E(n) Equivariant Graph Neural Networks



Víctor Garcia Satorras



Emiel Hogeboom



Max Welling

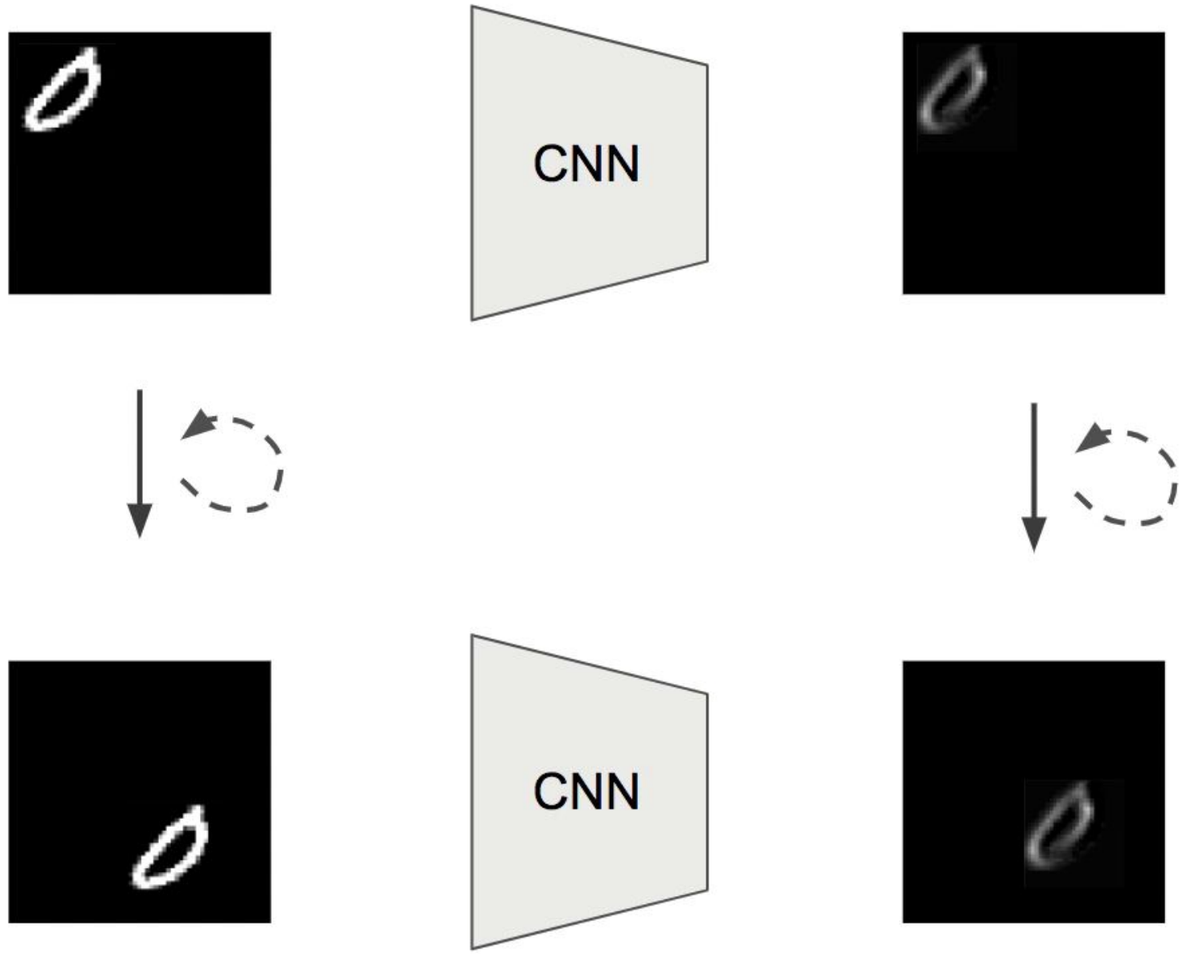


UNIVERSITY OF AMSTERDAM

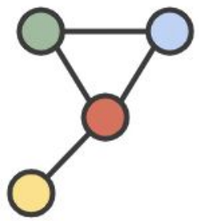


UvA - BOSCH
DELTA LAB

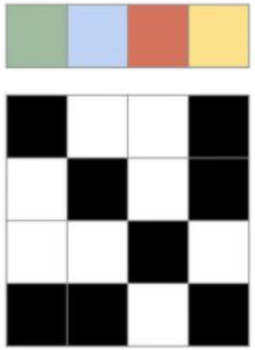
Background | Equivariance

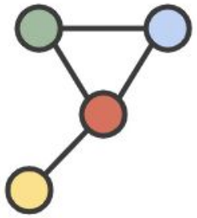


Background | Equivariance

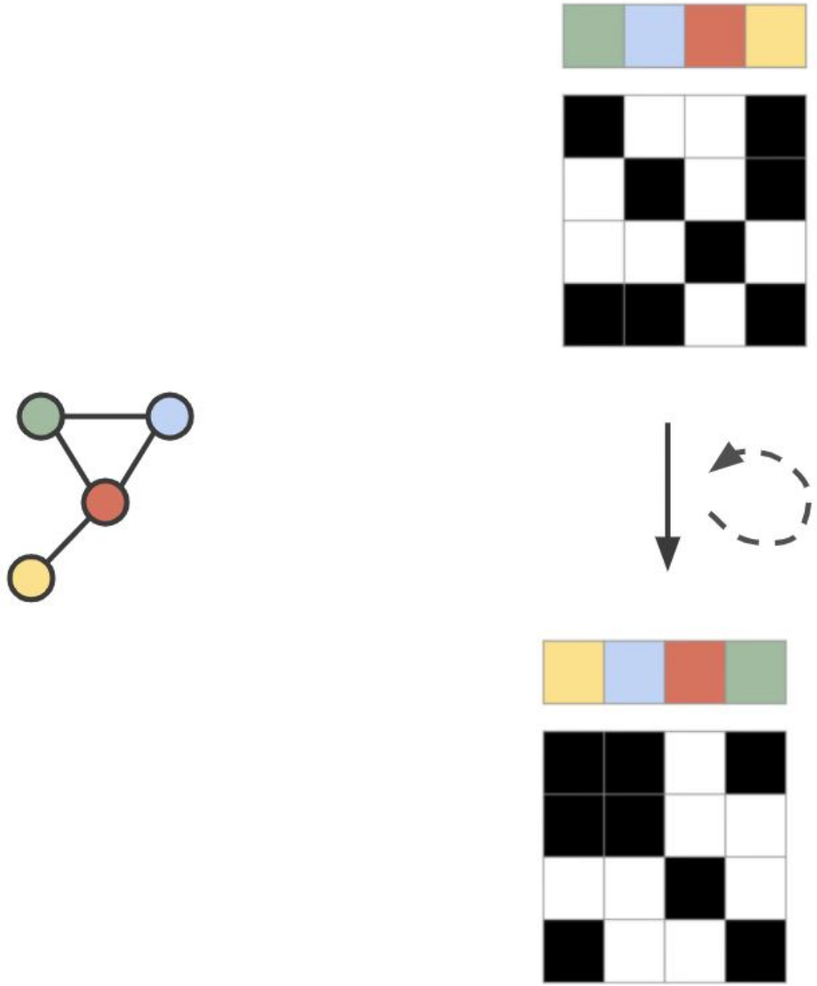


Background | Equivariance

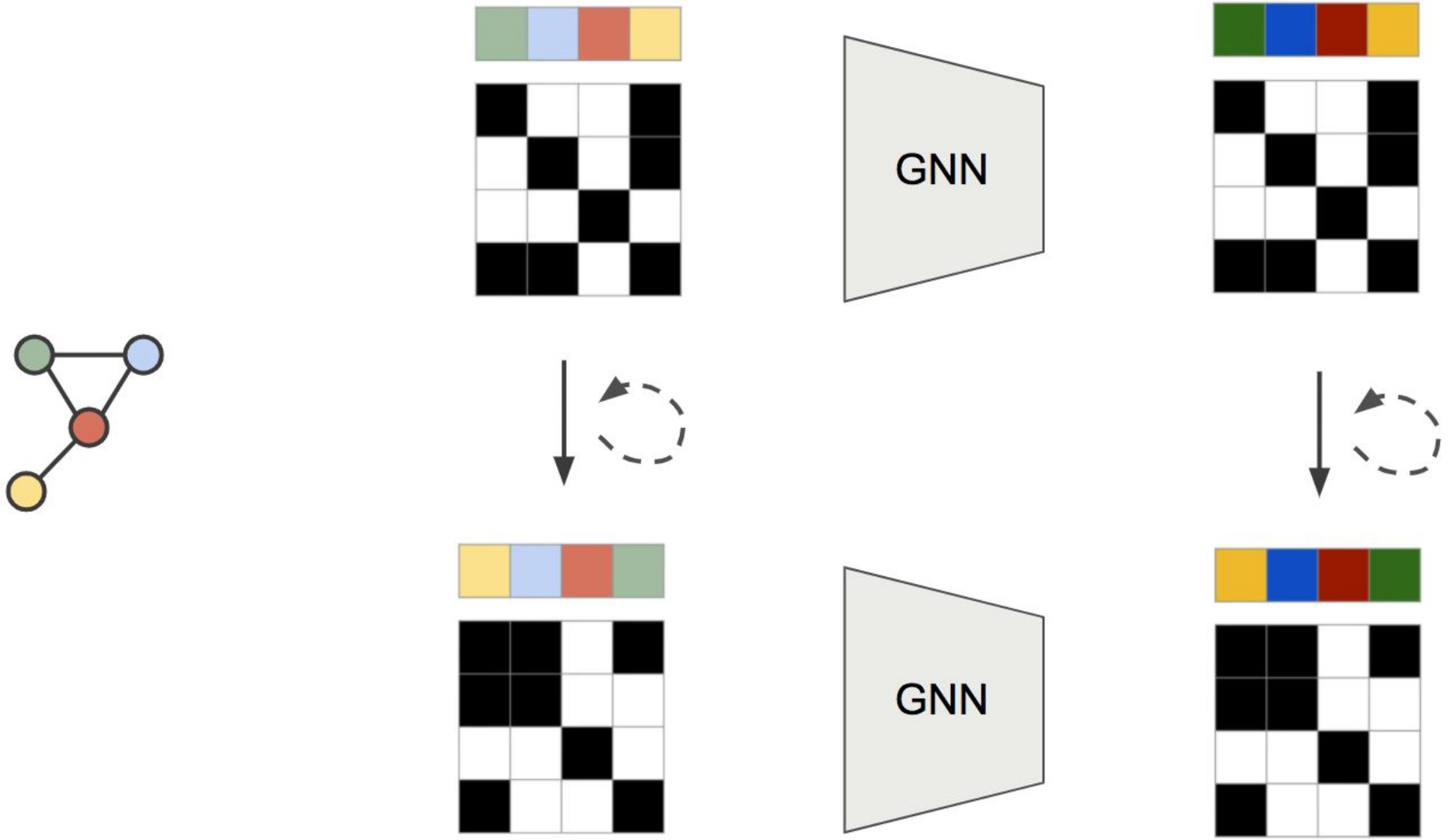




Background | Equivariance



Background | Equivariance



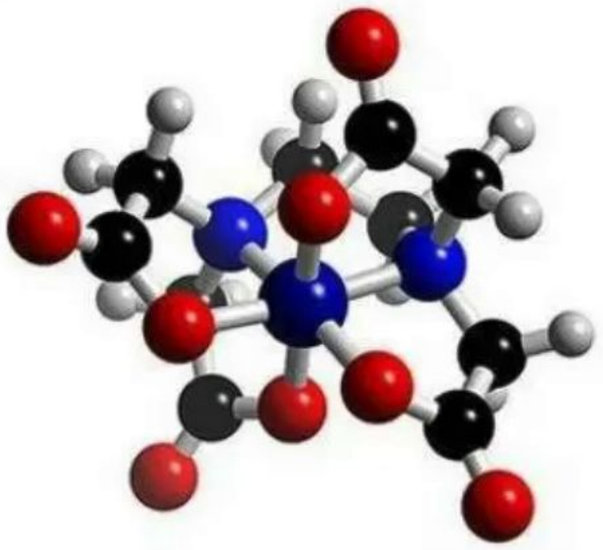
Background | The three symmetries in the paper

Inputs $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_M) \in \mathbb{R}^{M \times n}$ points in n dimensions, output $\mathbf{y}$. We want: $\mathbf{y} = \phi(\mathbf{x})$

Translation equivariance	$\mathbf{y} + g = \phi(\mathbf{x} + g)$	$g \in \mathbb{R}^n$ added to every point
Rotation and reflection equivariance	$Q\mathbf{y} = \phi(Q\mathbf{x})$	$Q \in \mathbb{R}^{n \times n}$ orthogonal, applied to every point
Permutation equivariance	$P(\mathbf{y}) = \phi\big(P(\mathbf{x})\big)$	P permutes the rows (the nodes)

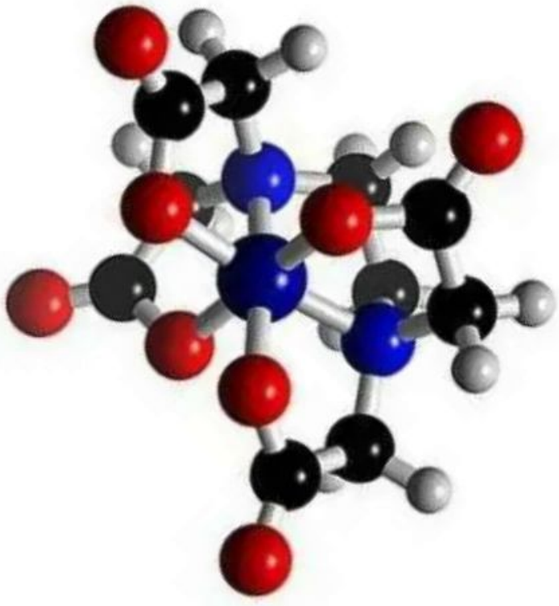
Velocities $\mathbf{v}$ are not affected by translations but rotate and permute like $\mathbf{x}$. EGNN satisfies all three for the coordinates, and the node features $\mathbf{h}$ stay invariant.

