E(n) Equivariant Graph Neural Networks

Vo Tuan Kiet

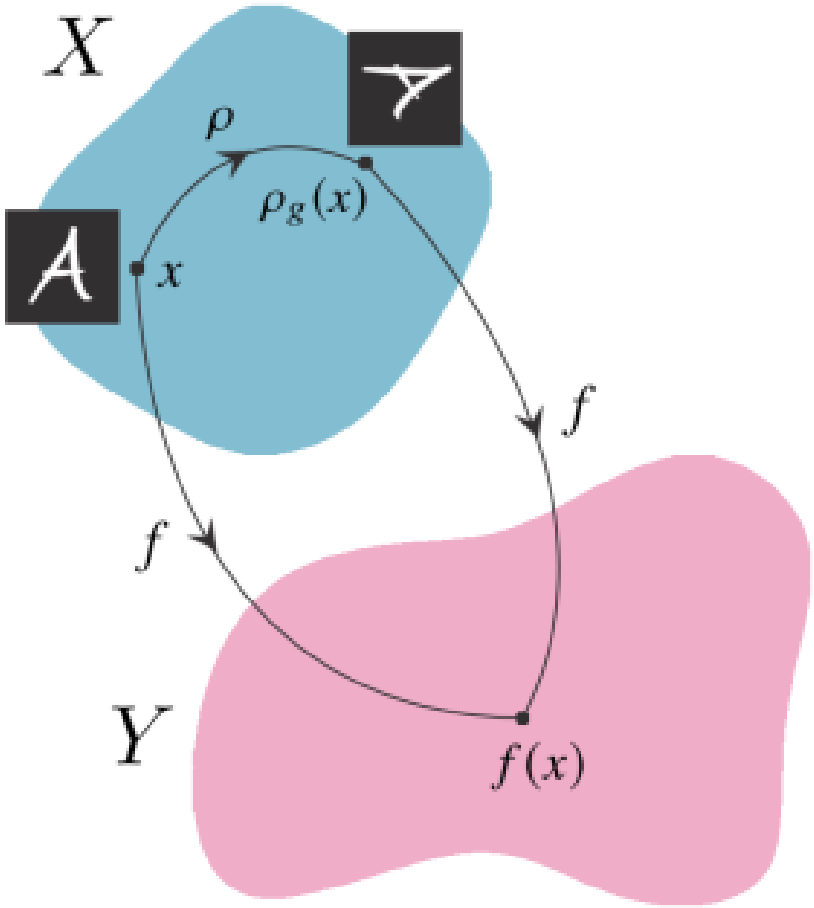
May 22, 2024

Transformation, Invariance, Equivariance

- Transformation: A function (usually geometrically meaningful) f that maps a set (X) to itself: f: $X \to X$
- Equivariance: A function $\phi : X \to Y$ is said to be equivariant if: $(S(\phi(x)) = \phi(T(x))$ where T is a transformation on X, and S is a transformation on Y.
- Invariance: A function $\phi : X \to Y$ is said to be invariant if: $S(\phi(T(x))) = \phi(x)$ where T is a transformation on X.

