E(n) Equivariant Graph Neural Networks

Abdelhamid Younes, Wangchen (Clark) Xu, Dominic Klukas

Presentation Roadmap

- **Motivation:** Why specialized architectures for 3D data are necessary.
- **Background:** Symmetries, Groups, and the definition of Equivariance.
- **Prior Work:** Comparison of TFN, SE(3)-Tr, and SchNet vs. EGNN.
- **Technical Deep Dive:** The Equivariant Graph Convolutional Layer (EGCL).
- **Proofs & Extensions:** Formal Equivariance Proof (Appx A) and Momentum (Appx B).
- **Experimental Results:** Performance on N-body systems and QM9 molecular data.
- **Conclusion:** Summary of strengths, limitations, and future outlook.

Problem Setup and Motivation

- Deep Learning needs good inductive biases for generalization and sample efficiency
    - This way, network doesn’t “relearn” trivially similar data
    - Networks with specific inductive biases only work on data that actually have these inductive biases

Problem Setup and Motivation

E(n) Equivariant Graph Neural Networks

Nodes have location/velocity/acceleration features in n-dimensional space, problem corresponding symmetry

Data is structured as a graph

Example Problems: Molecule Property Estimation

- Given:
    - Atom Types (C, H, O, N, F)
    - Locations of Atoms (E(3)-equivariant, to rotations and translations in molecule location)
    - Bonds Between Atoms (Makes our data a graph, to be processed by a GNN)
- Desired
    - Scalar features, such as Free Energy, Heat Capacity

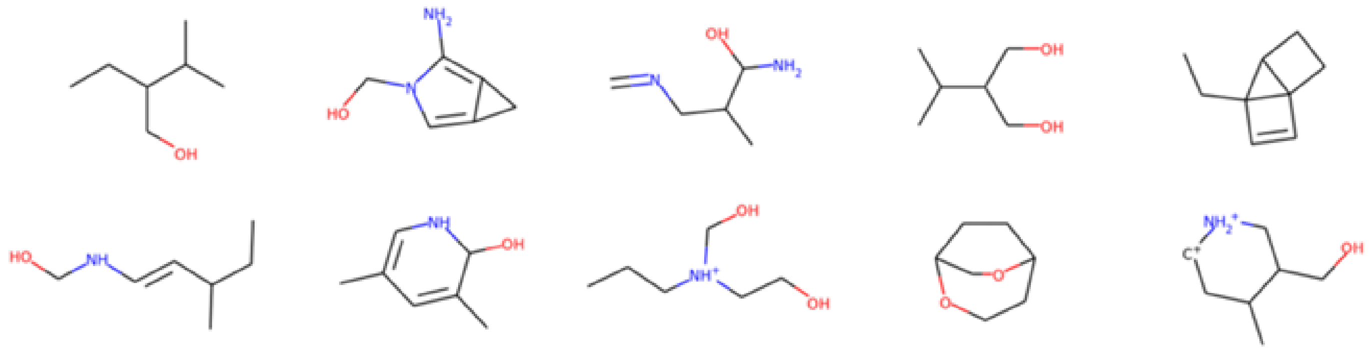
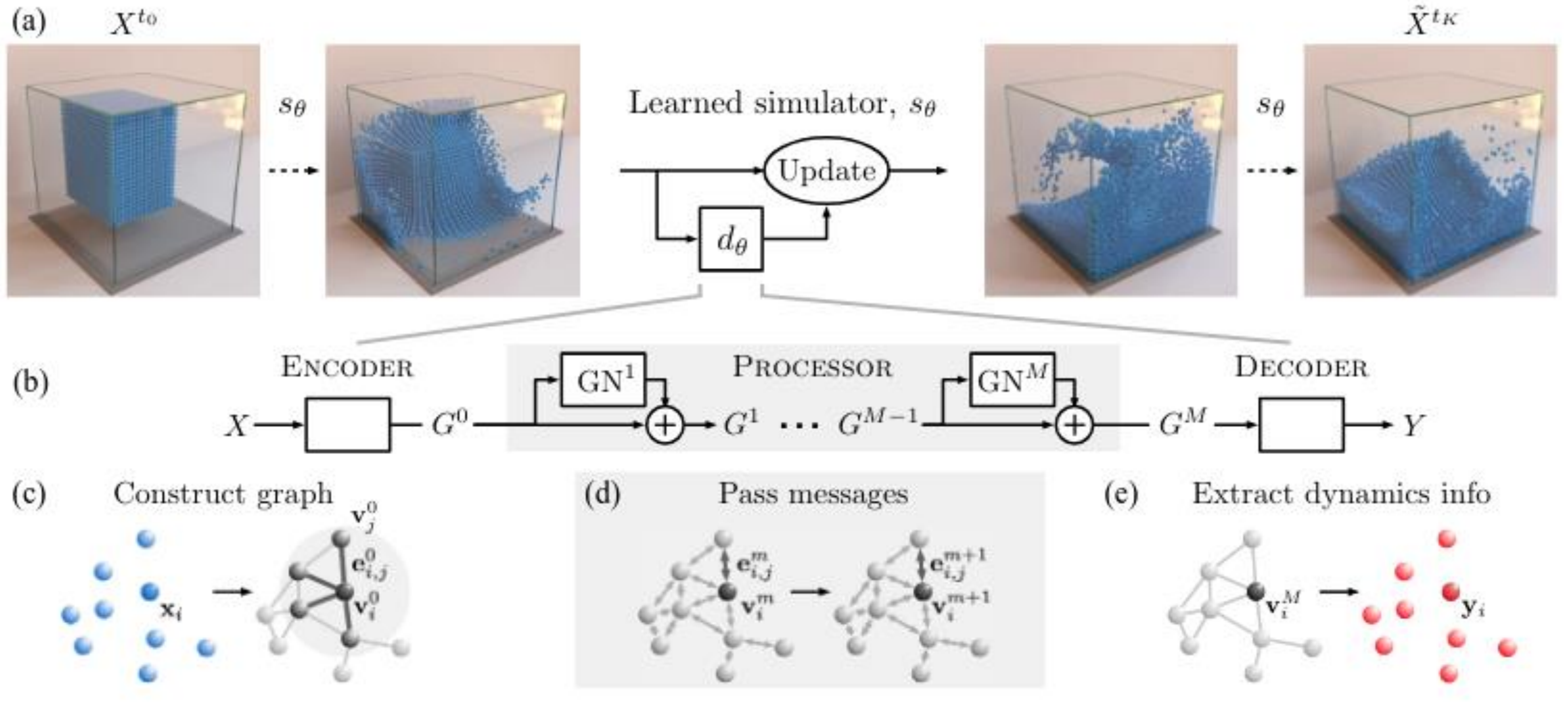