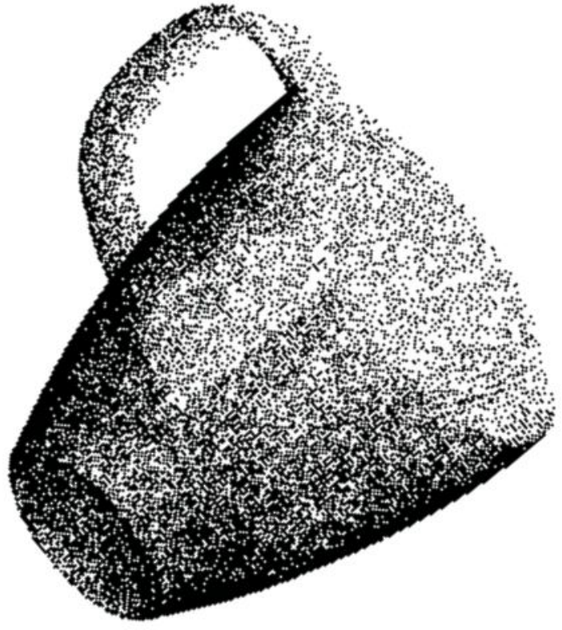
Background | Equivariance



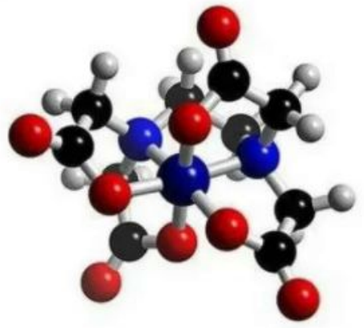


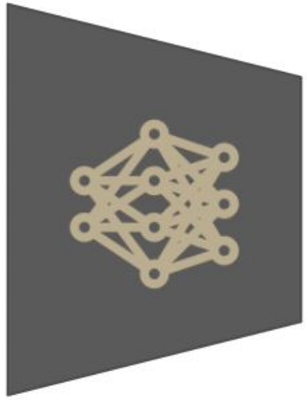
Background | Equivariance





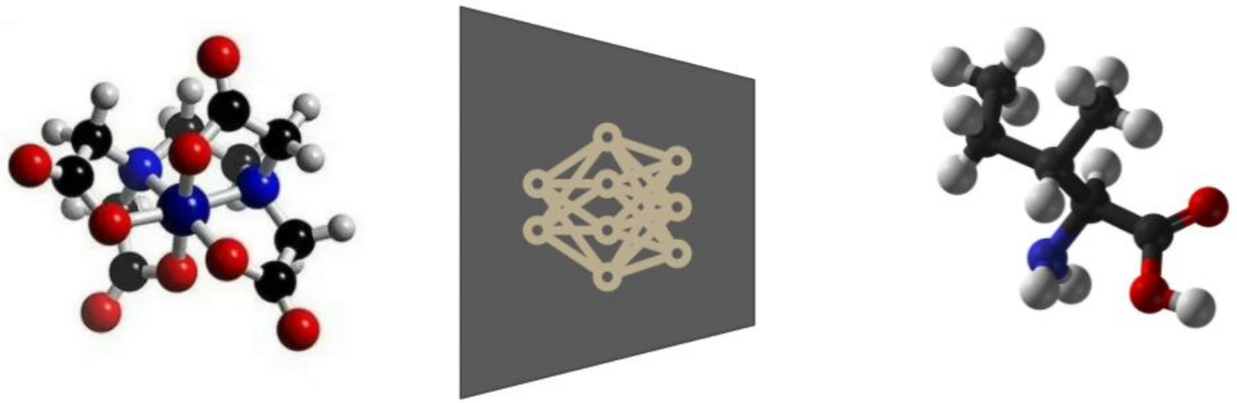
Background | Equivariance



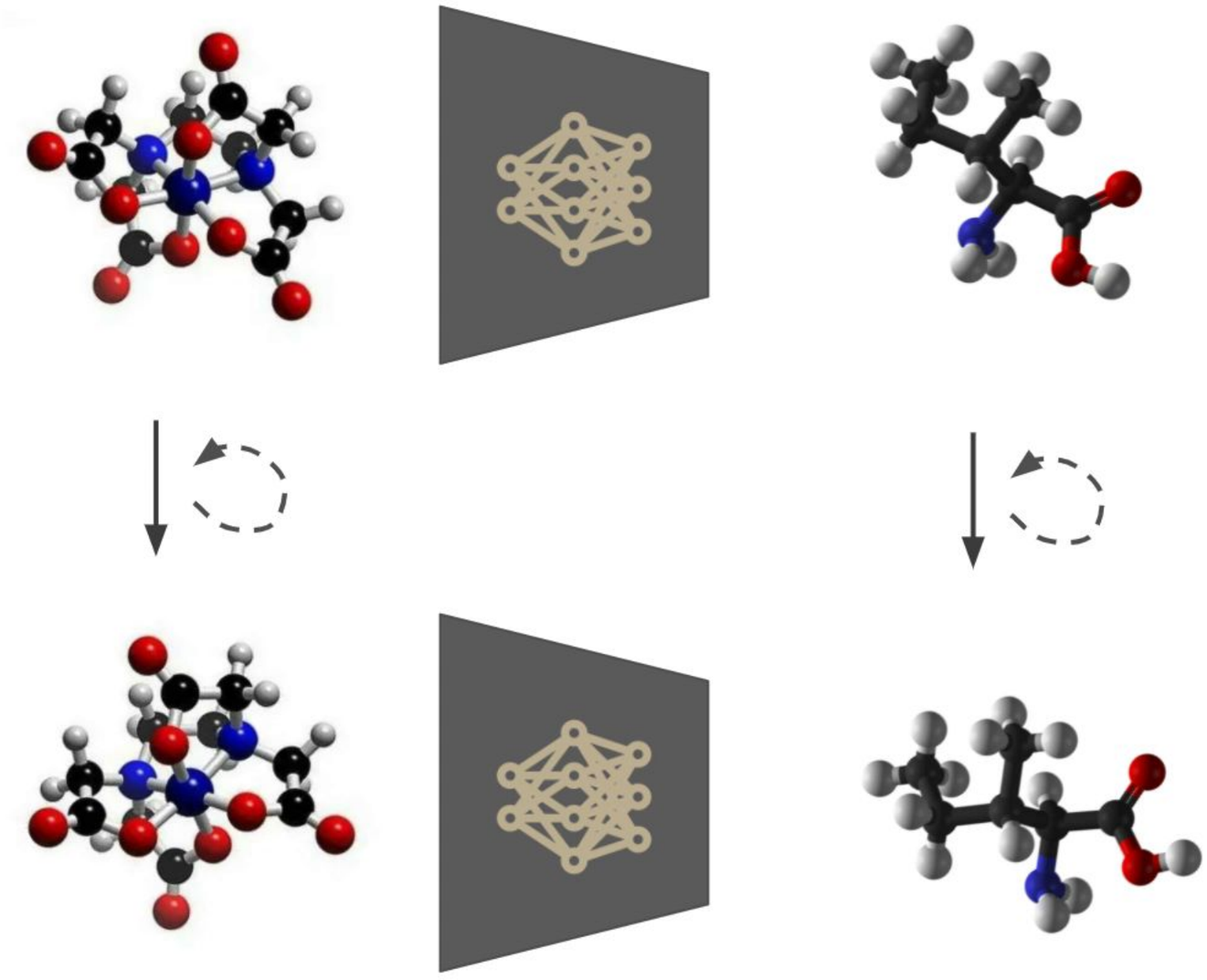


Property prediction

Background | Equivariance

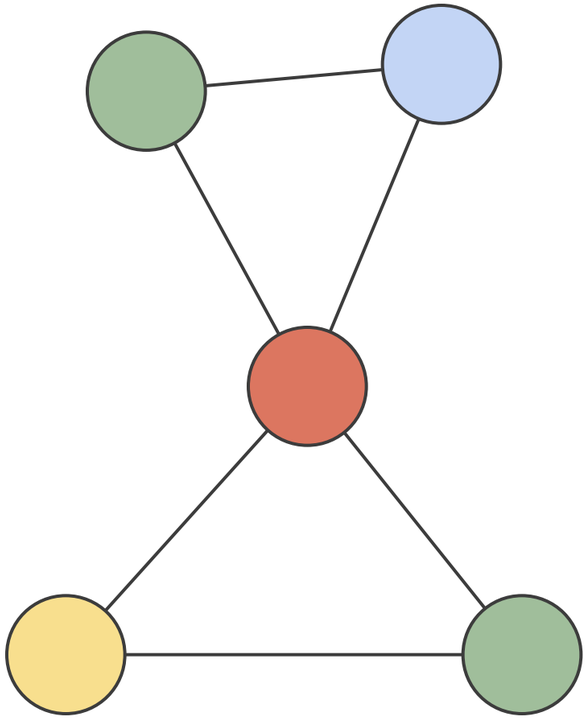


Background | Equivariance



Outline

1. Motivation: why build in symmetry? §1
2. Graph Neural Networks in a nutshell §2
3. A little group theory
4. Equivariance and invariance §2
5. E(n) Equivariant Graph Neural Networks §3
6. Proof of equivariance, velocities, learned edges §3.1–3.3



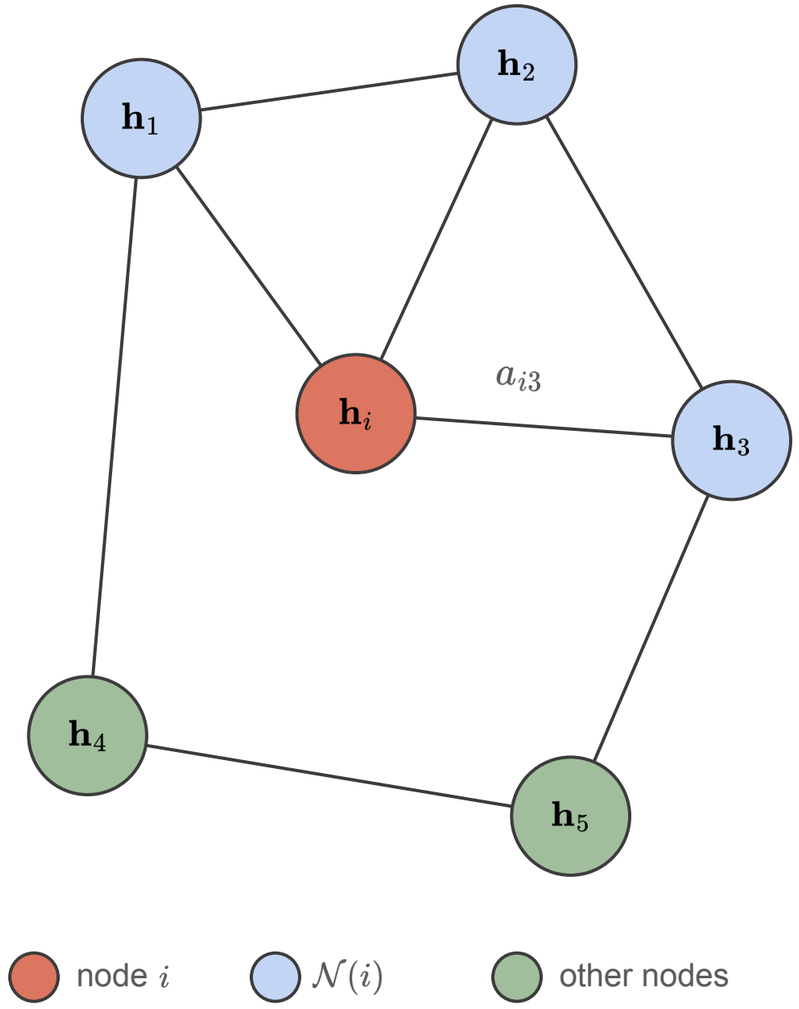
Motivation | Why build in symmetry?

- Many problems live in 3D space: **molecules**, **point clouds**, **N-body systems**
- The physics does not change if we rotate, translate or reflect the system, or relabel its particles
- Building these symmetries into the architecture is a strong **inductive bias**: less data, better generalisation
- Earlier E(3)/SE(3) equivariant models need higher-order representations (spherical harmonics): accurate but expensive, and limited to 3D