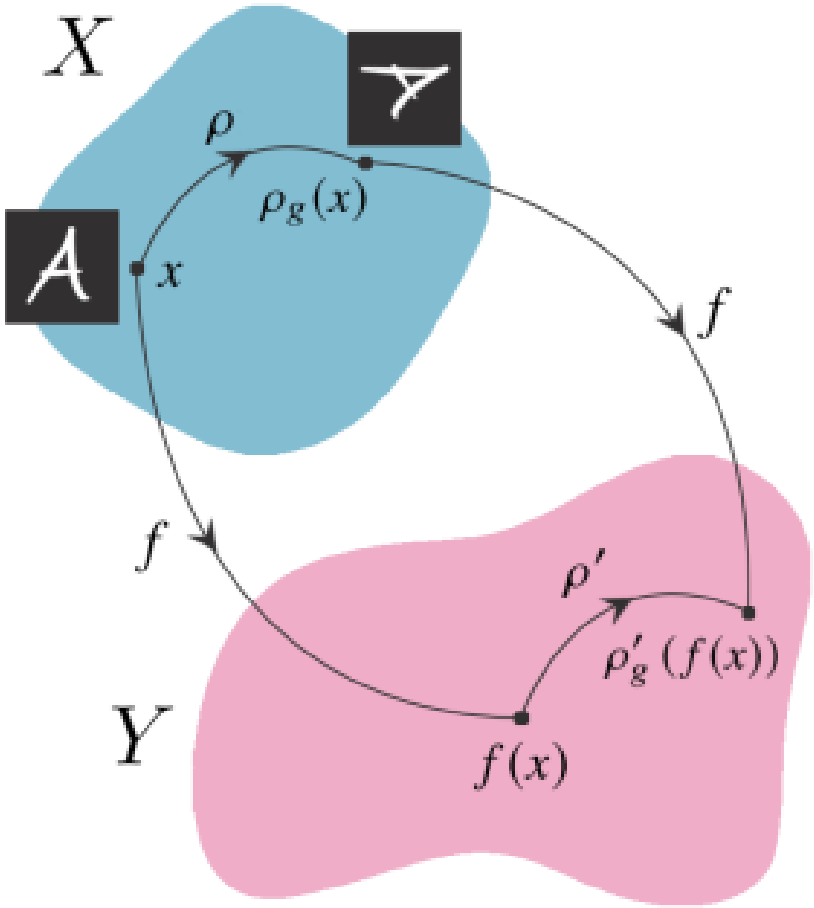
Invariance

$$f(\rho_g(x)) = f(x)$$



Equivariance

$$f(\rho_g(x)) = \rho'_g(f(x))$$



Equivariance

Three kind of equivariances are considered on $\phi(x) = y$:

- Translation equivariance : $T(x) = x + g, \phi(x + g) = y + g$
- Rotation equivariance : $T(x) = Qx, \phi(Qx) = Qy$
- Permutation equivariance : $T(x) = P(x), \phi(P(x)) = P(y)$

The Euclidean group $E(n)$ comprises all translations, rotations and reflections of Euclidean space E^n .

Graph Neural Networks

Given a graph $G = (V, E)$ with nodes $v_i \in V$ and edges $e_{ij} \in E$,$N(i) = \{j | e_{ij} \in E\}$, a graph convolutional layer is defined as:

