do not necessarily commute with the global $SO(3)$ action.

We study normalization design for force-supervised equivariant molecular learning through three contributions. First, we formalize the invariant-statistics condition underlying equivariant vector rescaling and its implications for energy-derived forces. Second, we introduce grouped vector root-mean-square scaling (GroupRMS), with tunable channel sharing and a correlated-channel variance analysis. Third, we evaluate this design on molecular-dynamics, transfer, out-of-distribution (OOD), and catalyst-scale benchmarks, including sharing-only controls and insertions into public PaiNN and MACE implementations. These controls distinguish gains from denominator sharing from those of changing the complete scalar-vector normalization block.

2 Background

2.1 Related Work

Equivariant molecular networks. Equivariant or geometry-aware graph neural networks (GNNs) are a standard choice for molecular energies and forces, ranging from early tensor and covariant models such as Tensor Field Networks and Cormorant [1, 37] and interaction models such as SchNet, DimeNet, DimeNet++, and SphereNet [23, 24, 28, 32] to modern scalar and vector architectures as well as higher-order irreducible representation (irrep) models, including $\mathrm{E}(n)$-equivariant GNNs, geometric vector perceptron, PaiNN, $\mathrm{SE}(3)$-Transformer, NequIP, MACE, GemNet, GemNet-OC, Equiformer, EquiformerV2, and ViSNet [3, 4, 14–16, 21, 26, 27, 31, 33, 42]. Recent large-scale molecular potential studies also emphasize architectures such as TorchMD-Net, Allegro, and M3GNet for broad chemistry and materials coverage [7, 30, 36].

Equivariant normalization. Invariant norm rescaling is established in Clebsch–Gordan networks, Vector Neurons, geometric vector perceptrons (GVPs), e3nn, and EquiformerV2 [11, 17, 22, 25, 27]. These operators differ in where statistics are collected and which channels share a denominator. For example, e3nn BatchNorm uses batch statistics per irrep copy, while GVP vector normalization and the $\ell = 1$ restriction of EquiformerV2 layer normalization share a node-local scale across vector channels. GeoNorm normalizes geometric coordinate states [29]. GroupRMS studies tunable denominator sharing within a fixed statistical support, including graph-wise grouped scaling, and relates its force-learning behavior to correlated-channel estimator variance. Appendix G.3 gives a two-axis comparison and exact operators.

2.2 Problem Setting

Scalar and Vector Equivariance We consider scalar features $s_i \in \mathbb{R}^{C_s}$ and vector features $v_i = \{v_{i,c}\}_{c=1}^{C_v}$ with $v_{i,c} \in \mathbb{R}^3$ at node i. The same object can be written as $v_i \in \mathbb{R}^{C_v} \otimes \mathbb{R}^3$. Implementations often store Cartesian components in an array of shape $(C_v, 3)$. For a rotation $R \in SO(3)$, we use the standard action on $\mathbb{R}^3$:

$$T_R(s_i) = s_i, \quad T_R(v_i)_c = Rv_{i,c}. \tag{1}$$

When a 3-vector is represented as a 1×3 row in the last axis, the same action is written as $v_{i,c} \mapsto v_{i,c}R^\top$. A normalization map $N = (N_s, N_v)$ is equivariant if $N(T_R(s, v)) = T_R(N(s, v))$, where $T_R(s, v) \coloneqq (s, T_R(v))$. Since $T_R(s) = s$, the scalar condition reduces to $N_s(s, T_R(v)) = N_s(s, v)$. Our scalar branches use rotation-invariant scalar statistics, so we focus on verifying the vector part.

Why standard normalization can be brittle Scalar normalization typically uses mean and variance statistics. For vectors, subtracting a fixed learned bias breaks $SO(3)$-equivariance, and data-dependent centering is equivariant only when the centering statistic transforms covariantly. Even when shifts are avoided, the way vector scales are estimated can strongly affect optimization. This sensitivity to vector-scale estimation is especially pronounced when force-supervised training differentiates force losses, requiring mixed second derivatives of the energy with respect to coordinates and parameters, and when the estimator choice differs between batch-wise and graph-wise schemes or between per-channel and coupled scaling. Appendix C makes these constraints explicit: Fixed affine vector normalization must reduce to a scalar multiple of the identity with zero bias (Proposition 3), and component-wise vector RMS has a concrete $SO(3)$-equivariance defect (Proposition 4). Figure 1

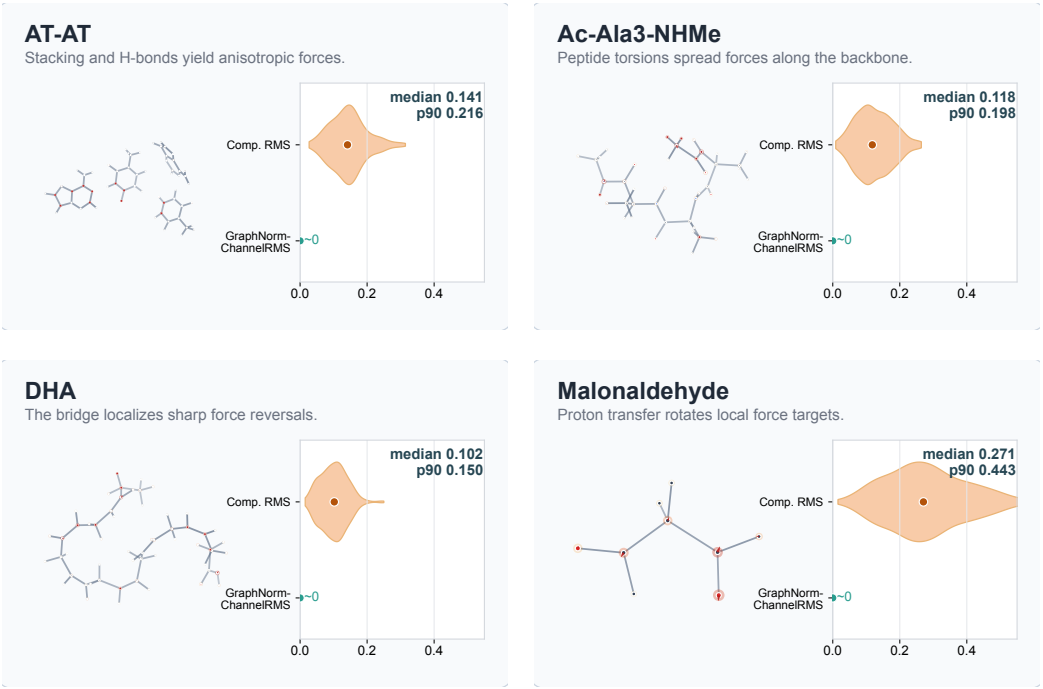


Figure 1: Real-force rotation-defect audit on actual MD22 and Molecular Dynamics 17 (MD17) labels. Each panel samples real target forces from the named trajectory and compares a scalar-style component-wise RMS rule against ChannelRMS under random global rotations. Component-wise RMS has a nonzero relative equivariance defect because the denominator changes with the Cartesian basis, whereas ChannelRMS stays at numerical zero by using an invariant scale. Component-wise RMS is an illustrative failure case, not the experimental BatchNorm-VectorRMS baseline, which is itself equivariant.

shows this defect on real force labels. Scalar-style component-wise RMS produces basis-dependent outputs under random rotations, while channel-wise RMS (ChannelRMS) remains at numerical zero because its denominator is built from rotation-invariant squared norms.

3 Proposed Method

Scope of the equivariance claim. Our claims concern the global $SO(3)$ action in Eq. 1. They hold for scalar statistics and covariant vector outputs under a common rotation. Extensions to local-frame gauge-equivariant models [19, 43] are left to future work.

Block naming. We name scalar-vector (SV) normalization blocks by their scalar branch and vector RMS branch. Table 1 summarizes the main single-branch combinations used below. In these names, GraphNorm denotes the graph-wise scalar mean and variance branch used below, GraphGroup denotes a graph-wise scalar group-normalization branch, ChannelRMS denotes graph-wise per-channel vector RMS, and GroupRMS denotes grouped vector RMS. The GraphNorm prefix is a descriptive label for our graph-wise scalar branch; the separate GraphNorm-SV baseline in the appendix keeps the baseline name following Cai et al. [6]. Our experimental baseline, *BatchNorm-VectorRMS*, uses scalar BatchNorm and invariant per-channel vector RMS over minibatch nodes; it does not normalize Cartesian components independently. Both branches use running statistics at evaluation.

3.1 Invariant-statistics recipe for equivariant application

Let $g(i)$ denote the graph containing node i. Consider any graph-wise scalar-valued statistic $\alpha_{g,c}(v)$ that may be shared across channels and satisfies $\alpha_{g,c}(T_R(v)) = \alpha_{g,c}(v)$ and $\alpha_{g,c}(v) \neq 0$. Define the

Table 1: Naming convention for the main single-branch scalar-vector normalization blocks. Checkmarks indicate which scalar statistics and vector RMS denominator each block uses.

Block	Scalar branch			Vector RMS branch		
	Graph mean/var	Graph group	Batch GroupNorm	Graph channel	Graph group	Batch group
GraphNorm-ChannelRMS	✓	–	–	✓	–	–
GraphNorm-GroupRMS	✓	–	–	–	✓	–
GraphGroup-GroupRMS	–	✓	–	–	✓	–
GroupNorm-GroupRMS	–	–	✓	–	–	✓

base vector normalization branch

$$\widetilde{N}_v(v)_{i,c} = (\gamma_v)_c \frac{v_{i,c}}{\alpha_{g(i),c}(v)}, \tag{2}$$

with learnable per-channel gain $\gamma_v \in \mathbb{R}^{C_v}$ and no vector shift. This base branch covers per-channel RMS scaling called ChannelRMS with $\alpha_{g,c} = \rho_{g,c}$ and grouped scaling called GroupRMS with $\alpha_{g,c} = \rho_{g,k(c)}$. In practice, ChannelRMS uses $\alpha_{g,c} = \sqrt{m_{g,c} + \varepsilon} > 0$, while graph-wise GroupRMS uses $\alpha_{g,c} = \sqrt{m_{g,k(c)} + \varepsilon} > 0$. Here $m_{g,c}$ and $m_{g,k}$ denote second moments, and ρ denotes the corresponding RMS scale, defined in Eq. 3 and Eq. 4. Positivity of the denominator is a design choice; equivariance does not require the learned gain to be positive.

Proposition 1 (Equivariance of invariant scaling)**.** *If $\alpha_{g,c}(T_R(v)) = \alpha_{g,c}(v)$ for all $R \in SO(3)$ and all channels c, and $\alpha_{g,c}(v) \neq 0$, then $\widetilde{N}_v(T_R(v)) = T_R(\widetilde{N}_v(v))$.*

Proof. Since $\alpha_{g,c}$ is invariant, $\alpha_{g(i),c}(T_R(v)) = \alpha_{g(i),c}(v)$, and therefore

$$\begin{aligned}\widetilde{N}_v(T_R(v))_{i,c} &= (\gamma_v)_c \frac{Rv_{i,c}}{\alpha_{g(i),c}(T_R(v))} = (\gamma_v)_c \frac{Rv_{i,c}}{\alpha_{g(i),c}(v)} \\ &= T_R(\widetilde{N}_v(v))_{i,c}.\end{aligned}$$

□

The default GraphNorm-ChannelRMS, GraphNorm-GroupRMS, and GraphGroup-GroupRMS blocks optionally mix this equivariant normalized branch with the identity using learnable channel-wise gates:

$$N_v^{\mathrm{res}}(v)_{i,c} = (a_v)_c \widetilde{N}_v(v)_{i,c} + (1 - (a_v)_c) v_{i,c},$$

where $a_v \in (0,1)^{C_v}$. Since the identity map is also equivariant and a_v is a rotation-invariant vector of scalar parameters, N_v^{res} remains equivariant. At the full energy-model level, Theorem 1 in Appendix C shows that inserting such $SO(3)$-equivariance-preserving normalizers into an equivariant energy network preserves energy invariance and the induced force equivariance away from graph-construction boundaries.

3.2 From invariant scaling to GraphNorm-ChannelRMS

Let $\mathcal{V}_g$ denote the set of nodes in graph g. Define the per-graph, per-channel second moment and induced RMS scale:

$$m_{g,c} = \frac{1}{3|\mathcal{V}_g|} \sum_{i \in \mathcal{V}_g} \|v_{i,c}\|_2^2, \quad \rho_{g,c} = \sqrt{m_{g,c} + \varepsilon}. \tag{3}$$

The factor 3 makes $m_{g,c}$ the mean squared value per Cartesian component, which is equivalent to the average over the $3|\mathcal{V}_g|$ scalar components of channel c. This second-moment definition uses only rotation invariants $\|Rv\|_2^2 = \|v\|_2^2$ and yields an equivariant normalizer by Proposition 1. For scalars, we use graph-wise mean and variance with instance-style normalization [38],

$$\widetilde{S}_i = \gamma_s \odot \frac{S_i - \mu_{g(i)}}{\sqrt{\sigma_{g(i)}^2 + \varepsilon}} + \beta_s.$$

In the default GraphNorm-ChannelRMS and GraphNorm-GroupRMS blocks, the scalar and vector normalized branches are then mixed with the input through learnable channel-wise gates $a_s = \sigma(\eta_s) \in (0,1)^{C_s}$ and $a_v = \sigma(\eta_v) \in (0,1)^{C_v}$:

$$S'_i = a_s \odot \widetilde{S}_i + (1 - a_s) \odot S_i,$$
$$V'_{i,c} = (a_v)_c \widetilde{N}_v(v)_{i,c} + (1 - (a_v)_c) v_{i,c}.$$

Consequently, GraphNorm-ChannelRMS stabilizes graph-wise feature magnitudes while preserving the equivariant vector geometry required for force prediction, consistent with the force-level result in Theorem 1.

3.3 Graph-wise GroupRMS for coupling vector scales across channel groups

Per-channel RMS scaling can be noisy in force-supervised regimes, especially with small batches and heavy-tailed gradients. We therefore group vector channels into G groups of equal size q with $Gq = C_v$, assume $G \mid C_v$, and estimate a group second moment and induced RMS scale:

$$m_{g,k} = \frac{1}{3|\mathcal{V}_g|q} \sum_{i \in \mathcal{V}_g} \sum_{c \in \mathcal{G}_k} \|v_{i,c}\|_2^2, \quad \rho_{g,k} = \sqrt{m_{g,k} + \varepsilon}. \tag{4}$$

Our default $G = 8$ at $C_v = 128$ gives group width $q = 16$, with $\varepsilon = 10^{-5}$. Width and stabilizer sweeps support this practical starting point (Appendix G.5). Let $k(c)$ denote the group index of channel c, so $c \in \mathcal{G}_{k(c)}$. All channels in group k are scaled by $\rho_{g,k}$, which corresponds to $\alpha_{g,c} = \rho_{g,k(c)}$ in Eq. 2, followed by a per-channel gain $(\gamma_v)_c$.

