Extended Abstract Track

# Equivariance by Local Canonicalization: A Matter of Representation

**Editors:** List of editors' names

## Abstract

Equivariant neural networks offer strong inductive biases for learning from molecular and geometric data but often rely on specialized, computationally expensive tensor operations. We present a framework to transfers existing tensor field networks into the more efficient local canonicalization paradigm, preserving equivariance while significantly improving the runtime. Within this framework, we systematically compare different equivariant representations in terms of theoretical complexity, empirical runtime, and predictive accuracy. We publish the `tensor_frames` package, a PyTorchGeometric based implementation for local canonicalization, that enables straightforward integration of equivariance into any standard message passing neural network.

**Keywords:** equivariance, local canonicalization, group representations, message passing

## 1. Introduction

Molecular systems in 3D space exhibit fundamental spatial symmetries — for instance, the energy of a molecule remains invariant under global rotations and reflections, while vectorial properties such as dipole moments transform accordingly. Learning models that respect these symmetry constraints is essential for both accuracy and generalization. This has led to the development of equivariant neural networks, which enforce consistent transformation behavior of outputs with respect to the input geometry. However, two key challenges remain: (a) comparing equivariant models with data augmentation is non-trivial (Lippmann et al., 2025; Brehmer et al., 2024), and (b) existing equivariant architectures often rely on specialized and computationally demanding building blocks (Passaro and Zitnick, 2023).

A recent line of work addresses both issues through local canonicalization (Lippmann et al., 2025; Spinner et al., 2025), offering a lightweight and efficient way to enforce exact equivariance. In this work, we build upon and extend this framework to molecular machine learning: We show how to transfer existing equivariant tensor field networks into the framework of local canonicalization, achieving improved runtime at competitive accuracy and additional flexibility in the choice of possible representations. We systematically compare different equivariant representations in terms of theoretical complexity, runtime, and predictive performance. We release a modular, efficient PyTorchGeometric based implementation that enables easy integration of our formalism into any standard message passing network. Please find our code supplementary at `https://tinyurl.com/tensor-frames-code`.

## 2. Background: group representations and equivariance

Symmetries in physical systems can be described using the mathematical foundations of group theory. Given a group $G$, a *group representation* $\rho$ on a vector space $V$ is a group homomorphism $\rho : G \to GL(V)$ such that

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2) \quad \forall g_1, g_2 \in G, \tag{1}$$

defining how elements $g \in G$ act on vectors $v \in V$, i.e. $(\rho(g)v)_i = \sum_j \rho(g)_{ij} v_j$. A function $\varphi : V \to W$ is *equivariant* under $G$ if $\rho_{\text{out}}(g)\varphi(x) = \varphi(\rho_{\text{in}}(g)x)$ for all $g \in G$ and $x \in V$, where $\rho_{\text{in}}, \rho_{\text{out}}$ are representations on $V$ and $W$ respectively.

**Representations of O(3).** We consider representations of the group O(3), the group of rotations and reflections in $\mathbb{R}^3$, to describe how geometric data transforms in equivariant networks. A vector $v$ transforms under $R \in \mathrm{O}(3)$ as $(Rv)_i = \sum_j R_{ij} v_j$. Correspondingly, higher-order *Cartesian tensors* transform as

$$T'_{i_1 \ldots i_n} = \sum_{j_1, \ldots, j_n} R_{i_1 j_1} \cdots R_{i_n j_n} T_{j_1 \ldots j_n} \quad \text{or} \quad P'_{i_1 \ldots i_n} = \det(R) R_{i_1 j_1} \cdots R_{i_n j_n} P_{j_1 \ldots j_n}. \tag{2}$$

We may distinguish between tensors $T$ and pseudotensors $P$ which behave differently under reflections, i.e. orientation-reversing transformations (Jeevanjee, 2011). The representations in Eq. (2) can be decomposed into so-called *irreducible representations* (see App. B). The irreducible representations of SO(3) are indexed by $l \in \mathbb{N}_0$ and described by the Wigner-$D$ matrices $D^{(l)}(R)$, which act on $(2l+1)$-dimensional tensors $x$ as $(D^{(l)}(R)x)_m = \sum_{m'} D^{(l)}_{mm'}(R) x_{m'}$. Internal representations in neural networks often combine multiple geometric types into a direct sum of representations.

**Equivariance via local canonicalization.** The key idea of equivariance by local canonicalization (Lippmann et al., 2025) is the following: Based on the Euclidean geometry of the network input, one predicts one equivariant local frame $R_i$ at each node $i$. The geometric input node features $F_i$ are transformed from the global frame of reference into the local frames, yielding coordinates $f_i = \rho_{in}(R_i)F_i$ invariant to the choice of global frame. After this canonicalization step, the node features can be processed using an arbitrary backbone architecture without breaking the invariance. However, in order to communicate geometric information during message passing between nodes with distinct local frames, it is crucial that *tensorial* messages are transformed from one local frame into the other. This yields the following general form of invariant message passing *with tensorial messages*, as proposed in (Lippmann et al., 2025):

$$f_i^{(k)} = \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \Big( \rho_{\text{f}}(R_i R_j^{-1}) f_j^{(k-1)}, R_i(x_i - x_j) \Big) \Big). \tag{3}$$

The internal message representation $\rho_{\text{f}}$ can be chosen freely as a hyperparameter.

