

Group Equivariant Deep Learning

Lecture 3 - Equivariant graph neural networks

Lecture 3.2 - Equivariant message passing as non-linear convolution

Neural Message Passing for Quantum Chemistry

Justin Gilmer 1 Samuel S. Schoenholz 1 Patrick F. Riley 2 Oriol Vinyals 3 George E. Dahl 1

Abstract

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Passing Neural Networks (MPNNs) and explore additional novel variations within this framework. Using MPNNs we demonstrate state of the art results on an important molecular property prediction benchmark; these results are strong enough that we believe future work should focus on datasets with larger molecules or more accurate ground truth labels.

1. Introduction

The past decade has seen remarkable success in the use of deep neural networks to understand and translate natural language (Wu et al., 2016), generate and decode complex audio signals (Hinton et al., 2012), and infer features from real-world images and videos (Krizhevsky et al., 2012). Although chemists have applied machine learning to many problems over the years, predicting the properties of molecules and materials using machine learning
(and especially deep learning) is still in its infancy. To
date, most research applying machine learning to chemistry
tasks (Hansen et al., 2015; Huang & von Lilienfeld, 2016;

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

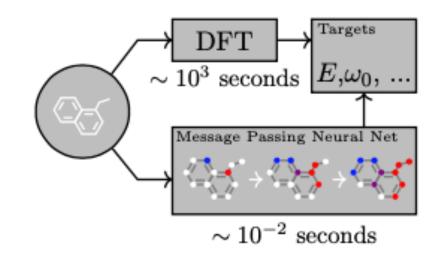


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Rupp et al., 2012; Rogers & Hahn, 2010; Montavon et al., 2012; Behler & Parrinello, 2007; Schoenholz et al., 2016) has revolved around feature engineering. While neural networks have been applied in a variety of situations (Merkwirth & Lengauer, 2005; Micheli, 2009; Lusci et al., 2013; Duvenaud et al., 2015), they have yet to become widely adopted. This situation is reminiscent of the state of image models before the broad adoption of convolutional neural networks and is due, in part, to a dearth of empirical evidence that neural architectures with the appropriate inductive bias can be successful in this domain.

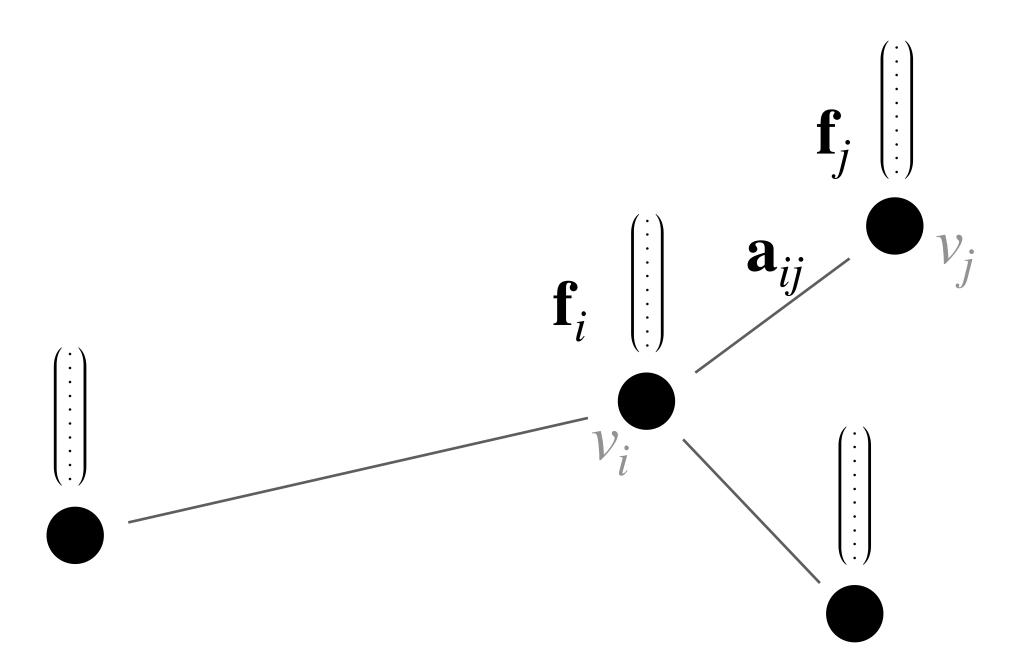
Recently, large scale quantum chemistry calculation and molecular dynamics simulations coupled with advances in high throughput experiments have begun to generate data at an unprecedented rate. Most classical techniques do not make effective use of the larger amounts of data that are now available. The time is ripe to apply more powerful and flexible machine learning methods to these problems, assuming we can find models with suitable inductive biases. The symmetries of atomic systems suggest neural networks that operate on graph structured data and are invariant to graph isomorphism might also be appropriate for molecules. Sufficiently successful models could someday help automate challenging chemical search problems in drug discovery or materials science.

In this paper, our goal is to demonstrate effective machine learning models for chemical prediction problems

Google Brain ²Google ³Google DeepMind. Correspondence to: Justin Gilmer < gilmer@google.com>, George E. Dahl < gdahl@google.com>.

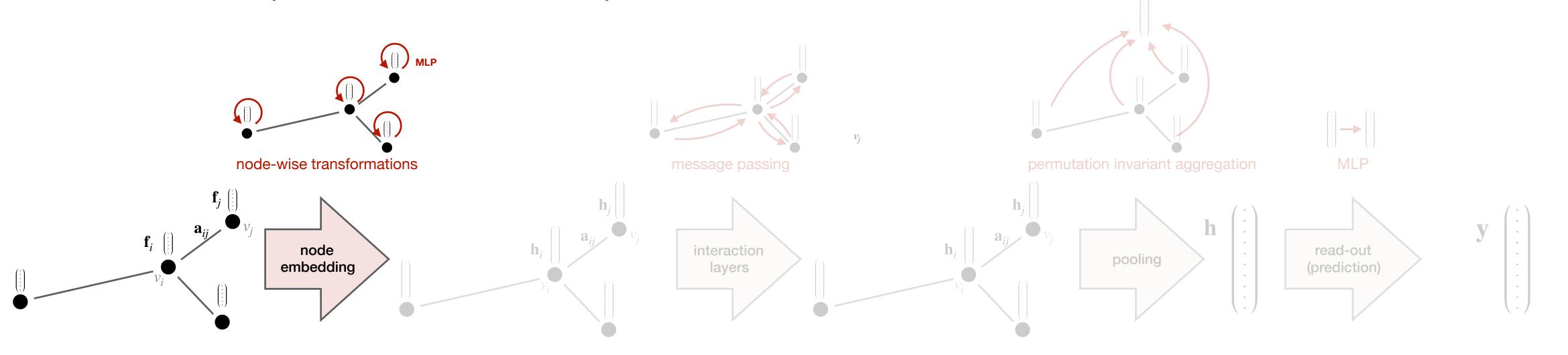
Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Input graph

Initial node embeddings

Final node embeddings

Graph embedding

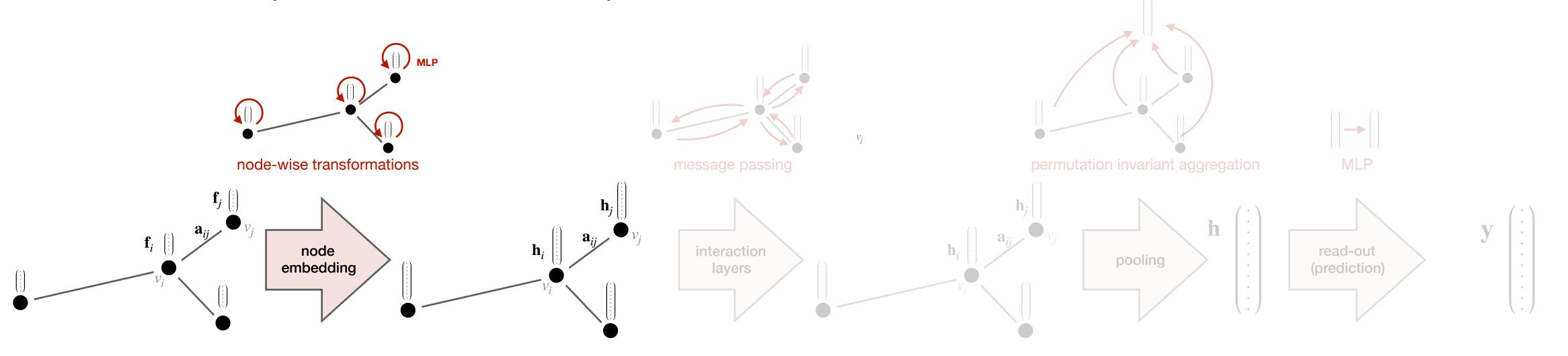
Graph-level prediction

read-out (prediction)

Node-level predictions

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Input graph

Initial node embeddings

Final node embeddings

Graph embedding

Graph-level prediction

read-out (prediction)

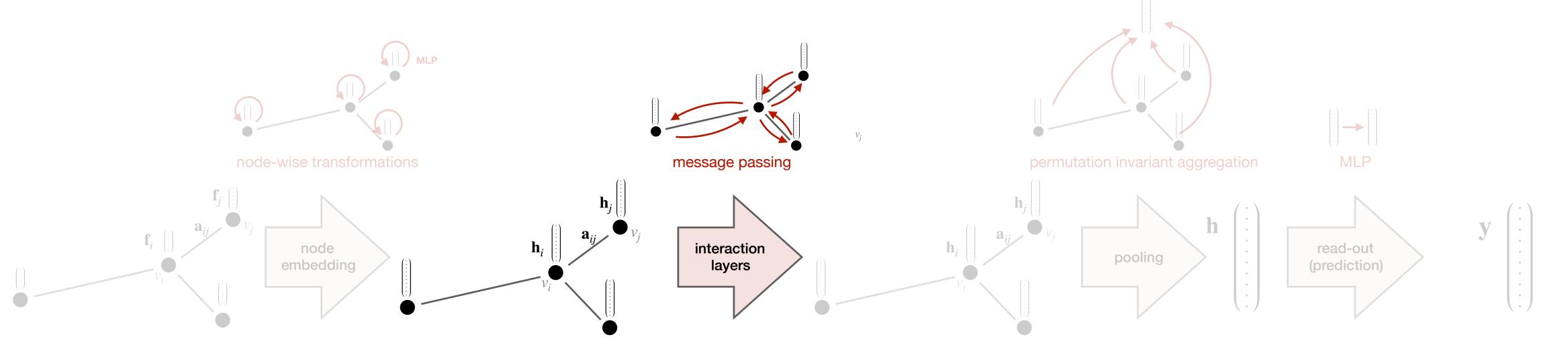
Node-level predictions

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Graph embedding



Input graph

Initial node embeddings

Final node embeddings

read-out (prediction)