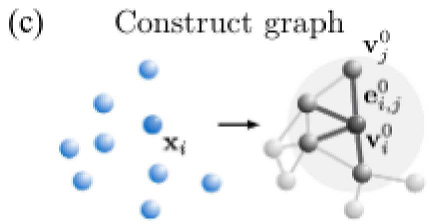
Image Credit: https://www.researchgate.net/figure/Examples-of-newly-generated-molecular-graphs-from-QM9_fig3_337431526

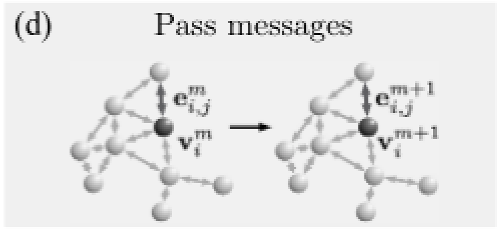
Example Problems: Particle Simulation



Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020). *Learning to Simulate Complex Physics with Graph Networks*. In H. Daumé III & A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning (Vol. 119, pp. 8459–8468). PMLR. https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html

Example Problems: Particle Simulation





Forces in particle physics are equivariant to rotation/translation (E(3))

Data is structured as a graph

Contributions

- New Architecture
    - Equivariance can be scaled to E(N) - not just E(3)
    - Fast inference time in comparison to TFN, SE(3) Transformer

Contributions

- New Architecture
  - Equivariance can be scaled to E(N) - not just E(3)
  - Fast inference time in comparison to TFN, SE(3) Transformer

Motivation: Generalization to E(N)

$$E(3) = \mathbb{R}^3 \rtimes O(3) \quad \Longrightarrow \quad E(n) = \mathbb{R}^n \rtimes O(n)$$

Motivation: Generalization to E(N)

$$E(3) \;=\; \mathbb{R}^3 \rtimes O(3) \qquad \Longrightarrow \qquad E(n) \;=\; \mathbb{R}^n \rtimes O(n)$$

$$v \in \mathbb{R}^n, \quad v = (a_1, a_2, \ldots, a_n)$$

Motivation: Generalization to E(N)

$$E(3) = \mathbb{R}^3 \rtimes O(3) \quad \Longrightarrow \quad E(n) = \mathbb{R}^n \rtimes O(n)$$

$$v \in \mathbb{R}^n, \quad v = (a_1, a_2, \ldots, a_n)$$

$$R \in O(n) \quad R^\top R = I \quad \Longrightarrow \quad ||Rv|| = ||v||$$

Motivation: Generalization to E(N)

$$E(3) = \mathbb{R}^3 \rtimes O(3) \quad \Rightarrow \quad E(n) = \mathbb{R}^n \rtimes O(n)$$

$$v \in \mathbb{R}^n, \quad v = (a_1, a_2, \ldots, a_n)$$

$$R \in O(n) \quad R^\top R = I \quad \Rightarrow \quad ||\boldsymbol{Rv}|| = ||v||$$

Why would we need this?

Motivation: Generalization to E(N)

9 Dimensional Vector?

$$u = \left(p_x,\ p_y,\ p_z,\ v_x,\ v_y,\ v_z,\ a_x,\ a_y,\ a_z\right)$$

$$u_1 = \left(0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1\right) \Longrightarrow Ru_1 = \left(\frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$$

6 Dimensional Vector?

$$v = \left(c_{\text{Salaries}},\ c_{\text{Inventory}},\ c_{\text{Bribes}},\ c_{\text{Golden Parachutes}},\ c_{\text{Research}},\ c_{\text{Raw Materials}}\right) \Longrightarrow Rv = ???$$

Motivation: Generalization to E(N)?

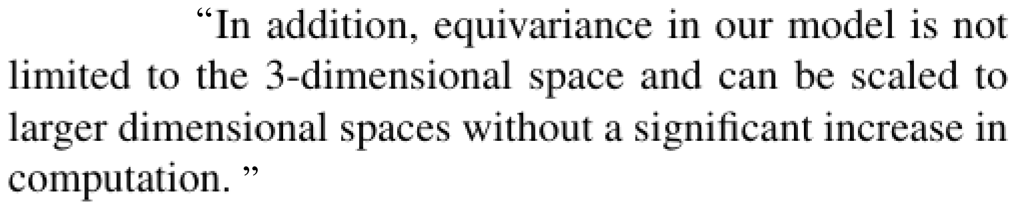
- E(N) almost never shows up in real life for N > 3
    - E(4) in physics, general relativity: (x, y, z, t)
    - Lattices… synthetic 4 dimensions

Motivation: Generalization to E(N)

- E(N) almost never shows up in real life for N > 3
  - E(4) in physics, general relativity: (x, y, z, t)
  - Lattices… synthetic 4 dimensions
- E(2) is useful however… rotations in plane, but very simple to implement vs E(3)
- E(1)... flipping signs of scalars… trivial.