- **EGNN**: equivariant to rotations, translations, reflections and permutations, simple, and works in any dimension n

Architecture	Symmetry built in
MLP	none
CNN	translations
GNN	permutations of nodes
EGNN	**permutations + E(n): rotations, translations, reflections**

Background | Graphs

- A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has nodes $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E}$
- Every node carries a feature vector $\mathbf{h}_i \in \mathbb{R}^{nf}$ (e.g. atom type)
- Edges may carry attributes a_{ij} (e.g. bond type)
- $\mathcal{N}(i)$ denotes the neighbours of node i
- In this paper nodes also have coordinates $\mathbf{x}_i \in \mathbb{R}^n$: a graph embedded in space
- Examples: molecules (atoms, bonds), particle systems, point clouds, social networks



Background | Message passing (Eq. 2)

1. Message (every edge)

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij}\right)$$

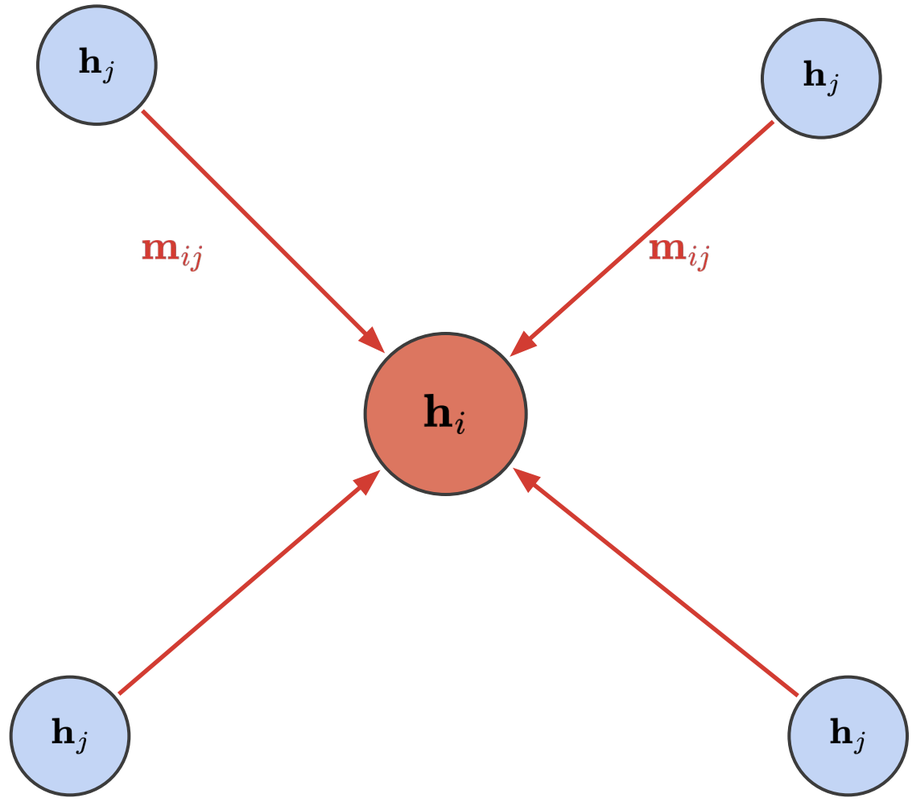
2. Aggregate (sum over neighbours)

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$

3. Update (every node)

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$

ϕ_e and ϕ_h are MLPs, shared by all edges and nodes. One application of these three steps is one GNN layer: features go from $\mathbf{h}^l$ to $\mathbf{h}^{l+1}$