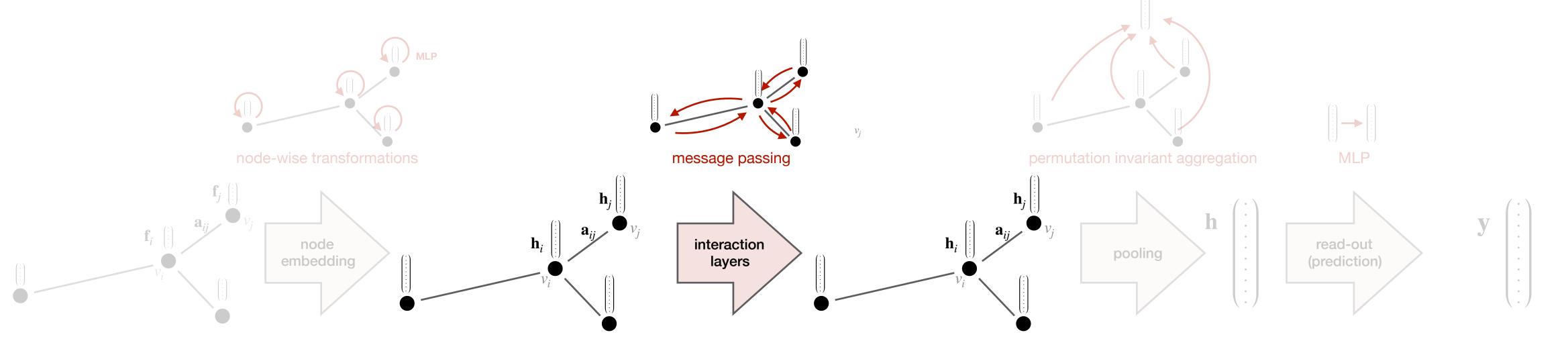
y_i y_i y_i

Node-level predictions

Graph-level prediction

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



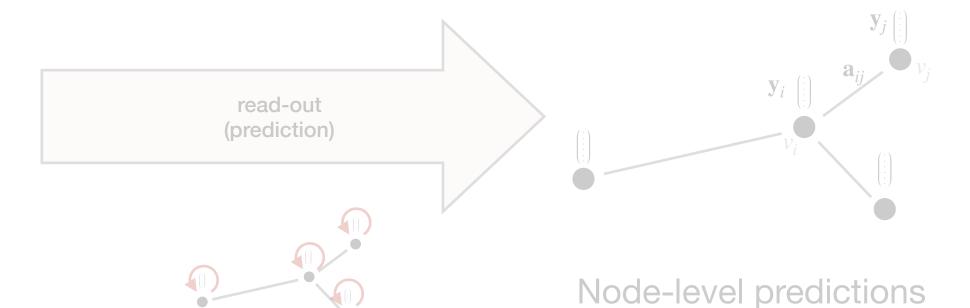
Input graph

Initial node embeddings

Final node embeddings

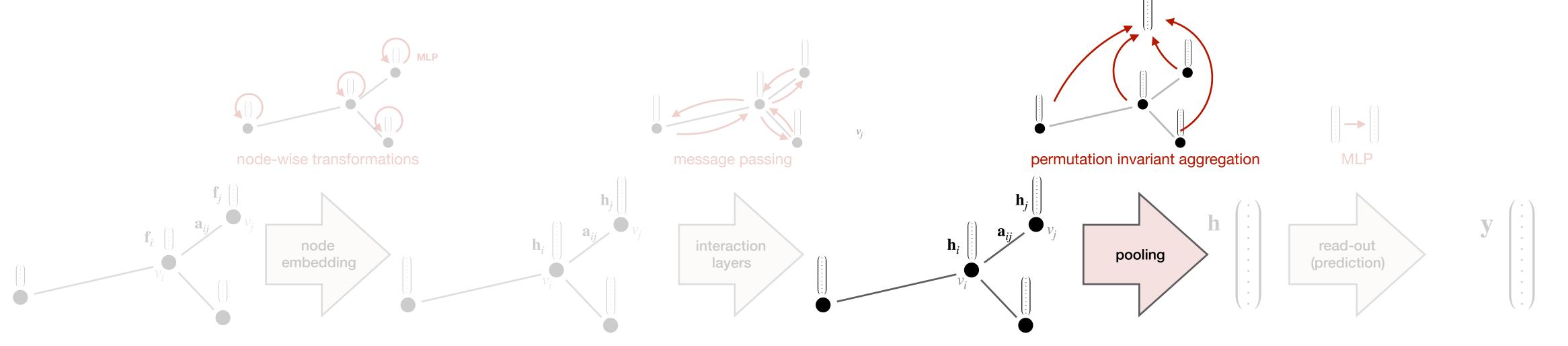
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



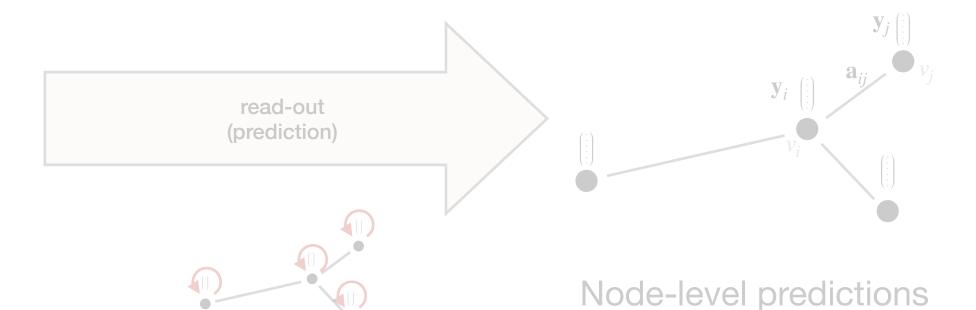
Input graph

Initial node embeddings

Final node embeddings

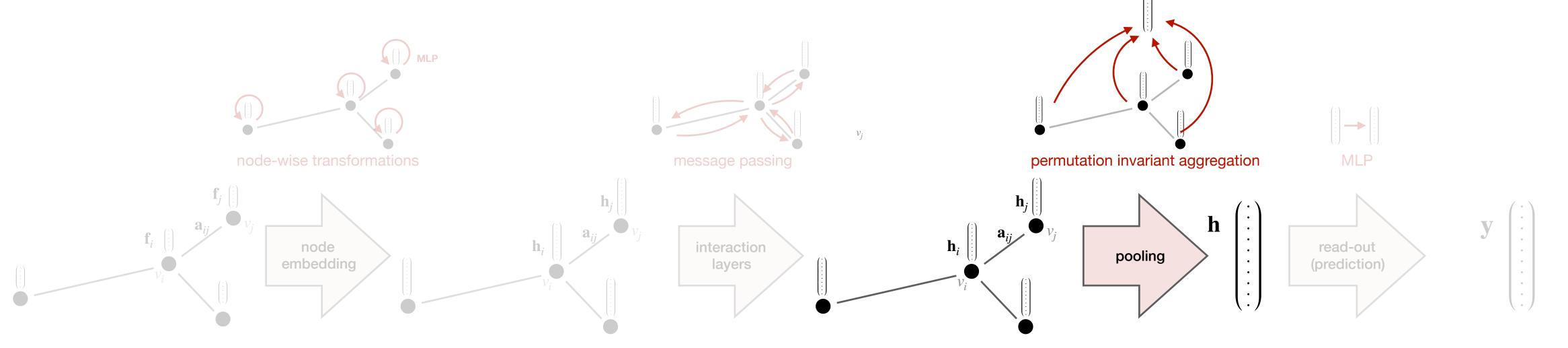
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



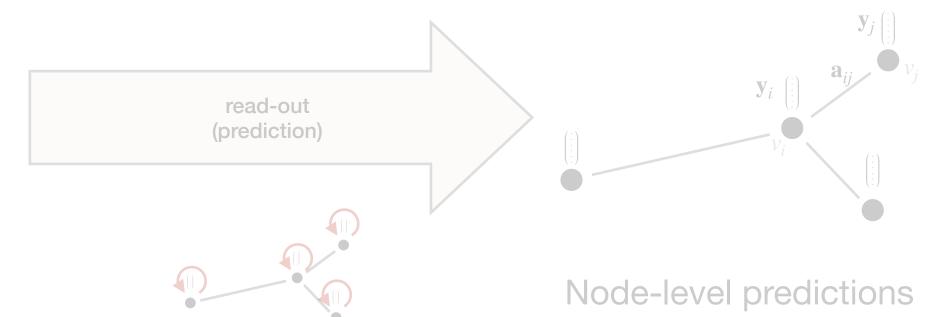
Input graph

Initial node embeddings

Final node embeddings

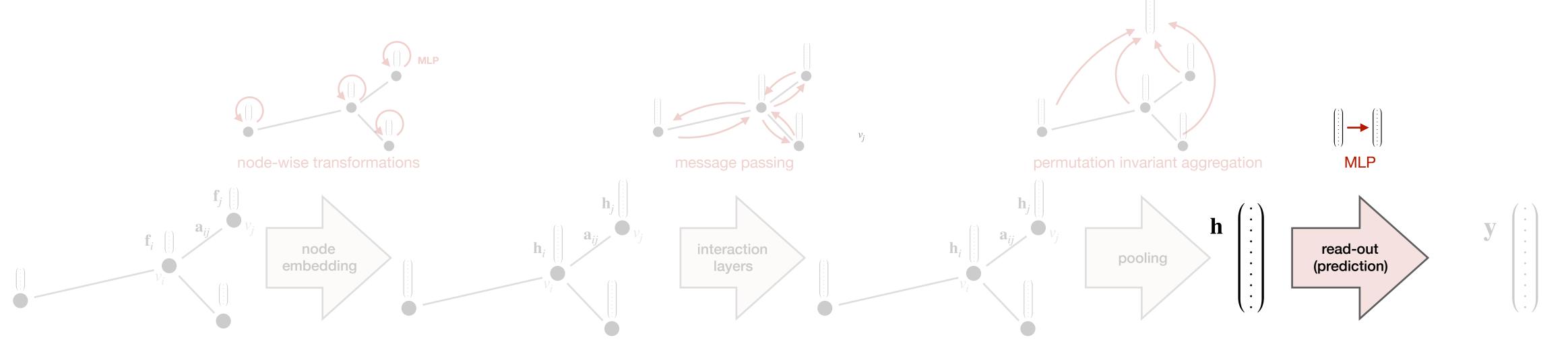
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



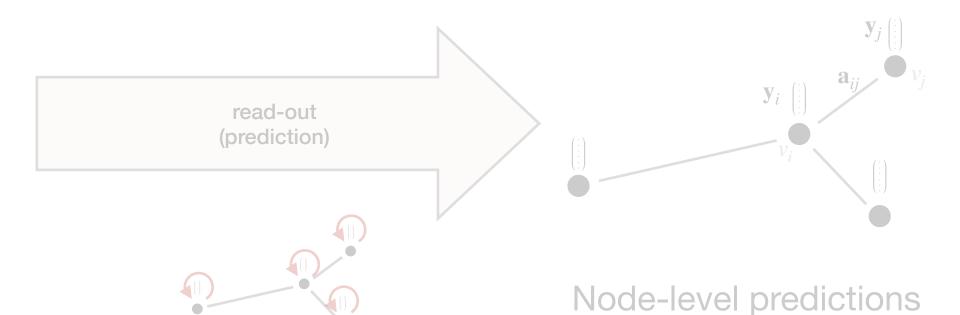
Input graph

Initial node embeddings

Final node embeddings

Graph embedding

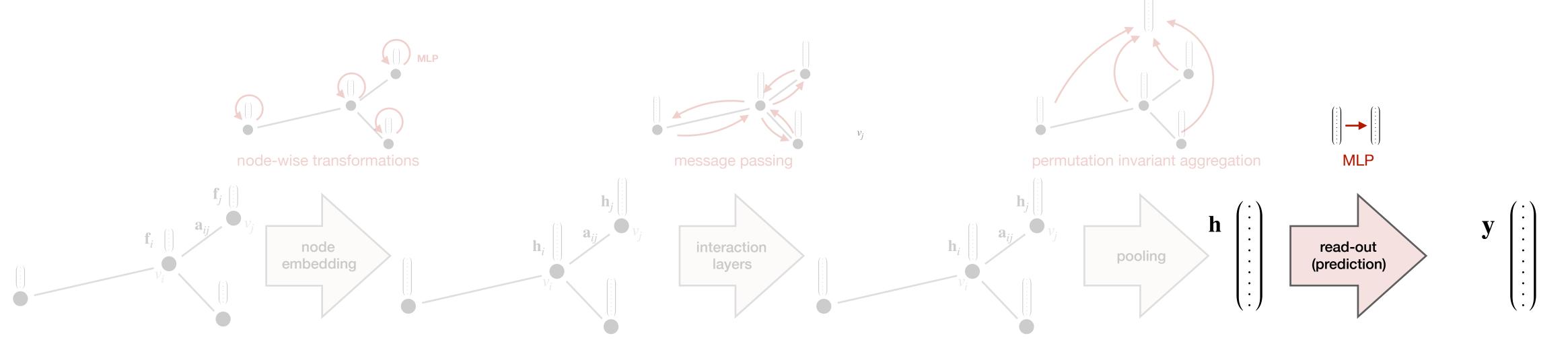
Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$



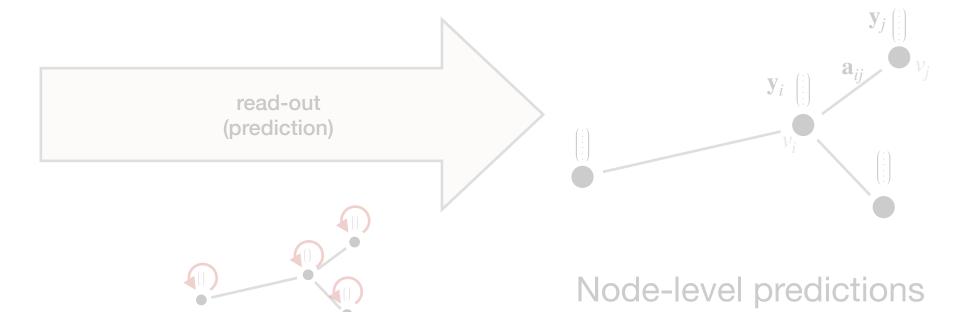
Input graph

Initial node embeddings

Final node embeddings

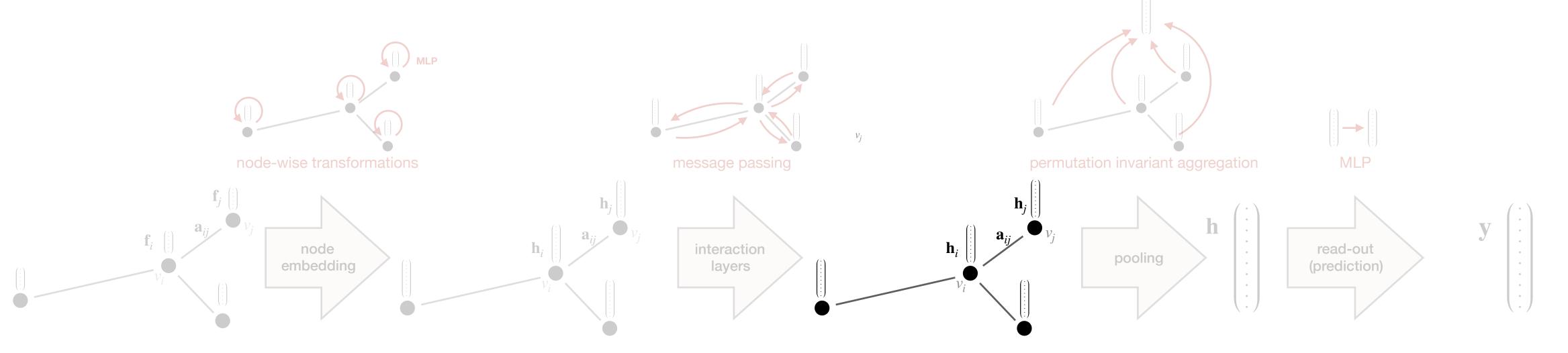
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