$$m_{ij} = \phi_e(h_i^l; h_j^l, a_{ij}) \tag{1}$$

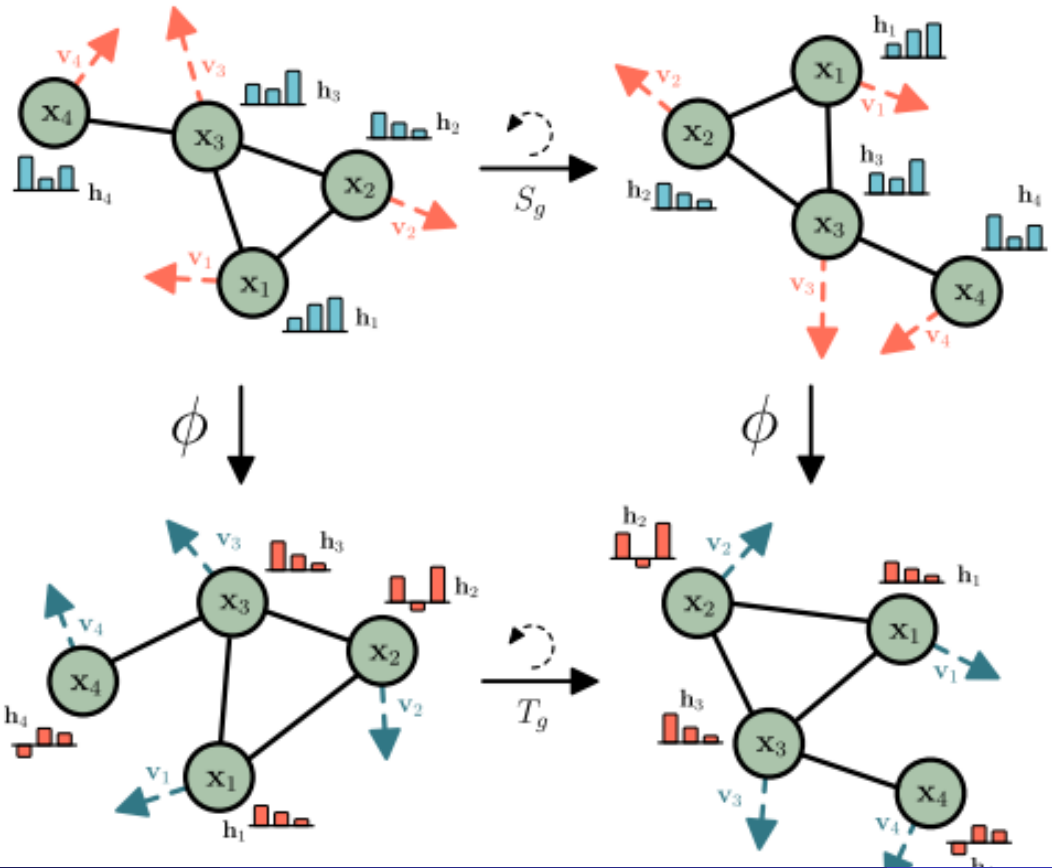
$$m_i = \sum_{j \in N(i)} m_{ij} \tag{2}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \tag{3}$$

GNN is permutation equivariance

E(n) Equivariant Graph Neural Networks

In addition to the feature node embeddings h_i, a n-dimensional coordinate $x_i \in R^n$ is introduced. Equivariant graph neural networks aim to preserve the equivariance w.r.t. **rotations** and **translations** on the set of coordinates while also preserving equivariance to **permutations** on the set of nodes V in the same fashion as GNNs



Each equivariant graph convolutional layer (EGCL) takes the form of $h^{l+1}, x^{l+1} = EGCL(h^l, x^l, E)$

$$m_{ij} = \phi_e\left(h_i^l, h_j^l, ||x_i - x_j||_2, a_{ij}\right) \tag{4}$$

$$x_i^l + 1 = x_i + C\sum_{j \neq i}(x_i^l - x_j^l)\phi_x(m_{ij}) \tag{5}$$

$$m_i = \sum_{j \neq i} m_{ij} \tag{6}$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \tag{7}$$

EGCL vs GCL

The major differences between EGCL and GCL is the computation of m_{ij} and x_i^{l+1}

- For each message m_{ij} along the edge between v_i and v_j, the squared Euclidean distance $||x_i - x_j||^2$ between the two nodes are used.
- For each coordinate transformation, $\phi_x : \mathbb{R}^f \to \mathbb{R}$ give the weight for each relative differences $x_i - x_j$. $C = 1/(M-1)$ averages the weighted sum.

Extend EGNNs for Velocity

Apart from equivariant to the coordinates x, EGNNs can be further extended to be equivariant to the velocity v and keep track of the momentum of each node.

$$h^{l+1}, x^{l+1}, v^{l+1} = EGCL(h^l, x^l, v^{init}, E)$$

$$v^{l+1} = \phi_v(h_i^l)v^{init} + C\sum_{j \neq i}(x_i^l - x_j^l)\phi_x(m_{ij})$$

$$x_i^{l+1} = x_i^l + v_i^{l+1}$$

The equivariance can be show w.r.t x and v similarly while the invariance w.r.t m_{ij} stays the same.