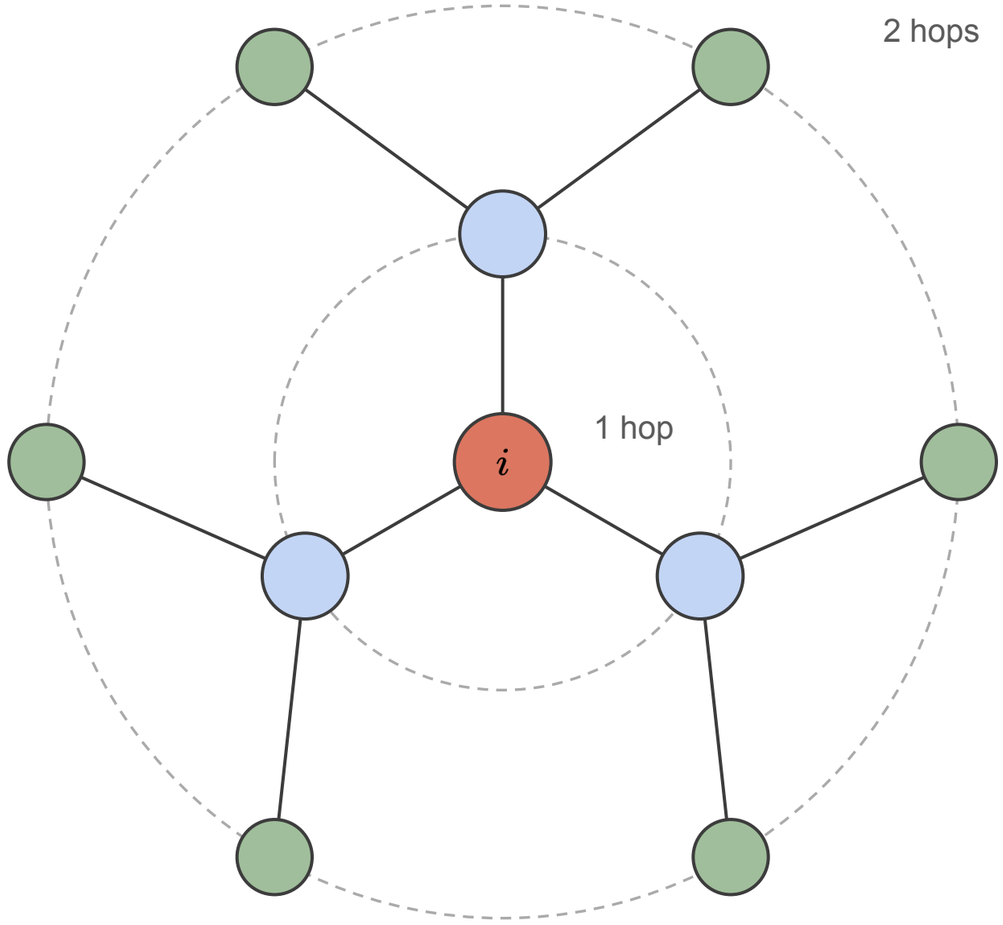
each neighbour sends a message;
node i sums them and updates itself

Background | Properties of GNNs

- **Receptive field**: after L layers, a node has information from its L hop neighbourhood
- **Weight sharing**: the same ϕ_e for ϕ_h every edge and node, so the same model works on graphs of any size
- **Permutation equivariant**: the sum over neighbours does not depend on node order, so relabelling the input nodes relabels the outputs
- **Graph-level outputs**: a readout over all nodes is permutation **invariant**:

$$\hat{y} = \phi_{\text{out}}\left(\sum_i \mathbf{h}_i^L\right)$$

- **Limitation**: if we feed coordinates $\mathbf{x}_i$ in as plain node features, the GNN is **not** equivariant to rotations or translations



Group Theory | What is a group?

A **group** $(G, \cdot)$ is a set with an operation that combines two elements into a third, such that:

Closure	Associativity	Identity	Inverse
$g \cdot h \in G$	$(g \cdot h) \cdot k = g \cdot (h \cdot k)$	$\exists\, e : \; e \cdot g = g \cdot e = g$	$\exists\, g^{-1} : \; g \cdot g^{-1} = e$

Examples

- $(\mathbb{Z}, +)$: integers under addition, identity 0, inverse $-n$
- Rotations of a square by 0°, 90°, 180°, 270°
- S_n: all permutations of n objects (relabelling nodes)
- $T(n)$: translations of $\mathbb{R}^n$
- $O(n)$: orthogonal $n \times n$ matrices (rotations and reflections)

Intuition

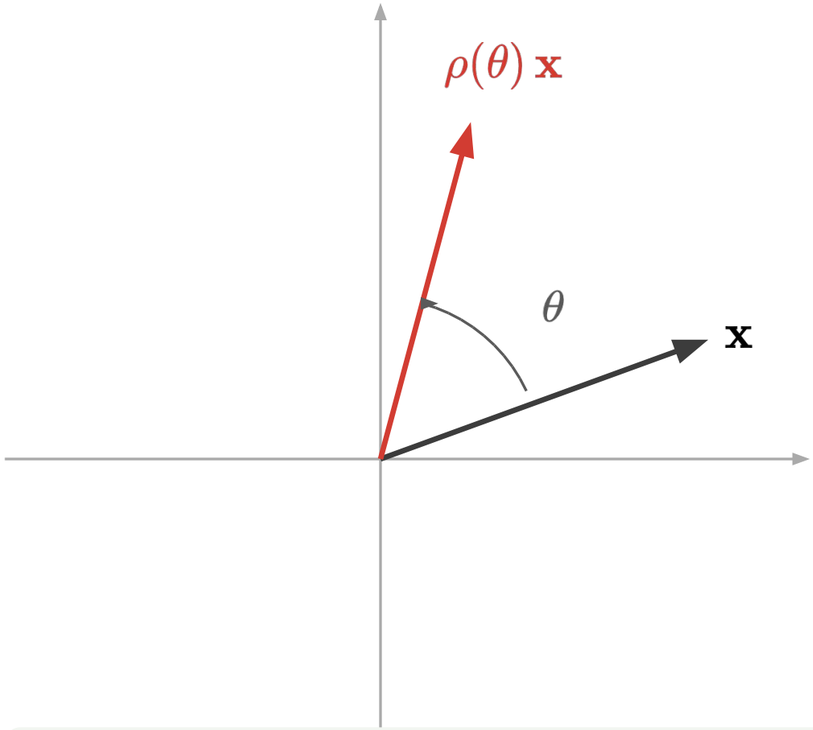
A group is a collection of transformations that can be composed and undone. Symmetries of an object always form a group.

Group Theory | Group actions and representations

- A group **acts** on a space X each g gives a map $T_g : X \to X$, with $T_e = \mathrm{id}$ and $T_g \circ T_h = T_{g \cdot h}$
- The same group can act differently on inputs ( ) and outputs ( )
- A **representation** $\rho(g) \in \mathbb{R}^{n \times n}$ is a linear action: $\rho(g \cdot h) = \rho(g)\rho(h)$, e.g. a rotation by θ acting on positions:

$$\rho(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- How a quantity transforms depends on **what it represents**:
  - a **location in space** (position $\mathbf{x}_i$, velocity $\mathbf{v}_i$) moves with the transformation
  - a **feature** of the node (atom type, charge, energy) describes the node itself, so it is unchanged
- EGNN keeps the two apart: features $\mathbf{h}$ stay **invariant**, coordinates $\mathbf{x}$ are **equivariant**



Location: $\mathbf{x} \mapsto \rho(\theta)\mathbf{x}$ (moves)

Feature, e.g. atom type: unchanged

Feature, e.g. energy E: $E \mapsto E$

Group Theory | The Euclidean group E(n)

E(n) contains all distance-preserving maps of $\mathbb{R}^n$

$$\mathbf{x} \mapsto Q\mathbf{x} + g, \qquad Q^\top Q = I, \quad g \in \mathbb{R}^n$$

Group	Elements	Action on a point $\mathbf{x}$
$T(n)$ translations	$g \in \mathbb{R}^n$	$\mathbf{x} + g$
$O(n)$ rotations + reflections	$Q^\top Q = I,\ \det Q = \pm 1$	$Q\mathbf{x}$
$SO(n)$ rotations	$Q^\top Q = I,\ \det Q = +1$	$Q\mathbf{x}$
$SE(n)$ rigid motions	$(Q, g),\ \det Q = +1$	$Q\mathbf{x} + g$
$E(n)$ Euclidean group	$(Q, g),\ \det Q = \pm 1$	$Q\mathbf{x} + g$
S_M permutations	permutation matrix P	reorders the M points

SE(3) (used by e.g. the SE(3)-Transformer) excludes reflections. EGNN is equivariant to the larger group E(n), together with permutations S_M, for any dimension n.

Group Theory | What survives an E(n) transformation?

Relative positions are equivariant

$$(Q\mathbf{x}_i + g) - (Q\mathbf{x}_j + g) = Q(\mathbf{x}_i - \mathbf{x}_j)$$

Translation cancels; the difference rotates with Q

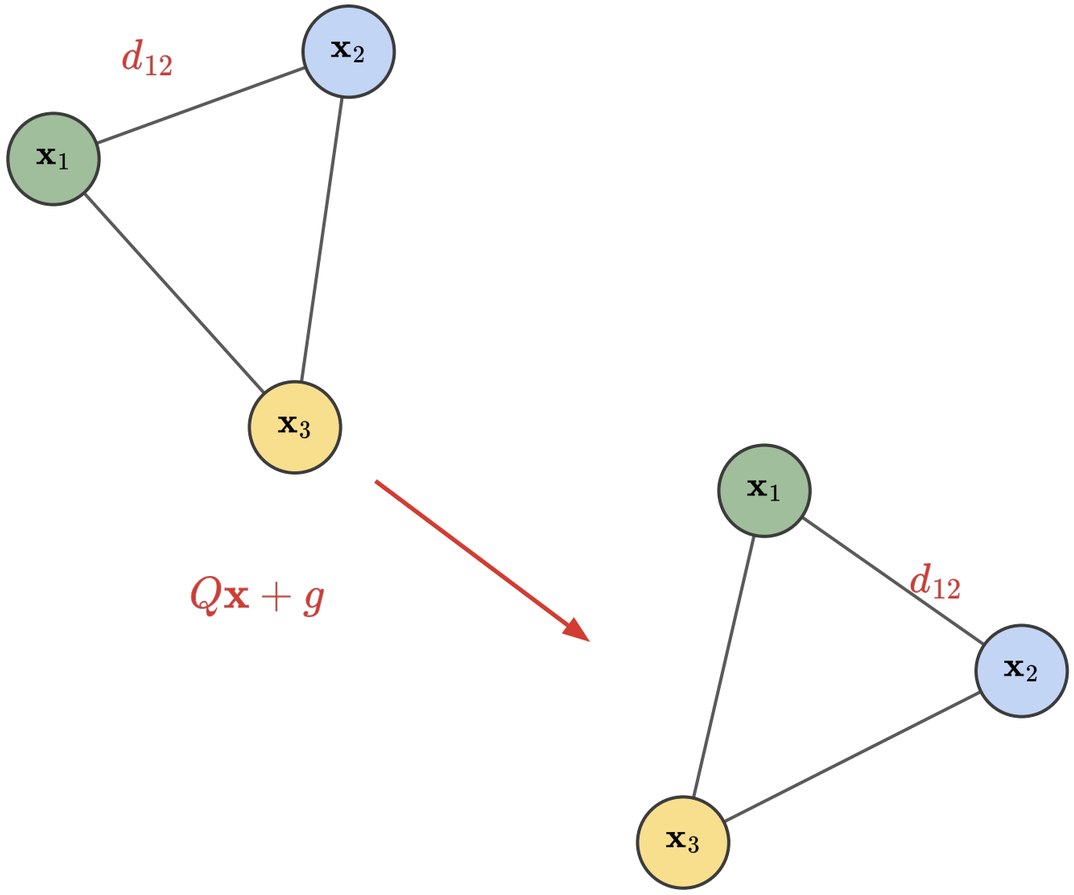
Distances are invariant

$$\|Q(\mathbf{x}_i - \mathbf{x}_j)\|^2 = (\mathbf{x}_i - \mathbf{x}_j)^\top Q^\top Q\,(\mathbf{x}_i - \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

Nothing changes at all.