Motivation: Generalization to E(N)

- E(N) almost never shows up in real life for N > 3
  - E(4) in physics, general relativity: (x, y, z, t)
  - Lattices… synthetic 4 dimensions
- E(2) is useful however… rotations in plane, but very simple to implement vs E(3)
- E(1)... flipping signs of scalars… trivial.



"In addition, equivariance in our model is not limited to the 3-dimensional space and can be scaled to larger dimensional spaces without a significant increase in computation. "

Contributions

- New Architecture
  - Equivariance can be scaled to E(N) - not just E(3)
  - Fast inference time in comparison to TFN, SE(3) Transformer
    - No expensive spherical harmonics
    - No attention mechanism

The Need for E(n) Equivariance & Current Bottlenecks

- **The Solution:** Enforcing equivariance restricts the network strictly to physically relevant functions.
- **The Goal:** Native equivariance to the Euclidean group E(n) (rotations, translations, and reflections).
- **Prior Bottlenecks:** Earlier methods relied heavily on computing spherical harmonics for higher-order representations, neglecting the spatial coordinates awareness.
- **The Drawbacks:** These approximations are computationally expensive and strictly limit the network to 3-dimensional spaces.

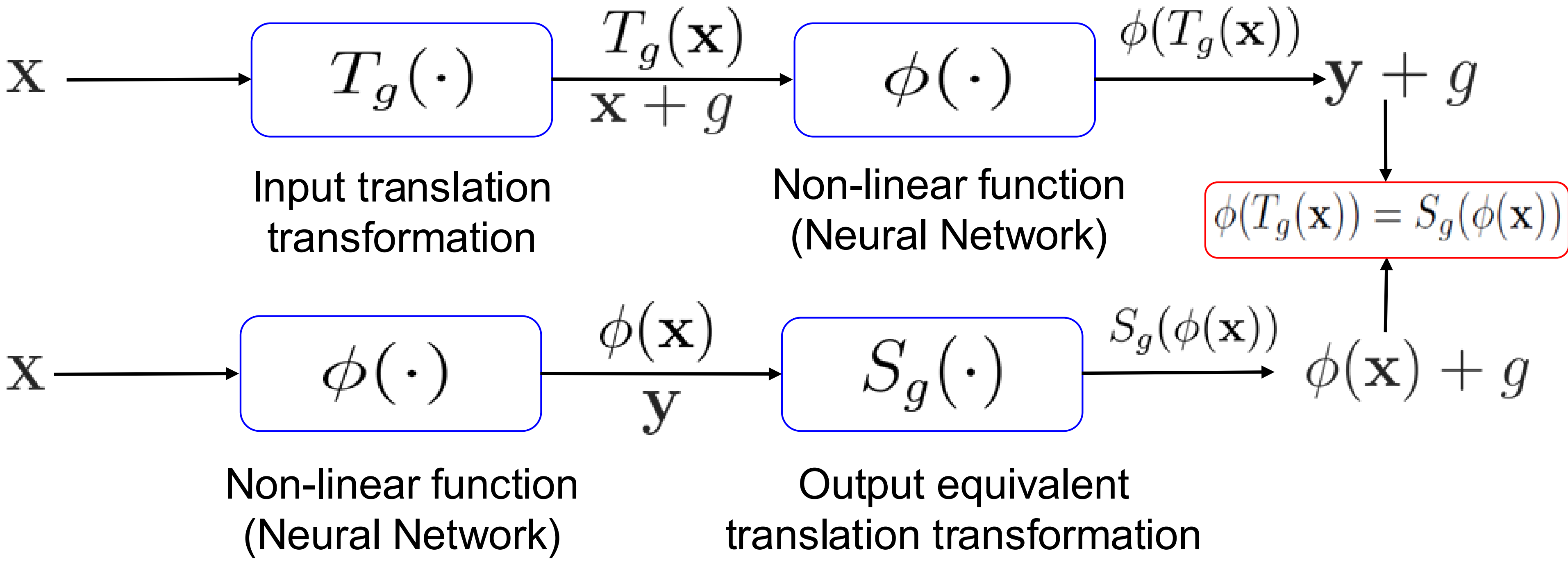
Comparison Between different networks architectures and proposed EGNN

Method	Symmetry	Complexity / Speed	Space	Implementation & Limitations
Standard GNN	Non-equivariant	Low (Fast)	n-D	Lacks physical inductive biases.
Radial Field	E(n)-Equivariant	Low (Fast)	n-D	Drops node features (only updates coordinates).
Tensor Field Network TFN / SE(3)-Tr.	SE(3)-Equivariant	High (Slow)	3D only	Requires complex spherical harmonics.
Schnet	E(n)-Invariant	Low (Fast)	n-D	Does not update coordinates.
EGNN	E(n)-Equivariant	Low (Fast)	n-D	Updates features, coordinates, & skips spherical harmonics.