Remark 1 (Variance reduction intuition)**.** Under an idealized independent and identically distributed (i.i.d.) model for the per-channel squared-norm statistics, averaging over q channels reduces $\mathrm{Var}(m_{g,k})$ by exactly a factor $1/q$. Let $X_c := (1/(3|\mathcal{V}_g|)) \sum_{i \in \mathcal{V}_g} \|v_{i,c}\|_2^2$, so that $m_{g,k} = (1/q) \sum_{c \in \mathcal{G}_k} X_c$. If $\{X_c\}_{c \in \mathcal{G}_k}$ are i.i.d. with $\mathrm{Var}(X_c) = \sigma^2$, then

$$\mathrm{Var}(m_{g,k}) = \mathrm{Var}((1/q) \sum_c X_c) = \frac{1}{q^2} \sum_c \mathrm{Var}(X_c)$$
$$= \frac{\sigma^2}{q}.$$

For $\rho_{g,k} = \sqrt{m_{g,k} + \varepsilon}$, a first-order delta-method approximation then gives $\mathrm{Var}(\rho_{g,k}) \approx (4(\mathbb{E}[m_{g,k}] + \varepsilon))^{-1} \mathrm{Var}(m_{g,k}) = O(1/q)$, so the standard deviation is $O(1/\sqrt{q})$.

Appendix C.1 extends this i.i.d. variance-reduction intuition to correlated channels through an effective sample-size formula, while Figure 2 visualizes the same scale-estimator variance reduction by contrasting ChannelRMS with GroupRMS.

This mechanism is measurable in learned activations: over 500 held-out graphs and all five layers, group width $q = 16$ gives a realized effective sample-size gain of 18.18 versus 17.12 predicted from measured channel correlations, with RMS-denominator coefficient of variation reduced by 73% (Appendix G.5). Figure 4 provides an additional geometry-only illustration.

3.4 Other variants

Although GroupRMS is the core proposal of this paper, we include several closely related variants to isolate the role of graph-wise versus batch-wise statistics and scalar normalization choices under the same $SO(3)$-equivariance-preserving vector-scaling principle. We additionally consider Hybrid-GroupRMS, which forms separate graph-wise and batch-wise normalized branches and mixes their outputs with learnable channel-wise gates, rather than interpolating the raw statistics before normalization. We also consider GraphGroup-GroupRMS, a graph-wise group-normalization variant with group mean and variance for scalars and graph-wise group RMS for vectors, followed by the same residual gate with the identity used in the default GraphNorm-GroupRMS block. GroupNorm-GroupRMS is a batch-global GroupRMS-family variant that combines scalar GroupNorm over

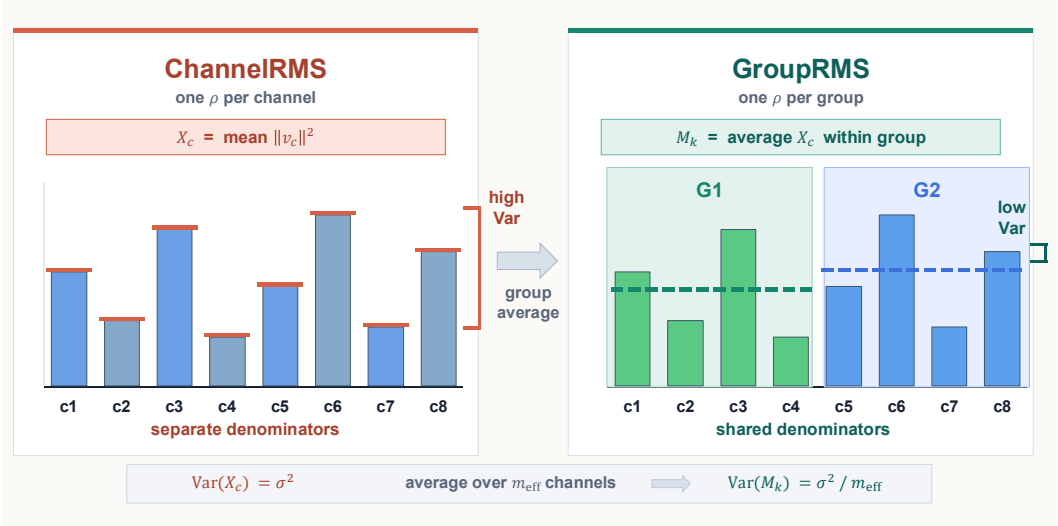


Figure 2: Illustration of GroupRMS variance reduction for vector RMS estimation. (Left) ChannelRMS computes one squared-norm statistic $X_c = (3|\mathcal{V}_g|)^{-1} \sum_{i \in \mathcal{V}_g} \|v_{i,c}\|_2^2$, equivalently $m_{g,c}$, and one RMS denominator $\rho_{g,c}$ per vector channel, so channel-wise denominator noise is inherited separately by each channel. (Right) GroupRMS partitions the C_v channels into groups and forms $M_k = (1/q) \sum_{c \in \mathcal{G}_k} X_c$, equivalently $m_{g,k}$, so all channels in group k share $\rho_{g,k} = \sqrt{M_k + \varepsilon}$. As discussed in Remark 1, Proposition 2, and Corollary 1, $\mathrm{Var}(M_k) = \sigma^2/q$ under the idealized i.i.d. model and $\mathrm{Var}(M_k) = \sigma^2/m_{\mathrm{eff}}$ under correlated channels with equal marginal variance, reducing denominator noise by averaging over m_{eff} effective channels and stabilizing vector scaling in small-batch force training.

channels and all minibatch nodes with a batch-global grouped vector RMS,

$$\bar{m}_k = \frac{1}{3Nq} \sum_{i=1}^{N} \sum_{c \in \mathcal{G}_k} \|v_{i,c}\|_2^2, \quad \bar{\rho}_k = \sqrt{\bar{m}_k + \varepsilon},$$

so its vector denominator is shared across the whole minibatch rather than computed separately for each graph. Because $\bar{m}_k$ also depends only on squared norms and is applied identically across Cartesian coordinates, this vector branch remains equivariant. Algorithm 2 states this batch-global GroupNorm-GroupRMS block. We also include SV baselines, namely GraphNorm-SV and RMSNorm-SV, with vector statistics computed from rotation invariants following Cai et al. [6] and Zhang and Sennrich [45]. Appendix tables provide implementation details. A graph-wise vector denominator is independent of unrelated co-batched graphs, but independence of the full block also requires graph-wise scalar statistics. GroupNorm-GroupRMS and the fixed-scalar-GroupNorm controls use batch-global scalar statistics.

4 Experiments

4.1 Experimental Setup

Datasets. Our evaluation uses MD22 [9] with 6 core trajectories and extends to MD17 [8], revised MD17 (rMD17) [10], QM9, ISO17, and Open Catalyst 2020 (OC20). Recent force-field evaluations also emphasize broader simulation reliability and chemistry coverage, for example, in recent molecular simulation stress-test benchmarks [12, 13]. Appendix F also includes partial comparisons among BatchNorm-VectorRMS, GraphGroup-GroupRMS, GraphNorm-GroupRMS, and GroupNorm-GroupRMS on auxiliary benchmarks.

Backbones and training. The MD22 and auxiliary benchmark tables use PaiNN-lite, a lightweight SV message-passing backbone inspired by PaiNN [33]. The auxiliary QM9 OOD tables additionally

Algorithm 1 GraphNorm-GroupRMS block.

Require: scalar features $S \in \mathbb{R}^{N \times C_s}$, vector features $V \in \mathbb{R}^{N \times C_v \times 3}$, batch indices $b \in \{0, \dots, B-1\}^N$
Require: group count $G \mid C_v$, group width $q = C_v/G$, $\varepsilon > 0$
Require: scalar affine (γ_s, β_s), vector gain $\gamma_v \in \mathbb{R}^{C_v}$
Require: scalar gate $a_s \in (0,1)^{C_s}$, vector gate $a_v \in (0,1)^{C_v}$

Store raw inputs $S^{\mathrm{raw}} \leftarrow S$, $V^{\mathrm{raw}} \leftarrow V$
For each graph index g, let $\mathcal{V}_g := \{i : b(i) = g\}$
Compute per-graph and channel-wise scalar mean and variance $\mu_g, \sigma_g^2 \in \mathbb{R}^{C_s}$ over $i \in \mathcal{V}_g$
$\widetilde{S}_i \leftarrow (S_i - \mu_{b(i)})/\sqrt{\sigma^2_{b(i)} + \varepsilon}$; $\widetilde{S} \leftarrow \gamma_s \odot \widetilde{S} + \beta_s$
Partition channels into groups $\{\mathcal{G}_k\}_{k=1}^G$
Compute the graph-wise group second moment:
$m_{g,k} \leftarrow (1/(3|\mathcal{V}_g|q)) \sum_{i \in \mathcal{V}_g} \sum_{c \in \mathcal{G}_k} \|V_{i,c,:}\|_2^2$
$\rho_{g,k} \leftarrow \sqrt{m_{g,k} + \varepsilon}$
$\widetilde{V}_{i,c,:} \leftarrow V_{i,c,:}/\rho_{b(i),k(c)}$; $\widetilde{V} \leftarrow \gamma_v \odot \widetilde{V}$
$S_i \leftarrow a_s \odot \widetilde{S}_i + (1 - a_s) \odot S_i^{\mathrm{raw}}$
$V_{i,c,:} \leftarrow (a_v)_c \widetilde{V}_{i,c,:} + (1 - (a_v)_c) V^{\mathrm{raw}}_{i,c,:}$
return normalized (S, V)

Table 2: MD22 core macro-average over three trajectories. Mean ± sample SD over three seed-level macro-averages; energy in eV and force in eV/Å. Lower is better.

Norm	Avg. Energy MAE	Avg. Force MAE
BatchNorm-VectorRMS	0.0675 ± 0.0053	0.1190 ± 0.0040
GroupNorm-GroupRMS	$\mathbf{0.0424 \pm 0.0004}$	$\mathbf{0.0383 \pm 0.0005}$

include SV-v2, a compact scalar-vector message-passing GNN; controlled insertion experiments use public SchNetPack PaiNN and official MACE. For force tasks, we train an energy model and obtain forces by autograd. Reported metrics use mean absolute error (MAE) for energies and forces, while training objectives use weighted mean squared error (MSE) terms. Additional dataset-specific hyperparameters, objectives, epoch budgets, and convergence criteria are given in Appendix D.

MD22 protocols. The long-budget MD22 results are in Tables 2 and 3, with paired statistics in Appendix Table 8; additional MD22 optimization summaries appear in Appendix F. The MD22 core suite uses six trajectories, namely Ac-Ala3-NHMe, AT-AT, AT-AT-CG-CG, Buckyball Catcher, docosahexaenoic acid (DHA), and stachyose. Core runs use training, validation, and test sizes of 1000, 200, and 500, with $k = 12$, batch size 8, max epochs 1536, patience 128, learning rate 2×10^{-4}, and $(\lambda_E, \lambda_F) = (0.2, 0.8)$. The main-text subset consists of AT-AT, Ac-Ala3-NHMe, and DHA. The DHA-large and Stachyose-large tasks use training, validation, and test sizes of 5000, 1000, and 1000, the same k and learning rate, 12 epochs, and patience 4.

4.2 Results

MD22 core subset. Tables 2 and 3 report the main MD22 core comparisons. On Ac-Ala3-NHMe, BatchNorm-VectorRMS reaches 0.1234 ± 0.0048 force MAE, while GroupNorm-GroupRMS reaches 0.0460 ± 0.0006 in Table 3. Across this subset, average energy MAE improves from 0.0675 to 0.0424, and average force MAE improves from 0.1190 to 0.0383 in Table 2. The paired random-effects summary remains negative for both metrics in Table 8, and Appendix Figure 3 shows the corresponding optimization-regime shift.

Auxiliary aggregate context. Table 4 summarizes molecular-dynamics, transfer, and catalyst-scale results. Detailed per-task tables and paired rMD17 statistics remain in Appendix F. QM9 gap prediction is a static electronic-property task trained without force labels or trajectories. On QM9

Table 3: MD22 core force MAE in eV/Å, mean ± SD over three training seeds per trajectory.

Task	Norm	Force MAE
AT-AT	BatchNorm-VectorRMS	0.1269 ± 0.0062
AT-AT	GroupNorm-GroupRMS	$\mathbf{0.0360 \pm 0.0009}$
Ac-Ala3-NHMe	BatchNorm-VectorRMS	0.1234 ± 0.0048
Ac-Ala3-NHMe	GroupNorm-GroupRMS	$\mathbf{0.0460 \pm 0.0006}$
DHA	BatchNorm-VectorRMS	0.1067 ± 0.0121
DHA	GroupNorm-GroupRMS	$\mathbf{0.0330 \pm 0.0007}$

Table 4: Auxiliary results for PaiNN-lite: mean ± sample SD over three seeds. MD17/rMD17 entries are seed-level macro-averages over ten molecules. Energy/force units are kcal/mol and kcal/mol/Å for MD17, and eV and eV/Å for rMD17/ISO17/OC20.

Benchmark	Norm	Energy MAE	Force MAE
MD17	GraphGroup-GroupRMS	$\mathbf{1.3176 \pm 0.1126}$	$\mathbf{2.1410 \pm 0.0886}$
MD17	GroupNorm-GroupRMS	1.6035 ± 0.1583	2.1590 ± 0.0379
MD17	GraphNorm-GroupRMS	1.7006 ± 0.1266	2.2214 ± 0.0291
MD17	BatchNorm-VectorRMS	1.4203 ± 0.2489	3.7806 ± 0.1148
rMD17	GraphGroup-GroupRMS	0.0664 ± 0.0117	$\mathbf{0.0962 \pm 0.0049}$
rMD17	GraphNorm-GroupRMS	$\mathbf{0.0661 \pm 0.0024}$	0.1009 ± 0.0012
rMD17	BatchNorm-VectorRMS	0.0861 ± 0.0129	0.1763 ± 0.0061
ISO17 full	BatchNorm-VectorRMS	0.0832 ± 0.0365	0.1356 ± 0.0030
ISO17 full	GraphGroup-GroupRMS	0.0606 ± 0.0199	$\mathbf{0.0965 \pm 0.0020}$
ISO17 full	GraphNorm-GroupRMS	$\mathbf{0.0605 \pm 0.0046}$	0.1023 ± 0.0068
OC20 2M	BatchNorm-VectorRMS	11.4575 ± 2.7620	0.9120 ± 0.0342
OC20 2M	GraphNorm-GroupRMS	$\mathbf{7.7454 \pm 1.0357}$	$\mathbf{0.6646 \pm 0.0448}$

size-OOD, extending SV-v2 to ten seeds gives MAE 0.2160 ± 0.0335 eV for BatchNorm-VectorRMS and 0.1900 ± 0.0058 eV for Hybrid-GroupRMS, a 12.0% reduction ($p = 0.0257$; Table 11).

rMD17 paired summary. The rMD17 random-effects CIs remain negative for both energy and force with GraphGroup-GroupRMS and GraphNorm-GroupRMS (Appendix Table 9).

Isolating denominator sharing. We fix scalar GroupNorm, disable residual gates, and match all training choices and per-channel gains while changing only graph-wise ChannelRMS to GroupRMS ($G = 8$, $q = 16$). Across four MD17 and four MD22 systems (9–87 atoms), GroupRMS improves all $40/40$ paired runs (five seeds per system), with an 8.24% geometric-mean force-MAE reduction (95% CI 7.06–9.41%, $p = 2.1 \times 10^{-16}$). Per-system reductions range from 4.9% to 12.9% (Appendix G.2).