## 3. Learning representations

In many problems, there is no canonical choice for the representation under which geometric messages should transform in Eq. (3). It can therefore be beneficial for the model to learn the transformation from frame $R_j$ to $R_i$. However, enforcing the strict mathematical properties of a group representation, cf. Eq. (1), for a learned transformation is challenging. A more flexible alternative is to relax this requirement and not enforce Eq. (1). As long as the transition matrix $R_i R_j^{-1}$ is accounted for in the message passing, geometric consistency can in principle be preserved (cf. Fig. 1 in (Lippmann et al., 2025)). The simplest approach would be to learn a linear transformation between local frames. However, this requires

outputting a full matrix of size $d_f \times d_f$ from an MLP, which scales poorly with feature dimension $d_f$. To address this, we propose learning the effect of the frame transition on the features directly: $\rho_f(R_i R_j^{-1}) f_j \Rightarrow \texttt{MLP}(R_i R_j^{-1}, f_j)$. This learned transformation allows the model to flexibly adapt the transformation behavior to the task, eliminating the need for manual tuning.

## 4. Transforming tensor field based architectures to local canonicalization

The typical tensor field network is based on internal features that transform under the irreducible representation and messages are computed via a tensor product convolution:

$$m_{ij,m_3}^{(l_3)} = \left[ f_j^{(l_1)} \otimes \mathcal{R}(r_{ij}) Y^{(l_2)}(\hat{r}_{ij}) \right]_{m_3} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C_{m_1 l_1, m_2 l_2}^{m_3 l_3} f_{j,m_1}^{(l_1)} \mathcal{R}(r_{ij}) Y_{m_2}^{(l_2)}(\hat{r}_{ij}), \quad (4)$$

where $\mathcal{R}(r_{ij})$ is a radial embedding of $r_{ij} = \|x_i - x_j\|$, $Y^{(l_2)}(\hat{r}_{ij})$ are spherical harmonics evaluated on the unit vector $\hat{r}_{ij} = \frac{x_i - x_j}{\|x_i - x_j\|}$, and $C$ are Clebsch–Gordan coefficients that couple irreducible representations of the angular momenta $l_1$, $l_2$, and $l_3$. This operation combines three ingredients: node features, a radial function of distance, and an angular component via spherical harmonics. Motivated by this, we design a neural network layer that mimics this structure using standard operations:

$$\texttt{EDGE}(\rho_{\text{f}}(R_i R_j^{\text{T}}) f_j, R_i(x_i - x_j)) = A(B\rho_{\text{f}}(R_i R_j^{-1}) f_j \odot \texttt{MLP}(\mathcal{R}(r_{ij}) || \Theta(R_i \hat{r}_{ij}))). \quad (5)$$

Following the form of Eq. 3, here, $\mathcal{R}$ and $\Theta$ are radial and angular embedding functions of the local relative distance vector $R_i(x_i - x_j)$, see App. D and $A$ and $B$ are learned linear transformations. $\odot$ denotes element-wise multiplication and $||$ indicates concatenation. The transformation $\rho_{\text{f}}(R_i R_j^{\text{T}})$ transforms features from node $j$ into the local frame of node $i$. This EDGE layer serves as a drop-in replacement for the tensor product in Eq. (4), enabling expressive equivariant message passing while relying on efficient, standard components.

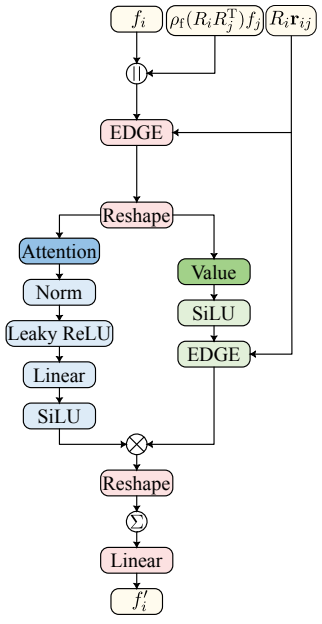



Figure 1: **LoCaFormer attention block.**

Using local canonicalization and the `EDGE` building block, we have implemented an attention-based message passing architecture, the LoCaFormer (Fig. 1), inspired by the widely used Equiformer architecture (Liao and Smidt, 2022), see App. E for details. Beyond an implementation of the LoCaFormer layer, our python package `tensor_frames` includes a module to predict equivariant local frames as proposed in (Lippmann et al., 2025), efficient implementations for the transformation of the irreducible representation, Cartesian tensor representation and the MLP representation described in Sec. 3. The main module of the package is `tensor_frames.nn.TFMessagePassing` which can be used as a drop-in replacement for the widely used message passing module in PytorchGeometric (Fey and Lenssen,

2019) to make any existing message passing layer equivariant using local canonicalization combined with tensorial message passing, see App. A.