1- Translation equivariance

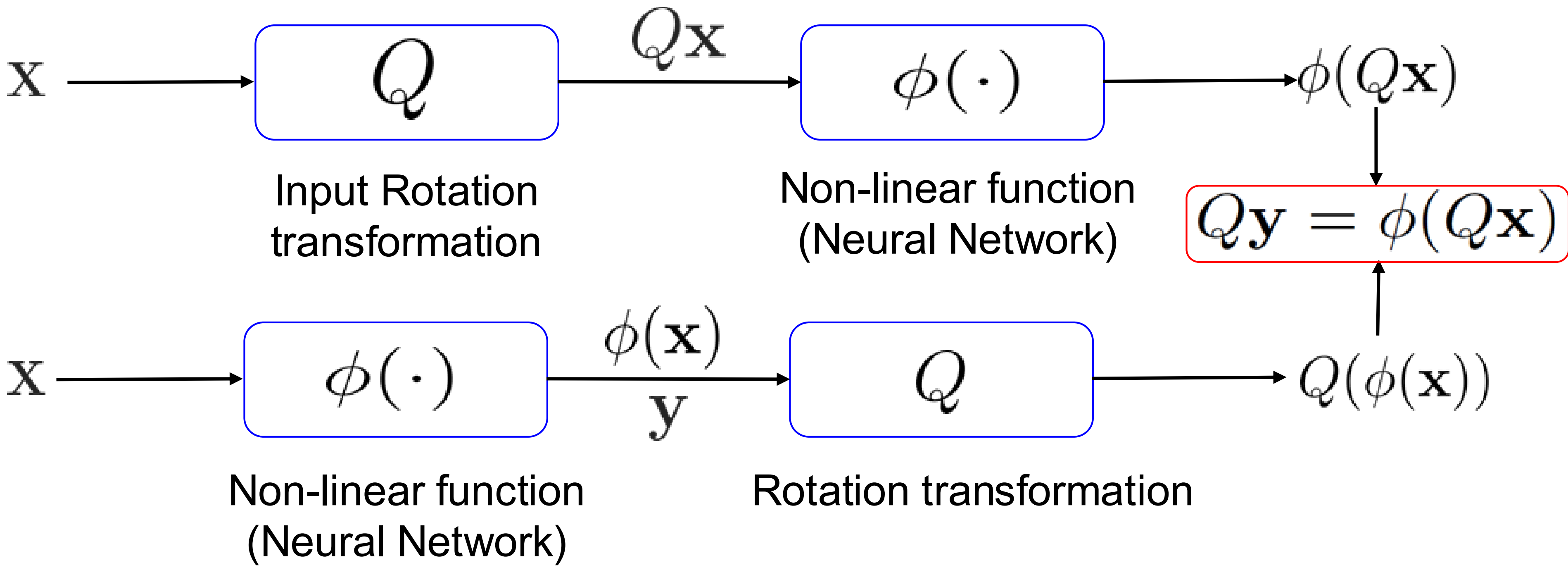
$$\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_M) \in \mathbb{R}^{M\times n}$$

$$\phi(\mathbf{x}+g) = \phi(\mathbf{x}) + g = \mathbf{y} + g.$$



2- Rotation equivariance

Q is orthogonal matrix of dimension $\mathbb{R}^{n\times n}$



Standard GNN

$$\boxed{\mathbf{h}^{l+1} = \mathrm{GCL}[\mathbf{h}^l, \mathcal{E}]}$$

1- Message passing

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

2- Message aggregation

$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$

3- node embedding update

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

Proposed Equivariant GNN (EGNN)

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

1- Message passing

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

2- Message aggregation

$$\mathbf{m}_i = \sum_{j \neq i} \mathbf{m}_{ij}$$

3- Particle Position update

$$\boxed{\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \underbrace{C \sum_{j \neq i} (\mathbf{x}_i^l - \mathbf{x}_j^l)\phi_x(\mathbf{m}_{ij})}_{\text{average weighted sum of the relative differences}}}$$

4- node embedding update

$$C = \frac{1}{M-1}.$$

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

1- Extensions of Equivariant GNN (EGNN) to Vector type representation

Introducing particle's momentum

$$\mathbf{h}^{l+1}, \mathbf{x}^{l+1}, \boxed{\mathbf{v}^{l+1}} = \mathrm{EGCL}[\mathbf{h}^{l}, \mathbf{x}^{l}, \boxed{\mathbf{v}^{\text{init}}}, \mathcal{E}]$$

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

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

2- Inferring the edges (at case on not knowing the adjacency matrix)

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

$$\boxed{e_{ij} \approx \phi_{inf}(\mathbf{m}_{ij})}$$

$$\boxed{\phi_{inf} : \mathbb{R}^{nf} \rightarrow [0,1]^1}$$

Invariant Variables (Type-0 / Scalars)

These remain unchanged under E(n) transformations

- $\boldsymbol{h}_i^l \in \mathbb{R}^{nf}, \boldsymbol{h}_i^l \rightarrow \boldsymbol{h}_i^l$.(node feature/embedding at layer l; E(n) invariant)
- $a_{ij}, a_{ij} \rightarrow a_{ij}$.(edge attributes between nodes i, j; E(n) invariant)
- $\left|\boldsymbol{X}_i^l - \boldsymbol{X}_j^l\right|^2 \in \mathbb{R}, \left|\boldsymbol{X}_i^l - \boldsymbol{X}_j^l\right|^2 \rightarrow \left|\boldsymbol{X}_i^l - \boldsymbol{X}_j^l\right|^2$.(squared distance; E(n) invariant)
- $\boldsymbol{m}_{ij} \in \mathbb{R}^{nf}, \boldsymbol{m}_{ij} \rightarrow \boldsymbol{m}_{ij}$.(edge message; E(n) invariant)
- $\boldsymbol{m}_i = \sum_{j \neq i} \boldsymbol{m}_{ij} \in \mathbb{R}^{nf}, \boldsymbol{m}_i \rightarrow \boldsymbol{m}_i$.(aggregated message; E(n) invariant)
- $e_{ij} \in \{0,1\}$(or $[0,1]$ when inferred), $e_{ij} \rightarrow e_{ij}$.(edge indicator/weight; E(n) invariant)

Equivariant Variables (Type-1 / Vectors)

These transform proportionally with the input E(n) transformations
(coordinates related variables : position and velocity)

- $\boldsymbol{X}_i^{l+1}=\boldsymbol{X}_i^l+C\sum_{j\neq i}(\boldsymbol{X}_i^l-\boldsymbol{X}_j^l)\,\phi_x(\boldsymbol{m}_{ij}),\ \boldsymbol{X}_i^{l+1}\rightarrow Q\boldsymbol{X}_i^{l+1}+g.\text{(coordinate update; E(n) equivariant)}$
- $\boldsymbol{X}_i^l\in\mathbb{R}^n,\ \boldsymbol{X}_i^l\rightarrow Q\boldsymbol{X}_i^l+g.\text{(coordinates at layer } l\text{; E(n) equivariant)}$
- $(X_i^l-X_j^l)\in\mathbb{R}^n,(X_i^l-X_j^l)\rightarrow Q(X_i^l-X_j^l).\text{(relative displacement; translation}-\text{invariant, SO(n) equivariant)}$
- $\boldsymbol{v}_i^{\text{init}}\in\mathbb{R}^n,\ \boldsymbol{v}_i^{\text{init}}\rightarrow Q\boldsymbol{v}_i^{\text{init}}.\text{(initial velocity; SO(n) equivariant)}$
- $\boldsymbol{v}_i^{l+1}\in\mathbb{R}^n,\ \boldsymbol{v}_i^{l+1}\rightarrow Q\boldsymbol{v}_i^{l+1}.\text{(updated velocity; SO(n) equivariant)}$