Comparison with related vector normalizers. Table 5 fixes the scalar branch and gate setting while comparing vector operators on MD22 Ac-Ala3-NHMe and MD17 aspirin. With 1k training frames, EquiformerV2-style layer normalization performs best; GroupRMS improves ChannelRMS by 9.43% across the six paired runs. At 4k MD22 frames, GroupRMS reaches 0.03412 ± 0.00051 eV/Å, improving over no vector normalization by 19.98% ($p = 0.047$). Its difference from EquiformerV2-style normalization is not statistically significant in this three-seed comparison ($p = 0.15$). Sharing benefits both node-local and graph-wise supports (Appendix G.3).

Transfer to public implementations. We insert matched graph-wise ChannelRMS or GroupRMS branches into official MACE product-block outputs, retaining the native scalar path. Across three conditions and eight seeds each, GroupRMS improves native MACE in $22/24$ pairs (6.34%, $p = 8.4 \times 10^{-6}$) and ChannelRMS in $22/24$ pairs (5.27%, $p = 3.9 \times 10^{-6}$; Table 6). Both inserted branches add the same 256 parameters. In public SchNetPack PaiNN, GroupRMS improves ChannelRMS in $19/20$ pairs (5.32%, $p = 5.3 \times 10^{-5}$). Appendix G.4 reports native baselines, exact insertion points, and identity/rotation audits.

Table 5: Controlled vector-normalizer comparison with fixed scalar GroupNorm and no gate. Force MAE, mean $\pm$ SD over three paired seeds; 1000 training frames. Operator support, affine conventions, and budgets are detailed in Appendix G.3.

Vector operation	MD22 Ac-Ala3-NHMe (eV/Å)	MD17 aspirin (kcal/mol/Å)
No vector normalization	0.05069 ± 0.00060	1.2448 ± 0.0347
e3nn BatchNorm	0.08894 ± 0.00826	3.0326 ± 0.6401
Vector Neurons-style norm BN	0.27960 ± 0.21010	5.2774 ± 2.6063
EquiformerV2-style LN ($\ell = 1$)	$\mathbf{0.04921 \pm 0.00201}$	$\mathbf{1.2260 \pm 0.0276}$
Graph-wise ChannelRMS	0.05615 ± 0.00155	1.4378 ± 0.0722
Graph-wise GroupRMS	0.05102 ± 0.00160	1.2972 ± 0.0171

Table 6: Official MACE: force MAE in kcal/mol/Å, mean $\pm$ SD over eight paired seeds per cell. The 4k protocol matches sample exposure to the 1k protocol. Native MACE has 381,840 parameters; both inserted variants have 382,096.

MACE variant	Aspirin 1k	Ethanol 1k	Aspirin 4k
Native	1.9481 ± 0.1064	0.7999 ± 0.0389	1.9640 ± 0.1011
+ ChannelRMS	1.9092 ± 0.1051	0.8092 ± 0.0411	1.9153 ± 0.1049
+ GroupRMS	$\mathbf{1.8058 \pm 0.0540}$	$\mathbf{0.7659 \pm 0.0290}$	$\mathbf{1.8145 \pm 0.0524}$

4.3 Discussion

Why GroupRMS helps. Force supervision backpropagates through all layers and requires higher-order derivatives, which amplifies optimization noise. Theoretically, GroupRMS aggregates across channels to increase the effective sample size for scale estimation. Under the idealized i.i.d. model with common variance in Remark 1 and the correlated-channel extension in Proposition 2 and Corollary 1, the resulting lower denominator variance can stabilize gradients.

Interpreting the normalization comparisons. Both BatchNorm-VectorRMS and GroupRMS preserve global rotation equivariance. Their different optimization behavior therefore concerns statistic support, scalar normalization, and denominator sharing. Fixed running statistics define a fixed evaluation mapping for BatchNorm-VectorRMS, while current-batch statistics can couple co-batched structures. The sharing-only experiments isolate the channel-coupling effect within a fixed support and scalar branch.

Energy and force trade-off. The full GroupNorm-GroupRMS block improves both energy and force on the long-budget MD22 subset. In MD17 aspirin NVE tests, it also reduces 300 K energy drift by 68.5% relative to BatchNorm-VectorRMS; all tested controls complete the 4 ps trajectories. Force-evaluation and force-training latency increase by 8.5% and 7.6%, respectively, with identical parameter counts. Appendix G.6 reports rollout controls and a separate 100 ps stress test; Appendix G.7 reports the full cost comparison.

5 Conclusion

We introduce GroupRMS for molecular force learning, combining invariant vector rescaling with tunable sharing of scale estimates across channel groups. The correlated-channel variance analysis and learned-feature measurements explain how sharing reduces denominator noise. The full GroupNorm-GroupRMS block improves average MD22 force MAE from 0.1190 to 0.0383 and energy MAE from 0.0675 to 0.0424. Sharing-only controls across eight systems and insertions into public PaiNN and MACE establish complementary benefits beyond the full-block comparison. These results make grouped scale estimation a practical normalization choice for equivariant molecular networks.

Acknowledgements

B.J.K. was supported by JSPS KAKENHI Grant Number JP25K24362. B.J.K. was supported by computational resources of the TSUBAME4.0 supercomputer provided by Institute of Science Tokyo through the HPCI System Research Project (Project ID: hp260049). The authors declare no competing interests.

References

[1] Brandon M. Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant Molecular Neural Networks. In *NeurIPS*, pages 14510–14519, 2019.

[2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *CoRR*, abs/1607.06450, 2016.

[3] Ilyes Batatia, Dávid Péter Kovács, Gregor N. C. Simm, Christoph Ortner, and Gábor Csányi. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. In *NeurIPS*, 2022.

[4] Simon L. Batzner, Tess E. Smidt, Lixin Sun, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, and Boris Kozinsky. SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *CoRR*, abs/2101.03164, 2021.

[5] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

[6] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. GraphNorm: A Principled Approach to Accelerating Graph Neural Network Training. In *ICML*, volume 139, pages 1204–1215, 2021.

[7] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table. *Nat. Comput. Sci.*, 2(11):718–728, 2022.

[8] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.

[9] 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. *Science Advances*, 9(2):eadf0873, 2023.

[10] Anders S. Christensen and O. Anatole von Lilienfeld. On the role of gradients for machine learning of molecular energies and forces. *Mach. Learn. Sci. Technol.*, 1(4):45018, 2020.

[11] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. In *ICCV*, pages 12180–12189, 2021.

[12] Peter K. Eastman, Pavan Kumar Behara, David L. Dotson, Raimondas Galvelis, John E. Herr, Josh T. Horton, Yuezhi Mao, John D. Chodera, Benjamin P. Pritchard, Yuanqing Wang, Gianni De Fabritiis, and Thomas E. Markland. SPICE, A Dataset of Drug-like Molecules and Peptides for Training Machine Learning Potentials. *CoRR*, abs/2209.10702, 2022.

[13] Xiang Fu, Zhenghao Wu, Wujie Wang, Tian Xie, Sinan Keten, Rafael Gómez-Bombarelli, and Tommi S. Jaakkola. Forces are not Enough: Benchmark and Critical Evaluation for Machine Learning Force Fields with Molecular Simulations. *Trans. Mach. Learn. Res.*, 2023, 2023.

[14] Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. In *NeurIPS*, 2020.

[15] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. GemNet: Universal Directional Graph Neural Networks for Molecules. In *NeurIPS*, pages 6790–6802, 2021.

[16] Johannes Gasteiger, Muhammed Shuaibi, Anuroop Sriram, Stephan Günnemann, Zachary W. Ulissi, C. Lawrence Zitnick, and Abhishek Das. GemNet-OC: Developing Graph Neural Networks for Large and Diverse Molecular Simulation Datasets. *Trans. Mach. Learn. Res.*, 2022, 2022.

[17] Mario Geiger and Tess E. Smidt. e3nn: Euclidean Neural Networks. *CoRR*, abs/2207.09453, 2022.

[18] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, volume 70, pages 1263–1272, 2017.

[19] Lingshen He, Yiming Dong, Yisen Wang, Dacheng Tao, and Zhouchen Lin. Gauge Equivariant Transformer. In *NeurIPS*, pages 27331–27343, 2021.

[20] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, volume 37, pages 448–456, 2015.

[21] Bowen Jing, Stephan Eismann, Pratham N. Soni, and Ron O. Dror. Equivariant Graph Neural Networks for 3D Macromolecular Structure. *CoRR*, abs/2106.03843, 2021.

[22] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron O. Dror. Learning from Protein Structure with Geometric Vector Perceptrons. In *ICLR*, 2021.

[23] Johannes Klicpera, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules. *CoRR*, abs/2011.14115, 2020.

[24] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *ICLR*, 2020.

[25] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network. In *NeurIPS*, pages 10138–10147, 2018.

[26] Yi-Lun Liao and Tess E. Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. In *ICLR*, 2023.

[27] Yi-Lun Liao, Brandon M. Wood, Abhishek Das, and Tess E. Smidt. EquiformerV2: Improved Equivariant Transformer for Scaling to Higher-Degree Representations. In *ICLR*, 2024.

[28] Yi Liu, Limei Wang, Meng Liu, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical Message Passing for 3D Graph Networks. *CoRR*, abs/2102.05013, 2021.

[29] Ziqiao Meng, Liang Zeng, Zixing Song, Tingyang Xu, Peilin Zhao, and Irwin King. Towards Geometric Normalization Techniques in SE(3) Equivariant Graph Neural Networks for Physical Dynamics Simulations. In *IJCAI*, pages 5981–5989, 2024.

[30] Albert Musaelian, Simon L. Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics. *CoRR*, abs/2204.05249, 2022.

[31] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. *CoRR*, abs/2102.09844, 2021.

[32] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NIPS*, pages 991–1001, 2017.

[33] Kristof Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*, volume 139, pages 9377–9388, 2021.

[34] Kristof T Schütt, Stefaan SP Hessmann, Niklas WA Gebauer, Jonas Lederer, and Michael Gastegger. SchNetPack 2.0: A neural network toolbox for atomistic machine learning. *The Journal of Chemical Physics*, 158(14), 2023.

[35] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017.

[36] Philipp Thölke and Gianni De Fabritiis. TorchMD-NET: Equivariant Transformers for Neural Network based Molecular Potentials. *CoRR*, abs/2202.02541, 2022.

[37] 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.

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022, 2016.

[39] Oliver T Unke and Markus Meuwly. PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of chemical theory and computation*, 15(6): 3678–3693, 2019.

[40] Oliver T. Unke, Stefan Chmiela, Michael Gastegger, Kristof T. Schütt, Huziel E. Sauceda, and Klaus-Robert Müller. SpookyNet: Learning Force Fields with Electronic Degrees of Freedom and Nonlocal Effects. *CoRR*, abs/2105.00304, 2021.

[41] Oliver T Unke, Stefan Chmiela, Huziel E Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T Schutt, Alexandre Tkatchenko, and Klaus-Robert Muller. Machine learning force fields. *Chemical reviews*, 121(16):10142–10186, 2021.

[42] Yusong Wang, Tong Wang, Shaoning Li, Xinheng He, Mingyu Li, Zun Wang, Nanning Zheng, Bin Shao, and Tie-Yan Liu. Enhancing geometric representations for molecules with equivariant vector-scalar interactive message passing. *Nature Communications*, 15(1):313, 2024.

[43] Maurice Weiler, Patrick Forré, Erik Verlinde, and Max Welling. Coordinate Independent Convolutional Networks - Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds. *CoRR*, abs/2106.06020, 2021.

[44] Yuxin Wu and Kaiming He. Group Normalization. In *ECCV (13)*, volume 11217, pages 3–19, 2018.

[45] Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *NeurIPS*, pages 12360–12371, 2019.

Appendix Table of Contents

A Limitations and Future Work **14**

B Notation **14**

C Additional Theoretical Analysis **15**

C.1 Correlated-channel variance analysis for GroupRMS 15

D Detailed Datasets, Tasks, and Hyperparameters **18**

E Reference PyTorch Implementation **20**

F Auxiliary Context and Additional Tables **21**

G Controlled Comparisons and Additional Validation **28**

G.1 Protocols and statistical reporting 28

G.2 Sharing-only controls across eight systems 28

G.3 Statistical support and sharing in related normalizers 29

G.4 Public PaiNN and MACE insertions 30

G.5 Learned-feature variance measurements and sensitivity 31

G.6 NVE trajectories and finite-horizon stability 33

G.7 Computational cost 34

A Limitations and Future Work

The evaluation includes full-block comparisons, sharing-only controls across eight molecular systems, and public PaiNN and MACE insertions. Each suite uses its stated training budget and statistical support; results across suites are not a single matched ranking. At 1k frames, the node-local EquiformerV2-style normalizer is strongest among the tested related operators, and the public PaiNN benefit over its native model varies by molecule. The NVE tests assess finite-horizon behavior rather than general long-time thermodynamic accuracy. Our MACE test uses scalar/vector hidden irreps, so extending the empirical study to richer higher-order hidden representations, wider chemistry coverage, and local-frame gauge equivariance remains future work.

B Notation

Table 7: List of notation.

Symbol	Meaning
$SO(3)$	Group of 3D rotations.
$R \in SO(3)$	Rotation matrix.
T_R	Rotation action on features.
$\mathbb{R}$	Real numbers.
$\|\cdot\|_2$	Euclidean norm.
$g, g(i)$	Graph and graph containing node i.
$\mathcal{V}_g, \|\mathcal{V}_g\|$	Node set and node count.
i, j, c, k	Node, neighbor, channel, and group indices.
N, B	Numbers of nodes and graphs in a minibatch.
$b \in \{0, \dots, B-1\}^N$	Node-to-graph map.
C_s, C_v	Numbers of scalar and vector channels.
s_i, $v_{i,c}$	Scalar feature and vector channel at node i.
S, V	Batched scalar and vector feature arrays.
$\odot$	Elementwise Hadamard product.
$\widetilde{N}_v(\cdot)$	Base vector normalizer.
$N_v^{\mathrm{res}}(\cdot)$	Residual-gated vector normalizer.
$\widetilde{S}_i, \widetilde{V}_{i,c,:}$	Normalized scalar and vector branches.
$\alpha_{g,c}(v)$	Rotation-invariant scale statistic.
$m_{g,c}, \rho_{g,c}$	Graph-wise ChannelRMS second moment and RMS scale.
$m_{g,k}, \rho_{g,k}$	Graph-wise GroupRMS second moment and RMS scale.
$\bar{m}_k, \bar{\rho}_k$	Batch-global GroupRMS second moment and RMS scale.
$G, q, \mathcal{G}_k, k(c)$	Group count, size, channel set, and channel-to-group map.
ε	Numerical stabilizer.
μ_g, σ_g^2	Per-graph scalar mean and variance.
$\gamma_s, \beta_s, \gamma_v$	Learnable gains and scalar bias.
a_s, a_v	Scalar and vector residual gates.
X_c	Per-channel squared-norm statistic.
M_k, m_{eff}	Group-average statistic and effective sample size.
$\mathbb{E}[\cdot], \mathrm{Var}(\cdot)$	Expectation and variance.
x	Atomic coordinates.
$\hat{E}, E, \hat{F}, F$	Predicted and ground-truth energies and forces.
λ_E, λ_F	Energy and force loss weights.
$\mathrm{MSE}(\cdot), \mathrm{MAE}$	Loss and evaluation metrics.
Δ, CI_{95}	Paired difference and 95% CI.
r, K	Cutoff radius and neighbor cap.