The EGNN recipe

- compute messages from **distances** (invariant)
- move points along $\mathbf{x}_i - \mathbf{x}_j$ (equivariant)



same triangle after a rotation and a translation: all pairwise distances are unchanged

Background | Equivariance (Eq. 1)

ϕ is **equivariant** to T_g if there is an S_g such that

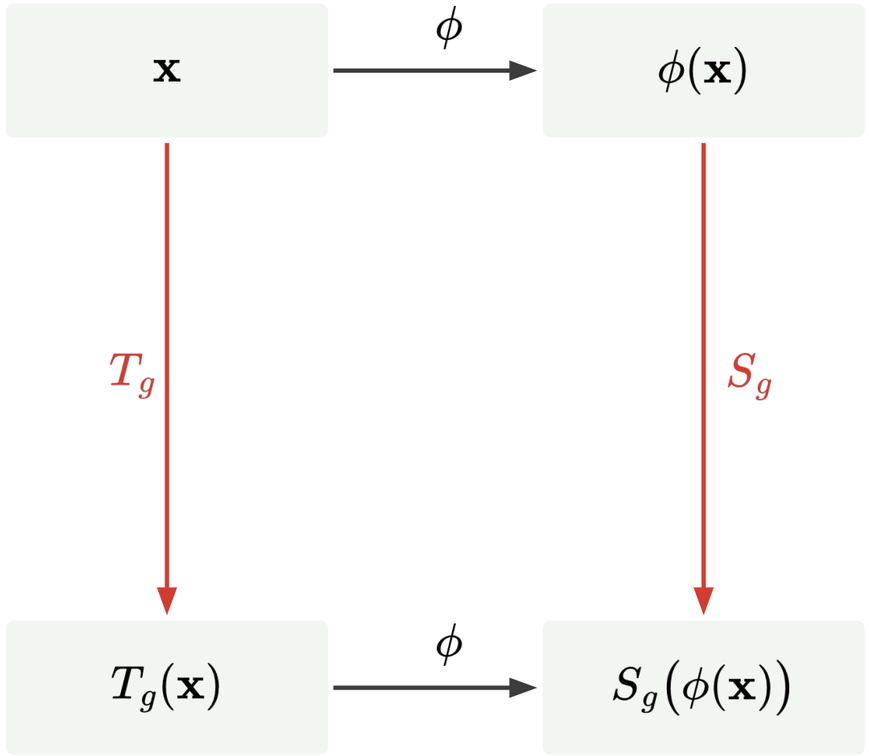
$$\phi\big(T_g(\mathbf{x})\big) = S_g\big(\phi(\mathbf{x})\big) \tag{1}$$

Invariance is the special case $S_g = \mathrm{id}$

$$\phi\big(T_g(\mathbf{x})\big) = \phi(\mathbf{x})$$

In words: transforming then applying ϕ = applying ϕ then transforming

- Image class label: **invariant** to translations
- Segmentation mask: **equivariant** to translations
- Energy of a molecule: **invariant** to rotations
- Forces on the atoms: **equivariant** to rotations



both paths around the square give the same result

Model

We present a model that is

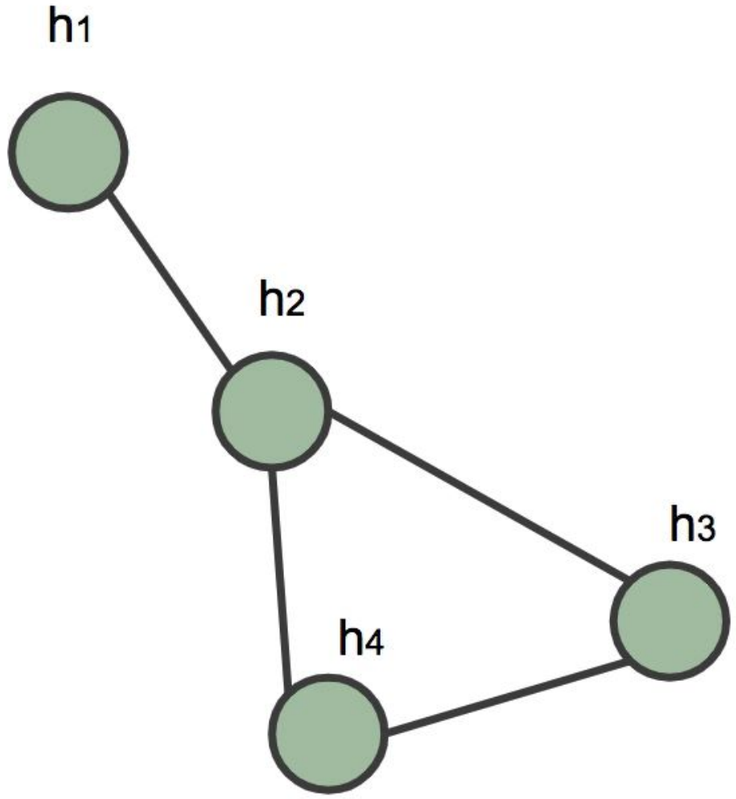
- **Equivariant** to euclidean transformations and permutations
- Achieves **competitive or better performance** in the provided experiments
- More **efficient** than previous models (computationally and memory wise)
- It is **simple** to implement.

E(n) Equivariant Graph Neural Networks

Model

Graph Neural Networks

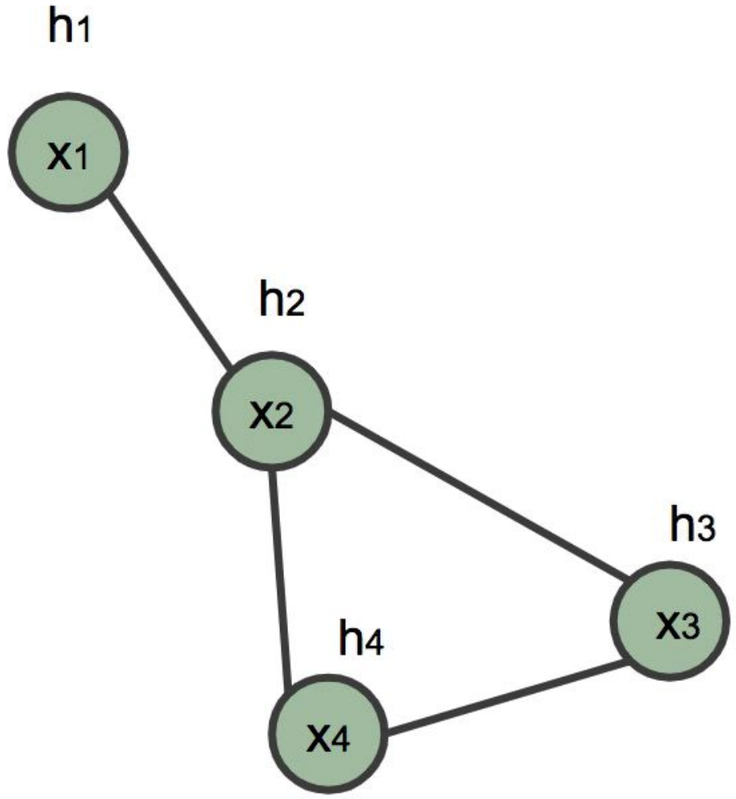
Given $\mathbf{h}^l, \mathcal{E} = (e_{ij})$



Model

Graph Neural Networks

Given $\mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$



Model

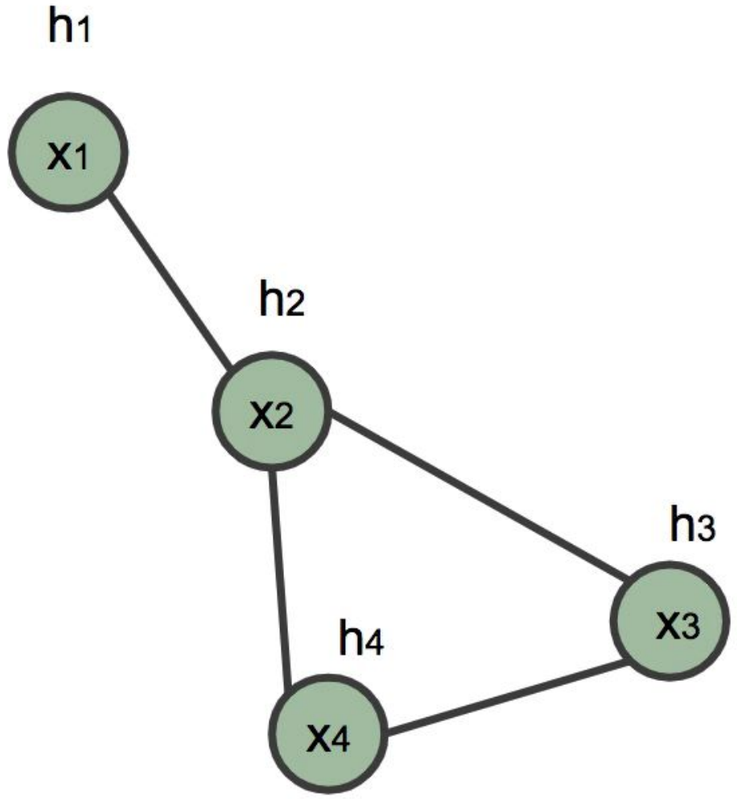
Graph Neural Networks

Given $\mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$

$$\mathbf{m}_{ij} = \phi_e \left(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij} \right)$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$