Experiments Overview

Evaluate EGNN on three tasks:

- N-body dynamical system

  → Learning physical interactions and trajectory prediction
- Graph autoencoder

  → Representation learning under structural symmetry
- Molecular property prediction (QM9)

  → Predicting invariant properties of molecules

Goals:

- Demonstrate accuracy compared to prior methods
- Show generality across domains
- Highlight computational efficiency and model simplicity

Experiment 1: N-body Dynamical System

Task

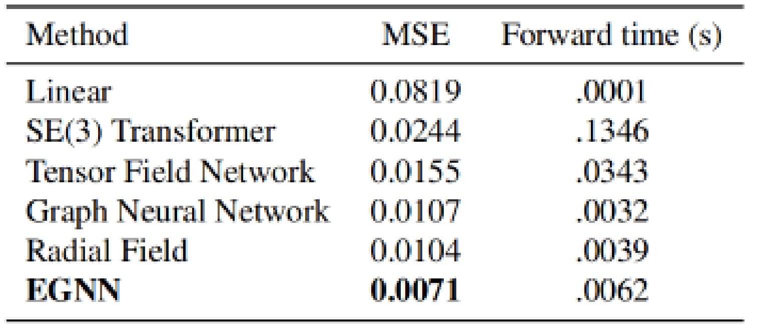
- Predict particle positions after 1000 timesteps
- Input:
    1. Initial positions
    2. Velocities
    3. Charges
- Physics-based interactions (attraction / repulsion)

Baselines

- Graph Neural Network (GNN)
- Radial Field
- Tensor Field Network (TFN)
- SE(3) Transformer

Results

- **EGNN achieves lowest prediction error**
- More efficient than higher-order equivariant models
- No spherical harmonics required



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

Table 2. Mean Squared Error for the future position estimation in the N-body system experiment, and forward time in seconds for a batch size of 100 samples running in a GTX 1080Ti GPU.

Experiment 2: Graph Autoencoder

Task

- Learn node embeddings to reconstruct graph structure
- Input: adjacency matrix (no node features)
- Output: reconstructed edges

Challenge: Symmetry Problem

- Nodes with identical structure → identical embeddings in GNN
- Makes reconstruction difficult

EGNN Solution

- Introduce coordinates (random noise as positional features)
- Preserve equivariance while breaking symmetry
- Enables distinct node representations

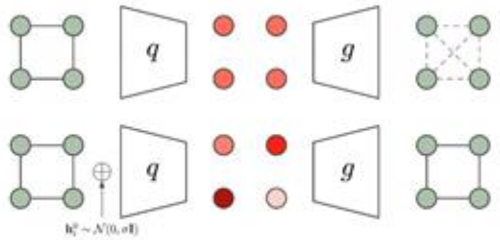
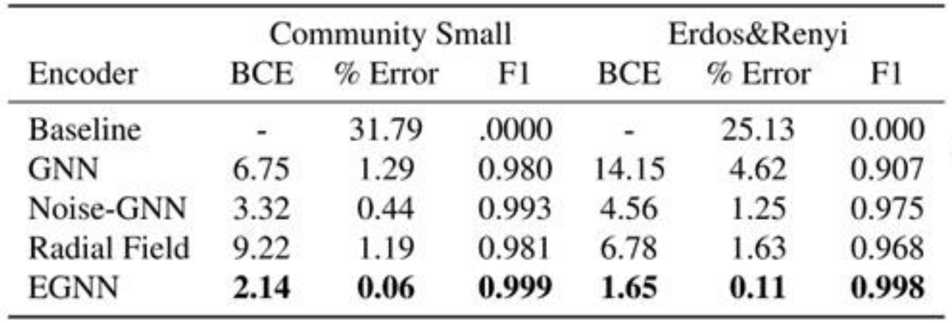


Figure 3. Visual representation of a Graph Autoencoder for a 4 nodes cycle graph. The bottom row illustrates that adding noise at the input graph breaks the symmetry of the embedding allowing the reconstruction of the adjacency matrix.

Experiment 2: Graph Autoencoder

Results

- EGNN achieves the lowest reconstruction error
- Near-perfect performance on multiple datasets
- Outperforms GNN and other baselines



	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**

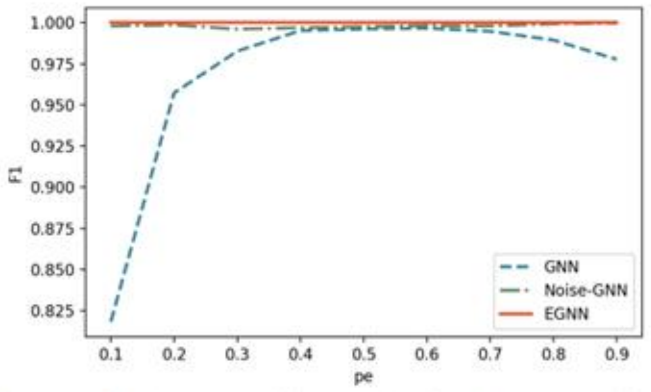


Figure 5. In the Table at the left we report the Binary Cross Entropy, % Error and F1 scores for the test partition on the Graph Autoencoding experiment in the Community Small and Erdos&Renyi datasets. In the Figure at the right, we report the F1 score when overfitting a training partition of 100 samples in the Erdos&Renyi dataset for different values of sparsity p_e. The GNN is not able to successfully auto-encode sparse graphs (small p_e values) for the Erdos&Renyi dataset even when training and testing on the same small subset.