C Additional Theoretical Analysis

This section collects the additional theoretical results used to support the $SO(3)$-equivariance-preserving normalization recipe in the main text. Section C.1 relaxes the i.i.d. variance intuition for GroupRMS to correlated channel statistics. The remaining results show why scalar-style vector normalization is structurally constrained and lift equivariant normalization to energy-derived force equivariance.

C.1 Correlated-channel variance analysis for GroupRMS

We relax the i.i.d. assumption and quantify the same variance-reduction mechanism for correlated channel statistics.

Proposition 2 (GroupRMS variance under correlated channels)**.** *Fix one graph with n nodes and one GroupRMS channel group $\mathcal{G}_k$ of size $q \geq 1$. Let*

$$X_c := \frac{1}{3n} \sum_{i=1}^{n} \|v_{i,c}\|_2^2, \qquad M_k := \frac{1}{q} \sum_{c \in \mathcal{G}_k} X_c, \tag{5}$$

where the randomness is over minibatch sampling, stochastic training dynamics, or both. Let Σ_k be the covariance matrix of the random vector formed by $\{X_c : c \in \mathcal{G}_k\}$. Then

$$\mathrm{Var}(M_k) = \frac{1}{q^2} \mathbf{1}^\top \Sigma_k \mathbf{1}. \tag{6}$$

If $q \geq 2$, $\mathrm{Var}(X_c) = \sigma^2 > 0$ for all $c \in \mathcal{G}_k$, and $\bar{r}_k$ denotes the average pairwise correlation inside the group,

$$\bar{r}_k := \frac{2}{q(q-1)} \sum_{\substack{c,d \in \mathcal{G}_k \\ c<d}} \mathrm{Corr}(X_c, X_d), \tag{7}$$

then

$$\mathrm{Var}(M_k) = \frac{\sigma^2}{q}(1 + (q-1)\bar{r}_k) \tag{8}$$

The positive-semidefinite covariance constraint implies $1 + (q-1)\bar{r}_k \geq 0$.

Proof. The identity in Eq. 6 follows directly from the variance of a linear combination:

$$\mathrm{Var}(M_k) = \mathrm{Var}((1/q) \sum_{c \in \mathcal{G}_k} X_c) = \frac{1}{q^2} \sum_{c,d \in \mathcal{G}_k} \mathrm{Cov}(X_c, X_d) = \frac{1}{q^2} \mathbf{1}^\top \Sigma_k \mathbf{1}.$$

When all marginal variances equal σ^2, the diagonal covariance terms sum to $q\sigma^2$, and the off-diagonal terms sum to $q(q-1)\bar{r}_k\sigma^2$. Therefore

$$\mathrm{Var}(M_k) = \frac{q\sigma^2 + q(q-1)\bar{r}_k\sigma^2}{q^2} = \frac{\sigma^2}{q}(1 + (q-1)\bar{r}_k),$$

which is Eq. 8. Since the displayed quantity is a variance, it also gives $1 + (q-1)\bar{r}_k \geq 0$. □

Corollary 1 (Effective sample size of GroupRMS)**.** *Under the equal-marginal-variance setting of Proposition 2, define*

$$m_{\text{eff}} := \begin{cases} \dfrac{q}{1 + (q-1)\bar{r}_k}, & 1 + (q-1)\bar{r}_k > 0, \\ +\infty, & 1 + (q-1)\bar{r}_k = 0. \end{cases} \tag{9}$$

Then $\mathrm{Var}(M_k) = \sigma^2 / m_{\text{eff}}$. Thus, GroupRMS recovers the i.i.d. $1/q$ variance reduction when $\bar{r}_k = 0$, still reduces variance relative to the common single-channel variance σ^2 whenever $\bar{r}_k < 1$, and loses this reduction relative to a single channel only when $\bar{r}_k = 1$, which under the correlation bound means

all pairwise correlations are one. At the lower positive-semidefinite boundary $1 + (q-1)\bar{r}_k = 0$*, the group average has zero variance.*

Proof. The corollary follows by rewriting Eq. 8. The interpretations are the corresponding cases of the same formula together with the bound $1 + (q-1)\bar{r}_k \geq 0$ from Proposition 2. □

Remark 2 (RMS denominator variance)**.** For the RMS denominator $\rho_k = \sqrt{M_k + \varepsilon}$, a first-order delta approximation around $\mu_k = \mathbb{E}[M_k]$ gives

$$\mathrm{Var}(\rho_k) \approx \frac{\mathrm{Var}(M_k)}{4(\mu_k + \varepsilon)}. \tag{10}$$

Thus the variance reduction for M_k in Proposition 2 and Corollary 1 carries to the RMS scale under this local approximation.

Proposition 3 (Necessary form of affine vector normalization)**.** *Let* $\Phi : \mathbb{R}^3 \to \mathbb{R}^3$ *be an affine operation on one vector channel,*

$$\Phi(u) = Au + b, \tag{11}$$

where $A \in \mathbb{R}^{3\times 3}$ *and* $b \in \mathbb{R}^3$ *are held fixed under the comparison* $u \mapsto Ru$*; for example, they may be learned Cartesian parameters or non-rotating statistics. Then,* Φ *is* $SO(3)$*-equivariant,*

$$\Phi(Ru) = R\Phi(u) \qquad \textit{for all } u \in \mathbb{R}^3, R \in SO(3), \tag{12}$$

if and only if there exists a scalar $\lambda \in \mathbb{R}$ *such that*

$$A = \lambda I_3, \qquad b = 0. \tag{13}$$

In particular, a component-wise vector normalizer $u \mapsto \mathrm{diag}(a_1, a_2, a_3)u + b$ *is equivariant only when* $a_1 = a_2 = a_3$ *and* $b = 0$*.*

Proof. If $A = \lambda I_3$ and $b = 0$, then $\Phi(Ru) = \lambda Ru = R(\lambda u) = R\Phi(u)$, so the condition is sufficient. For necessity, set $u = 0$ in Eq. 12. This substitution gives $b = Rb$ for every $R \in SO(3)$, hence $b = 0$ because the only vector fixed by all rotations is the zero vector.

With $b = 0$, Eq. 12 becomes $ARu = RAu$ for all u and R, so $AR = RA$ for every rotation R. Fix a nonzero vector x. For every rotation R around the axis spanned by x, we have $Rx = x$, and therefore

$$R(Ax) = A(Rx) = Ax.$$

The only vectors fixed by all rotations around the axis x are scalar multiples of x, so Ax is parallel to x. Hence every nonzero vector is an eigenvector of A. Applying this eigenvector property to e_1, e_2, e_3 gives $Ae_j = \lambda_j e_j$. Applying it to $e_1 + e_2$ gives $\lambda_1 = \lambda_2$, and applying it to $e_1 + e_3$ gives $\lambda_1 = \lambda_3$. Therefore $A = \lambda I_3$. □

Consequently, coordinate-wise affine operations used for scalar features become valid vector-channel operations only after the Cartesian scales are tied and the non-rotating vector shift is removed. This statement does not rule out subtracting a vector statistic that itself rotates covariantly with the input, nor does it rule out scalar scale factors that depend on rotation invariants; those are different operations from the fixed or non-rotating Cartesian affine normalization considered here. For multiple vector channels, the one-channel result should be read as a restriction on the Cartesian factor: fixed channel mixing by scalar coefficients is compatible with equivariance, but any fixed affine operation acting on the 3-vector coordinates must use a scalar multiple of I_3 and no fixed vector bias.

Proposition 4 (Explicit equivariance defect of component-wise vector RMS)**.** *Let a scalar-style component-wise RMS normalizer act on a collection $U = (u_1, \dots, u_N) \in (\mathbb{R}^3)^N$ by*

$$s_j(U) := \sqrt{\frac{1}{N}\sum_{\ell=1}^{N}(u_{\ell,j})^2 + \varepsilon}, \qquad j = 1, 2, 3,$$
$$C(U)_\ell := \operatorname{diag}(s_1(U), s_2(U), s_3(U))^{-1} u_\ell, \qquad \varepsilon > 0. \tag{14}$$

Then C is not $SO(3)$-equivariant in general, even though it uses no vector shift. In particular, take $N = 1$, $u_1 = e_1 = (1, 0, 0)^\top$, and let R_θ be the rotation in the x-y plane with angle $\theta \in (0, \pi/2)$. Then

$$C(R_\theta U)_1 - R_\theta C(U)_1 = \begin{pmatrix} \cos\theta \left[\frac{1}{\sqrt{\cos^2\theta+\varepsilon}} - \frac{1}{\sqrt{1+\varepsilon}}\right] \\ \sin\theta \left[\frac{1}{\sqrt{\sin^2\theta+\varepsilon}} - \frac{1}{\sqrt{1+\varepsilon}}\right] \\ 0 \end{pmatrix} \neq 0. \tag{15}$$

Proof. For $U = (e_1)$,

$$s(U) = ((1+\varepsilon)^{1/2}, \varepsilon^{1/2}, \varepsilon^{1/2}), \qquad C(U)_1 = ((1+\varepsilon)^{-1/2}, 0, 0)^\top.$$

Since $R_\theta e_1 = (\cos\theta, \sin\theta, 0)^\top$, the rotated collection has

$$s(R_\theta U) = ((\cos^2\theta + \varepsilon)^{1/2}, (\sin^2\theta + \varepsilon)^{1/2}, \varepsilon^{1/2}).$$

Substituting these two expressions into Eq. 14 gives Eq. 15. For $\theta \in (0, \pi/2)$, both $\cos\theta$ and $\sin\theta$ are positive, while $\cos^2\theta < 1$ and $\sin^2\theta < 1$. Hence at least one displayed component, indeed both, is nonzero. □

Thus coordinate-wise RMS normalization imported from scalar neural networks is not merely a different estimator of vector scale. Its Cartesian denominator depends on the chosen basis and creates a genuine equivariance defect. This basis dependence is the basic risk for coordinate-specific scale rules. Unless the effective Cartesian denominator is tied to a scalar invariant, the operation need not commute with rotations. Rotation-compatible vector RMS avoids this defect by replacing the three coordinate denominators with a scalar statistic built from $\|u_\ell\|_2^2$ and applying that scalar identically to all coordinates.

Regularity convention for the force theorem. The next statement is local on an open rigid-motion-stable differentiability region. If a discrete geometric graph construction is present, x and $T_{R,t}x$ are assumed to lie in graph cells on which the constructed graph is fixed, the rigid-motion transformed map remains in the corresponding cells for nearby configurations, and the resulting energy map is C^1. Configurations on cutoff or tie-breaking boundaries are excluded.

Definition 1 ($SO(3)$-compatible normalization)**.** A scalar-vector normalization layer is $SO(3)$-compatible if its scalar outputs are functions only of scalar features and rotation-invariant vector statistics, and its vector outputs use no fixed or non-covariant vector shift and are finite sums of vector inputs multiplied by coordinate-shared scalar coefficients that are invariant functions of scalar features and rotation-invariant vector statistics.

Theorem 1 (Force equivariance under $SO(3)$-compatible normalization)**.** *Let $x = (x_1, \dots, x_n) \in (\mathbb{R}^3)^n$ denote atomic coordinates and let $T_{R,t}x = (Rx_1 + t, \dots, Rx_n + t)$ for $R \in SO(3)$ and $t \in \mathbb{R}^3$. Consider an energy model $\widehat{E}_\theta(x, z)$ whose hidden features consist of scalar channels and vector channels, with z unchanged by rigid motions. Assume that the network depends on x only through the graph construction, geometric inputs, and hidden features generated from them. Under the regularity convention above, assume further that:*

1. *the graph incidence is rigid-motion invariant;*
2. *every geometric input is translation invariant and transforms under rotations as either an invariant scalar or an equivariant vector;*

3. *every non-normalization message and update block is $SO(3)$-equivariant, with scalar outputs invariant and vector outputs transforming as $v_{i,c} \mapsto Rv_{i,c}$;*

4. *every normalization layer is $SO(3)$-compatible in the sense of Definition 1;*

5. *the final energy readout is $SO(3)$-invariant.*

Then

$$\widehat{E}_\theta(T_{R,t}x, z) = \widehat{E}_\theta(x, z), \tag{16}$$

$$\widehat{F}_{\theta,i}(T_{R,t}x, z) = R\widehat{F}_{\theta,i}(x, z), \qquad \widehat{F}_{\theta,i}(x, z) := -\nabla_{x_i}\widehat{E}_\theta(x, z). \tag{17}$$

Thus, under the equivariant-block and invariant-readout assumptions above, inserting $SO(3)$-equivariance-preserving vector normalization preserves the rigid-motion force law when forces are defined by $\widehat{F}_\theta = -\nabla_x \widehat{E}_\theta$.

Proof. Translation invariance follows because the graph construction and the geometric inputs used by the model are unchanged by global translations; typical examples are relative displacements and distances. For rotations, apply induction over the network layers. The graph incidence is unchanged by the rigid motion, while the geometric scalar inputs are invariant and the geometric vector inputs transform as $v_{i,c} \mapsto Rv_{i,c}$. Equivariant message and update blocks preserve this transformation rule by assumption. For a normalization layer, scalar outputs are invariant because they depend only on invariant quantities. Vector outputs commute with R because all vector inputs entering the sum transform by R, the scalar denominators, gains, gates, or mixing coefficients are invariant and shared across Cartesian coordinates, and no fixed or otherwise non-covariant vector shift is added. Hence all hidden features remain correctly transformed, and the final scalar readout is invariant, which proves Eq. 16.

It remains to differentiate the invariant energy at a differentiable configuration. Let $Q_R := I_n \otimes R$ be the block-diagonal action on all coordinates and write $y = Q_R x + \mathbf{1} \otimes t$. The preceding invariance argument gives the local identity $\widehat{E}_\theta(Q_R x' + \mathbf{1} \otimes t, z) = \widehat{E}_\theta(x', z)$ for x' in a neighborhood of the differentiable configuration under consideration. Differentiating this identity at x gives

$$Q_R^\top \nabla_y \widehat{E}_\theta(y, z) = \nabla_x \widehat{E}_\theta(x, z).$$

Since $Q_R^\top = Q_R^{-1}$, multiplying by Q_R yields $\nabla_y \widehat{E}_\theta(y, z) = Q_R \nabla_x \widehat{E}_\theta(x, z)$. Negating both sides gives Eq. 17 componentwise. □

Remark 3 (Batch-statistic normalizers)**.** For batch-statistic normalizers, Theorem 1 applies to the complete batch forward map used to compute the statistics. The single-graph identity holds when the external context is fixed or transformed so that the scalar coefficients are unchanged. More generally, the scalar coefficients must be unchanged under the rigid motion being compared, and any vector input appearing in a vector sum must transform by the same rotation as the corresponding output block. For the graph-wise normalizers used in the main text this caveat is vacuous.

D Detailed Datasets, Tasks, and Hyperparameters

This appendix gives the detailed protocols referenced from the main text, followed by additional tables for QM9 out-of-distribution (OOD) settings, MD17 and rMD17 force benchmarks, ISO17 transfer, OC20 scale experiments, notation, implementation controls, and a higher-sample-count MD22 subset.