## 5. Discussion and results on molecular property prediction

We have trained four LoCaFormer models with different internal representations on the QM9 dataset (Wu et al., 2018), see Tab. 1. The models achieve competitive performance, while being between 4-and 5-times faster than the widely used Equiformer, see App. E for experimental details. Tensorial messages with Cartesian tensors or irreducible representations improve predictions for the isotropic polarizability $\alpha$ and magnetic dipole moment norm $\mu$, likely because these properties depend more strongly on geometric information. To test this, we have trained the same models on the dataset of Zhang et al. (2023), containing 10,000 N-methylacetamide configurations with dipole moments $\mu$ (vectors) and polarizabilities $\alpha$ (rank-2 tensors). Here, tensorial message passing notably outperforms both scalar message passing and the learned MLP representation, confirming its advantage for tensorial targets. This experiment highlights the flexibility of our framework, which allows straightforward comparison across representations to decide whether the extra cost of tensorial features is justified or invariant features suffice. Notably, the less common Cartesian tensor representation can be computationally more efficient than irreducible representations when the transformation of the latter is limited by the Wigner matrix computation, see App. C.

Table 1: **MAE and runtime on QM9 property prediction.**

| Method | $\alpha$ [$a_0$] | $\epsilon_{\text{HOMO}}$ [meV] | $\epsilon_{\text{LUMO}}$ [meV] | $\mu$ [D] | it/s [$s^{-1}$] |
|---|---|---|---|---|---|
| Equiformer (Liao and Smidt, 2022) | <u>.050</u> | **14** | **13** | **.010** | 0.8 |
| MACE (Batatia et al., 2022) | **.038** | 22 | 19 | <u>.015</u> | n/a |
| LoCaFormer: Scalar messages | .057 | 20.8 | 18.2 | .030 | 4.5 |
| LoCaFormer: Cart. tensor rep. | .052 | 20.6 | <u>17.7</u> | .018 | 3.8 |
| LoCaFormer: Irreducible rep. | .054 | <u>19.1</u> | 19.4 | .020 | 3.3 |
| LoCaFormer: MLP rep. | .057 | 20.6 | 17.9 | .030 | 4.0 |

Table 2: **RMSE of magn. dipole moment $\mu$ and polarizability $\alpha$.**

| Method | $\mu$ [a.u.] | $\alpha$ [a.u.] |
|---|---|---|
| EANN (Zhang et al., 2023, 2020) | .004 | .020 |
| LoCaFormer: Scalar messages | .005 | .050 |
| LoCaFormer: Cart. tensor rep. | .003 | .036 |
| LoCaFormer: Irreducible rep. | .003 | .042 |
| LoCaFormer: MLP rep. | .004 | .158 |

Furthermore, we have compared our best performing LoCaFormer models (according to Tab. 1) against models trained without local canonicalization but using data augmentation instead. In our experiments (App. E.1), the models with built-in equivariance are more data efficient, as expected, meaning that the prediction error decreases faster as more training data becomes available. However, perhaps surprisingly, in the low data regime the prediction error of the data augmented model is smaller. This challenges the common belief that models with built-in equivariance should prevail in the low data regime since they do not need any extra data to model the symmetries; and aligns with results in (Lippmann et al., 2025) for a completely different architecture and dataset.

To summarize, the framework of local canonicalization offers a promising alternative for equivariant architectures in molecular ML. Our comparative study highlights trade-offs between different internal representations. While learned representations remain less effective than exact group representations, we believe that learning representations can be an interesting avenue for future research with numerous design choices yet to be explored. By open-sourcing our efficient implementation we would like to aid the development of the field of local canonicalization.

## References

Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35:11423–11436, 2022.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Johann Brehmer, Sönke Behrends, Pim De Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. Texturing and modeling: A procedural approach, with contributions from wr mark and jc hart, 2003.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1eWbxStPH.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Nadir Jeevanjee. *An introduction to tensors and group theory for physicists*. Springer, 2011.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *arXiv preprint arXiv:2206.11990*, 2022.

Peter Lippmann, Gerrit Gerhartz, Roman Remme, and Fred A Hamprecht. Beyond canonicalization: How tensorial messages improve equivariant message passing. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Representation Learning*, volume 2025, pages 88067–88087, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/db7534a06ace69f4ec95bc89e91d5dbb-Paper-Conference.pdf.

