Equivariance by Local Canonicalization: A Matter of Representation

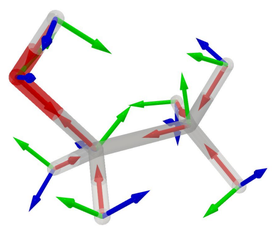
Gerrit Gerhartz*, Peter Lippmann*, Fred A. Hamprecht
Interdisciplinary Center for Scientific Computing (IWR),
Heidelberg University, Germany

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386 · IWR Interdisciplinary Center for Scientific Computing · STRUCTURES CLUSTER OF EXCELLENCE · NEURAL INFORMATION PROCESSING SYSTEMS

Exact & expressive O(3) equivariance

- ➔ Equivariance made easy by **canonicalization & tensorial messages**
- ➔ Works for all representations & with any message passing NN
- ➔ Our **tensor_frames library** offers an easy-to-use implementation

Local canonicalization & Tensorial messages

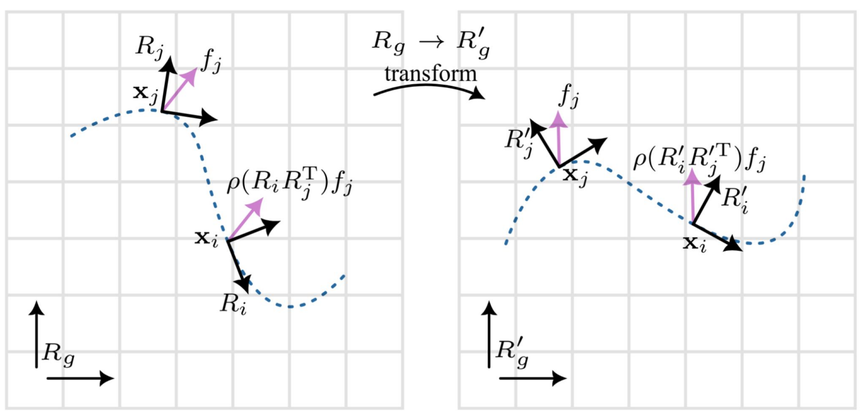


- Predict local frames at each node
  → Canonicalize the node features locally

$$f_i = \rho_{\text{in}}(R_i) F_i$$

- Use frame-to-frame transitions to send tensorial messages:

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



Easily build new architectures

- No architecture constraints
- Our **tensor_frames library** transforms any PyG message passing module into a tensorial message passing module:

```python
from tensor_frames.nn.tfmessage_passing import TFMessagePassing
from tensor_frames.reps.tensorreps import TensorReps

class GCNConv(TFMessagePassing):
    def __init__(self, in_reps: TensorReps, out_reps: TensorReps):
        super().__init__(
            params_dict={
                "x": {"type": "local", "rep": in_reps}
            }
        )
        self.linear = torch.nn.Linear(in_reps.dim, out_reps.dim)

    def forward(self, edge_index, x, lframes):
        return self.propagate(edge_index, x=x, lframes=lframes)

    def message(self, x_j):
        return self.linear(x_j)

module = GCNConv(TensorReps("16x0n+8x1n"), TensorReps("4x0n+1x1n"))
```

Compare different geometric representations

Equivariance: $\rho_{out}(g)\phi(x) = \phi(\rho_{in}(g)x)$

Group representation: $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$

Cartesian tensor rep. **or** ***Irreducible rep.***

$$T'_{i_1 \dots i_n} = \sum_{j_1,\dots,j_n} R_{i_1 j_1} \cdots R_{i_n j_n} T_{j_1 \dots j_n} \qquad (D^{(l)}(R)x)_m = \sum_{m'=-l}^{l} D^{(l)}_{mm'}(R) x_{m'}$$

- Both have different computational advantages

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	$\mathcal{O}(n\,3^n)$	$\mathcal{O}(n)$

- Must the transformation be a group rep.?
  → Learned "representation", let the network decide how to transform:

$$\rho_f(R_i R_j^{-1}) f_j \;\Rightarrow\; \texttt{MLP}(R_i R_j^{-1}, f_j)$$

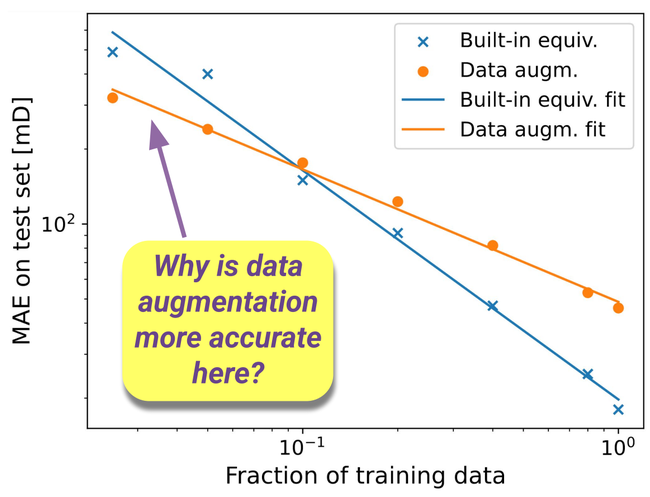
- Performance of different transformations:

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

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

Fair comparison against data augmentation

- Data augmentation by choosing random global frames, keeping everything else as is!
  → **Exact equivariance is more data efficient**



Can YOU solve this puzzle?

Transferable across domains

Achieves SOTA results in several tasks:

- **3D Computer vision**
- **Quantum Chemistry: Orbital-free DFT**
- **Particle Physics: Lorentz Equivariance**