Shared optimization defaults. Unless stated otherwise, runs use a hidden width of 128, 5 layers, radial basis size 24, Fourier feature count 8, cutoff radius 6.0, dropout 0.05, normalization $\varepsilon = 10^{-5}$, vector grouping $G = 8$, and AdamW with weight decay 10^{-6}. Training uses global gradient clipping at norm 5.0, ReduceLROnPlateau with factor 0.5 and scheduler patience set to half of early-stopping patience, and early stopping on validation metrics. For force tasks, we optimize $\lambda_E \mathrm{MSE}(E) + \lambda_F \mathrm{MSE}(F)$, with default $(\lambda_E, \lambda_F) = (0.2, 0.8)$ throughout the reported comparison suite.

Seeds and split protocol. Unless explicitly stated, all train/validation/test splits and sampled subsets are fixed with split seed 42. Unless a table specifies otherwise, means and sample standard deviations use run seeds $(0, 1, 2)$ on that fixed split. Sharing-only controls use five seeds, MACE eight, and the public PaiNN and expanded QM9 comparisons ten. Macro-average SDs are calculated across per-seed task averages. Paired comparisons retain task–seed runs completed for every method. The OC20 100k table has one run per method, the uncertainty-weighted OC20 table two, and the ISO17 unseen-isomer InvariantWhiten row two; these exceptions are identified in their captions. Additional controlled protocols and statistical tests are specified in Appendix G.

Main MD22 protocol. The MD22 core protocol used for the main tables is specified in Section 4 and follows the shared optimization and split conventions above. The remaining paragraphs specify auxiliary protocols not described in the main text.

QM9 size-OOD protocol. Relevant table is Table 11. We predict QM9 target `gap` under the size-OOD split with train molecules of ≤ 18 atoms and test molecules of ≥ 19 atoms. These auxiliary runs use training, validation, and test sizes of 40k, 4k, and 8k; batch size 128; 5 layers; dropout 0.05; learning rate 3×10^{-4}; patience 4; and 6 epochs. The table reports both PaiNN-lite and SV-v2 under the dense radius-graph fallback without fixed-k graph construction.

QM9 held-out-fluorine protocol. Relevant table is Table 12. We predict QM9 target `gap` under an element-OOD split that holds out fluorine, denoted held-out F with atomic number $Z = 9$: training and validation molecules contain no fluorine atoms, and test molecules contain at least one fluorine atom. These runs use the same QM9 OOD sizes and optimization settings as the size-OOD protocol above.

ISO17 full-scale protocols. Relevant table is Table 4. ISO17 uses official reference training and validation identifiers for training and validation and uses `test_within` for the within-isomer full-scale evaluation. This setting uses requested training, validation, and test sizes of 50k, 10k, and 10k with batch size 128, $k = 12$, learning rate 2×10^{-4}, patience 5, 8 epochs, and the same default force objective; the effective validation size is 4k because the official validation identifier list contains 4k structures.

OC20 auxiliary scale protocols. Relevant tables are Tables 19, 20, 21, and 4. For the OC20 structure-to-energy-and-forces (S2EF) val-id setting, we train on public parquet mirrors and evaluate on disjoint val-id samples. The 20k table uses training, validation, and test sizes of 20k, 5k, and 5k from the 200k train pool together with val-id, with $k = 12$, batch size 8, 3 epochs, learning rate 2×10^{-4}, patience 1, and the default force objective. The 100k setting uses one run per method, batch size 8, and 2 epochs. The 200k table uses train size 200k, validation and test sizes of 5k each, batch size 32, and 1 epoch. The 2M table uses the full 2M parquet mirror, validation and test sizes of 5k each, batch size 32, 1 epoch, and a streaming train loader. All OC20 tables keep a learning rate of 2×10^{-4}, $k = 12$, and patience 1.

Graph backend equivalence protocol. Relevant table is Table 10. We compare dense cutoff fallback against `radius_graph` on 1024 QM9 samples with seed 1234 and batch size 128, sweeping cutoff radii $\{4.0, 6.0\}$ and neighbor caps $\{16, 32\}$. We report exact edge-set match fraction, Jaccard similarity, and cap-activation statistics.

Algorithm 2 GroupNorm-GroupRMS block.

Require: scalar features $S \in \mathbb{R}^{N \times C}$, vector features $V \in \mathbb{R}^{N \times C \times 3}$, optional batch indices b unused by this batch-global block

Require: group count G with $G \mid C$, group size q with $Gq = C$, $\varepsilon > 0$, scalar GroupNorm affine (γ_s, β_s), vector gain $\gamma_v \in \mathbb{R}^C$

Ignore b so that both scalar and vector statistics are computed over the current minibatch

Partition channels into groups $\{\mathcal{G}_k\}_{k=1}^{G}$

Compute batch-global scalar group mean $\bar{\mu}_k \leftarrow (1/(Nq)) \sum_{i=1}^{N} \sum_{c \in \mathcal{G}_k} S_{i,c}$

Compute batch-global scalar group variance $\bar{\sigma}_k^2 \leftarrow (1/(Nq)) \sum_{i=1}^{N} \sum_{c \in \mathcal{G}_k} (S_{i,c} - \bar{\mu}_k)^2$

$\widetilde{S}_{i,c} \leftarrow (S_{i,c} - \bar{\mu}_{k(c)}) / \sqrt{\bar{\sigma}_{k(c)}^2 + \varepsilon}$; $S' \leftarrow \gamma_s \odot \widetilde{S} + \beta_s$

Compute the batch-global group second moment:

$\bar{m}_k \leftarrow (1/(3Nq)) \sum_{i=1}^{N} \sum_{c \in \mathcal{G}_k} \|V_{i,c,:}\|_2^2$

$\bar{\rho}_k \leftarrow \sqrt{\bar{m}_k + \varepsilon}$

$V'_{i,c,:} \leftarrow (\gamma_v)_c V_{i,c,:} / \bar{\rho}_{k(c)}$

return normalized (S', V')

E Reference PyTorch Implementation

Listing 1 gives a minimal implementation of Algorithm 2, the GroupNorm-GroupRMS block used in the main MD22 comparison. The scalar branch applies standard `nn.GroupNorm` over channels with the nodes in the minibatch treated as the spatial axis. The vector branch computes the batch-global grouped denominator $\bar{\rho}_k$ from Section 3.4; it uses only squared vector norms and applies one scalar denominator identically to all Cartesian coordinates in a channel group.

Listing 1: Reference `nn.Module` implementation of GroupNorm-GroupRMS.

```python
import torch
from torch import nn
from typing import Optional, Tuple


class GroupNormGroupRMS(nn.Module):
    """Scalar GroupNorm plus batch-global grouped vector RMS.

    Inputs:
        scalar: Tensor with shape (N, C)
        vector: Tensor with shape (N, C, 3)
        batch: optional node-to-graph ids; unused by this batch-global block
    """

    def __init__(self, channels: int, groups: int = 8, eps: float = 1e-5):
        super().__init__()
        channels = int(channels)
        groups = int(groups)
        if channels <= 0:
            raise ValueError("channels must be positive")
        if groups <= 0 or channels % groups != 0:
            raise ValueError("groups must be a positive divisor of channels")

        self.channels = channels
        self.groups = groups
        self.group_size = channels // groups
        self.eps = float(eps)

        self.scalar_norm = nn.GroupNorm(groups, channels, eps=self.eps)
        self.vector_gain = nn.Parameter(torch.ones(channels))

    def forward(
        self,
        scalar: torch.Tensor,
        vector: torch.Tensor,
        batch: Optional[torch.Tensor] = None,
    ) -> Tuple[torch.Tensor, torch.Tensor]:
        del batch  # GroupNorm-GroupRMS uses minibatch-wide statistics.

        if scalar.ndim != 2:
            raise ValueError("scalar must have shape (N, C)")
```

```
        if vector.ndim != 3 or vector.size(-1) != 3:
            raise ValueError("vector must have shape (N, C, 3)")
        if scalar.shape != vector.shape[:2]:
            raise ValueError("scalar and vector must share (N, C)")
        if scalar.size(1) != self.channels:
            raise ValueError("input channel count does not match module")

        # nn.GroupNorm expects (batch, channels, *spatial).
        scalar_t = scalar.transpose(0, 1).unsqueeze(0)  # (1, C, N)
        scalar_out = self.scalar_norm(scalar_t).squeeze(0).transpose(0, 1)

        # Batch-global GroupRMS: mean over nodes, channels in group, and xyz.
        vec_g = vector.reshape(-1, self.groups, self.group_size, 3)
        rms = torch.sqrt(vec_g.square().mean(dim=(0, 2, 3)) + self.eps)
        vector_out = vec_g / rms.view(1, self.groups, 1, 1)
        vector_out = vector_out.reshape_as(vector)
        vector_out = vector_out * self.vector_gain.view(1, self.channels, 1)

        return scalar_out, vector_out
```

F Auxiliary Context and Additional Tables

Scope of the auxiliary tables. The main text highlights the MD22 core comparison and also reports aggregate auxiliary results. The appendix collects the corresponding detailed tables for QM9 OOD and failure-mode settings, MD17 and rMD17 force benchmarks, ISO17 transfer and paired-test settings, and OC20 scale and objective-balancing checks. Together, these tables broaden the evaluation beyond the MD22 core trajectories while preserving each benchmark's native protocol and metric scale. Each auxiliary experiment is introduced immediately before its corresponding table or tables.

Table 8: MD22 core random-effects meta-analysis over three trajectories, each with three paired seeds. Energy in eV; force in eV/Å. We report the mean paired difference Δ relative to BatchNorm-VectorRMS with 95% confidence interval (CI) bounds. Negative is better.

Metric	Compare	Δ	CI_{95} low	CI_{95} high
Energy MAE	GroupNorm-GroupRMS	**-0.0236**	-0.0352	-0.0121
Force MAE	GroupNorm-GroupRMS	**-0.0814**	-0.0919	-0.0709

Table 9: rMD17 random-effects meta-analysis over ten molecules, each with three paired seeds. Energy in eV; force in eV/Å. We report the mean paired difference Δ for the comparison to BatchNorm-VectorRMS with 95% CI. Negative is better.

Metric	Compare	Δ	CI_{95} low	CI_{95} high
Test Energy MAE	GraphGroup-GroupRMS	-0.0161	-0.0292	-0.0030
Test Energy MAE	GraphNorm-GroupRMS	**-0.0188**	-0.0233	-0.0143
Test Force MAE	GraphGroup-GroupRMS	**-0.0701**	-0.0839	-0.0564
Test Force MAE	GraphNorm-GroupRMS	-0.0662	-0.0825	-0.0500

MD22 paired statistical and optimization context. Table 8 reports the paired random-effects summary for the main MD22 subset. The aggregate gains are accompanied by a different optimization regime. Figure 3 shows that BatchNorm-VectorRMS seeds cluster at earlier best checkpoints with more clipping, while GroupNorm-GroupRMS moves each task toward later best checkpoints, lower clipping, and lower final force error.

Graph construction backend check. Table 10 records a QM9 implementation control showing that the dense cutoff fallback and PyTorch Geometric `radius_graph` backend agree exactly once the neighbor cap is non-binding. At the default configuration $(r, K) = (6.0, 32)$, the edge sets match exactly with Jaccard similarity 1.000 and zero mean degree discrepancy, which supports the claim that the dense fallback used in some auxiliary runs does not itself explain the normalization differences.

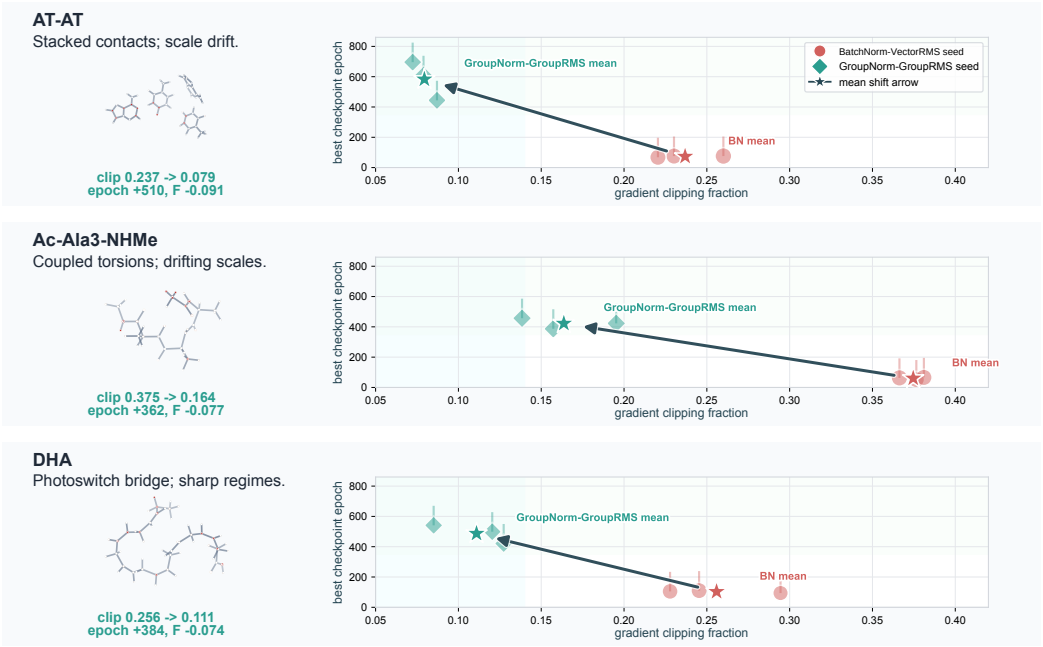


Figure 3: Seed-level optimization regime map on the completed long-budget MD22 subset. BN denotes BatchNorm-VectorRMS. Horizontal position is gradient-clipping fraction, vertical position is best-checkpoint epoch, marker size reflects final test force MAE, points are individual seeds, and vertical segments continue to the early-stop epoch. GroupNorm-GroupRMS consistently shifts the completed seeds away from the early, high-clipping BatchNorm-VectorRMS regime and toward later, lower-clipping, lower-error training behavior.

Table 10: Graph backend equivalence on QM9 between dense cutoff fallback and PyTorch Geometric `radius_graph`. We report the fraction of graphs with exact edge-set match for Exact, mean Jaccard similarity across graphs for Jaccard, and the fraction of nodes that hit the neighbor cap for Cap frac. Higher Cap frac. indicates that the cap is active.

r	K	Cap frac.	Exact	Jaccard	Mean $\lvert \deg_{\text{dense}} - \deg_{\text{radius}} \rvert$
4.0	16	0.202	0.508	0.974	0.0000
4.0	32	0.000	1.000	1.000	0.0000
6.0	16	0.683	0.446	0.890	0.0000
6.0	32	0.000	1.000	1.000	0.0000

QM9 size-OOD context. Table 11 reports size-OOD prediction of the HOMO–LUMO gap in eV. The SV-v2 BatchNorm-VectorRMS, GraphNorm-GroupRMS, and Hybrid-GroupRMS comparisons use ten paired seeds; the other rows retain the three-seed protocol. Hybrid-GroupRMS reduces mean MAE by 12.0% and seed SD by $5.7\times$ relative to BatchNorm-VectorRMS. The paired MAE difference is -0.02595 eV (95% CI $[-0.04794, -0.00396]$, $p = 0.0257$).