EGNN : Comparison to previous works

	GNN	Radial Field	TFN	Schnet	EGNN
Edge	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$	$\mathbf{m}_{ij} = \phi_{\text{rf}}(\|\mathbf{r}_{ij}^l\|)\mathbf{r}_{ij}^l$	$\mathbf{m}_{ij} = \sum_k \mathbf{W}^{lk}\mathbf{r}_{ji}^l\mathbf{h}_i^{lk}$	$\mathbf{m}_{ij} = \phi_{\text{cf}}(\|\mathbf{r}_{ij}^l\|)\phi_{\text{s}}(\mathbf{h}_j^l)$	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, \|\mathbf{r}_{ij}^l\|^2, a_{ij})$ $\hat{\mathbf{m}}_{ij} = \mathbf{r}_{ij}^l\phi_x(\mathbf{m}_{ij})$
Agg	$\mathbf{m}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$ $\hat{\mathbf{m}}_i = C\sum_{j\neq i} \hat{\mathbf{m}}_{ij}$
Node	$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$	$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \mathbf{m}_i$	$\mathbf{h}_i^{l+1} = w^{ll}\mathbf{h}_i^l + \mathbf{m}_i$	$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$	$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$ $\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \hat{\mathbf{m}}_i$
	Non-equivariant	$\mathrm{E}(n)$-Equivariant	$\mathrm{SE}(3)$-Equivariant	$\mathrm{E}(n)$-Invariant	$\mathrm{E}(n)$-Equivariant

Figure: Comparison over different works from the literature under the message passing framework notation.

Inferring The Edges

In certain graphs, there might be no adjacency matrix, such as a point cloud. Such graphs are often assumed to be fully connected, thus greatly increasing the complexity in message aggregation. This can be alleviated by adding another neural network $\phi_{inf} : \mathbb{R}^f \to [0, 1]$

$$m_i = \sum_{j \in N(i)} m_{ij} = \sum_{j \neq i} e_{ij} m_{ij}$$

$$e_{ij} = \begin{cases} 1, (v_i, v_j) \in E \\ 0, o.w \end{cases} \quad \text{for graphs with E,}$$

$$e_{ij} = \phi_{inf}(m_{ij}) \text{ for graphs without E}$$

Experiments : N-body System

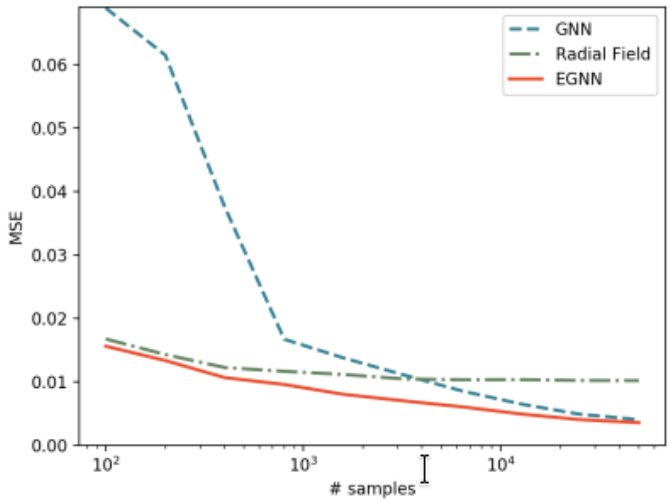
In a dynamic system, a fucntion defines the time-dependence of a point or a set of points in geometrical space. This paper considers the charged particles N-body experiment in 3D space. There are five particles carrying either a positive or negative charge. The input is the initial positions $x(0) = \{x_1^{(0)}, ..., x_5^{(0)}\}$, initial velocities $v^{(x)} = \{v_1^{(0)}, ..., v_5^{(0)}$, and charges $c = \{c_1, ..., c_5\} \in \{-1, 1\}^5$

N-body System

- **Implementation**. The EGNNs with velocity is used. The norm of velocity is transformed into h, and the attraction/repulsion between points is transformed into a_{ij}.
- **Result**

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

(a) Mean squared errors of the position prediction.



(b) Mean Squared Error of the position prediction when sweeping over different amounts of training data.

Graph Autoencoder

This paper test EGNNs as a graph autoencoder. The autoecoder represents a graph in adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{X}$ into $\mathbf{Z} \in \mathbb{R}^{|V| \times d_h}$, and the reconstruct $\mathbf{A}$ from $\mathbf{Z}$.
The following example shows a special case of a symmetric plain graph. GNNs on this graph is unable to distinguish different nodes since their neighbor topology are the same.