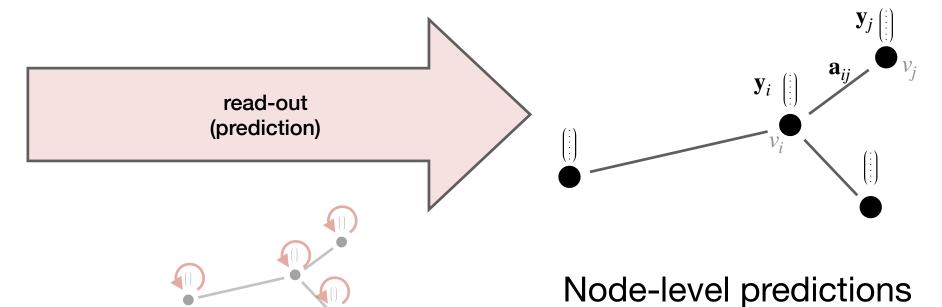
Input graph

Initial node embeddings

Final node embeddings

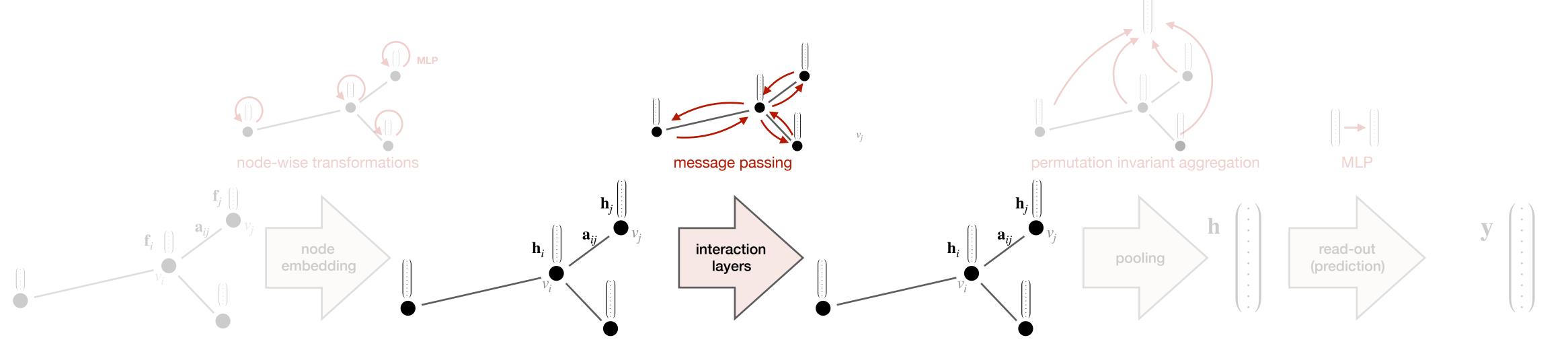
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



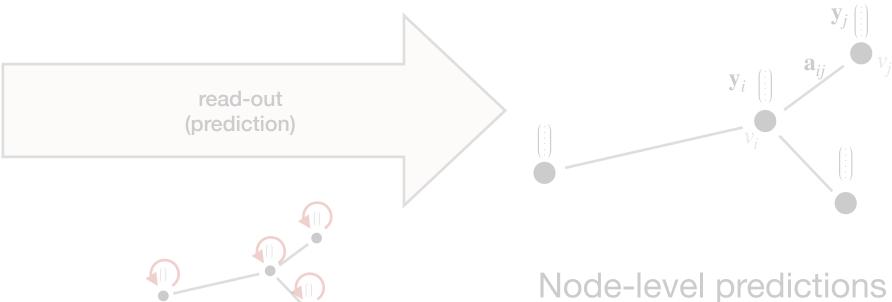
Input graph

Initial node embeddings

Final node embeddings

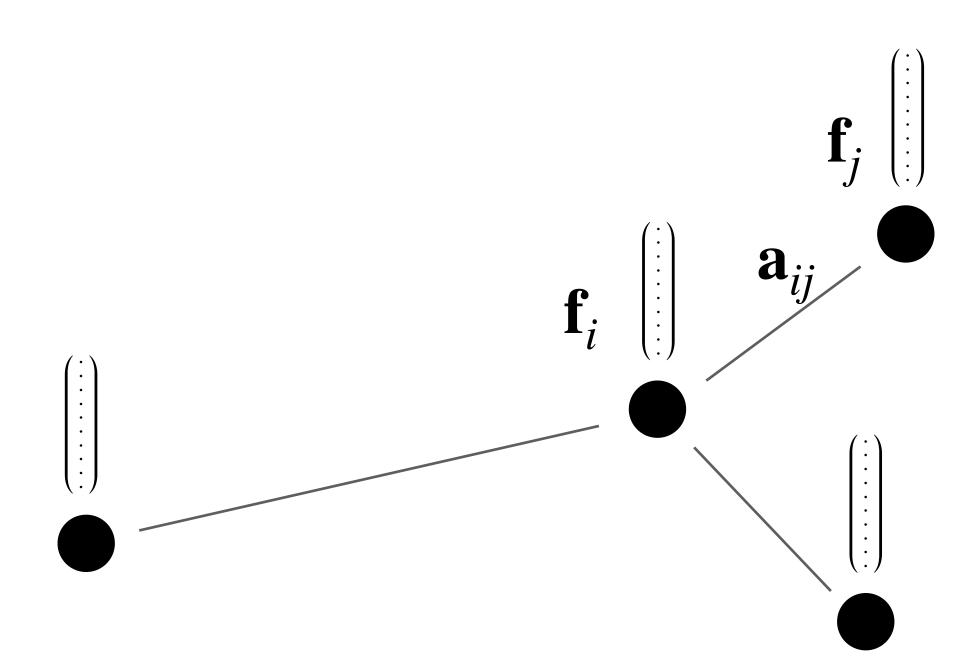
Graph embedding

Graph-level prediction



Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

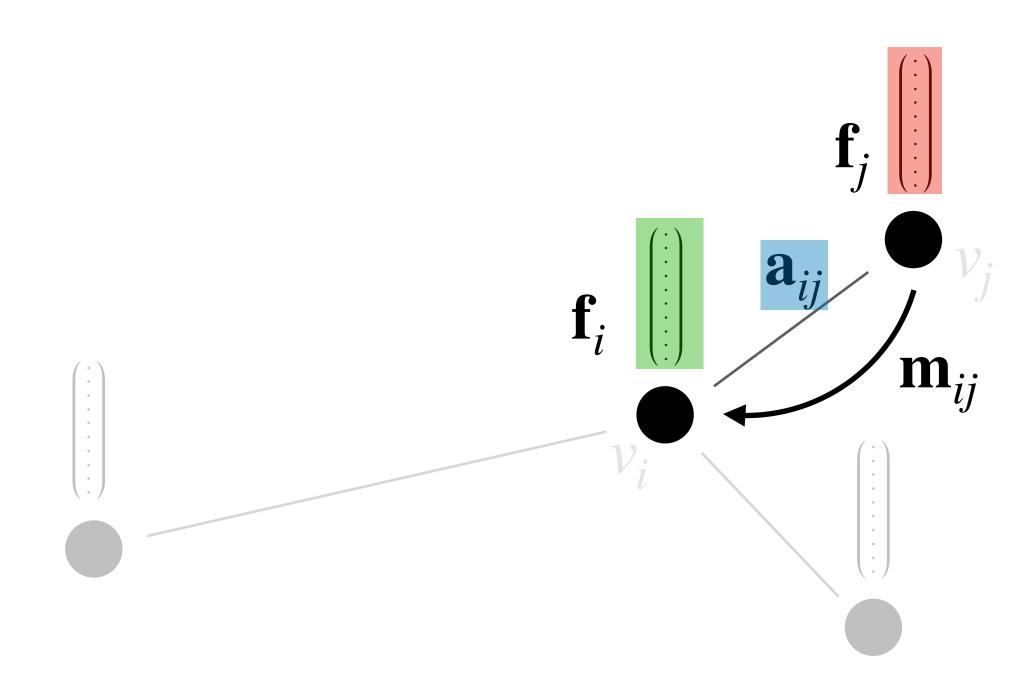
Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

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

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

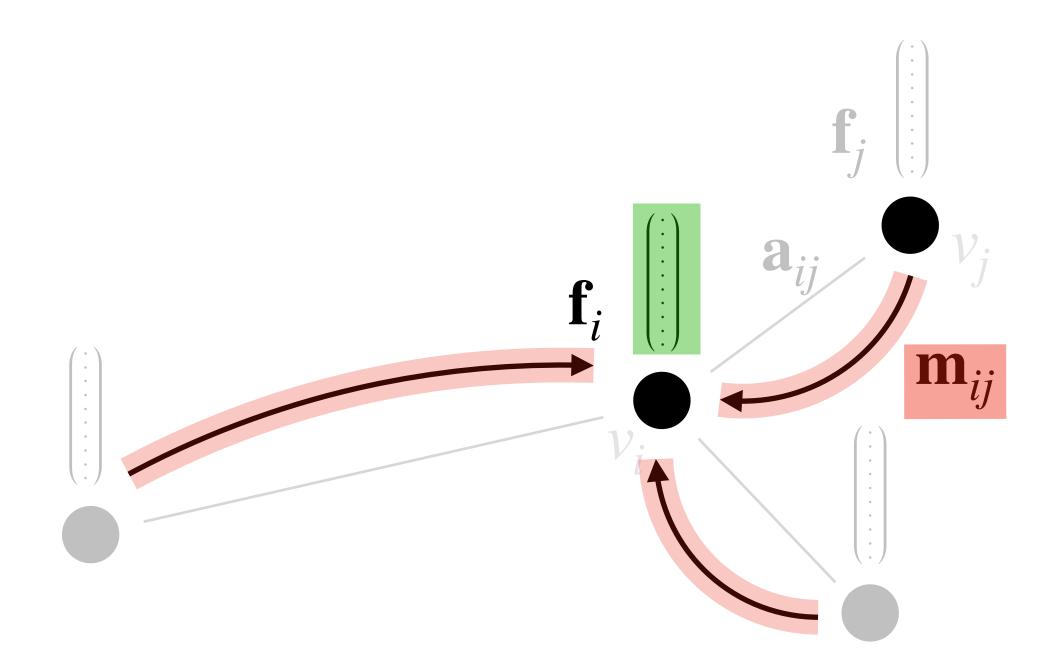
Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

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

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

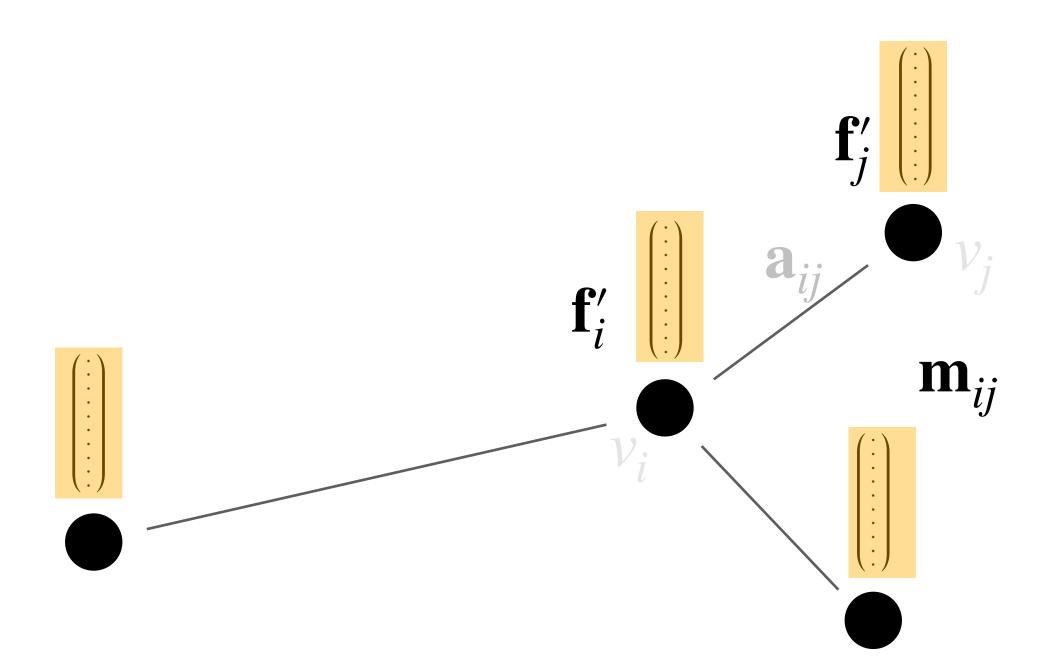
Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