$$\mathbf{h}_i^{l+1} = \phi_h \left(\mathbf{h}_i^l, \mathbf{m}_i \right)$$



Model

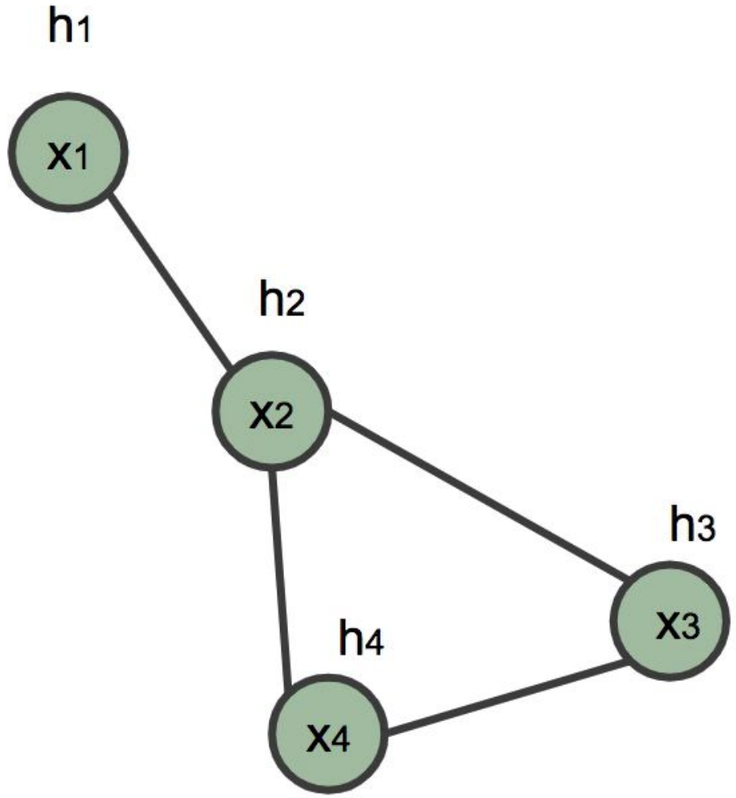
E(n) Invariant Graph Neural Networks

Given $\mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$



Model

E(n) Equivariant Graph Neural Networks

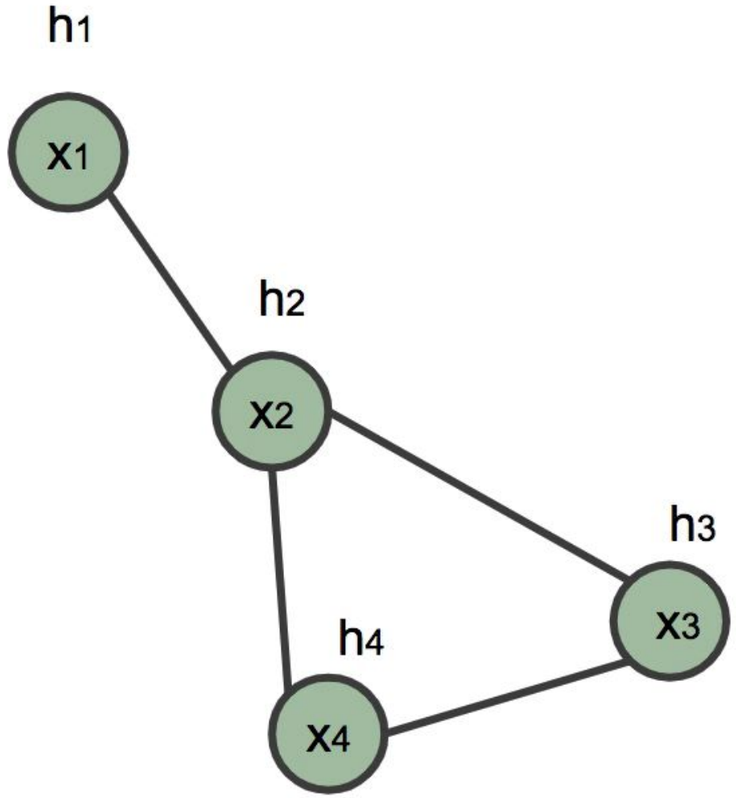
Given $\mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \sum_{j \neq i} \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) \phi_x\left(\mathbf{m}_{ij}\right)$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$



Model

E(n) Equivariant Graph Neural Networks

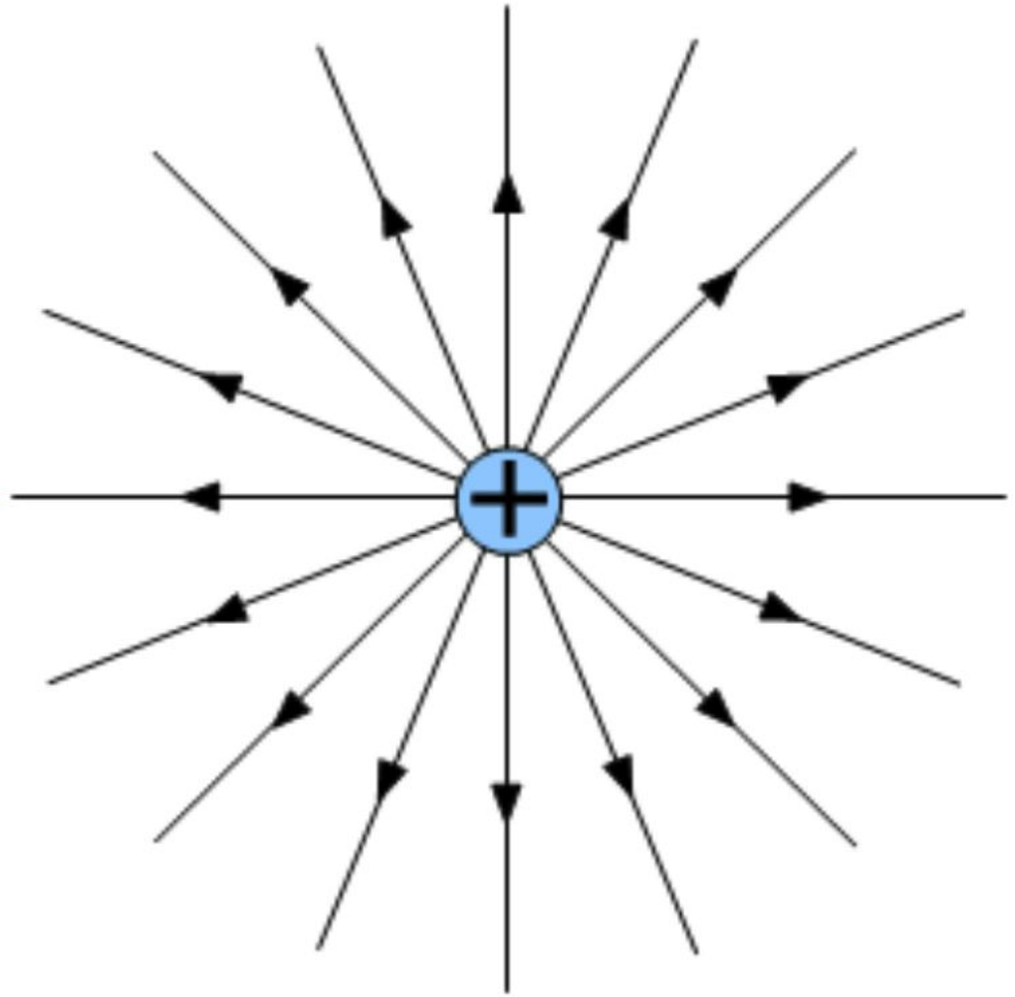
Given $\mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right)$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x\left(\mathbf{m}_{ij}\right)$$

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}$$

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$



Model | The Equivariant Graph Convolutional Layer (EGCL)

$$\mathbf{h}^{l+1}, \mathbf{x}^{l+1} = \mathrm{EGCL}\left[\mathbf{h}^{l}, \mathbf{x}^{l}, \mathcal{E}\right]$$

new compared to a GNN

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, a_{ij}\right) \quad (3)$$

edge message, gets the **invariant** squared distance

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x(\mathbf{m}_{ij}) \quad (4)$$

coordinate update along relative vectors, $C = \frac{1}{M-1}$

$$\mathbf{m}_i = \sum_{j\neq i}\mathbf{m}_{ij} \quad (5)$$

aggregate messages

$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right) \quad (6)$$

node feature update

- ϕ_e, ϕ_x, ϕ_h are MLPs; ϕ_x outputs **one scalar weight** per edge
- Features $\mathbf{h}$ (what a node is) and coordinates $\mathbf{x}$ (where it is) are kept separate: $\mathbf{h}$ stays invariant, $\mathbf{x}$ is equivariant. No spherical harmonics, any dimension n
- Sums run over all $j \neq i$ in the paper; with a known graph you can restrict them to $\mathcal{N}(i)$

Model | §3.1 Why the EGCL is E(n) equivariant

Goal: if the input is $(\mathbf{h}^l, Q\mathbf{x}^l + g)$, the output is $(\mathbf{h}^{l+1}, Q\mathbf{x}^{l+1} + g)$, for any orthogonal Q and translation g

Step 1: messages are invariant (Eq. 3)

$$\left\|(Q\mathbf{x}_i^l + g) - (Q\mathbf{x}_j^l + g)\right\|^2 = \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2$$

$\Rightarrow \mathbf{m}_{ij}$ is unchanged (given invariant $\mathbf{h}^l$)

Step 2: coordinates are equivariant (Eq. 4)

$$Q\mathbf{x}_i^l + g + C\sum_{j\neq i}\left(Q\mathbf{x}_i^l + g - Q\mathbf{x}_j^l - g\right)\phi_x(\mathbf{m}_{ij})$$

$$= Q\left(\mathbf{x}_i^l + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x(\mathbf{m}_{ij})\right) + g \;=\; Q\mathbf{x}_i^{l+1} + g$$

Step 3: features are invariant (Eqs. 5–6)

$\mathbf{m}_{ij}$ and $\mathbf{h}_i^l$ only depend on invariant quantities $\Rightarrow$ $\mathbf{h}^{l+1}$ is unchanged