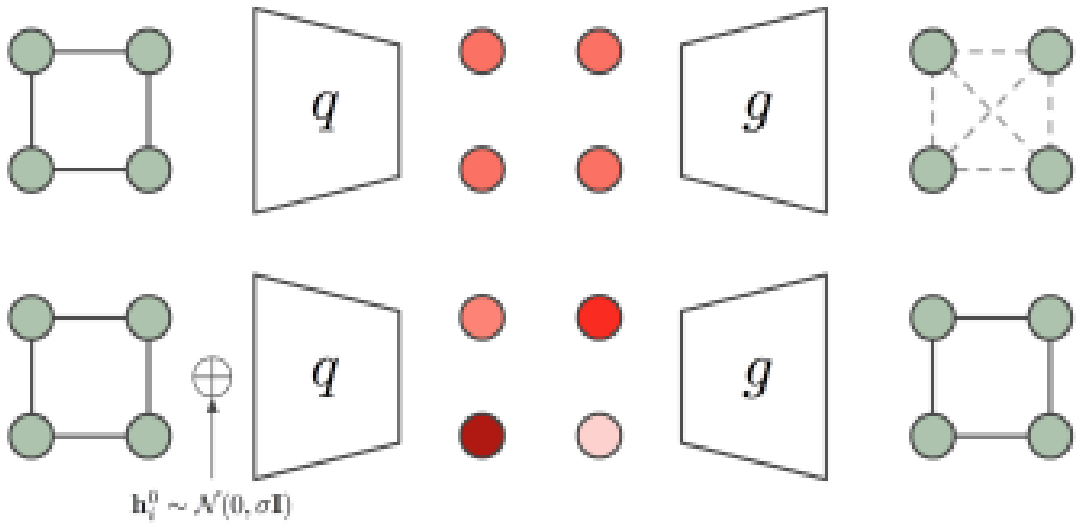


Figure: A symmetric cycle graph where all nodes have the same topology.

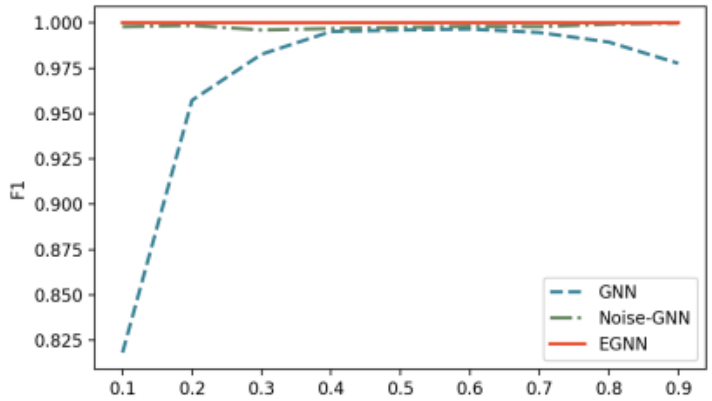
Graph Autoencoder

The symmetric problem above can be solved by associated each node with a vector sampled from N $(0, \sigma I)$. However, GNNs need to generalize to the noise distribution.

By using the sampled noises as x in EGNNs, they become translation and rotation equivariant, which makes the generalization easier. The authors provide another explanation that this makes the node representations from structural to positional, where equivariance helps.

	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**

(a) Mean squared errors of the position prediction.



(b) Mean Squared Error of the position prediction when sweeping over different amounts of training data.

QM9

EGNNs are used to predict molecular properties associated with quantum chemistry. These properties are invariant to translations and rotations. Since in this dataset, each molecule is in a stationary state, no position updates on x are performed. By simply using the distances, EGNNs can achieve on par performance compared to complicated models that consider angles or spherical harmonics.

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

Figure: Mean absolute errors of twelve prediction targets from QM9 dataset. EGNNs have similar performance to state-of-the-art DimeNet++