Experiment 3: Molecular Property Prediction (QM9)

Task

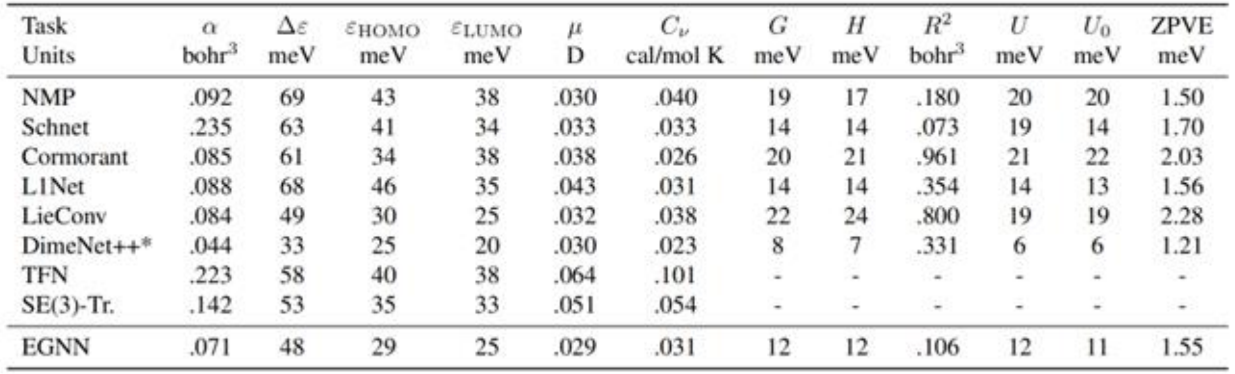
- Predict molecular properties from 3D structures
- Input:
  - Atom types
  - Atomic coordinates
- Properties are invariant to rotati

Baselines

- SchNet
- MPNN
- Tensor Field Network (TFN)
- SE(3) Transformer

Results

- EGNN achieves competitive or superior performance
- Comparable to more complex equivariant models
- Does not require higher-order tensor representations



Task	α	$\Delta\varepsilon$	$\varepsilon_{\mathrm{HOMO}}$	$\varepsilon_{\mathrm{LUMO}}$	μ	C_ν	G	H	R^2	U	U_0	ZPVE
Units	bohr^3	meV	meV	meV	D	cal/mol K	meV	meV	bohr^3	meV	meV	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

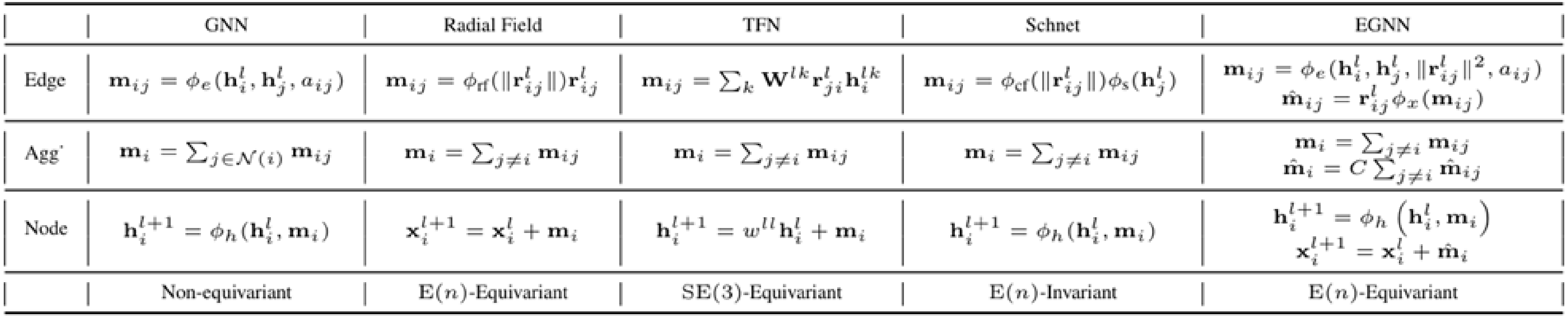
Table 3. Mean Absolute Error for the molecular property prediction benchmark in QM9 dataset. *DimeNet++ uses slightly different train/val/test partitions than the other papers listed here.

Take-Home Messages: Strengths and Weaknesses

- Strengths: theoretical grounding, makes sense of surrounding research, theoretical proofs make sense.

Take-Home Messages and Additional Comments

- Strengths: theoretical grounding, makes sense of surrounding research, theoretical proofs make sense.



	GNN	Radial Field	TFN	Schnet	EGNN
Edge	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$	$\mathbf{m}_{ij} = \phi_{\mathrm{rf}}(\lVert\mathbf{r}_{ij}^l\rVert)\mathbf{r}_{ij}^l$	$\mathbf{m}_{ij} = \sum_k \mathbf{W}^{lk}\mathbf{r}_{ji}^l\mathbf{h}_i^{lk}$	$\mathbf{m}_{ij} = \phi_{\mathrm{cf}}(\lVert\mathbf{r}_{ij}^l\rVert)\phi_s(\mathbf{h}_j^l)$	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, \lVert\mathbf{r}_{ij}^l\rVert^2, a_{ij})$ $\hat{\mathbf{m}}_{ij} = \mathbf{r}_{ij}^l\phi_x(\mathbf{m}_{ij})$
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

Take-Home Messages and Additional Comments

- Strengths: theoretical grounding, makes sense of surrounding research, theoretical proofs make sense.
- Weaknesses: Not particularly novel or significant
    - Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020). Learning to Simulate Complex Physics with Graph Networks. In H. Daumé III & A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning (Vol. 119, pp. 8459–8468). PMLR. https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html

Take-Home Messages and Additional Comments

- Strengths: theoretical grounding, makes sense of surrounding research, theoretical proofs make sense.
- Weaknesses: Not particularly novel or significant
  - Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020). Learning to Simulate Complex Physics with Graph Networks. In H. Daumé III & A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning (Vol. 119, pp. 8459–8468). PMLR. https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html
- Relatively well written.