Saro Passaro and C Lawrence Zitnick. Reducing SO(3) convolutions to SO(2) for efficient equivariant gnns. In *International Conference on Machine Learning*, pages 27420–27438. PMLR, 2023.

Didier Pinchon and Philip E Hoggan. Rotation matrices for real spherical harmonics: general rotations of atomic orbitals in space-fixed axes. *Journal of Physics A: Mathematical and Theoretical*, 40(7):1597, 2007.

Jonas Spinner, Luigi Favaro, Peter Lippmann, Sebastian Pitz, Gerrit Gerhartz, Tilman Plehn, and Fred A Hamprecht. Lorentz local canonicalization: How to make any network lorentz-equivariant. *arXiv preprint arXiv:2505.20280*, 2025.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Yaolong Zhang, Sheng Ye, Jinxiao Zhang, Ce Hu, Jun Jiang, and Bin Jiang. Efficient and accurate simulations of vibrational and electronic spectra with symmetry-preserving neural network models for tensorial properties. *The Journal of Physical Chemistry B*, 124(33):7284–7290, 2020.

Yaolong Zhang, Jun Jiang, and Bin Jiang. Learning dipole moments and polarizabilities. In *Quantum Chemistry in the Age of Machine Learning*, pages 453–465. Elsevier, 2023.

## Appendix A. Software contribution

We have implemented the general local canonicalization framework of Lippmann et al. (2025) as a modular, publicly available Python library. All components, such as learning local coordinate frames and handling different representations, are provided as reusable `PyTorch` modules, enabling direct integration into existing architectures.

`PyTorchGeometric` (Fey and Lenssen, 2019) is one of the most commonly used Python libraries to implement graph neural networks. For concreteness, let us consider a standard message passing network with message passing layers of the following form

$$f_i^{(k+1)} = \bigoplus_{j \in \mathcal{N}(i)} \phi(f_j^{(k)}, x_i - x_j), \tag{6}$$

where $f_i$ are the local node features at node $i$ and $x_i - x_j$ is the relative distance vector. Using local canonicalization with local frames $R_i$ at each node $i$ and tensorial messages, the modified message passing formula reads

$$f_i^{(k+1)} = \bigoplus_{j \in \mathcal{N}(i)} \phi(\rho_{\mathrm{f}}(R_i R_j^{-1}) f_j^{(k)}, R_i(x_i - x_j)). \tag{7}$$

The above can be implemented effortlessly in our `tensor_frames` packages whose main module `TFMessagePassing` is a wrapper around `torch_geometric.nn.MessagePassing` that handles all transformations automatically after providing the used representations. The following minimal code example in Listing 1 and 2 illustrates how easily one can transfer existing non-equivariant message passing layers to layers which exhibit exact built-in equivariance using the framework of local canonicalization. Notably, the message function, which in this example is very simple but often contains the most complicated logic of the layer, must not be altered at all. For the purpose of this submission, our full Python code is available at https://www.dropbox.com/scl/fo/1kkixfusw900ltoarihdx/ALg4UAGF7W4H5wLdin3M_yU?rlkey=wh8422h1xcer31e8v7171qpyb&st=lkwwd522&dl=0.

Listing 1: Example of a variant of the `EdgeConv` layer (Wang et al., 2019) implemented in `PyTorchGeometric`.

```
from torch_geometric.nn import MessagePassing as MP
from torch_geometric.models import MLP

class EdgeConv(MP):
    def __init__(self, in_dim, out_dim):
        super().__init__()
        self.mlp = MLP(in_dim + 3, out_dim, hidden_channels=[256])

    def forward(self, edge_index, f, pos):
        return self.propagate(edge_index, f=f, pos=pos)

    def message(self, f_j, pos_i, pos_j):
        return self.mlp(torch.cat([f_j, pos_i - pos_j], dim=-1))

message_passing_layer = EdgeConv(32, 32)
```

Listing 2: Example of equivariant adaptation of the `EdgeConv` layer variant using our `TFMessagePassing` for tensorial message passing in local canonicalization. The module is a wrapper around `torch_geometric.nn.MessagePassing`.

```
-    from torch_geometric.nn import MessagePassing
+    from tensor_frames.nn.tfmessage_passing import TFMessagePassing as MP
+    from tensor_frames.reps Irreps, MLPReps, TensorReps
+    from tensor_frames.lframes.lframes import LFrames
     from torch_geometric.models import MLP

     class EdgeConv(MP):
-        def __init__(self, in_dim, out_dim):
+        def __init__(self, in_reps: Irreps | TensorReps | MLPReps, out_reps):
-            super().__init__()
+            super().__init__(
                params_dict={
                    "x": {"type": "local", "rep": in_reps}
                    "positions": {"type": "global", "rep": TensorReps("1x0n")},
                }
            )
-            self.mlp = MLP(in_dim + 3, out_dim, hidden_channels=[256])
+            self.mlp = MLP(in_reps.dim + 3, out_reps.dim, hidden_channels=[256])

-        def forward(self, edge_index, f, pos):
+        def forward(self, edge_index, f, pos, lframes: LFrames):
-            return self.propagate(edge_index, f=f, pos=pos)
+            return self.propagate(edge_index, f=f, pos=pos, lframes=lframes)

        def message(self, f_j, pos_i, pos_j):
            return self.mlp(torch.cat([f_j, pos_i - pos_j], dim=-1))

-    message_passing_layer = EdgeConv(32, 32)
+    message_passing_layer = EdgeConv(Irreps("8x0n+8x1n"), Irreps("1x0n+1x1n"))
```