Table 11: QM9 size-OOD test MAE in eV, mean ± SD. Rows marked $n = 10$ use ten seeds; all others use three.

Backbone	Norm	Test MAE
PaiNN-lite	GraphNorm-GroupRMS	**0.1936 ± 0.0082**
PaiNN-lite	Hybrid-GroupRMS	0.2000 ± 0.0168
PaiNN-lite	GraphGroup-GroupRMS	0.2223 ± 0.0220
PaiNN-lite	BatchNorm-VectorRMS	0.2413 ± 0.0269
SV-v2 ($n = 10$)	Hybrid-GroupRMS	**0.1900 ± 0.0058**
SV-v2 ($n = 10$)	GraphNorm-GroupRMS	0.2020 ± 0.0194
SV-v2	GraphGroup-GroupRMS	0.2095 ± 0.0034
SV-v2 ($n = 10$)	BatchNorm-VectorRMS	0.2160 ± 0.0335

QM9 held-out-element OOD context. Table 12 shows the held-out-fluorine comparison and highlights a sharp BatchNorm-VectorRMS failure mode. On PaiNN-lite, BatchNorm-VectorRMS reaches 59.27 ± 100.80, while the GroupRMS-family variants remain in the sub-1 regime; on SV-v2, GraphNorm-GroupRMS is also best among the reported rows. This table adds an OOD robustness check beyond the force-trajectory benchmarks.

Table 12: QM9 held-out-fluorine OOD ($Z = 9$), test MAE in eV, mean ± SD over three seeds.

Backbone	Norm	Test MAE
PaiNN-lite	Hybrid-GroupRMS	**0.7271 ± 0.0447**
PaiNN-lite	GraphNorm-GroupRMS	0.8000 ± 0.0417
PaiNN-lite	GraphGroup-GroupRMS	0.8326 ± 0.0810
PaiNN-lite	BatchNorm-VectorRMS	59.2715 ± 100.8009
SV-v2	GraphNorm-GroupRMS	**0.5753 ± 0.0951**
SV-v2	Hybrid-GroupRMS	0.5940 ± 0.0995
SV-v2	BatchNorm-VectorRMS	0.6464 ± 0.2913
SV-v2	GraphGroup-GroupRMS	0.7177 ± 0.0619

MD17 force context. Table 4 reports the MD17 aggregate in the main text. Table 13 shows that the force gains are broad rather than confined to one molecule.

Table 13: MD17 force MAE in kcal/mol/Å, mean ± SD over three seeds per molecule for PaiNN-lite.

Task	Norm	Force MAE
MD17 Aspirin	GraphGroup-GroupRMS	**3.4022 ± 0.1161**
MD17 Aspirin	GroupNorm-GroupRMS	3.4358 ± 0.1257
MD17 Aspirin	GraphNorm-GroupRMS	3.5808 ± 0.3394
MD17 Aspirin	BatchNorm-VectorRMS	4.7345 ± 0.0930
MD17 Azobenzene	GraphNorm-GroupRMS	**2.3645 ± 0.0518**
MD17 Azobenzene	GroupNorm-GroupRMS	2.5013 ± 0.3367
MD17 Azobenzene	GraphGroup-GroupRMS	2.5298 ± 0.2949
MD17 Azobenzene	BatchNorm-VectorRMS	3.7095 ± 0.1465
MD17 Benzene	GraphNorm-GroupRMS	**0.9555 ± 0.1571**
MD17 Benzene	GraphGroup-GroupRMS	0.9571 ± 0.3318
MD17 Benzene	GroupNorm-GroupRMS	1.1270 ± 0.1480
MD17 Benzene	BatchNorm-VectorRMS	3.3844 ± 0.8508
MD17 Ethanol	GraphGroup-GroupRMS	**1.4107 ± 0.0949**
MD17 Ethanol	GroupNorm-GroupRMS	1.4243 ± 0.1594
MD17 Ethanol	GraphNorm-GroupRMS	1.4866 ± 0.2246
MD17 Ethanol	BatchNorm-VectorRMS	3.0097 ± 0.2275

Task	Norm	Force MAE
MD17 Malonaldehyde	GroupNorm-GroupRMS	**1.9166 ± 0.1293**
MD17 Malonaldehyde	GraphGroup-GroupRMS	1.9484 ± 0.0819
MD17 Malonaldehyde	GraphNorm-GroupRMS	1.9910 ± 0.0670
MD17 Malonaldehyde	BatchNorm-VectorRMS	4.6885 ± 0.1796
MD17 Naphthalene	GraphGroup-GroupRMS	**2.0072 ± 0.2069**
MD17 Naphthalene	GroupNorm-GroupRMS	2.0404 ± 0.1334
MD17 Naphthalene	GraphNorm-GroupRMS	2.0448 ± 0.1789
MD17 Naphthalene	BatchNorm-VectorRMS	2.8253 ± 0.1529
MD17 Paracetamol	GroupNorm-GroupRMS	**2.6548 ± 0.2548**
MD17 Paracetamol	GraphGroup-GroupRMS	2.7631 ± 0.1420
MD17 Paracetamol	GraphNorm-GroupRMS	2.8205 ± 0.1034
MD17 Paracetamol	BatchNorm-VectorRMS	4.1219 ± 0.8352
MD17 Salicylic Acid	GroupNorm-GroupRMS	**2.3931 ± 0.0523**
MD17 Salicylic Acid	GraphGroup-GroupRMS	2.5202 ± 0.1503
MD17 Salicylic Acid	GraphNorm-GroupRMS	2.7358 ± 0.0977
MD17 Salicylic Acid	BatchNorm-VectorRMS	3.9252 ± 0.2085
MD17 Toluene	GraphGroup-GroupRMS	**1.9360 ± 0.0820**
MD17 Toluene	GraphNorm-GroupRMS	2.0112 ± 0.0587
MD17 Toluene	GroupNorm-GroupRMS	2.0475 ± 0.1092
MD17 Toluene	BatchNorm-VectorRMS	3.3735 ± 0.1103
MD17 Uracil	GraphGroup-GroupRMS	**1.9354 ± 0.1253**
MD17 Uracil	GroupNorm-GroupRMS	2.0494 ± 0.1318
MD17 Uracil	GraphNorm-GroupRMS	2.2237 ± 0.0644
MD17 Uracil	BatchNorm-VectorRMS	4.0332 ± 0.0819

rMD17 force context. Table 4 reports the rMD17 aggregate; Appendix Table 9 gives its random-effects summary. Table 14 provides the corresponding per-molecule force summary.

Table 14: rMD17 force MAE in eV/Å, mean ± SD over three seeds per molecule for PaiNN-lite.

Task	Norm	Force MAE
rMD17 Aspirin	GraphGroup-GroupRMS	**0.1492 ± 0.0098**
rMD17 Aspirin	GraphNorm-GroupRMS	0.1630 ± 0.0169
rMD17 Aspirin	BatchNorm-VectorRMS	0.1981 ± 0.0083
rMD17 Azobenzene	GraphGroup-GroupRMS	**0.1001 ± 0.0055**
rMD17 Azobenzene	GraphNorm-GroupRMS	0.1192 ± 0.0051
rMD17 Azobenzene	BatchNorm-VectorRMS	0.1547 ± 0.0077
rMD17 Benzene	GraphNorm-GroupRMS	**0.0473 ± 0.0101**
rMD17 Benzene	GraphGroup-GroupRMS	0.0504 ± 0.0107
rMD17 Benzene	BatchNorm-VectorRMS	0.1410 ± 0.0071
rMD17 Ethanol	GraphGroup-GroupRMS	**0.0650 ± 0.0060**
rMD17 Ethanol	GraphNorm-GroupRMS	0.0736 ± 0.0016
rMD17 Ethanol	BatchNorm-VectorRMS	0.1476 ± 0.0127
rMD17 Malonaldehyde	GraphNorm-GroupRMS	**0.0829 ± 0.0034**
rMD17 Malonaldehyde	GraphGroup-GroupRMS	0.0868 ± 0.0042
rMD17 Malonaldehyde	BatchNorm-VectorRMS	0.2928 ± 0.0632
rMD17 Naphthalene	GraphGroup-GroupRMS	**0.0869 ± 0.0070**
rMD17 Naphthalene	GraphNorm-GroupRMS	0.0883 ± 0.0048
rMD17 Naphthalene	BatchNorm-VectorRMS	0.1331 ± 0.0079
rMD17 Paracetamol	GraphGroup-GroupRMS	**0.1293 ± 0.0101**
rMD17 Paracetamol	GraphNorm-GroupRMS	0.1323 ± 0.0052
rMD17 Paracetamol	BatchNorm-VectorRMS	0.1918 ± 0.0373
rMD17 Salicylic	GraphGroup-GroupRMS	**0.1142 ± 0.0096**
rMD17 Salicylic	GraphNorm-GroupRMS	0.1254 ± 0.0141
rMD17 Salicylic	BatchNorm-VectorRMS	0.1735 ± 0.0138

Task	Norm	Force MAE
rMD17 Toluene	GraphNorm-GroupRMS	**0.0876 ± 0.0025**
rMD17 Toluene	GraphGroup-GroupRMS	0.0905 ± 0.0054
rMD17 Toluene	BatchNorm-VectorRMS	0.1431 ± 0.0050
rMD17 Uracil	GraphNorm-GroupRMS	**0.0892 ± 0.0046**
rMD17 Uracil	GraphGroup-GroupRMS	0.0894 ± 0.0065
rMD17 Uracil	BatchNorm-VectorRMS	0.1870 ± 0.0079

MD17 auxiliary ablations. Table 15 provides a supporting analysis around small-data force behavior. The coupled-cluster singles, doubles, and perturbative triples (CCSD(T)) benzene table shows a stronger small-molecule force improvement over BatchNorm-VectorRMS.

Table 15: MD17 CCSD(T) benzene for PaiNN-lite, mean ± SD over three seeds. Energy in kcal/mol and force in kcal/mol/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	1.6146 ± 0.4158	4.1329 ± 0.1332
GraphGroup-GroupRMS	0.3902 ± 0.1072	1.3213 ± 0.0617
GraphNorm-GroupRMS	0.9965 ± 0.4795	1.0110 ± 0.0706
GroupNorm-GroupRMS	1.0928 ± 0.5821	1.3555 ± 0.1002

Higher-sample-count MD22 subset context. Table 16 reports an MD22 subset study on the DHA-large and Stachyose-large tasks. Force MAE improvements persist at higher sample counts. DHA-large drops from 0.1019 ± 0.0061 under BatchNorm-VectorRMS to 0.0646 ± 0.0042 with GraphGroup-GroupRMS, and Stachyose-large drops from 0.1160 ± 0.0018 to 0.0915 ± 0.0036 with GroupNorm-GroupRMS.

Table 16: Higher-sample-count MD22 tasks, mean ± SD over three seeds. Energy in eV and force in eV/Å.

Task	Norm	Energy MAE	Force MAE
DHA	GraphGroup-GroupRMS	0.0652 ± 0.0168	**0.0646 ± 0.0042**
DHA	GroupNorm-GroupRMS	0.0619 ± 0.0103	0.0689 ± 0.0058
DHA	GraphNorm-GroupRMS	0.0749 ± 0.0167	0.0710 ± 0.0108
DHA	BatchNorm-VectorRMS	0.0950 ± 0.0224	0.1019 ± 0.0061
Stachyose	GroupNorm-GroupRMS	0.0722 ± 0.0031	**0.0915 ± 0.0036**
Stachyose	GraphGroup-GroupRMS	0.1225 ± 0.0528	0.0993 ± 0.0047
Stachyose	GraphNorm-GroupRMS	0.1024 ± 0.0196	0.1074 ± 0.0059
Stachyose	BatchNorm-VectorRMS	0.1170 ± 0.0405	0.1160 ± 0.0018

Full-scale ISO17 transfer context. Table 4 provides a larger within-isomer ISO17 reference point beyond the main MD22-core tables. In this full-scale setting, BatchNorm-VectorRMS reaches 0.1356 ± 0.0030 force MAE, while GraphGroup-GroupRMS and GraphNorm-GroupRMS reduce this to 0.0965 ± 0.0020 and 0.1023 ± 0.0068, respectively.

ISO17 transfer context. Tables 17 and 18 report smaller-scale ISO17 transfer results. In these tables, GraphNorm-GroupRMS is best on within-isomer force, while GraphGroup-GroupRMS is best on across-isomer energy and force.

Table 17: ISO17 within-isomer transfer, mean ± SD over three seeds. Energy MAE in eV and force MAE in eV/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	0.0876 ± 0.0190	0.1708 ± 0.0068
GraphGroup-GroupRMS	$\mathbf{0.0827 \pm 0.0146}$	0.1266 ± 0.0064
Hybrid-GroupRMS	0.0922 ± 0.0229	0.1406 ± 0.0067
GraphNorm-GroupRMS	0.0892 ± 0.0190	$\mathbf{0.1238 \pm 0.0040}$
InvariantWhiten	0.0926 ± 0.0214	0.1525 ± 0.0073

Table 18: ISO17 unseen-isomer transfer, mean ± SD over three seeds except InvariantWhiten ($n = 2$). Energy MAE in eV and force MAE in eV/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	0.1437 ± 0.0272	0.1938 ± 0.0086
GraphGroup-GroupRMS	$\mathbf{0.1120 \pm 0.0158}$	$\mathbf{0.1554 \pm 0.0028}$
Hybrid-GroupRMS	0.1385 ± 0.0204	0.1678 ± 0.0049
GraphNorm-GroupRMS	0.1161 ± 0.0054	0.1588 ± 0.0025
InvariantWhiten	0.1944 ± 0.0362	0.1902 ± 0.0004

OC20 20k-scale context. Table 19 provides catalyst-scale evidence at 20k training structures. GraphNorm-GroupRMS improves energy MAE from 60.5284 ± 23.3085 to 27.5186 ± 5.8906 and force MAE from 2.8632 ± 1.0073 to 1.5800 ± 0.2479.

Table 19: OC20 S2EF 20k, mean ± SD over three seeds. Energy MAE in eV and force MAE in eV/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	60.5284 ± 23.3085	2.8632 ± 1.0073
GraphGroup-GroupRMS	31.5158 ± 8.8225	1.9537 ± 0.1137
GraphNorm-GroupRMS	$\mathbf{27.5186 \pm 5.8906}$	$\mathbf{1.5800 \pm 0.2479}$
GroupNorm-GroupRMS	31.4760 ± 4.9314	2.0330 ± 0.1962

OC20 larger-scale context. Tables 20 and 21 continue the OC20 scale check, while Table 4 reports the 2M aggregate in the main text. Across the 100k, 200k, and 2M settings, GraphNorm-GroupRMS remains lower than BatchNorm-VectorRMS for both reported energy and force MAE.