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

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

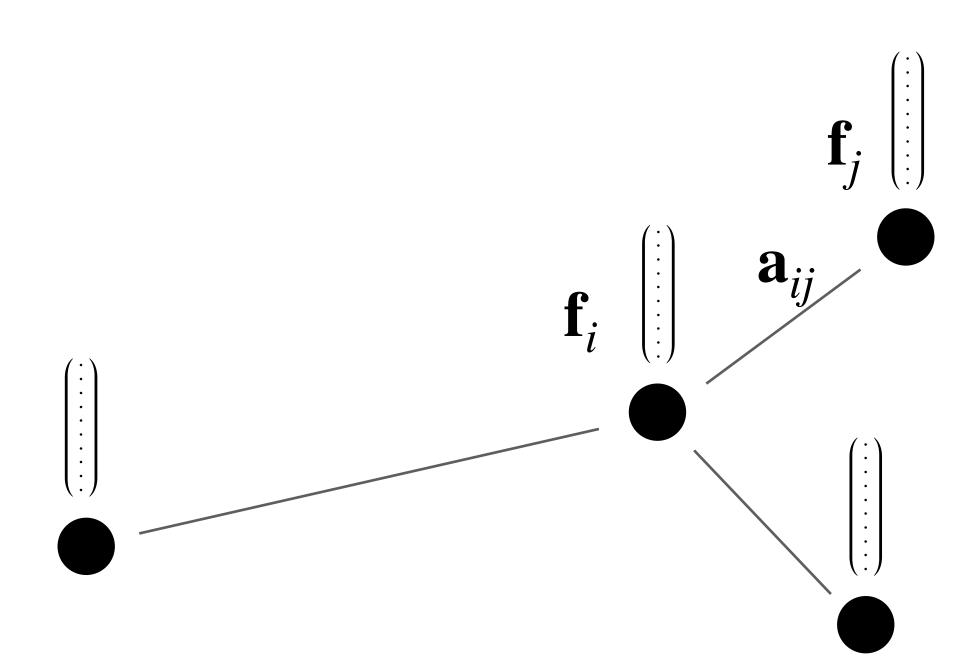
Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

$$\mathbf{f}_{i}' = \phi_{f}(\mathbf{f}_{i}, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij})$$

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



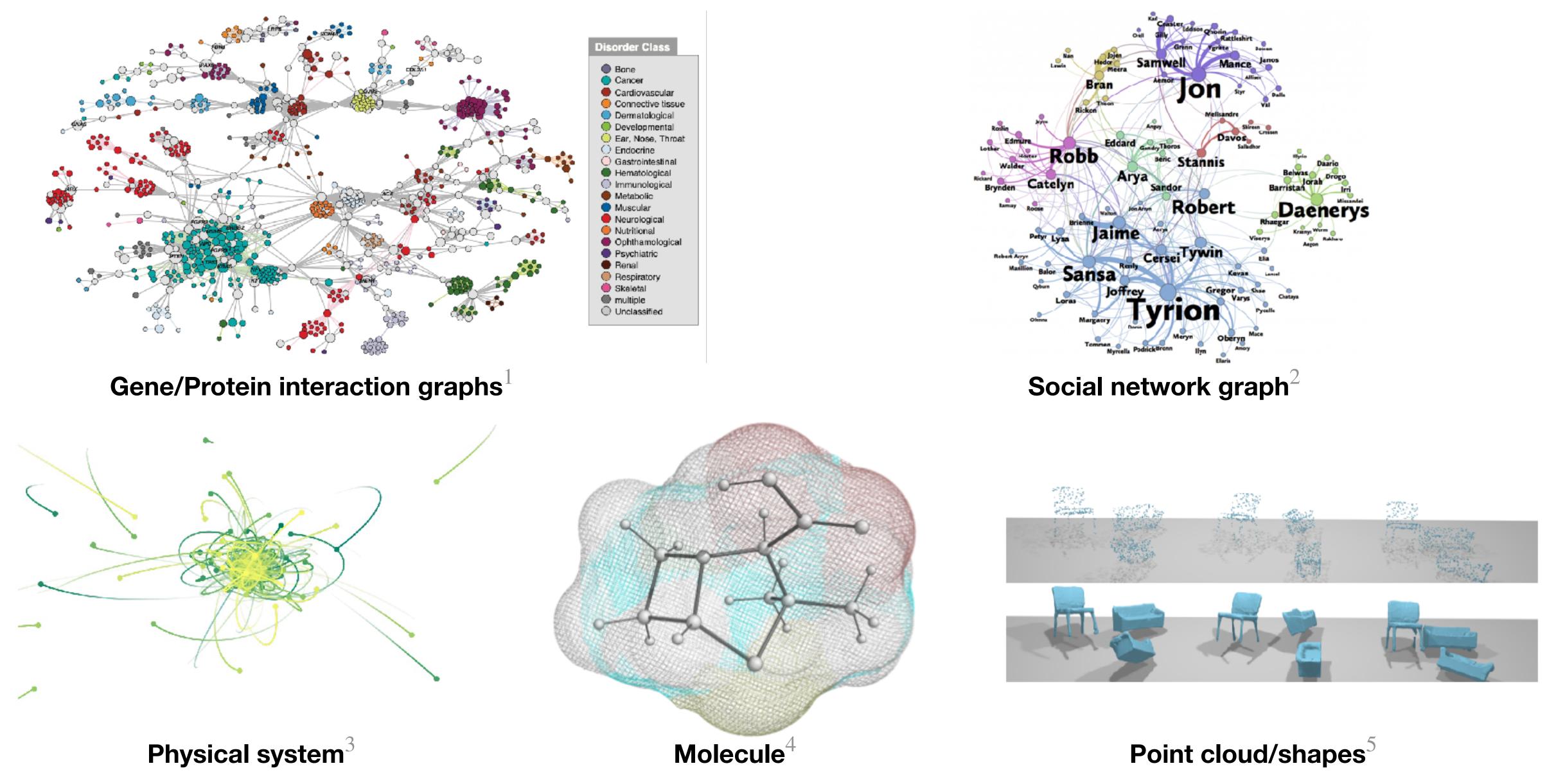
Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

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

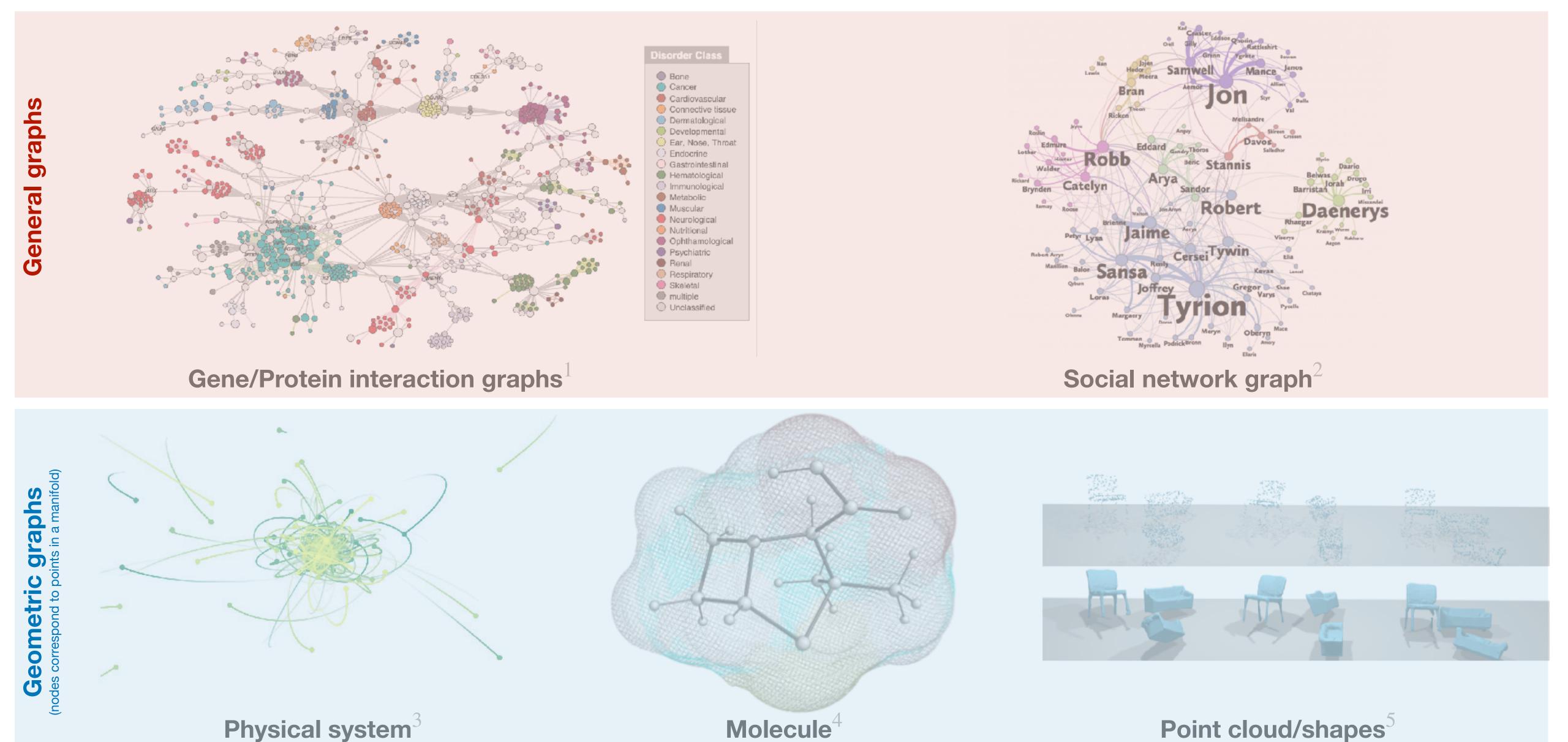


Figures from: ¹Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690. ²https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

³Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

⁴Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

⁵Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.



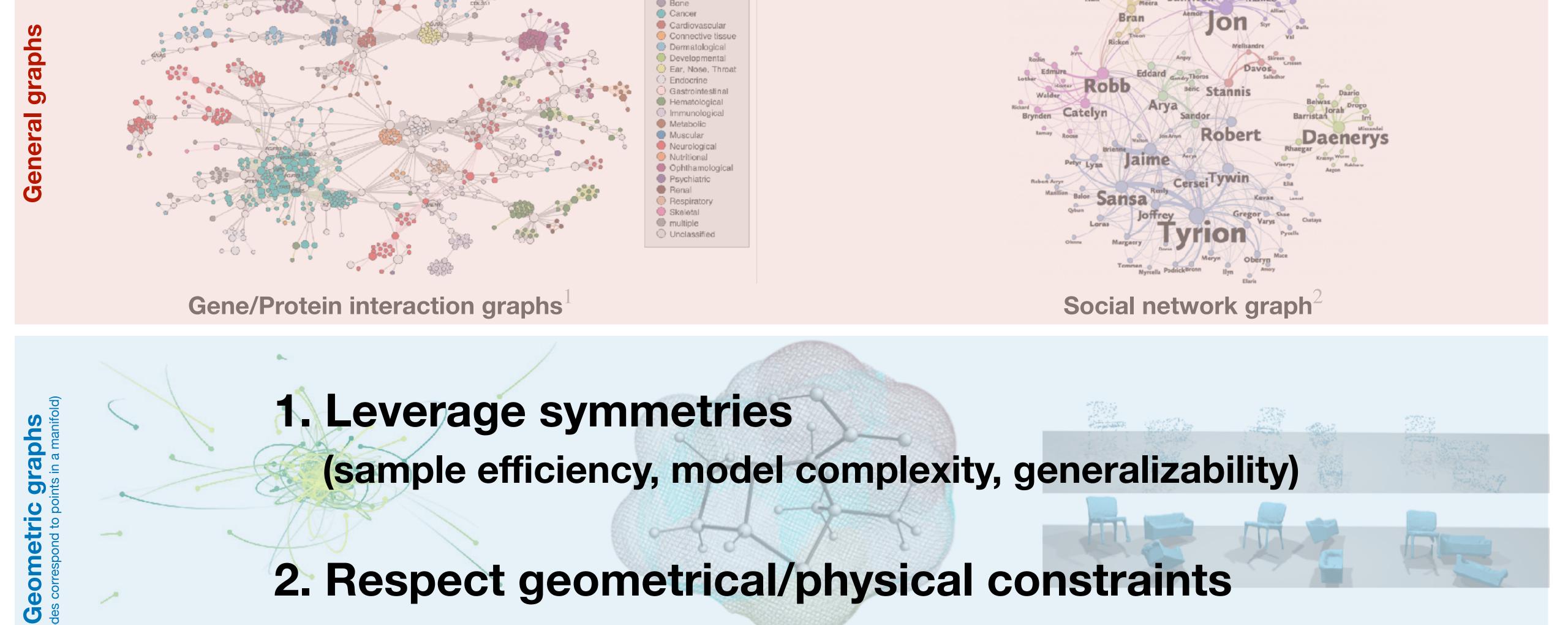
Figures from: ¹Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690.

²https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

³Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

⁴Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

⁵Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.