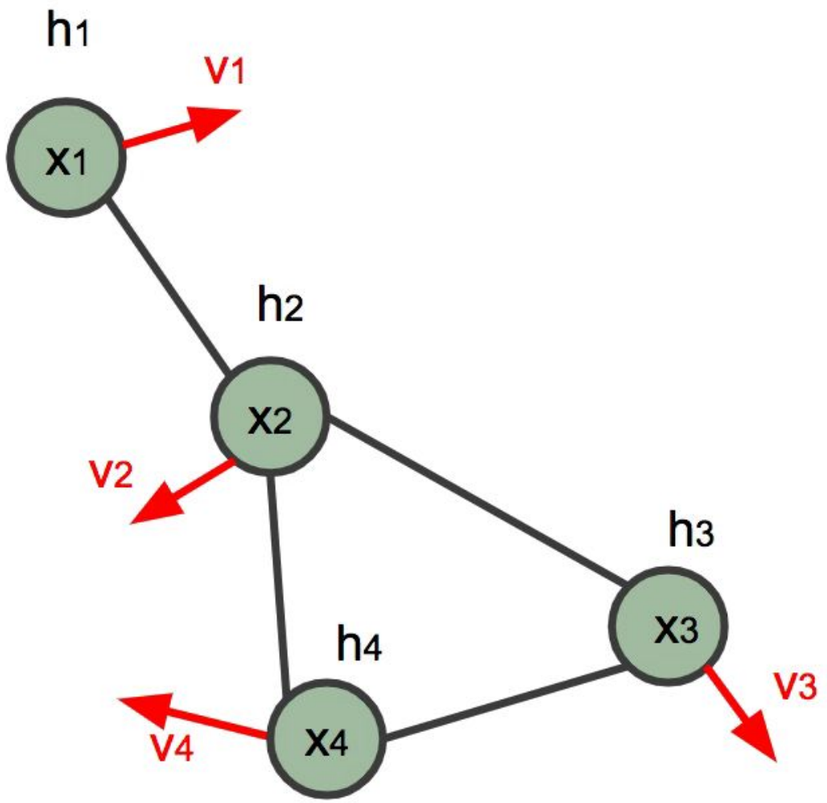
$\Rightarrow$ By induction, a stack of EGCLs is E(n) equivariant in $\mathbf{x}$ and invariant in $\mathbf{h}$. Permutation equivariance follows because every operation is a sum over nodes. (Formal proof: Appendix A.)

Model

E(n) Equivariant Graph Neural Networks

Vector inputs

Given $\boxed{\mathbf{v}^l}, \mathbf{x}^l, \mathbf{h}^l, \mathcal{E} = (e_{ij})$

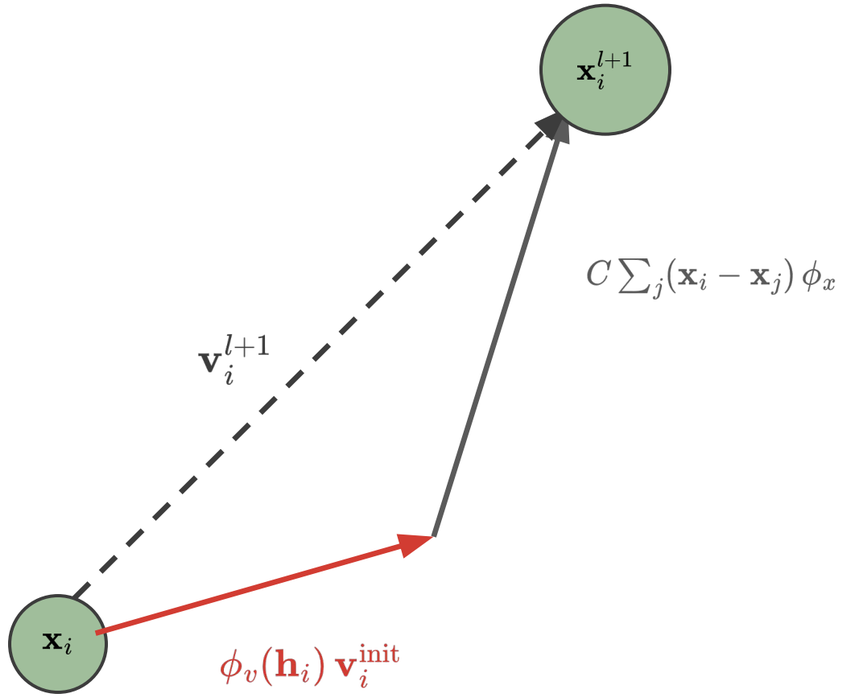


Model | §3.2 Vector inputs: tracking velocity (Eq. 7)

$$\mathbf{v}_i^{l+1} = \phi_v\left(\mathbf{h}_i^l\right)\mathbf{v}_i^{\text{init}} + C\sum_{j\neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)\phi_x(\mathbf{m}_{ij})$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \mathbf{v}_i^{l+1}$$

- Replaces Eq. 4 when the initial velocity is known, e.g. for **N-body** dynamics
- $\phi_v : \mathbb{R}^{nf} \to \mathbb{R}^1$ is an MLP giving a **scalar** that rescales the initial velocity
- With $\mathbf{v}_i^{\text{init}} = 0$ this reduces exactly to Eq. 4
- A velocity is a direction in space: unaffected by translations, rotated by Q. So the same proof still works and equivariance is preserved

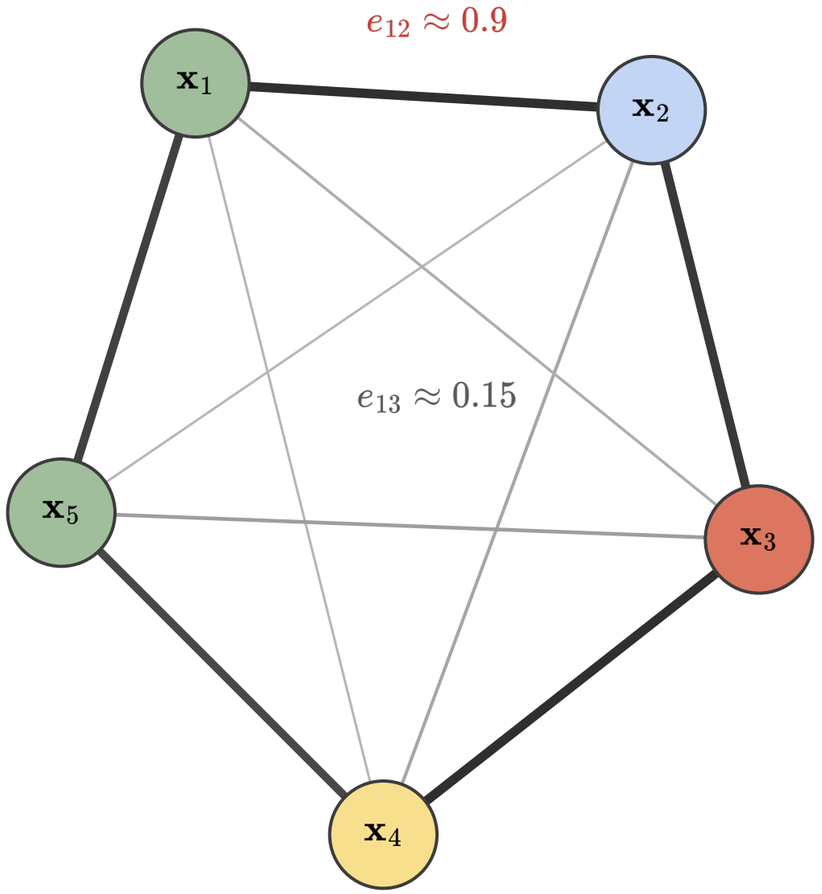


Model | §3.3 Inferring the edges (Eq. 8)

- Point clouds and particle systems often come **without** an adjacency matrix
- Option 1: a fully connected graph, where every node sends a message to every other node
- Option 2: **learn** which edges matter. Rewrite the aggregation with an edge indicator:

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} = \sum_{j \neq i} e_{ij}\, \mathbf{m}_{ij}$$

- Approximate $e_{ij} \approx \phi_{\text{inf}}(\mathbf{m}_{ij}) \in (0, 1)$ a linear layer followed by a sigmoid
- ϕ_{inf} only sees **invariant** messages, so E(n) equivariance is preserved



edge thickness = learned weight e_{ij}

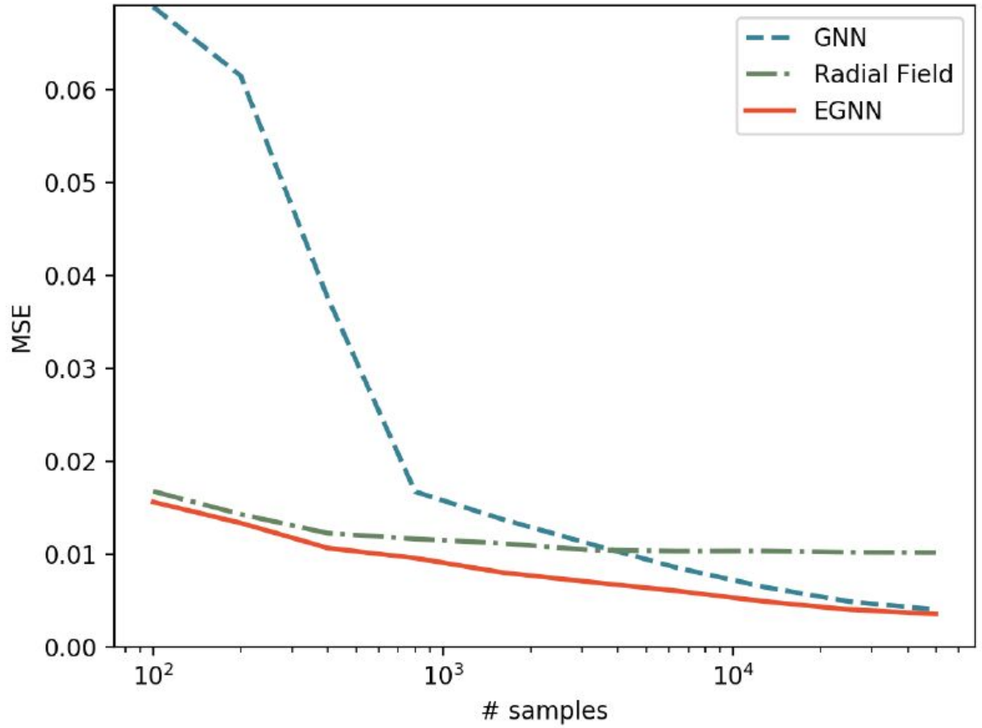
Particle dynamics

Predict five charged particles' positions 1,000 timesteps ahead.