Table 20: OC20 S2EF 100k, one run per method ($n = 1$). Energy MAE in eV and force MAE in eV/Å; no seed dispersion is estimated.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	19.9764	1.6278
GraphNorm-GroupRMS	**15.5054**	**0.8946**

Table 21: OC20 S2EF 200k, mean ± SD over three seeds. Energy MAE in eV and force MAE in eV/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	25.7145 ± 9.1409	1.9222 ± 0.1748
GraphNorm-GroupRMS	$\mathbf{14.5558 \pm 1.6439}$	$\mathbf{1.1052 \pm 0.0740}$

OC20 objective-balancing check. Table 22 reports the uncertainty-weighted objective check. BatchNorm-VectorRMS becomes unstable under this setting, whereas the GroupRMS-family variants remain in the same error scale as the fixed-weight OC20 runs.

Table 22: OC20 S2EF uncertainty-weighted objective, mean ± SD over two seeds. Energy MAE in eV and force MAE in eV/Å.

Norm	Energy MAE	Force MAE
BatchNorm-VectorRMS	5049.3017 ± 7091.2021	516.9659 ± 727.5572
GraphGroup-GroupRMS	26.4982 ± 5.4310	1.9281 ± 0.1353
GraphNorm-GroupRMS	**19.8610 ± 4.7102**	**1.2437 ± 0.1282**
GroupNorm-GroupRMS	22.5576 ± 0.4062	1.8392 ± 0.1163

G Controlled Comparisons and Additional Validation

This section reports the completed controlled experiments that complement the long-budget MD22 study. Each comparison retains its own data, optimization budget, and statistical support. No results from different training budgets are combined into a single ranking.

G.1 Protocols and statistical reporting

Shared controls. Unless specified below, controlled PaiNN-lite runs use width 128, five layers, 24 radial basis functions, cutoff 6 Å, neighbor cap 12, dropout 0.05, AdamW with learning rate 2×10^{-4} and weight decay 10^{-6}, gradient clipping at 5, and the energy/force MSE weights $(0.2, 0.8)$. Splits contain $1000/200/500$ train/validation/test frames with split seed 42. Graph-wise GroupRMS uses $G = 8$, $q = 16$, and $\varepsilon = 10^{-5}$. The fixed scalar GroupNorm branch treats all minibatch nodes as its spatial axis; its support is batch-global even when the vector operator is graph-wise. No residual gate is used in the sharing-only or related-normalizer comparisons. ChannelRMS and GroupRMS use identical learned per-channel gains and parameter counts. Force-task checkpoint selection and learning-rate scheduling use the validation combined energy/force loss.

Table 23: Budgets for the additional controlled suites. Max/patience/min denotes maximum epochs, early-stopping patience, and minimum epochs. All compared methods in a row use the same budget and validation rule.

Suite	Training frames	Batch size	Max/patience/min	Seeds
Sharing-only, MD22	1000	8	256/48/96	5
Sharing-only, MD17	1000	24	256/48/96	5
Related normalizers, MD22	1000	8	192/40/72	3
Related normalizers, MD17	1000	24	256/48/96	3
Public SchNetPack PaiNN	1000	24	256/48/96	10
Official MACE	1000	24	128/24/48	8
Group/epsilon sweeps, MD17	1000	8	512/64/128	3
Batch-size sweep, MD17	1000	2, 8, 32	256/48/96	3

For the 4k training-frame extensions, epoch budgets and stopping windows are scaled by $1000/4000$ to match sample exposure; validation/test sizes remain $200/500$. Thus these experiments compare performance under fixed exposure, not fully converged learning curves. Seed sets are $0, \ldots, n-1$ for each stated n. The width-scaling check uses the related-normalizer MD17 budget, with the GraphNorm-GroupRMS block and its default residual gates.

Statistics. Tables report means and sample SDs over training seeds, not standard errors. Paired tests match molecule, data split, and run seed. For pooled sharing-only and related-normalizer comparisons spanning MD17 and MD22, we use unit-free log ratios $\log(\mathrm{MAE}_{\mathrm{GroupRMS}}/\mathrm{MAE}_{\mathrm{reference}})$, a two-sided one-sample t-test against zero, and a t-interval transformed back to percentage reduction. Pooled MACE reductions use the same log-ratio convention. These intervals describe the observed task–seed pairs on the fixed splits. Public PaiNN reductions instead use the ratio of pooled arithmetic means; its paired t-test and CI use MAE differences in kcal/mol/Å. Rollout starts are first averaged within each trained seed before computing SDs or paired tests. Reported p-values are unadjusted exploratory comparisons.

G.2 Sharing-only controls across eight systems

Table 24 changes only the graph-wise vector denominator from per-channel to grouped estimation. GroupRMS improves all five paired seeds on each of eight systems, giving $40/40$ wins overall. The geometric-mean force reduction is 8.24% (95% CI 7.06–9.41%, $p = 2.1 \times 10^{-16}$).

Table 24: Sharing-only control, mean ± SD over five seeds. MD22 force MAE is in eV/Å; MD17 force MAE is in kcal/mol/Å. Scalar GroupNorm, gains, disabled gate, and all training choices are fixed.

Dataset	System	Atoms	ChannelRMS	GroupRMS
MD22	Ac-Ala3-NHMe	42	0.05331 ± 0.00026	0.04910 ± 0.00110
MD22	DHA	56	0.04180 ± 0.00119	0.03820 ± 0.00060
MD22	AT-AT	60	0.04660 ± 0.00092	0.04059 ± 0.00116
MD22	Stachyose	87	0.06704 ± 0.00157	0.06376 ± 0.00086
MD17	Ethanol	9	0.4844 ± 0.0134	0.4575 ± 0.0152
MD17	Salicylic acid	16	0.8767 ± 0.0450	0.8060 ± 0.0212
MD17	Naphthalene	18	0.5338 ± 0.0229	0.4839 ± 0.0174
MD17	Aspirin	21	1.4194 ± 0.0360	1.2984 ± 0.0298

G.3 Statistical support and sharing in related normalizers

Two useful design axes are the support over which statistics are collected (node, graph, or minibatch) and the set of channels sharing a scale (one channel, a group, or all vector channels). The distinction concerns the vector operator; scalar branches and residual gates must be specified separately.

Table 25: Vector-normalization design axes. "Full" means all vector channels at the stated support. Entries describe the specified operator, not every normalization option in the cited framework.

Operator	Statistical support	Sharing	Running statistics
Vector Neurons norm BN [11]	Minibatch of vector norms	Per channel	Yes
e3nn BatchNorm [17]	Minibatch	Per irrep copy	Yes
GVP vector normalization [22]	Node	Full	No
EquiformerV2 LN ($\ell = 1$) [27]	Node	Full	No
Node-local channel RMS	Node	Per channel	No
Graph-wise ChannelRMS	Graph	Per channel	No
Graph-wise GroupRMS	Graph	Tunable group	No
Batch-global GroupRMS	Current minibatch	Tunable group	No

GVP uses $V/\sqrt{\|V\|_F^2/C_v}$ at each node, up to its affine/stabilization conventions; it therefore belongs to the node/full-sharing entry. Clebsch–Gordan networks also normalize covariant features through invariant magnitudes [25]. These precedents motivate studying the support and sharing choices explicitly.

Exact tested operators. All rows of Table 5 keep scalar GroupNorm fixed. The e3nn arm uses the official `e3nn.nn.BatchNorm` class (e3nn 0.6.0), with $128 \times 1o$ irreps, component normalization, mean reduction, affine gains, momentum 0.1, and no nonscalar bias. The Vector Neurons-style arm computes $r_{i,c} = \sqrt{\|v_{i,c}\|_2^2 + \varepsilon}$ and returns $v_{i,c}\,\mathrm{BN}(r)_{i,c}/r_{i,c}$, using the published norm-normalize-rescale construction. Its norm BatchNorm retains its scalar affine gain and bias. The EquiformerV2-style $\ell = 1$ restriction divides every vector channel at a node by $\sqrt{(3C_v)^{-1}\sum_c \|v_{i,c}\|_2^2 + \varepsilon}$, followed by a per-channel gain. Node-local per-channel RMS instead uses $\sqrt{\|v_{i,c}\|_2^2/3 + \varepsilon}$. Graph-wise ChannelRMS and GroupRMS use Eq. 3 and Eq. 4. The no-normalization arm is the identity. Published affine conventions are retained, so the entire related-normalizer table is not parameter matched; the ChannelRMS/GroupRMS pair is.

GeoNorm was introduced for coordinate states [29]. Our additional *GeoNorm-style latent center-and-scale* control is a transplant, not an official GeoNorm model: it subtracts the covariant per-graph, per-channel center $\mu_{g,c} = |\mathcal{V}_g|^{-1}\sum_i v_{i,c}$ and divides by $\sqrt{(3|\mathcal{V}_g|)^{-1}\sum_i \|v_{i,c} - \mu_{g,c}\|_2^2 + \varepsilon}$, with per-channel gain. It has graph-wise, per-channel statistics. Table 26 reports this control and the node-local per-channel endpoint alongside the six primary rows in Table 5.

Table 26: Additional controlled operators with 1k frames: force MAE, mean ± SD over three seeds. Units match Table 5.

Vector operation	MD22 Ac-Ala3-NHMe	MD17 aspirin
GeoNorm-style latent center-and-scale	0.05865 ± 0.00074	1.4269 ± 0.0357
Node-local per-channel RMS	0.05524 ± 0.00199	1.3909 ± 0.0376

Table 27: MD22 Ac-Ala3-NHMe with 4k training frames and matched sample exposure. Force MAE in eV/Å, mean ± SD over three seeds; fixed scalar GroupNorm and no gate.

Vector operation	Force MAE
No vector normalization	0.04264 ± 0.00295
Node-local per-channel RMS	0.03943 ± 0.00305
EquiformerV2-style LN ($\ell = 1$)	0.03284 ± 0.00137
Graph-wise ChannelRMS	0.03754 ± 0.00196
Graph-wise GroupRMS	0.03412 ± 0.00051

Sharing reduces mean force error on both supports: 16.73% for node-local per-channel RMS to EquiformerV2-style LN, and 9.11% for graph-wise ChannelRMS to GroupRMS. GroupRMS improves the no-vector-normalization control in all three seeds (19.98%, $p = 0.047$). The three-seed GroupRMS/EquiformerV2 comparison does not resolve a difference ($p = 0.15$).

Scalar BatchNorm control. A separate three-seed MD17 aspirin control retains scalar BatchNorm and removes vector normalization entirely. Under the 512-epoch, batch-size-8 protocol, BatchNorm-VectorRMS gives 4.1415 ± 0.0578 kcal/mol/Å, while scalar BatchNorm with the vector identity gives 4.1930 ± 0.1729. The latter also removes the vector gain parameters; it tests removal of the vector-normalization branch. The matched sharing-only experiment in Table 24 provides the more direct test of denominator sharing.

G.4 Public PaiNN and MACE insertions

SchNetPack PaiNN. We use the public SchNetPack 2.1.1 PaiNN implementation [33, 34], width 128, five interactions, 20 radial basis functions, and cutoff 6 Å. The vector-only ChannelRMS/GroupRMS branches follow each mixing block and leave native scalar features unchanged. Both use graph-wise statistics, the same gain and residual gate initialized to 0.25, and identical parameter counts (968,961 versus 968,321 for native PaiNN). Disabling insertion reproduces the native model to a maximum absolute difference of 2×10^{-6} in float32. Table 28 includes the native model and the full GraphNorm-GroupRMS block to distinguish vector-only and full-block effects.

Table 28: Public SchNetPack PaiNN, mean ± SD over ten paired seeds. Energy in kcal/mol; force in kcal/mol/Å. Both vector-only insertions use the native scalar path.

	Aspirin		Ethanol	
Variant	Energy MAE	Force MAE	Energy MAE	Force MAE
Native	$.4696 \pm .1685$	$.9294 \pm .0436$	$.1292 \pm .0344$	$.4614 \pm .0196$
+ ChannelRMS	$.4476 \pm .1231$	$1.0561 \pm .0218$	$.1486 \pm .0546$	$.4414 \pm .0399$
+ GroupRMS	$.4396 \pm .1468$	$1.0108 \pm .0364$	$.1220 \pm .0422$	$.4071 \pm .0383$
+ GraphNorm-GroupRMS	$.4666 \pm .1795$	$.9735 \pm .0405$	$.1204 \pm .0283$	$.3824 \pm .0198$

GroupRMS improves ChannelRMS in $19/20$ pairs, with a pooled arithmetic-mean reduction of 5.32% and paired difference -0.03981 kcal/mol/Å (95% CI $[-0.05591, -0.02372]$, $p = 5.3 \times 10^{-5}$). Improvements are 4.29% on aspirin and 7.78% on ethanol. Relative to native PaiNN, benefits vary by molecule, as the explicit native rows show.

Official MACE. We wrap outputs of the official `mace-torch` 0.3.16 product blocks [3], retaining native interactions, product bases, radial machinery, scalar irreps, and readouts. The configuration uses hidden irreps $64 \times 0e + 64 \times 1o$, three interactions, spherical harmonics up to $\ell = 3$, eight Bessel functions, polynomial cutoff order five, correlation order three, cutoff 5 Å, and MLP irreps $16 \times 0e$. Thus the hidden nonscalar channels tested here are vectors; the implementation's degree-wise extension uses $2\ell + 1$ in place of the factor 3 for an irrep of degree ℓ. GroupRMS uses eight groups of eight multiplicity channels, $\varepsilon = 10^{-5}$, per-channel gains, and residual gates initialized to 0.25. The final scalar-only block is unchanged.

The disabled wrapper agrees with native MACE to 6.7×10^{-16} in float64; energy-invariance and force-equivariance residuals are below 10^{-15}. The inserted arms both add 256 parameters. Across the 24 paired runs in Table 6, GroupRMS reduces force MAE by 6.34% versus native MACE (95% CI 4.08–8.54%) and 5.27% versus ChannelRMS (95% CI 3.49–7.02%), winning 22/24 pairs in each comparison. On aspirin 4k alone, it improves native MACE in all eight seeds (7.61%, $p = 0.0035$).

G.5 Learned-feature variance measurements and sensitivity

Measured effective sample size. We probe a trained MD22 Ac-Ala3-NHMe model on 500 held-out graphs and collect channel moments at all five layers. Table 29 averages layer-wise statistics. Realized effective sample size is the single-channel-to-group second-moment variance ratio; the prediction uses the measured within-group correlations in Corollary 1. Denominator CV is standard deviation divided by mean across held-out graphs. The layer-wise calculation precedes averaging; inserting an averaged correlation into the nonlinear formula need not give the reported average prediction. Slightly negative correlations can give effective sample size larger than q.

Table 29: Learned-feature probe over 500 graphs and five layers. CV and effective sample-size gains are descriptive layer averages, not independent training-seed estimates.

Group width q	RMS denominator CV	Realized m_{eff}	Predicted m_{eff}
1	.07314	1.00	1.00
2	.05461	2.01	2.00
4	.04021	4.02	4.01
8	.02780	8.78	8.40
16	.01971	18.18	17.12
32	.01506	32.34	29.55
64	.01210	50.12	44.49
128	.01038	69.06	58.81

At $q = 16$, denominator CV is 73% lower than at $q = 1$, with a 6% discrepancy between realized and predicted variance gains. In a separate resampling probe, reducing batch size from 24 to 4 increases the CV of a cross-graph scale estimator by 149%. A graph-wise operator's denominator for fixed features is independent of this co-batch size.