Figures from: ¹Goh, K. I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A. L. (2007). The human disease network. Proceedings of the National Academy of Sciences, 104(21), 8685-8690.

Physical system

Molecule

Point cloud/shapes

²https://predictivehacks.com/social-network-analysis-of-game-of-thrones/

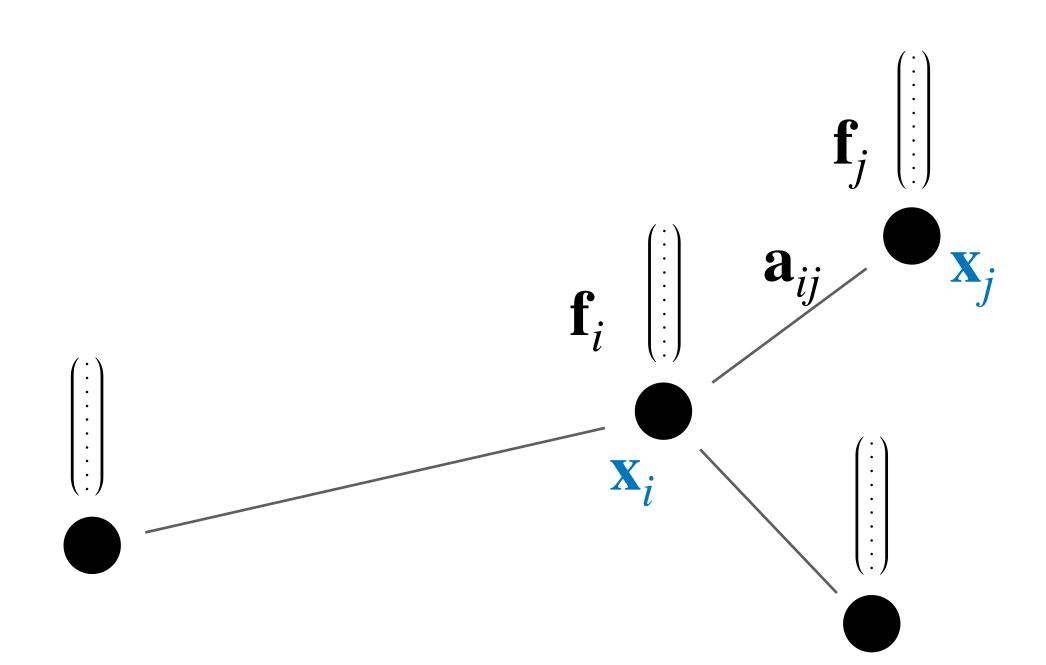
³Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E., & Welling, M. (2021). Geometric and Physical Quantities improve E (3) Equivariant Message Passing. In ICLR 2022

⁴Atz, K., Grisoni, F., & Schneider, G. (2021). Geometric deep learning on molecular representations. Nature Machine Intelligence, 1-10.

⁵Chatzipantazis, E., Pertigkiozoglou, S., Dobriban, E., & Daniilidis, K. (2022). SE (3)-Equivariant Attention Networks for Shape Reconstruction in Function Space. arXiv preprint arXiv:2204.02394.

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

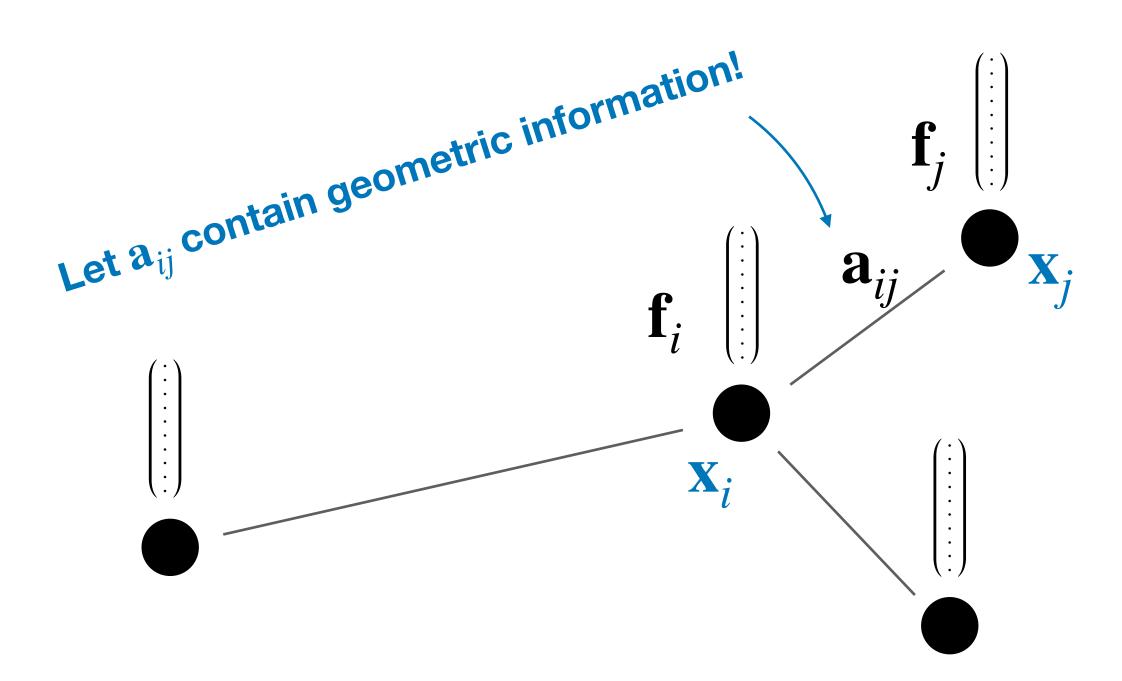
Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

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

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

Messages

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$
"Condition" messages on geometry

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

Graph
$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$

Graph
$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$ Only equivariant to translations...

Graph
$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$ Only equivariant to translations...

$$(X = \mathbb{R}^d) \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$ only equivariant to translations...

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$

$$\text{Full } E(3) \text{ equivariance, but a bifine the problem of the problem$$

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ii} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d) \ \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$

$$\text{Only equivariant to translations...}$$

$$(X = \mathbb{R}^d) \ \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|)$$

$$\text{Full } E(3) \text{ equivariance, but a bit restrictive...}$$

$$(X = G) \ \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1}g_i)$$

$$\text{Solution 1: Lift to the group!}$$

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{g}_j^{-1} \mathbf{g}_i)$$
Solution 1: Lift to the group!

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ii} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$



Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$(X = \mathbb{R}^d)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$ only equivariant to translations...

$$(X = \mathbb{R}^{d}) \quad \mathbf{m}_{ij} = \phi_{m}(\mathbf{f}_{i}, \mathbf{f}_{j}, ||\mathbf{x}_{j} - \mathbf{x}_{i}||)$$

$$\mathbf{Full} \quad E(3) \quad \text{equivariance, but a bit}$$

$$\mathbf{restrictive...}$$

$$\mathbf{m}_{ij} = \phi_{m}(\mathbf{f}_{i}, \mathbf{f}_{j}, g_{j}^{-1} g_{i})$$

$$\mathbf{solution 1: Lift to the group!}$$

$$(X = G)$$
 $\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{g}_j^{-1} \mathbf{g}_i)$
Solution 1: Lift to the group!

Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ii} \in \mathscr{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$

Scalar fields: attributes need to be invariants!

Steerable feature fields: attributes can be *covariants*! Goal: iteratively update node features to obtain useful hidden representations $\mathbf{h} \in \mathbb{R}^{C_h}$

Message passing layer:

$$\begin{cases} (\mathbf{X} = \mathbb{R}^d) & \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i) \\ & \text{Only equivariant to translations...} \end{cases} \\ (\mathbf{X} = \mathbb{R}^d) & \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) \\ (\mathbf{X} = \mathbf{G}) & \mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{g}_j^{-1}\mathbf{g}_i) \\ & \text{Solution 1: Lift to the group!} \\ \\ \{ (\mathbf{X} = \mathbb{R}^d) & \hat{\mathbf{m}}_{ij} = \hat{\phi}_m(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, Y(\mathbf{x}_j - \mathbf{x}_i)) \\ & \text{Solution 2: work with steerable} \\ & \text{Solution 2: work with steerable} \end{cases}$$

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ii} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this wavenum k, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongang Qi, Li Fuxin CORIS Institute, Oregon State University wuwen, qiz, lifforegonetate.edu

Abstrac

Unlike images which are represented in regular dense grids, 3D point clouds are irregular and anordered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density fune tions. With respect to a given point, the weight functions are learned with muitt-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions. which allowed us to dramatically scale up the network and significantly improve its performance. The learned convolution kernel can be used to compute translation-invariant. and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able. to achieve state of the art on challenging semantic segmen tation henchmarks on 3D paint clouds. Resides, our expertracets converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a similar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can dinearly obtain 3D data are increasingly obtaining. This includes indoor sensors such as laser scanners, time of flight sensors such as the Kineet, RealSense or Google Pango, structural light sensors such as those on the iPhoneX, as well as outdoor sensors such as LIDAR and MEMS sensors. The capability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation ambiguities from 2D imagery, and surface normals provide important cues of the scene geometry.

In 2D images, convolutional neural networks (CNNs) have fundamentally changed the landscape of computer vision by greatly improving results on almost every vision task. CNNs succeed by utilizing translation invariance, so that the same set of convolutional filters can be applied on all the locations in an image, reducing the number of paramcters and improving generalization. We would hope such successes to be transferred to the analysis of 3D data. However, 3D data often come in the form of point clouds, which is a set of unordered 3D points, with or without additional features (e.g. RGB) on each point. Point clouds are unordered and do not conform to the regular lattice grids as in 2D images. It is difficult to apply conventional CNNs on such unordered input. An alternative approach is to treat the 3D space as a volumetric grid, but in this case, the volume will be sparse and CNNs will be computationally intractable on high-resolution volumes.

In this paper, we propose a novel approach to perform convolution on 3D point clouds with non-uniform sampling. We note that the convolution operation can be viewed as a discrete approximation of a continuous convolution operator. In 3D space, we can treat the weights of this convolution operator to be a (Lipschitz) continuous function of the local 3D point coordinates with respect to a reference 3D point. The continuous function can be approximated by a multilayer perceptron(MLP), as done in [33] and [15]. But these algorithms did not take non-uniform sampling into account. We propose to use an inverse density scale to re-weight the continuous function learned by MLP, which corresponds to the Monte Carle approximation of the continuous convelution. We call such an operation PointConv. PointConv. involves taking the positions of point clouds as input and learning an MLP to approximate a weight function, as well as applying a inverse density scale on the learned weights to compensate the non-uniform sampling.

The naive implementation of PointCoov is memory inefficient when the channel size of the output features is very large and hence hard to train and scale up to large networks. In order to reduce the memory consumption of PointCoov. res to obtain \mathbb{R}^{C_h}

nessages on geometry

904.1

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$
$$= k(\mathbf{x}_i - \mathbf{x}_i)\mathbf{f}_i$$



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this water mark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongang Qi, Li Fuxin CORIS Institute, Oregon State University wuwen, qiz, lifforeyonetate.edu

Abstract

Unlike images which are represented in regular dense grids, 3D point clouds are irregular and anumbered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density fine tions. With respect to a given point, the weight functions are learned with multi-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions. which allowed us to dramatically scale up the network and significantly improve its performance. The learned comolution kernel can be used to compute translation-invariant and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able. to achieve state of the art on challenging semantic segmen tation henchmarks on 3D point clouds. Resides, our expertracets converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a similar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can dinearly obtain 3D data are increasingly obtaining. This includes indoor sensors such as laser scanners, time of flight sensors such as the Kineet, RealSense or Google Tango, structural light sensors such as those on the iPhoneX, as well as outdoor sensors such as LIDAR and MEMS sensors. The capability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation ambiguities from 2D imagery, and surface normals provide important cues of the scene geometry.

In 2D images, convolutional neural networks (CNNs) have fundamentally changed the landscape of computer vision by greatly improving results on almost every vision task. CNNs succeed by utilizing translation invariance, so that the same set of convolutional filters can be applied on all the locations in an image, reducing the number of paramcters and improving generalization. We would hope such successes to be transferred to the analysis of 3D data. However, 3D data often come in the form of point clouds, which is a set of unordered 3D points, with or without additional features (e.g., RGB) on each point. Point clouds are unordered and do not conform to the regular lattice grids as in 2D images. It is difficult to apply conventional CNNs on such unordered input. An alternative approach is to treat the 3D space as a volumetric grid, but in this case, the volume will be sparse and CNNs will be computationally intractable on high-resolution volumes.

In this paper, we propose a novel approach to perform convolution on 3D point clouds with non-uniform sampling. We note that the convolution operation can be viewed as a discrete approximation of a continuous convolution operator. In 3D space, we can treat the weights of this convolution operator to be a (Lipschitz) continuous function of the local 3D point coordinates with respect to a reference 3D point. The continuous function can be approximated by a multilayer perceptron(MLP), as done in [33] and [16]. But these algorithms did not take non-uniform sampling into account. We propose to use an inverse density scale to re-weight the continuous function learned by MLP, which corresponds to the Monte Carle approximation of the continuous convelution. We call such an operation PointConv. PointConv. involves taking the positions of point clouds as input and learning an MLP to approximate a weight function, as well as applying a inverse density scale on the learned weights to compensate the non-uniform sampling.