## Appendix B. Connection between irreducible and Cartesian tensor representation

In this section, we demonstrate how the Cartesian tensor representations used in our experiments can be decomposed into irreducible representations to allow for a fair comparison.

A representation is called *irreducible* if there is no non-trivial subspace of the vector space that the representation acts on that is closed under the action of the group (Jeevanjee, 2011). In other words, the irreducible representations can be thought of as the smallest building blocks of a representation, which cannot be further decomposed into smaller representations.

The irreducible representations of SO(3) are given by the Wigner-$D$ matrices, which act on $2l+1$-dimensional vector spaces. Therefore, $l$ is an index that labels the irreducible representations. An element of the vector space is a vector with $2l+1$ components, which

is called a spherical tensor. For a spherical tensor $x$, the Wigner-$D$ matrix acts as follows:

$$(D^{(l)}(R)x)_m = \sum_{m'=-l}^{l} D^{(l)}_{mm'}(R)x_{m'}, \quad m \in \{-l, \dots, l\}. \tag{8}$$

Inside equivariant neural networks, it is common to use features that combine several different representations, such as vectors and scalars.

Let us illustrate for a rank-2 tensor representation $T_{ij}$ of SO(3) the decomposition into irreducible representations. First the symmetric $S_{ij}$ and the antisymmetric $A_{ij}$ parts of the rank-2 tensor $T_{ij}$

$$S_{ij} = \frac{1}{2}(T_{ij} + T_{ji}), \quad A_{ij} = \frac{1}{2}(T_{ij} - T_{ji}), \tag{9}$$

are defined, which form representations themselves. The symmetric part lives in a subspace closed under the action of the group, which can be seen by

$$S'_{ij} = \frac{1}{2}(T'_{ij} + T'_{ji}) = \frac{1}{2}(R_{ik}R_{jl}T_{kl} + R_{jk}R_{il}T_{kl}) = R_{ik}R_{jl}\frac{1}{2}(T_{kl} + T_{lk}) = R_{ik}R_{jl}S_{kl}. \tag{10}$$

This proof also holds for the antisymmetric part. Counting degrees of freedom, antisymmetric part is a three-dimensional subspace, which can be shown to be equivalent to the irreducible representation of $l = 1$. The symmetric part can be further decomposed into a trace and the traceless part. Both forming its own subspace:

$$\mathrm{Tr}(S') = \sum_i S'_{ii} = \sum_{j,k}\sum_i R_{ij}R_{ik}S_{jk} = \sum_i S_{ii} = \mathrm{Tr}(S), \tag{11}$$

where we have used that $RR^\top = 1$. The trace is the subspace, which corresponds to the irreducible representation of $l = 0$. The last part, which is left, is the traceless symmetric part

$$S_{ij} - \delta_{ij}\frac{\mathrm{Tr}(S)}{3}, \tag{12}$$

which is an irreducible subspace and corresponds to the irreducible representation of $l = 2$ with dimension 5. Note that $T_{ij}$ transforms like the outer product of two vectors $v$ and $w$, e.g. $T'_{ij} = v'_i w'_j = R_{ik}R_{jl}v_k w_l = R_{ik}R_{jl}T_{kl}$. This justifies that the Cartesian tensors of rank 2 is labeled by its dimensions $\underline{3}\otimes\underline{3}$, where $\underline{3}$ corresponds to the irreducible representation of $l = 1$ that has dimension 3. In terms of dimensions the decomposition of a rank-2 Cartesian tensor into irreducible representations is given by:

$$\underline{3} \otimes \underline{3} = \underline{1} \oplus \underline{3} \oplus \underline{5}. \tag{13}$$

The general rule to decompose the outer product or tensor product of two spherical tensors with angular momenta $l_1$ and $l_2$ is:

$$\underline{2l_1 + 1} \otimes \underline{2l_2 + 1} = \bigoplus_{L=|l_1-l_2|}^{l_1+l_2} \underline{2L + 1} \tag{14}$$

Following this rule, we can decompose higher-order Cartesian tensor representations:

$$\begin{aligned}\underline{3}\otimes\underline{3}\otimes\underline{3} &= (\underline{1}\oplus\underline{3}\oplus\underline{5})\otimes\underline{3}\\ &= \underline{1}\otimes\underline{3}\oplus\underline{3}\otimes\underline{3}\oplus\underline{5}\otimes\underline{3}\\ &= \underline{3}\oplus(1\oplus\underline{3}\oplus\underline{5})\oplus(\underline{3}\oplus\underline{5}\oplus\underline{7})\\ &= \underline{1}\oplus\underline{3}\oplus\underline{3}\oplus\underline{3}\oplus\underline{5}\oplus\underline{5}\oplus\underline{7}.\end{aligned} \tag{15}$$

Using these decompositions, the Cartesian tensor representations used in the experiments can be decomposed into irreducible representations to allow for a fair comparison.