Group count and stabilizer. Tables 30 and 31 report independent three-seed MD17 aspirin sweeps of the batch-global vector GroupRMS variant, with scalar GroupNorm fixed at eight groups and no gate. Smaller G means larger group width $q = C_v/G$. The default $G = 8$, $\varepsilon = 10^{-5}$ was specified before these sweeps; we did not select a separate value per test set. Each sweep has its own repeated runs, so their default rows need not be numerically identical.

Table 30: Group-count sweep at $C_v = 128$, $\varepsilon = 10^{-5}$. Force MAE in kcal/mol/Å, mean ± SD over three seeds.

G	1	2	4	8	16	32	64	128
q	128	64	32	16	8	4	2	1
Mean	.9475	.9344	.9383	.9375	.9266	.9426	.9335	.9669
SD	.0144	.0427	.0252	.0180	.0217	.0165	.0176	.0160

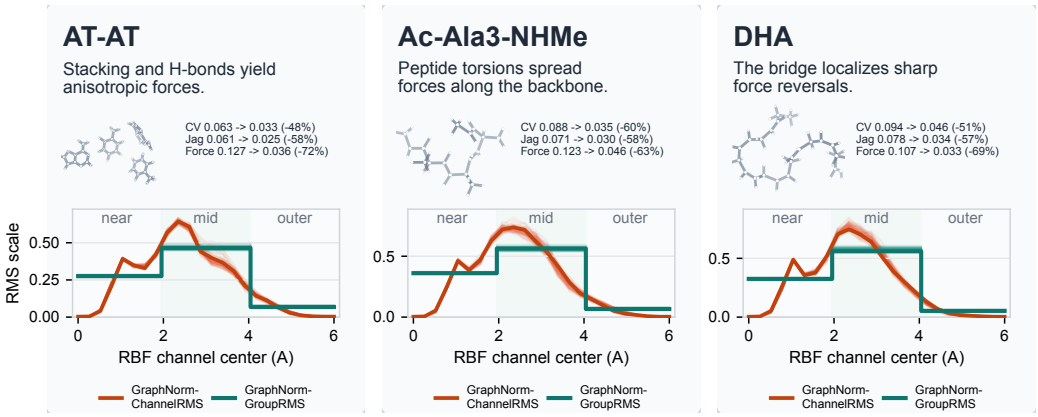


Figure 4: Actual MD22 vector-scale spectra for a transparent geometry-only PaiNN-style first-layer proxy rather than learned hidden activations. For each task, we use the shared proxy $v_{i,c} = \sum_j \phi_c(r_{ij})\hat{r}_{ij}$ with 24 radial channels over 96 real frames. Orange curves are per-channel ChannelRMS denominators, while teal steps share one graph-wise GroupRMS denominator within each 8-channel block. Across the three main MD22 tasks, GroupRMS reduces frame-to-frame coefficient of variation and spectral jaggedness, making the denominator-smoothing mechanism concrete on real molecular graphs.

Table 31: Independent stabilizer sweep at $G = 8$. Force MAE in kcal/mol/Å, mean $\pm$ SD over three seeds.

ε	10^{-8}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
Force MAE	$.9269 \pm .0162$	$.9226 \pm .0367$	$.9277 \pm .0161$	$.9147 \pm .0197$	$.9352 \pm .0242$

The unshared endpoint $G = C_v$ has the highest error in the group sweep. The default $G = 8$ is within 1.2% of the minimum, and the default stabilizer is within 1.4% of the minimum in its sweep. A practical starting point is a moderate group width, e.g., $q = 16$, choosing $G = C_v/q$ when divisible and validating nearby divisors.

Table 32: Graph-wise width-scaling check on MD17 aspirin, three seeds. Force MAE in kcal/mol/Å. The graph-wise scalar branch and default residual gates are fixed; only vector grouping changes.

C_v	No sharing ($G = C_v$)	Fixed $G = 8$	Fixed $q = 16$
64	$1.7548 \pm .0227$	$1.6526 \pm .0357$	$1.6313 \pm .0825$
256	$1.2755 \pm .0390$	$1.1964 \pm .0344$	$1.2131 \pm .0268$

At widths 64 and 256, keeping $q = 16$ improves the unshared endpoint by 7.04% and 4.90%, respectively, in all three seeds. Neither width resolves a significant difference between fixed $q = 16$ and fixed $G = 8$.

Batch and graph sizes. Table 33 compares batch-global ChannelRMS and GroupRMS at each batch size, with the same scalar branch. Evaluation batch size is fixed at eight. The 256-epoch cap fixes data passes, so the number of optimizer steps changes with batch size; comparisons should be made within each row. GroupRMS wins seven of nine paired seeds and has lower mean error at each batch size.

Table 33: Batch-size sweep, MD17 aspirin. Force MAE in kcal/mol/Å, mean ± SD over three seeds; matched budgets within each batch size.

Training batch size	Batch-global ChannelRMS	Batch-global GroupRMS
2	$.9191 \pm .0420$	$.8943 \pm .0273$
8	$1.2295 \pm .0462$	$1.2079 \pm .0352$
32	$2.0087 \pm .1272$	$1.9083 \pm .0573$

A separate three-seed, 18-epoch MD22 comparison spans 42–148 atoms (Table 34). GraphNorm-GroupRMS improves the full BatchNorm-VectorRMS baseline by 11.9–26.2% across these systems. This is a full-block comparison rather than the sharing-only study in Table 24.

Table 34: Graph-size context, force MAE in eV/Å, mean ± SD over three seeds under the same 18-epoch budget.

System	Atoms	BatchNorm-VectorRMS	GraphNorm-GroupRMS
Ac-Ala3-NHMe	42	$.15651 \pm .00822$	$.11598 \pm .00343$
DHA	56	$.15603 \pm .02518$	$.12173 \pm .00783$
AT-AT	60	$.20873 \pm .01817$	$.15520 \pm .00644$
Stachyose	87	$.20392 \pm .01915$	$.17969 \pm .01404$
AT-AT-CG-CG	118	$.24234 \pm .00347$	$.20260 \pm .01049$
Buckyball catcher	148	$.17686 \pm .00431$	$.13045 \pm .00414$

Rotation and co-batch audits. Float64 energy-invariance and force-equivariance residuals for the six primary related-normalizer arms in Table 5 are below 10^{-14}. Including the additional node-local per-channel control gives maximum absolute energy and force residuals 5.1×10^{-13} and 7.2×10^{-14}, respectively. A graph-wise denominator audit comparing a 13-node graph alone and co-batched with a 79-node graph changes its output by less than 7×10^{-9} in relative norm. This last test concerns the graph-wise vector operator for fixed inputs; it does not imply co-batch independence of models containing batch-global scalar statistics.

G.6 NVE trajectories and finite-horizon stability

Four-picosecond and high-temperature tests. We run MD17 aspirin NVE trajectories with velocity Verlet at 0.25 fs: 16,000 steps (4 ps) at 300 K and 6000 steps (1.5 ps) at 500 K. Each of three trained seeds uses four matched initial structures at each temperature, giving 12 trajectories per method and temperature. Initial velocities are sampled from a Maxwell distribution, center-of-mass motion is removed, and kinetic energy is rescaled using $3N - 3$ degrees of freedom. Forces are energy gradients and neighbor graphs are rebuilt each step. The models use the controlled MD17 training budget in Table 23.

Energy drift is the absolute fitted linear slope of total energy (potential plus kinetic), divided by atom count, in meV/atom/ps. Maximum deviation is the largest absolute change from initial total energy in meV/atom. The radial-distribution diagnostic is the L_1 difference between normalized pair-distance histograms. A trajectory completes stably when coordinates and energy remain finite, minimum pair distance stays above 0.35 Å, and initially close pairs ($0.4 < r < 1.9$ Å) stretch by no more than 2.5×. All methods complete both horizons.

Table 35: Aspirin NVE results. Mean ± SD over three seed averages, each from four starts. Force MAE is in kcal/mol/Å; drift in meV/atom/ps; maximum deviation in meV/atom. "Fixed GN" retains the same batch-global scalar GroupNorm; its vector controls use graph-wise statistics.

		300 K, 4 ps		500 K, 1.5 ps
Normalization	Force MAE	Drift	Max. deviation	Drift
BatchNorm-VectorRMS	$3.3264 \pm .0725$	$7.561 \pm .483$	32.60 ± 3.23	7.413 ± 2.898
Fixed GN / no vector norm	$1.2244 \pm .0201$	$1.773 \pm .687$	11.01 ± 3.63	2.374 ± 1.564
Fixed GN / ChannelRMS	$1.3947 \pm .0657$	4.673 ± 2.556	19.41 ± 8.91	4.259 ± 1.209
Fixed GN / GroupRMS	$1.2928 \pm .0309$	2.701 ± 1.158	12.18 ± 2.97	3.955 ± 1.679
GroupNorm-GroupRMS	$1.2854 \pm .0176$	$2.382 \pm .596$	11.66 ± 2.60	4.716 ± 2.173

The batch-global GroupNorm-GroupRMS block reduces force MAE by 61.4% ($p = 4.4 \times 10^{-4}$), 300 K energy drift by 68.5% ($p = 0.0012$), maximum energy deviation by 64.2% ($p = 0.00038$), and 500 K drift by 36.4% ($p = 0.0245$) relative to BatchNorm-VectorRMS. At 300 K, the pair-distance histogram difference is 0.05274 ± 0.00306 for the baseline and 0.05500 ± 0.00112 for GroupNorm-GroupRMS ($p = 0.42$). The no-vector-normalization control is also stable and competitive at this training size. These trajectories complement the controlled accuracy and variance measurements without establishing a uniform ranking of long-time dynamics.

Separate 100-ps stress test. A longer test uses 300 K, timestep 0.5 fs, and up to 200,000 steps, with three paired model seeds and one matched starting structure per seed. Here the treatment uses fixed scalar GroupNorm with *batch-global* vector GroupRMS under the 512-epoch, batch-size-8 training protocol. The termination rule checks finite coordinates/energy, minimum distance above 0.5 Å, and a maximum initial-bond stretch ratio of 2; bonds are defined using covalent radii. Mean survival is 25.78 ± 8.03 ps for BatchNorm-VectorRMS and 41.56 ± 13.31 ps for the GroupRMS treatment, a 61.2% increase. This protocol has a different timestep, stopping rule, and training budget from Table 35; survival statistics are reported separately.

G.7 Computational cost

We measure a batch of 24 aspirin graphs (21 atoms each) on one NVIDIA GH200 with 120 GB memory, using 10 warm-up steps and 100 timed steps in each of five repeats. Force evaluation includes energy differentiation with respect to coordinates; force training includes the parameter-gradient computation. Memory is peak allocated GPU memory in MiB. The controlled training suites and public-backbone experiments use GH200 GPUs; the separate batch-global sweeps and 100-ps stress tests use NVIDIA A100 GPUs with 40 GB memory.

Table 36: Full-block cost comparison on PaiNN-lite. Latencies are mean ± SD over five repeats. Both methods have 1,726,593 parameters.

	Force evaluation		Force training	
Block	ms/step	Peak MiB	ms/step	Peak MiB
BatchNorm-VectorRMS	$16.250 \pm .060$	488.70	$44.705 \pm .076$	960.76
GroupNorm-GroupRMS	$17.624 \pm .060$	491.64	$48.081 \pm .111$	960.77

The full GroupNorm-GroupRMS block adds 8.5% evaluation latency and 7.6% training latency with less than 3 MiB extra peak memory and no additional parameters relative to BatchNorm-VectorRMS. Grouping alone also adds no parameters relative to ChannelRMS.

Table 37: Separate controlled vector-operator timing with fixed scalar GroupNorm and no gate; five repeats, same device and batch shape. Latencies in ms/step; peak memory is for force training.

Vector operator	Parameters	Eval. ms	Train ms	Peak MiB
None	1,725,953	$17.504 \pm .051$	$46.191 \pm .346$	927.54
e3nn BatchNorm	1,726,593	$18.199 \pm .035$	$52.072 \pm .163$	942.32
Vector Neurons-style	1,727,233	$19.295 \pm .030$	$51.626 \pm .193$	952.17
EquiformerV2-style LN	1,726,593	$18.605 \pm .024$	$49.792 \pm .219$	949.72
Graph-wise ChannelRMS	1,726,593	$20.265 \pm .070$	$51.933 \pm .229$	953.44
Graph-wise GroupRMS	1,726,593	$19.898 \pm .023$	$52.367 \pm .139$	957.25

In this operator-level benchmark, graph-wise GroupRMS evaluates 1.8% faster than ChannelRMS, while force-training latency is slightly higher. Relative to no vector normalization, its evaluation/training overhead is $13.7\%/13.4\%$. These measurements distinguish the cost of the complete block from that of sharing the vector denominator.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Introduction state the paper's contributions: an $SO(3)$-equivariance-preserving vector normalization recipe, GroupRMS, and empirical evaluation for molecular force learning. These claims are supported by the method section, theoretical results in the appendix, and the MD22 and auxiliary benchmark results, while the stated scope is qualified in Appendix A.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix A discusses suite-specific training budgets, the tested scalar/vector MACE configuration, finite-horizon rollout scope, and local-frame gauge-equivariant extensions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The main invariant-scaling proposition and the additional propositions, corollary, and force-equivariance theorem in Appendix C state their assumptions and provide proofs or derivations. The force theorem also states the differentiability and graph-construction regularity convention under which it applies.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Sections 4 and D specify datasets, splits, seeds, model dimensions, objectives, optimizer settings, epoch budgets, early stopping, and evaluation metrics for the reported experiments. The supplementary material contains the accompanying code, while Algorithms 1–2 and Appendix E provide implementation detail for the proposed normalization blocks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The paper uses public benchmark datasets, and the supplementary material provides the accompanying code and instructions for reproducing the main experimental results. The paper also includes algorithms, hyperparameters, and a reference PyTorch implementation of the main normalization block.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4 gives the main MD22 protocol, metrics, and benchmark coverage, and Appendices D and G.1 give shared defaults, split seeds, run seeds, optimizers, learning rates, batch sizes, epoch budgets, and dataset-specific protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Main and appendix tables report means and standard deviations over training seeds where applicable, and Tables 8 and 9 report paired random-effects summaries with 95% confidence intervals. Appendices D and G.1 specify seed counts, sample SDs, paired tests, and CI construction; table captions identify the one- and two-seed exceptions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: Appendix G.7 reports hardware, memory, and repeated per-step latency measurements for the additional studies, and the paper specifies dataset sizes and optimization budgets. A complete per-experiment and project-wide compute accounting is not available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work is methodological research on normalization for molecular force learning using public scientific benchmarks and does not involve human subjects, private data, or deployment of a high-risk system.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The paper studies a foundational normalization method for molecular force learning and does not introduce a deployed system, user-facing application, or high-risk model. Any societal effects would be indirect through downstream molecular simulation or catalyst-modeling applications rather than a direct impact of the work itself.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained models, scraped datasets, language models, or image generators. The contribution is a normalization method and benchmark evaluation on scientific datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [No]

    Justification: The paper credits the datasets and related models through citations, including MD22, MD17, rMD17, QM9, ISO17, and OC20, but it does not explicitly list the licenses or terms of use for each existing asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The supplementary material includes the code asset for the proposed normalization method and experimental workflow, together with documentation needed to use it. The paper does not introduce a new dataset or trained model asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development, experiments, and analysis do not use LLMs as an important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.