Lessons Learned

- More complete/expressive function classes are not always necessary for good performance
  - Recall: ChebyNet

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

Lessons Learned

- More complete/expressive function classes are not always necessary for good performance
    - Recall: ChebyNet

$$h_\theta * X \approx \sum_{n=0}^{K} \theta_n T_n(\tilde{X})$$

$$T_0(\tilde{X}) = X$$

$$T_1(\tilde{X}) = 2LX/\lambda_{\max} - X$$

$$T_{n+1}(\tilde{X}) = 2\left(\frac{2L}{\lambda_{\max}} - I\right) T_n(\tilde{X}) - T_{n-1}(\tilde{X})$$

What if we truncate to 1$^{\text{st}}$ order?

Lessons Learned

- More complete/expressive function classes are not always necessary for good performance

Tensor Field Networks

$$Y_m^{(l)}(\mathcal{R}(g)\hat{r}) = \sum_{m'} D_{mm'}^{(l)}(g) Y_{m'}^{(l)}(\hat{r})$$

$$F_{cm}^{(l_f, l_i)}(\vec{r}) = R_c^{(l_f, l_i)}(r) Y_m^{(l_f)}(\hat{r})$$

EGNN

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, \|\mathbf{r}_{ij}^l\|^2, a_{ij})$$
$$\hat{\mathbf{m}}_{ij} = \mathbf{r}_{ij}^l \phi_x(\mathbf{m}_{ij})$$

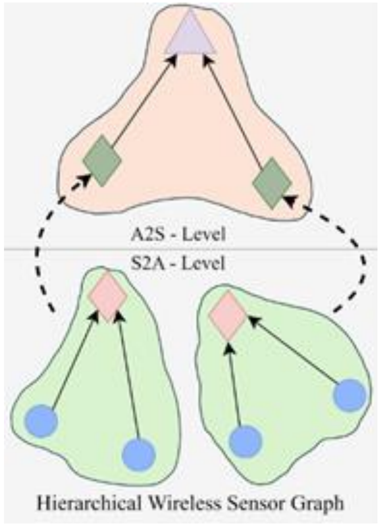
$$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$$
$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \hat{\mathbf{m}}_i$$

Questions

- Why is higher dimension E(n) ever useful?
- Practical applications… have they tried scaling it, and seeing how it performs on bigger applications?
  - Power Grids
  - Wind Farms
  - Sensor Networks

Suggestions for improvement

- Methods to deal with the long range problem
  - Approximately equivariant clustering/pooling algorithms / Hierarchical GNN



Source: Hwang, W.-J., Giang, L. T., Nguyen, X. T., Viet, V. H., Chien, T. V., & Hoa, N. T. (2024). *HGNN: A hierarchical graph neural network architecture for joint resource management in dynamic wireless sensor networks.* **IEEE Sensors Journal, 24**(24), 352–364.

Suggestions for improvement

- Methods to deal with the long range problem
  - Approximately equivariant clustering/pooling algorithms / Hierarchical GNN

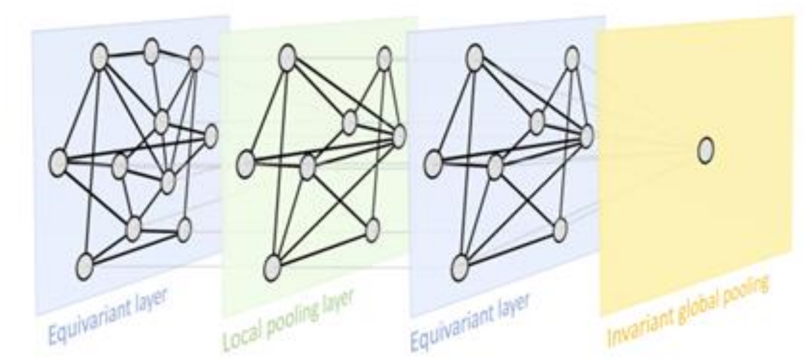


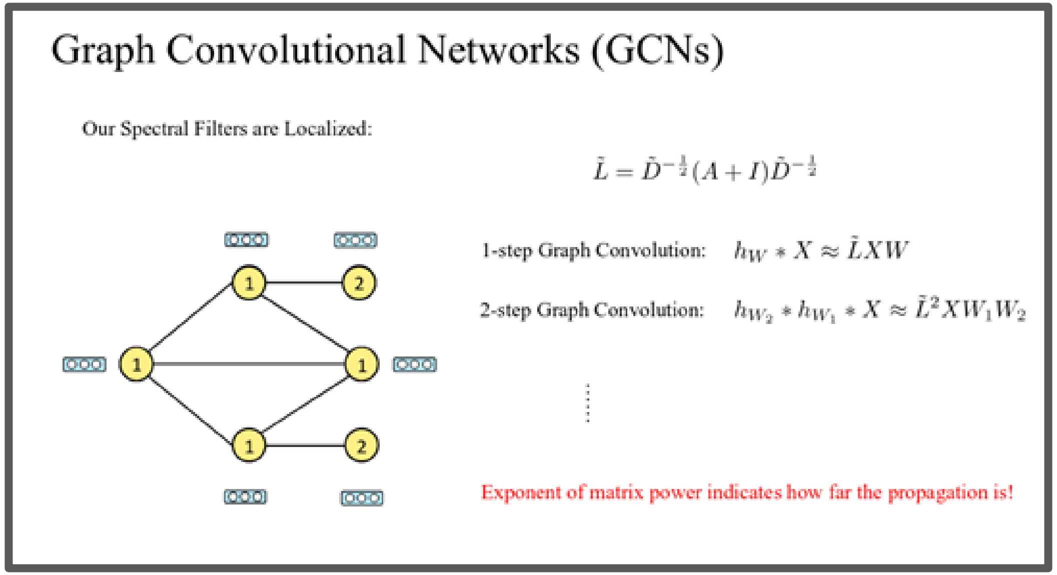
Figure 8: Geometric Deep Learning blueprint, exemplified on a graph. A typical Graph Neural Network architecture may contain permutation equivariant layers (computing node-wise features), local pooling (graph coarsening), and a permutation-invariant global pooling layer (readout layer).