The naive implementation of PointConv is memory inefficient when the channel size of the output features is very large and hence hard to train and scale up to large networks. In order to reduce the memory consumption of PointConv. res to obtain \mathbb{R}^{C_h}

nessages on geometry

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$
$$= k(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{x}_j - \mathbf{x}_i) \mathbf{f}_j)$$



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this water mark, it is identical to the accepted version;

the final published version of the proceedings is available on IEEE Xplore.

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongang Qi, Li Fuxin CORIS Institute, Oregon State University wuwen, çiz, lifforeyenetate.edu

Abstract

Unlike images which are represented in regular dense grids, 3D point clouds one irregular and anordered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density fine tions. With respect to a given point, the weight functions are learned with multi-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions. which allowed us to dramatically scale up the network and significantly improve its performance. The learned comolution kernel can be used to compute translation-invariant. and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able. to achieve state of the art on challenging semantic segmen tation henchmarks on 3D point clouds. Resides, our expertracets converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a shullar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can directly obtain 3D data are increasingly objiquitous. This includes indoor sensors such as laser scanners, time of flight sensors such as the Kinect, RealSense or Google Tango, structural light sensors such as those on the iPhoneX, as well as outdoor sensors such as those on the iPhoneX, as well as outdoor sensors such as LIDAR and MEMS sensors. The supability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation ambiguities from 2D imagery, and surface normals provide important cues of the scene geometry.

In 2D images, convolutional neural networks (CNNs) have fundamentally changed the landscape of computer vision by greatly improving results on almost every vision task. CNNs succeed by utilizing translation invariance, so that the same set of convolutional filters can be applied on all the locations in an image, reducing the number of paramcters and improving generalization. We would hope such successes to be transferred to the analysis of 3D data. However, 3D data often come in the form of point clouds, which is a set of unordered 3D points, with or without additional features (e.g. RGB) on each point. Point clouds are unordered and do not conform to the regular lattice grids as in 2D images. It is difficult to apply conventional CNNs on such unordered input. An alternative approach is to treat the 3D space as a volumetric grid, but in this case, the volume. will be sparse and CNNs will be computationally intractable on high-resolution volumes.

In this paper, we propose a novel approach to perform convolution on 3D point clouds with non-uniform sampling. We note that the convolution operation can be viewed as a discrete approximation of a continuous convolution operator. In 3D space, we can treat the weights of this convolution operator to be a (Lipschitz) continuous function of the local 3D point coordinates with respect to a reference 3D point. The continuous function can be approximated by a multilayer perceptron(MLP), as done in [33] and [16]. But these algorithms did not take non-uniform sampling into account. We propose to use an inverse density scale to re-weight the continuous function learned by MLP, which corresponds to the Monte Carle approximation of the continuous convelution. We call such an operation PointConv. PointConv. involves taking the positions of point clouds as input and learning an MLP to approximate a weight function, as well as applying a inverse density scale on the learned weights to compensate the non-uniform sampling.

The naive implementation of PointCoov is memory inefficient when the channel size of the output features is very large and hence hard to train and scale up to large networks. In order to reduce the memory consumption of PointCoov. res to obtain \mathbb{R}^{C_h}

ressages on geometry

¹Schütt, K., Kindermans, P. J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., & Müller, K. R. (2017).

Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. NeurIPS ²Wu, W., Qi, Z., & Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. CVPR

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$
$$= k(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_{i} = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\mathbf{x}_{j} - \mathbf{x}_{i})\mathbf{f}_{j})$$

$$(k \star f)(\mathbf{x}_{i})$$



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this wave mark, it is identical to the accepted version;

the final published version of the proceedings is available on IEEE Xplore.

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongang Qi, Li Fuxin CORIS Institute, Oregon State University wuwen, çiz, lifforeyenetate.edu

Abstract

Unlike images which are represented in regular dense grids, 3D point clouds one irregular and anordered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density functions. With respect to a given point, the weight functions are learned with multi-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions. which allowed us to dramatically scale up the network and significantly improve its performance. The learned comolution kernel can be used to compute translation-invariant and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able. to achieve state of the art on challenging semantic segmen tation henchmarks on 3D point clouds. Resides, our expertracets converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a similar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can directly obtain 3D data are increasingly obtainious. This includes indoor sensors such as laser scanners, time of flight sensors such as the Kineet, ResiSense or Google Tango, structural light sensors such as those on the iPhoneX, as well as outdoor sensors such as LIDAR and MEMS sensors. The capability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation ambiguities from 2D imagery, and surface normals provide important cuts of the scene geometry.

In 2D images, convolutional neural networks (CNNs) have fundamentally changed the landscape of computer vision by greatly improving results on almost every vision task. CNNs succeed by utilizing translation invariance, so that the same set of convolutional filters can be applied on all the locations in an image, reducing the number of paramcters and improving generalization. We would hope such successes to be transferred to the analysis of 3D data. However, 3D data often come in the form of point clouds, which is a set of unordered 3D points, with or without additional features (e.g. RGB) on each point. Point clouds are unordered and do not conform to the regular lattice grids as in 2D images. It is difficult to apply conventional CNNs on such unordered input. An alternative approach is to treat the 3D space as a volumetric grid, but in this case, the volume. will be sparse and CNNs will be computationally intractable on high-resolution volumes.

In this paper, we propose a novel approach to perform convolution on 3D point clouds with non-uniform sampling. We note that the convolution operation can be viewed as a discrete approximation of a continuous convolution operator. In 3D space, we can treat the weights of this convolution operator to be a (Lipschitz) continuous function of the local 3D point coordinates with respect to a reference 3D point. The continuous function can be approximated by a multilayer perceptron(MLP), as done in [33] and [16]. But these algorithms did not take non-uniform sampling into account. We propose to use an inverse density scale to re-weight the continuous function learned by MLP, which corresponds to the Monte Carle approximation of the continuous convelution. We call such an operation PointConv. PointConv. involves taking the positions of point clouds as input and learning an MLP to approximate a weight function, as well as applying a inverse density scale on the learned weights to compensate the non-uniform sampling.

The naive implementation of PointCoov is memory inefficient when the channel size of the output features is very large and hence hard to train and scale up to large networks. In order to reduce the memory consumption of PointCoov. res to obtain \mathbb{R}^{C_h}

ressages on geometry

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$
$$= k(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{x}_j - \mathbf{x}_i) \mathbf{f}_j)$$

• Only E(d) equivariant when (Lecture 1.7)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j)$$

GyF

This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Facept for this water mark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongang Qi, Li Fuxin CORIS Institute, Oregon State University wowen, qiz, lifforeyenetate.edu

Abstract

Unlike images which are represented in regular dense grids, 3D point clouds one irregular and anordered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density fine tions. With respect to a given point, the weight functions are learned with multi-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions. which allowed us to dramatically scale up the network and significantly improve its performance. The learned comolution kernel can be used to compute translation-invariant. and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able. to achieve state of the art on challenging semantic segmen tation henchmarks on 3D point clouds. Resides, our expertracets converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a shullar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can dinearly obtain 3D data are increasingly obtainism. This includes indoor sensors such as baser scanners, time of flight sensors such as the Kinect, RealSense or Google Tango, structural light sensors such as those on the iPhoneX, as well as outdoor sensors such as LIDAR and MEMS sensors. The capability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation ambiguities from 2D imagery, and surface normals provide important cues of the scene geometry.

In 2D images, convolutional neural networks (CNNs) have fundamentally changed the landscape of computer vision by greatly improving results on almost every vision task. CNNs succeed by utilizing translation invariance, so that the same set of convolutional filters can be applied on all the locations in an image, reducing the number of paramcters and improving generalization. We would hope such successes to be transferred to the analysis of 3D data. However, 3D data often come in the form of point clouds, which is a set of unordered 3D points, with or without additional features (e.g. RGB) on each point. Point clouds are unordered and do not conform to the regular lattice grids as in 2D images. It is difficult to apply conventional CNNs on such unordered input. An alternative approach is to treat the 3D space as a volumetric grid, but in this case, the volume. will be sparse and CNNs will be computationally intractable on high-resolution volumes.

In this paper, we propose a novel approach to perform convolution on 3D point clouds with non-uniform sampling. We note that the convolution operation can be viewed as a discrete approximation of a continuous convolution operator. In 3D space, we can treat the weights of this convolution operator to be a (Lipschitz) continuous function of the local 3D point coordinates with respect to a reference 3D point. The continuous function can be approximated by a multilayer perceptron(MLP), as done in [33] and [16]. But these algorithms did not take non-uniform sampling into account. We propose to use an inverse density scale to re-weight the continuous function learned by MLP, which corresponds to the Monte Carle approximation of the continuous convelution. We call such an operation PointConv. PointConv. involves taking the positions of point clouds as input and learning an MLP to approximate a weight function, as well as applying a inverse density scale on the learned weights to compensate the non-uniform sampling.

The naive implementation of PointConv is memory inefficient when the channel size of the output features is very large and hence hard to train and scale up to large networks. In order to reduce the memory consumption of PointConv. res to obtain \mathbb{R}^{C_h}

essages on geometry

¹Schütt, K., Kindermans, P. J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., & Müller, K. R. (2017). Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. NeurIPS ²Wu, W., Qi, Z., & Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. CVPR

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Point convolutions 1,2 , $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{x}_j - \mathbf{x}_i)$$
$$= k(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{x}_j - \mathbf{x}_i) \mathbf{f}_j)$$

• Only E(d) equivariant when (Lecture 1.7)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j)$$



This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this wave mark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

res to obtain \mathbb{R}^{C_h}

PointCony: Deep Convolutional Networks on 3D Point Clouds

Wenxuan Wu, Zhongong Qi, Li Fuxin CORIS Institute, Oregon State University

wuwen, qiz, lifforegonetate.edu

Abstract

Unlike images which are represented in regular dense grids, 3D point clouds one irregular and anordered, hence applying convolution on them can be difficult. In this paper, we extend the dynamic filter to a new convolution operation, named PointCom: PointCom: can be applied on point ciouds to build deep convolutional networks. We treat convolution kernels as nonlinear functions of the local coordinates of 3D points comprised of weight and density functions. With respect to a given point, the weight functions: are learned with multi-layer perceptron networks and density functions through kernel density estimation. The most important contribution of this work is a novel reformulation proposed for efficiently computing the weight functions, which allowed us to dramatically scale up the network and significantly improve its performance. The learned comolution kernel can be used to compute translation-invariant and permutation-invariant convolution on any point set in the 3D space. Besides, PointConv can also be used as deconvolution operators to propagate features from a subsampled point cloud back to its original resolution. Experiments on ModelNet#0, ShapeNet, and ScanNet show that deep convolutional neural networks built on PointConv are able to achieve state of the art on challenging semantic segmen tation henchmarks on 3D point clouds. Resides, our expertraints converting CIFAR-10 into a point cloud showed that networks built on PointConv can match the performance of convolutional networks in 2D images of a shullar structure.

1. Introduction

In recent robotics, autonomous driving and virtual/augmented reality applications, sensors that can dinearly obtain 3D data are increasingly obliquitous. This includes indoor sensors such as laser scanners, time of flight sensors such as the Kinect, RealSense or Google Tango, structural light sensors such as those on the iPhoneX, as well as purdoor sensors such as LIDAR and MEMS sensors. The capability to directly measure 3D data is invaluable in those applications as depth information could remove a lot

of the segmentation of face normals provide:

face normals provide if In 2D images, co have fundamentally ch sion by greatly impr task. CNNs succeed b that the same set of co all the locations in an ir ctors and improving g successes to be transfe ever, 3D data often con is a set of unordered 3 features (e.g., RGB) o ordered and do not co in 2D images. It is diff such unordered input. 3D space as a volume will be sparse and CNN on high-resolution volu In this paper, we p convolution on 3D poir We note that the come discrete approximation tor. In 3D space, we can operator to be a (Lipso

> involves taking the potearning an MLP to apas applying a inverse a to compensate the non-The naive implement federat when the chann large and hence hard to in order to reduce the

3D point coordinates v

The continuous function

layer perceptron (MLP)

algorithms did not take

We propose to use an i

continuous function le

the Monte Carlo appro-

lution. We call such a

SchNet – a deep learning architecture for molecules and materials

- K.T. Schütt, ^{1, 2)} H.E. Sauceda, ² P. J. Kindermans, ¹ A. Tkatchenko, ^{3, 2)} and K. R. Müller^{1,4,6, 2)}

 ¹⁾ Machine Learnine Group, Technische Universität Berlin, 1958 Ferlin, Cermany
- ²¹ Frita-Hober-Pastitut der Mac-Flanck-Gesellschaft, 14:26 Berlin, Germany
 ¹¹ Physics and Material's Science Research, Unit. University of Legendram. 1-151. Europela.
- ³⁾ Physics and Materials Science Research Unit, University of Luxumbourg, L-1511 Luxumbourg, Luxumbourg
- Mac Fielick Institut für Informatik, Saarbrücken. Germany
 Department of Brom and Cognitive Engineering, Korea University. Anam-dong, Secagbuk-gu, Swaii 136-713,

(Dated: 28 March 2018

Deep learning has led to a paradigm shift in artificial intelligence, including web, text and image search, speech recognition, as well as bioinformatics, with growing impact in chemical physics. Machine learning in general and deep learning in particular is ideally suited for representing quantum-mechanical interactions, enabling to model nonlinear potential-energy surfaces or enhancing the exploration of chemical compound space. Here we present the deep learning architecture SchNet that is specifically designed to model atomistic systems by making use of continuous-lifter convolutional layers. We demonstrate the capabilities of bichNet by accurately predicting a range of properties across chemical space for molecules and materials where our model learns chemically plausible embeddings of atom types across the periodic table. Finally, we employ SchNet to predict potential energy surfaces and energy conserving force fields for molecular dynamics simulations of small molecules and perform an exemplary study of the quantum-mechanical properties of C₂₀-fullerene that would have been infeasible with regular ob halt's molecular dynamics.