## Appendix C. Computational complexity of Cartesian tensors and irreducible representations

Let $n$ denote the order of a Cartesian tensor representation in 3-dimensional Euclidean space. The highest irreducible representation contained in such a tensor is $l = n$, as discussed in App. B. A tensor of order $n$ has $3^n$ components, and the transformation of a single component under a group element $R \in \mathrm{O}(3)$ is given by

$$T'_{i_1\ldots i_n} = \sum_{j_1,\ldots,j_n} R_{i_1 j_1}\ldots R_{i_n j_n} T_{j_1\ldots j_n}, \tag{16}$$

involving $n$ contracted indices, each running over $d = 3$ values. Updating one component therefore requires $\mathcal{O}(n)$ operations, and the full tensor transformation scales as $\mathcal{O}(n\,3^n)$.

For comparison, consider a 3D irreducible representation of angular momentum $l$. Such a representation has $2l+1$ components, which transform according to the Wigner-$D$ matrix $D^{(l)} \in \mathbb{R}^{(2l+1)\times(2l+1)}$, cf. Eq. (8). Applying $D^{(l)}$ to a single component requires $\mathcal{O}(2l+1)$ operations. For the irreducible representation, the number of components grows linearly in $l$, while for the Cartesian tensor representation the number of components are $3^n$ for $n = l$. While Cartesian tensor representations grow much faster in components, they avoid the need to compute Wigner-$D$ matrices, whose computation is non-trivial.

The most efficient approach for computing the Wigner-$D$ matrices for a given $R \in \mathrm{SO}(3)$ is described in (Pinchon and Hoggan, 2007), in which one uses different precomputed building blocks for every $l$. The approach scales like $\mathcal{O}(l^3)$ for the computation of $D^{(l)}(R)$. In practice, only one Wigner matrices must be computed for each combination of $l$ and

Table 3: **Computational complexity for Cartension tensor representation and irreps of SO(3)** "Per entry" refers to the cost of transforming a single feature component. The total transform cost for the irreducible representation includes the computation of the Wigner-D matrix, which is the complexity bottleneck.

| Representation | # Components | Total Transform Cost | Per-Entry Cost |
|---|---|---|---|
| Irrep ($l$) | $2l+1$ | $\mathcal{O}(l^3)$ | $\mathcal{O}(l^2)$ |
| Cart. tensor rep ($n$) | $3^n$ | $\mathcal{O}(n\,3^n)$ | $\mathcal{O}(n)$ |

$R$, so that it can be reused across feature channels of the same $l$. Nonetheless, for large $l$ the $\mathcal{O}(l^3)$ scaling may become a bottleneck. Therefore, if each feature component carries comparable information content, Cartesian tensor representations can be computationally more efficient when the computation of the Wigner matrix dominates the runtime. However, the need for large $l$ in practical applications is not yet fully explored, most likely related to this computational bottleneck.

## Appendix D. Radial and angular embedding of the molecule geometry

The radial embedding of the relative distance $r_{ij} = \|x_i - x_j\|$ is calculated from Bessel functions of the first kind:

$$\mathcal{R}^{(m)}(r_{ij}) = \frac{\omega(r_{ij})}{r_{ij}} \sin\left(\frac{r_{ij}}{r_c}\lambda^{(m)}\right) \tag{17}$$

Here, $\omega$ is a smooth cutoff function (Ebert et al., 2003) that goes smoothly to 0 at the cutoff radius $r_c$ and $\lambda^{(m)}$ are learnable frequencies. The embedding functions are depicted in Fig. 2. In principle, one could use other functions as the embedding functions such as Gaussians. We ablated this choice in preliminary experiments, and found Bessel functions performed equally well with fewer frequencies than Gaussian embedding functions. This coincides with the results of Gasteiger et al. (2020). We embed the angular part of the relative distance vector similarly to the radial part. Each component of the normalized