Source: Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). *Geometric deep learning: Grids, groups, graphs, geodesics, and gauges*. **IEEE Signal Processing Magazine, 38**(5), 18–42. https://doi.org/10.1109/MSP.2021.3051845

Suggestions for improvement

- Methods to deal with the long range problem
  - Approximately equivariant clustering/pooling algorithms / Hierarchical GNN
  - Equivariant Laplacian for long range relationships

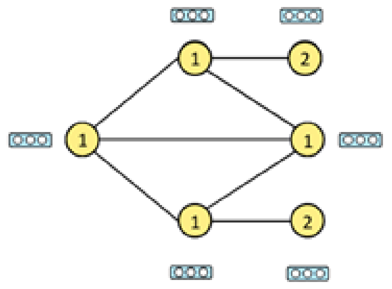
Recall:



Graph Convolutional Networks (GCNs)

Our Spectral Filters are Localized:

$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}$$



1-step Graph Convolution: $h_W * X \approx \tilde{L}XW$

2-step Graph Convolution: $h_{W_2} * h_{W_1} * X \approx \tilde{L}^2 X W_1 W_2$

$\vdots$

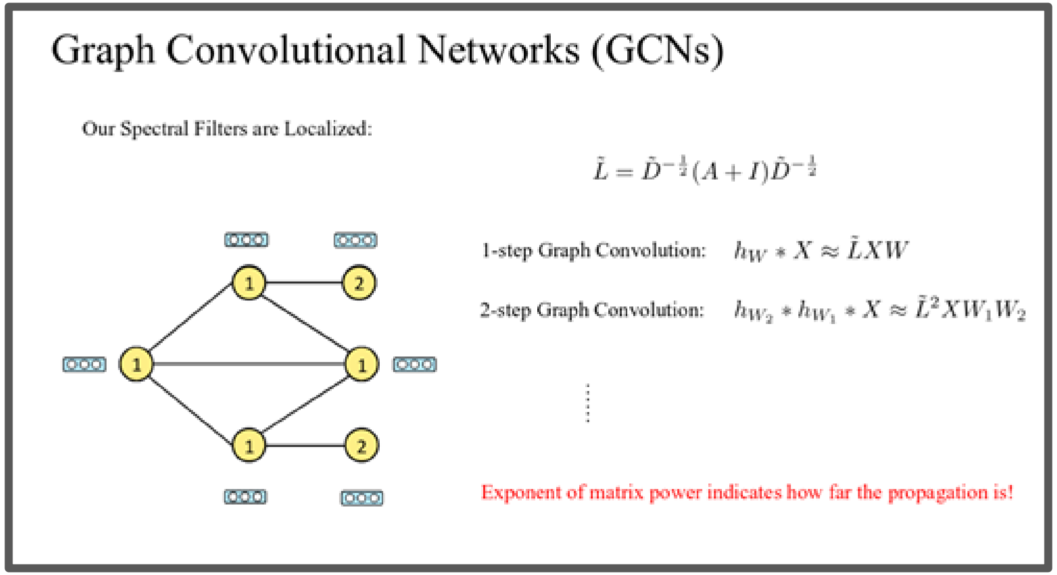
Exponent of matrix power indicates how far the propagation is!

Source: Liao, R. (2025). *EECE 571F: Advanced topics in deep learning* [Lecture slides]. University of British Columbia.

Suggestions for improvement

- Methods to deal with the long range problem
  - Approximately equivariant clustering/pooling algorithms / Hierarchical GNN
  - Equivariant Laplacian for long range relationships… (Or stacking even more layers)

Recall:



Graph Convolutional Networks (GCNs)

Our Spectral Filters are Localized:

$$\tilde{L} = \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}$$

1-step Graph Convolution: $h_W * X \approx \tilde{L}XW$

2-step Graph Convolution: $h_{W_2} * h_{W_1} * X \approx \tilde{L}^2 X W_1 W_2$

⋮

Exponent of matrix power indicates how far the propagation is!

Source: Liao, R. (2025). *EECE 571F: Advanced topics in deep learning* [Lecture slides]. University of British Columbia.

Suggestions for improvement

- Methods to deal with the long range problem
  - Approximately equivariant clustering/pooling algorithms / Hierarchical GNN
  - Equivariant Laplacian for long range relationships
  - Powers of the adjacency matrix as input to network (perhaps for sparse high diameter graphs)

Appendix

Proof of Invariance

Translation Invariance (square relative displacement)

$$\|\mathbf{x}_i^l + \cancel{g} - [\mathbf{x}_j^l + \cancel{g}]\|^2 = \|\mathbf{x}_i^l - \mathbf{x}_j^l\|^2$$

Rotation Invariance (square relative displacement)

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

Rotation and Translation Invariance (edge message)

$$\mathbf{m}_{i,j} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \|Q\mathbf{x}_i^l + \cancel{g} - [Q\mathbf{x}_j^l + \cancel{g}]\|^2, a_{ij}\right) = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \|\mathbf{x}_i^l - \mathbf{x}_j^l\|^2, a_{ij}\right)$$

Proof of Equivariance

Rotation and Translation Equivariance (coordinates position of node)

*** Similarly extended to coordinates velocity

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

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

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

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