I. INTRODUCTION

Accelerating the discovery of molecules and materials with desired properties is a long standing challenge. in computational chemistry and the materials sciences. However, the computational cost of accurate quantumchemical calculations proves prohibitive in the explaration of the vast chemical space. In recent years, there have been increased efforts to overcome this bottleneck using machine learning, where only a reduced set of reference calculations is required to accurately predict chemical properties 1913 or potential-energy surfaces 19125 While these approaches make use of painstakingly handcrafted descriptors, deep learning has been applied to predict properties from molecular structures using graph neural networks [2012] However, these are restricted to predictions for equilibrium structures due to the lack of atomic positions in the input. Only recently, approaches that learn a representation directly from atom types and positions have been developed 22100 While neural networks are often considered a 'black box', there has recently been an increased effort to explain their predictions in order to understand how they operate or even extract scientific insight. This can either be done by analyzing a trained model and or by directly designing interpretable models 15 For quantum chemistry, some of us have proposed such an interpretable architecture with Deep Tensor Neural Networks (DTNN) that not only learns a representation of atomic environments but

²³k, etof, sincett@ta-berlin.ce ^b alexandre tlastehenko@uni.lu ^Ok aus-robert.mueller@tu-berlin.ce allows for spatially and chemically resolved insights into quantum-machanical observables²⁸.

Here we build upon this work and present the deep learning architecture SchNet that allows to model complex atomic interactions in order to predict potential energy surfaces or speeding up the exploration of themical space. SchNet, being a variant of DTNNs, is able to learn representations for molecules and materials that follow fundamental symmetries of atomistic systems by construction, e.g., rotational and translational invariance as well as invariance to atom indexing. This enables accurate predictions throughout compositional and configurational chemical space where symmetries of the potential energy surface are captured by design. Interactions between atoms are modeled using continuous-filter convolutional layers³⁰⁰ being able to incorporate further chamical knowledge and constraints using specifically designed filter-generating neural networks. We demonstrate that those allow to efficiently incorporate periodic boundary conditions enabling accurate predictions of formation en ergies for a diverse set of bulk crystals. Beyond that both SchNet and DTNNs provide local chemical potentials to analyze the obtained representation and allow for chemical insights 24. An analysis of the obtained representation shows that SchNet learns chemically plausible embeddings of atom types that capture the structure of the periodic table. Finally, we present a path-integral molecular dynamics (PIMD) simulation using an energy conserving force field learned by SchNet trained on reference data from a classical MD at the PBE+vdW¹ level of theory effectively accelerating the simulation by three orders of magnitude. Specifically, we employ the recently developed perturned path-integral approach 41 for carrying out imaginary time PIMD, which allows quick emvergence of quantum-mechanical properties with re-

¹Schütt, K., Kindermans, P. J., Sauceda Felix, H. E., Chmiela, S., Tkatchenko, A., & Müller, K. R. (2017). Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. NeurIPS ²Wu, W., Qi, Z., & Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. CVPR

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ii} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$

Special case (EGNN¹, $X = \mathbb{R}^d$):

Messages (non-linear transformations)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, ||\mathbf{x}_j - \mathbf{x}_i||)$$

Aggregate + node updates

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

$$\mathbf{x}_{i}' = \mathbf{x}_{i} + C \sum_{j \neq i} (\mathbf{x}_{j} - \mathbf{x}_{i}) \phi_{x}(\mathbf{m}_{ij})$$

E(n) Equivariant Graph Neural Networks

Victor Garcia Satorras 1 Emiel Hoogeboom 1 Max Welling 1

Abstract

This paper introduces a new model to learn graph neural networks equivariant to rotations, translations, reflections and permutations called E(n)-Equivariant Graph Neural Networks (EGNNs). In contrast with existing methods, our work does not require computationally expensive higher-order representations in intermediate layers while it still achieves competitive or better performance. In addition, whereas existing methods are limited to equivariance on 3 dimensional spaces, our model is easily scaled to higher-dimensional spaces. We demonstrate the effectiveness of our method on dynamical systems modelling, representation learning in graph autoencoders and predicting molecular properties.

1. Introduction

Although deep learning has largely replaced hand-crafted features, many advances are critically dependent on inductive biases in deep neural networks. An effective method to restrict neural networks to relevant functions is to exploit the symmetry of problems by enforcing equivariance with respect to transformations from a certain symmetry group. Notable examples are translation equivariance in Convolutional Neural Networks and permutation equivariance in Graph Neural Networks (Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2016a).

Many problems exhibit 3D translation and rotation symmetries. Some examples are point clouds (Uy et al., 2019), 3D molecular structures (Ramakrishnan et al., 2014) or N-body particle simulations (Kipf et al., 2018). The group corre sponding to these symmetries is named the Euclidean group: SE(3) or when reflections are included E(3). It is often desired that predictions on these tasks are either equivariant or invariant with respect to E(3) transformations.

UvA-Bosch Delta Lab, University of Amsterdam, Netherlands. Correspondence to: Victor Garcia Satorras <v.garciasatorras@uva.nl> Emiel Hoogeboom <e.hoogsboom@uva.rl>, Max Welling <m.welling@uva.nl>.

Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

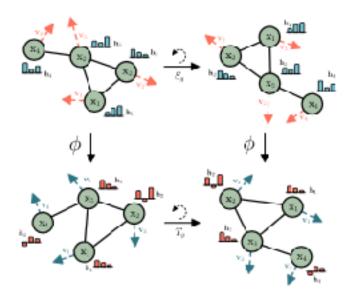
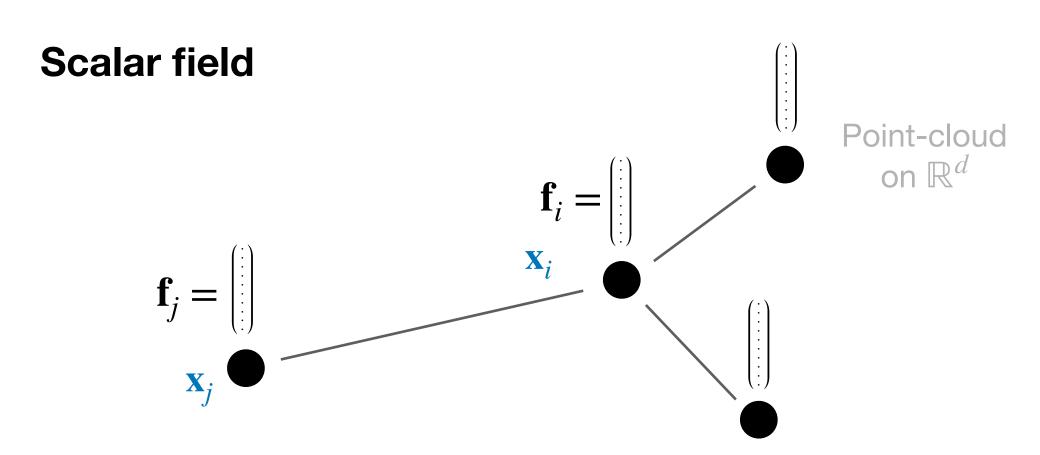


Figure 1. Example of rotation equivariance on a graph with a graph neural network φ

Recently, various forms and methods to achieve E(3) or SE(3) equivariance have been proposed (Thomas et al., 2018; Fuchs et al., 2020; Finzi et al., 2020; Köhler et al., 2020). Many of these works achieve innovations in studying types of higher-order representations for intermediate network layers. However, the transformations for these higher-order representations require coefficients or approximations that can be expensive to compute. Additionally, in practice for many types of data the inputs and outputs are restricted to scalar values (for instance temperature or energy, referred to as type-0 in literature) and 3d vectors (for instance velocity or momentum, referred to as type-1 in

In this work we present a new architecture that is translation, rotation and reflection equivariant (E(n)), and permutation equivariant with respect to an input set of points. Our model is simpler than previous methods in that it does not require the spherical harmonics as in (Thomas et al., 2018; Fuchs et al., 2020) while it can still achieve competitive or better results. In addition, equivariance in our model is not Proceedings of the 38th International Conference on Machine larger dimensional spaces without a significant increase in

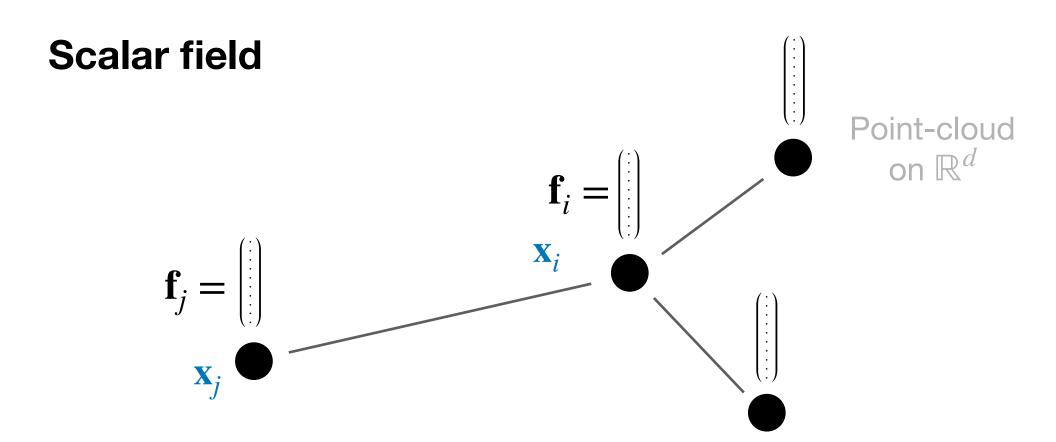




Point convolutions

Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$



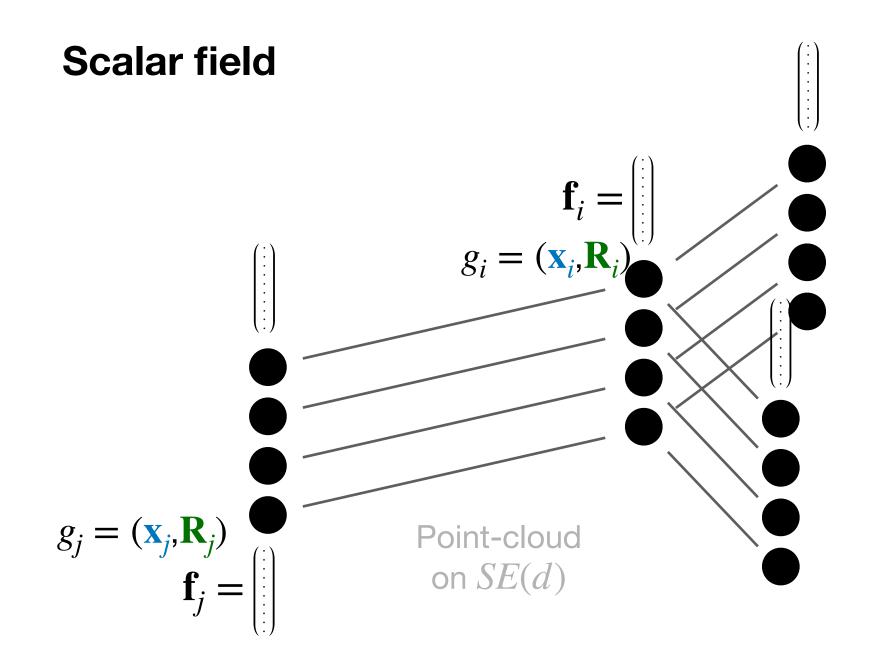
Point convolutions

Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

Regular group convolution (Lecture 1)

$$\mathbf{f}'_{i}(\mathbf{R}) = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}^{-1}(\mathbf{x}_{j} - \mathbf{x}_{i}))\mathbf{f}_{j})$$