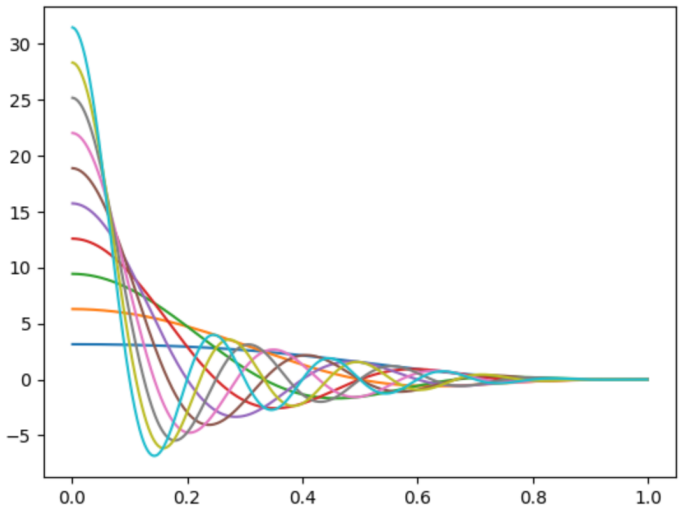



Figure 2: **Bessel embedding functions.** Here, Bessel functions of the first kind are depicted with ten different frequencies. The Bessel functions are also multiplied by the envelope function with a cutoff radius of 1.

relative distance vector $\hat{r}_{ij} = \frac{x_i - x_j}{\|x_i - x_j\|}$ is embedded separately:

$$\theta^{(m)}(\hat{r}_{ij}^k) = \frac{\omega(\hat{r}_{ij}^k)}{\hat{r}_{ij}^k} \sin\left(\hat{r}_{ij}^k \lambda^{(m)}\right) \text{sign}(\hat{r}_{ij}^k), \quad k \in \{x, y, z\}, \quad \Theta = \theta(\hat{r}_{ij}^x) \mid\mid \theta(\hat{r}_{ij}^y) \mid\mid \theta(\hat{r}_{ij}^z) \tag{18}$$

Each component $k \in \{x, y, z\}$ of the normalized relative distance vector is embedded by Bessel functions of the first kind, as for the radial embedding. This embedding needs to be multiplied with the sign of the component of the relative distance vector, since the Bessel functions are symmetric around the origin, but in the angular case negative and positive $x, y$ or $z$ values need to be embedded separately. Including the sign functions introduces a non-continuity if $x = 0$, $y = 0$ or $z = 0$ in the angular embedding, but in experiments we not found found this to be a problem for the training dynamics of the model. The full angular embedding $\theta_{ij}$ is the concatenation of the three components.

## Appendix E. Experimental details

**Hyperparameter choices.** The hyperparameters for the models trained on QM9 property prediction are summarized in Tab. 4. Further, let us introduce the following notation to specify the feature representation used during message passing: The feature representation is given by a direct sum of Cartesian tensor and pseudotensor representations. A Cartesian tensor representation is characterized by its order (i.e. the number of indices, see Eq. (2)) and its behavior under parity (`n` for tensors and `p` for pseudotensors). Furthermore, we specify the multiplicities, that is, how often each representation appears in a direct sum representation. For instance, the representation denoted as `8x0p+4x1n` is the direct sum of 8 pseudoscalars and 4 vectors. For a fair representation comparison, the decomposition of the Cartesian tensor representation into irreducible representations (cf. App. B) can be used to build the corresponding irreducible representation. For the MLP representation the same feature dimension as in the other representations is used.

**Architectural design.** In the Equiformer (Liao and Smidt, 2022), special care is required to use the appropriate specialized linear layers, normalization layers and activation

Table 4: **Hyperparameters for training our LoCaFormer models on QM9 property prediction.**

| | LoCaFormer |
|---|---|
| optimizer | AdamW |
| weight decay | 5e-3 |
| learning rate | 5e-4 |
| scheduler | Cosine-LR |
| epochs | 600 |
| warm up epochs | 5 |
| gradient clip | 0.5 |
| loss | Smooth-L1 |

Table 5: **Architecture of the LoCaFormer model.**

| Parameter | Value |
|---|---|
| Number of radial Bessel functions | 32 |
| Number of angular Bessel functions | 20 |
| Number of layers | 5 |
| Number of heads | 4 |
| Attention branch dimension | 48 |
| Value branch dimension | 96 |
| Hidden Layer MLP | 512 |
| Attention score dropout | 0.1 |
| Stochastic depth | 0.05 |

function, whereas in our case a standard MLP and LayerNorm can be employed without breaking equivariance. Beside the LoCaFormer attention block illustrated in Fig. 1, the full LoCaFormer layer consists of the attention block followed by an MLP, together with two vanilla LayerNorm layers, one inserted before the MLP and one before the attention block. Further, we have added skip connections around both modules to improve gradient flow. The hyperparameters used in our experiments are summarized in Tab. 5. All LoCaFormer models use radial and angular embedding of the relative distance vector with a Bessel function (App. D). The learned local frames are predicted using the procedure described in (Lippmann et al., 2025). As an output head, we use an MLP with hidden dimensions $[512, 128, 32]$, followed by sum-pooling used to aggregate the final node-wise predictions. During training, we use a dropout rate of 0.1 for the attention scores and a stochastic depth of 5% during training.