Prediction error and runtime

Method	MSE	Forward time (s)
Linear	0.0819	.0001
SE(3) Transformer	0.0244	.1346
Tensor Field Network	0.0155	.0343
Graph Neural Network	0.0107	.0032
Radial Field	0.0104	.0039
EGNN	**0.0071**	.0062

Error vs. training data



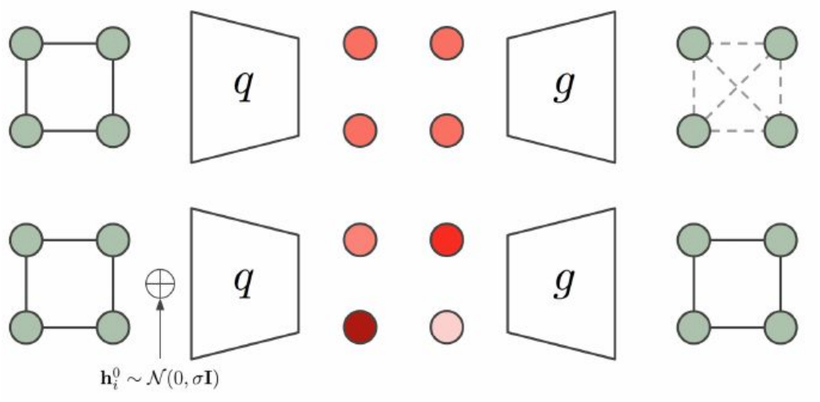
EGNN reduces MSE by 32% versus Radial Field.

Satorras et al. · E(n) Equivariant Graph Neural Networks · §5.1, Table 2 and Figure 2

Graph autoencoding

Add random coordinates, then learn node positions that reconstruct the graph.

Why add noise?



Test reconstruction results

	Community Small			Erdos&Renyi		
Encoder	BCE	% Error	F1	BCE	% Error	F1
Baseline	-	31.79	.0000	-	25.13	0.000
GNN	6.75	1.29	0.980	14.15	4.62	0.907
Noise-GNN	3.32	0.44	0.993	4.56	1.25	0.975
Radial Field	9.22	1.19	0.981	6.78	1.63	0.968
EGNN	**2.14**	**0.06**	**0.999**	**1.65**	**0.11**	**0.998**

EGNN edge error: 0.06% on Community Small and 0.11% on Erdős–Rényi.

Satorras et al. · E(n) Equivariant Graph Neural Networks · §5.2, Figures 3 and 5

5.3 Molecular properties

Predict 12 QM9 properties from atom types and pairwise distances.

Mean absolute error (lower is better)

*DimeNet++ uses a different train/validation/test split.

Task Units	α bohr^3	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_ν cal/mol K	G meV	H meV	R^2 bohr^3	U meV	U_0 meV	ZPVE meV
NMP	.092	69	43	38	.030	.040	19	17	.180	20	20	1.50
Schnet	.235	63	41	34	.033	.033	14	14	.073	19	14	1.70
Cormorant	.085	61	34	38	.038	.026	20	21	.961	21	22	2.03
L1Net	.088	68	46	35	.043	.031	14	14	.354	14	13	1.56
LieConv	.084	49	30	25	.032	.038	22	24	.800	19	19	2.28
DimeNet++*	.044	33	25	20	.030	.023	8	7	.331	6	6	1.21
TFN	.223	58	40	38	.064	.101	-	-	-	-	-	-
SE(3)-Tr.	.142	53	35	33	.051	.054	-	-	-	-	-	-
EGNN	.071	48	29	25	.029	.031	12	12	.106	12	11	1.55

Competitive accuracy with fixed coordinates and distance-based messages.

Associated Works

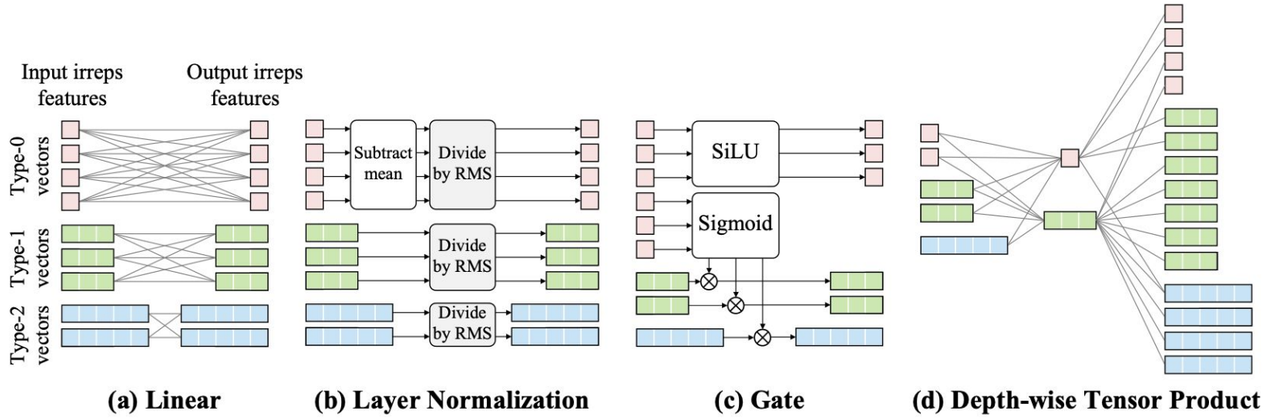
Matbench Discovery! GNNs with varying methods of equivariance used for simulating atoms



Associated Works

EQUIFORMER: EQUIVARIANT GRAPH ATTENTION TRANSFORMER FOR 3D ATOMISTIC GRAPHS

Yi-Lun Liao, Tess Smidt
Massachusetts Institute of Technology
{ylliao, tsmidt}@mit.edu
https://github.com/atomicarchitects/equiformer



L = 0 · Scalar



$s' = s$

1 component · unchanged

L = 1 · Vector



$v' = Rv$

3 components · rotate together

L = 2 · Tensor



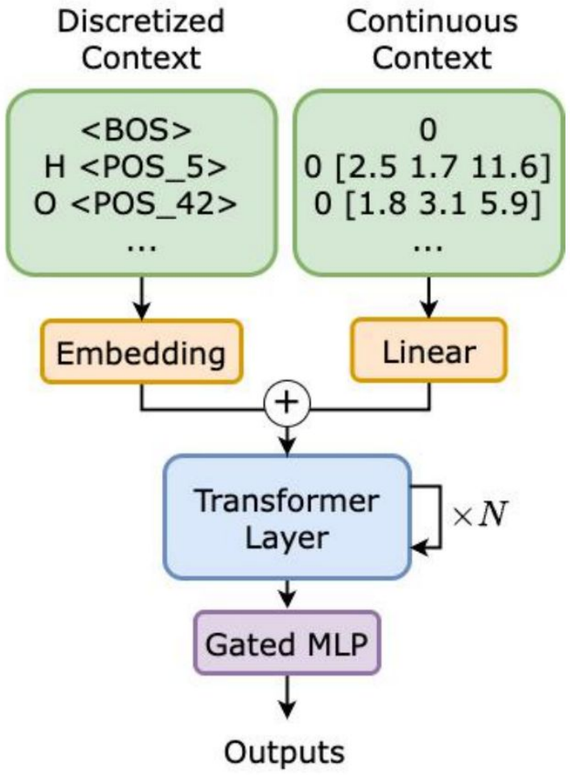
$Q' = RQR^T$

5 components · symmetric, traceless

Associated Work - Vanilla Transformer vs SOTA EquiGNN

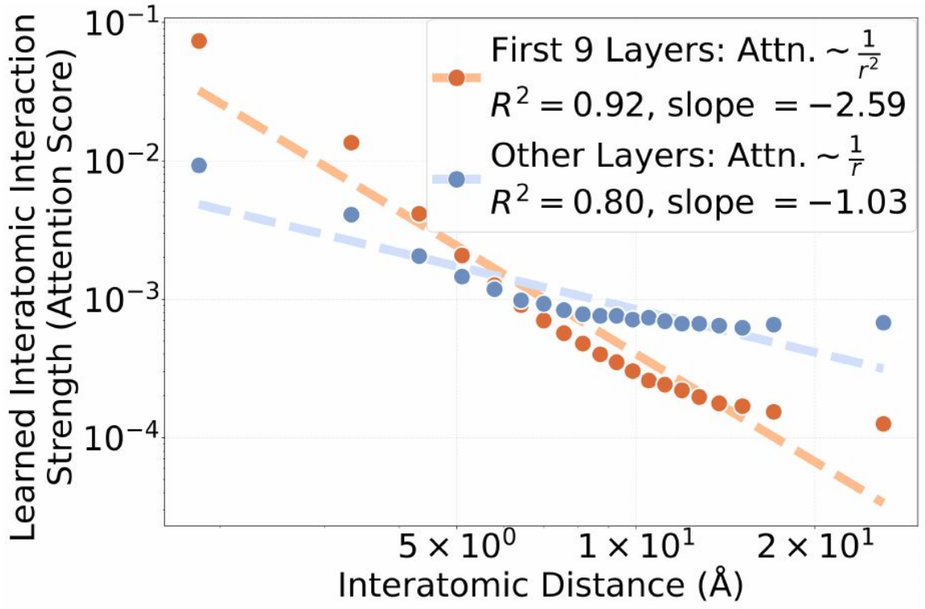
LLaMA2 backbone trained on OMol25 with no predefined interaction graph

Architecture



Pretrain on discrete tokens, then fine-tune energy and force heads.

Learned interactions and prediction results



OMol25 4M · comparable training FLOPs · lower is better

Metric	Transformer 1B	eSEN-sm-d 6M
Energy MAE (meV)	117.99	129.77
Force MAE (meV/Å)	18.35	13.01
Forward latency (ms)	17.2	26.3

Kreiman et al. · arXiv:2510.02259v3 (2026) · Figs. 1, 5; Table I. Latency: 100 atoms on one A6000.