**Tensorial property prediction.** For the tensorial property prediction task on the dataset taken from (Zhang et al., 2023) the same architecture are used as for property prediction on QM9, except for the following small modifications: The intermediate representations are

Table 6: **Influence of the representations on train time.** We report the iterations per second during training of the model with the QM9 regression task. The timings are averaged over 1 epoch. Batch size is set to 128. We also measure the runtime of the Equiformer (Liao and Smidt, 2022), which uses computationally more demanding tensor products to achieve equivariance. The models were trained on a single NVIDIA RTX 6000 GPU (CPU: 2x AMD Epyc 7452, 1024 GB RAM).

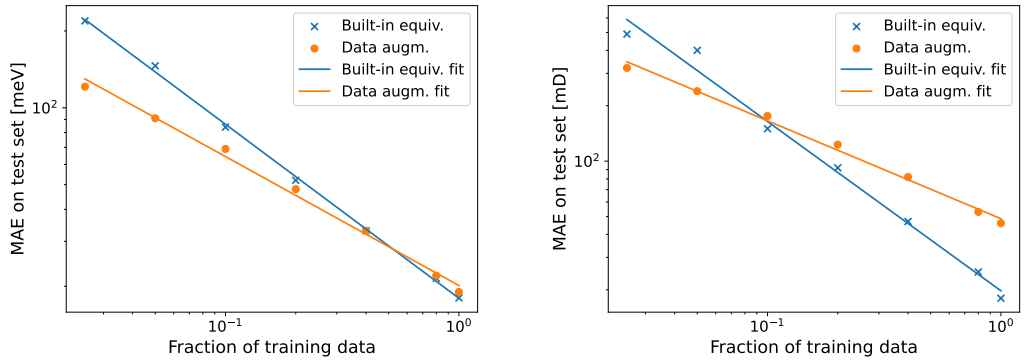
| Representation | LoCaFormer [it/s] | Diff. to scalar | Equiformer [it/s] |
|---|---|---|---|
| Scalar | 4.5 | 0% | n/a |
| Cart. Tensor | 3.8 | -16% | n/a |
| Irreducible | 3.3 | -27% | 0.8 |
| MLP | 4.0 | -11% | n/a |

based on the Cartesian tensor representation "94x0n + 32x1n + 16x2n" and the number of message passing layers is reduced to three due to the smaller size of the tensorial property prediction dataset.

**Influence of the representations on train time.** In Tab. 6 we report the number of training iterations on QM9 property prediction. While the LoCaFormer with scalar messages still predicts local frames to leverage local canonicalization, it does not perform any (non-trivial) frame-to-frame transitions. Using learned MLP representations (cf. Sec. 3) or a proper group representation in the frame-to-frame transition introduces a computational overhead but notably improves the predictions in particular on geometric and tensorial targets (see Tabs. 1 and 2). Moreover, the combination of local canonicalization paired with tensorial message passing offers an efficient framework for implementing exact equivariance with optimized standard deep learning building blocks. This can be seen in the direct runtime comparison against the popular Equiformer architecture (Liao and Smidt, 2022), trained on the same hardware and with the same batch size. Depending on the internal representations used in the message passing of the LoCaFormer, our model is between 4 and 5 times faster than the Equiformer, which employs only irreducible representations; and, unlike our approach, relies on specialized tensor operations to achieve exact equivariance, which are computationally demanding (Passaro and Zitnick, 2023).

### E.1. Relation to data augmentation.

The framework of local canonicalization makes it straightforward to compare data augmentation (by choosing the same random local frame for each node) with built-in equivariance. Equivariant models are often considered more data-efficient because they do not need to spend model capacity or additional data to learn symmetries (Batzner et al., 2022), implying faster performance gains as training data increases (Hestness et al., 2017). To test




(*a*) Data efficiency plot with $\epsilon_{\text{LUMO}}$ as target

(*b*) Data efficiency plot with $\mu$ as target

Figure 3: **Equivariance increases data efficiency compared to data augmentation on QM9.** However, in the low data regime the data augmentation yields superior accuracy.

this, we trained the best equivariant model (according to Tab. 1) and its data-augmented counterpart (same architecture and hyperparameters) on varying fractions of the training set and compared test accuracies. We evaluated two targets: the norm of the magnetic dipole moment $\mu$, where tensorial messages improve performance, and $\epsilon_{\mathrm{LUMO}}$, where tensorial messages have little effect. In both cases, the model with built-in equivariance exhibits steeper error–data scaling, cf. Fig. 3, confirming higher data efficiency (Hestness et al., 2017). However, interestingly, the equivariant model did not consistently achieve lower error across all data fractions: in the low-data regime, the augmented model sometimes outperformed it. This challenges the common assumption that exact equivariance would dominate in low-data settings.