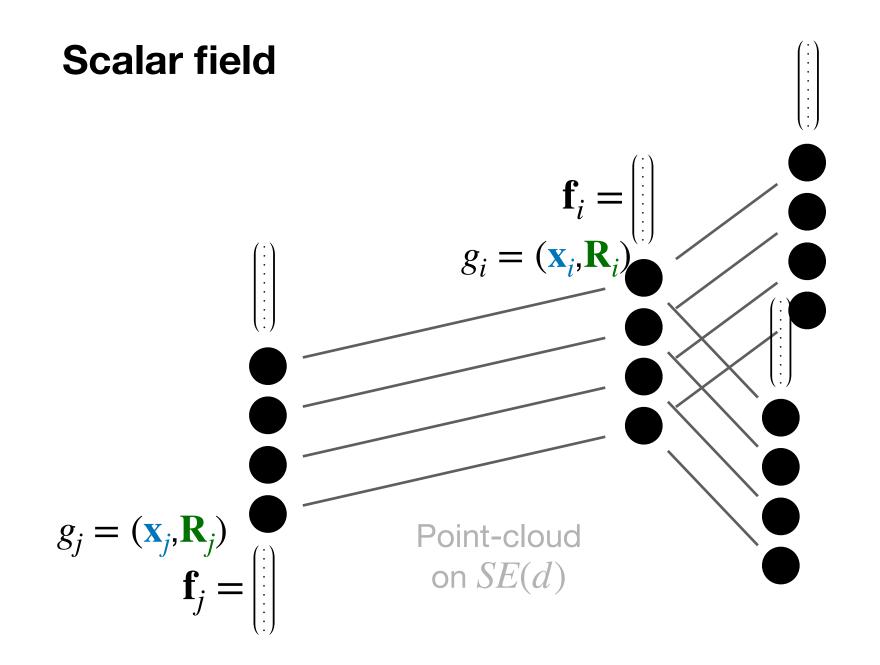
Point convolutions

Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

Regular group convolution (Lecture 1)

$$\mathbf{f}'_{i}(\mathbf{R}) = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}^{-1}(\mathbf{x}_{j} - \mathbf{x}_{i}))\mathbf{f}_{j})$$



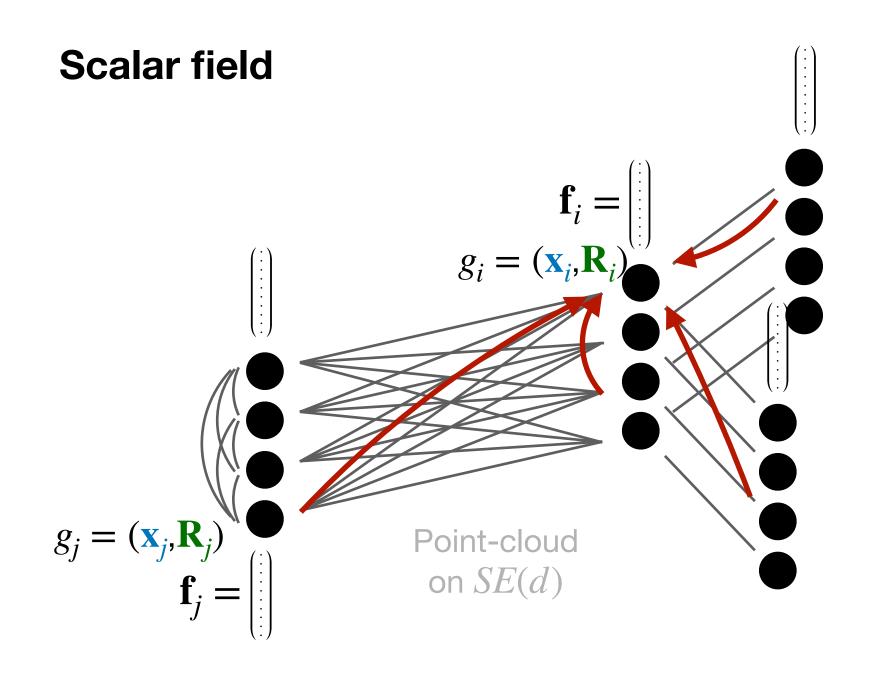
Point convolutions

Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

Regular group convolution (Lecture 1)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}_i^{-1}(\mathbf{x}_j - \mathbf{x}_i), \mathbf{R}_i^{-1}\mathbf{R}_j)\mathbf{f}_j)$$



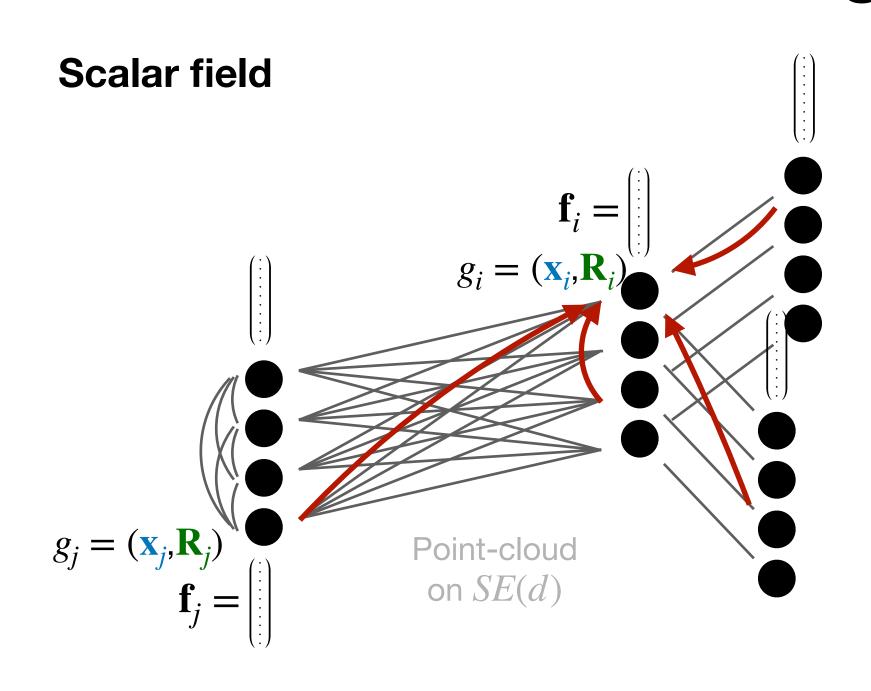
Point convolutions

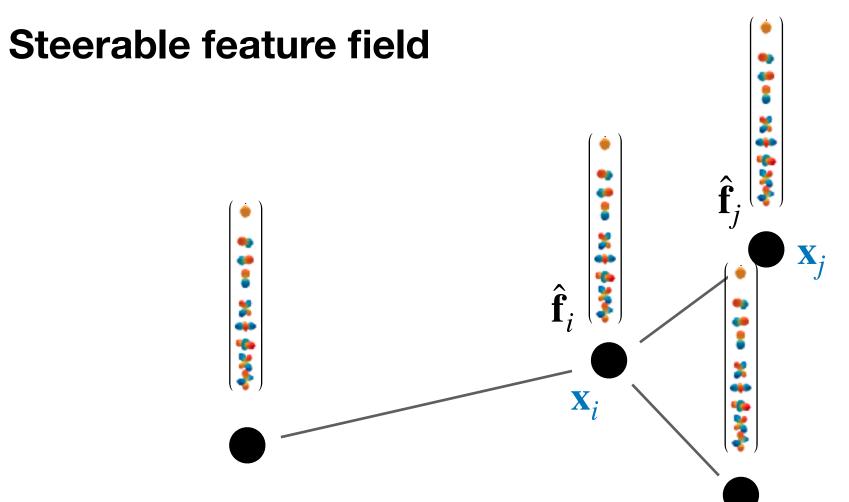
Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

Regular group convolution (Lecture 1)

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}_i^{-1}(\mathbf{x}_j - \mathbf{x}_i), \mathbf{R}_i^{-1}\mathbf{R}_j)\mathbf{f}_j)$$





Point convolutions

Translation equivariant, but not rotation equivariant

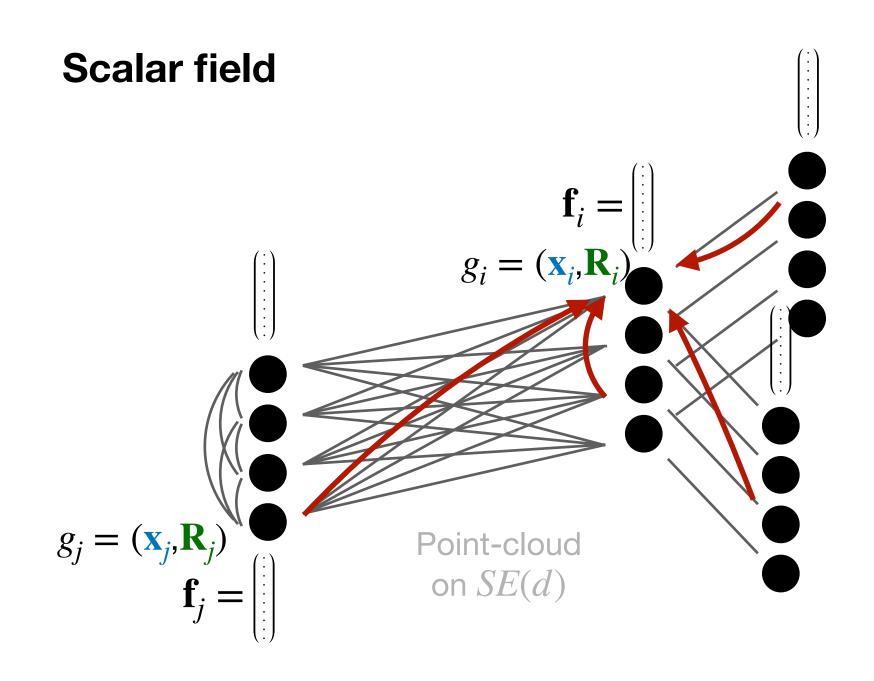
$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

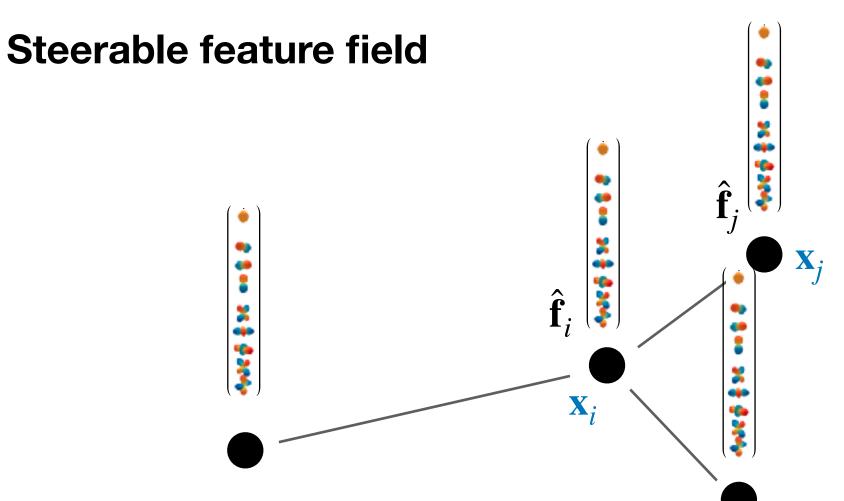
Regular group convolution (Lecture 1)

$$\mathbf{f}_i'(\mathbf{R}) = \phi_f(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}^{-1}(\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$
 Fourier transformation
$$\mathbf{\hat{f}}_i'$$

Fourier

transformation





Point convolutions

Translation equivariant, but not rotation equivariant

$$\mathbf{f}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} k((\mathbf{x}_j - \mathbf{x}_i))\mathbf{f}_j)$$

Regular group convolution (Lecture 1)

Rotation equivariant, but requires grid on SO(d)

$$\mathbf{f}_{i}'(\mathbf{R}) = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\mathbf{R}^{-1}(\mathbf{x}_{j} - \mathbf{x}_{i}))\mathbf{f}_{j})$$

Steerable group convolution (Lecture 2)

$$\hat{\mathbf{f}}'_i = \phi_f(\sum_{j \in \mathcal{N}(i)} \hat{k}(\mathbf{x}_j - \mathbf{x}_i) \tilde{\mathbf{f}}_j)$$

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in \mathcal{I}$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Lie group convolutions 1,2 , X = G):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1} g_i)$$
$$= k(\text{Log}(g_i^{-1} g_j))\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_{i} = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\operatorname{Log}(g_{i}^{-1}g_{j}))\mathbf{f}_{j})$$

$$(k \star_{G} f)(g_{i})$$

Published as a conference paper at ICLR 2020

B-SPLINE CNNS ON LIE GROUPS

Erik J. Bekkers

Centre for Analysis and Scientific Computing
Department of Applied Mathematics and Computer Science
Findhoven University of Technology, Eindhoven, the Netherlands
e. j.bekkers@tne.nl

ABSTRACT

Group convolutional neural networks (G-CNNs) can be used to improve classical CNNs by equipping them with the geometric structure of groups. Central in the success of G-CNNs is the lifting of feature maps to higher dimensional disentangled representations in which data characteristics are effectively learned. geometric data-augmentations are made obsolete, and predictable behavior under geometric transformations (equivariance) is guaranteed via group theory. Currently, however, the practical implementations of G-CNNs are limited to either discrete groups (that leave the grid intact) or continuous compact groups such as rotations (that enable the use of Fourier theory). In this paper we lift these limitations and propose a modular framework for the design and implementation of G-CNNs for arbitrary Lie groups. In our approach the differential structure of Lie groups is used to expand convolution kernels in a generic basis of B splines that is defined on the Lie algebra. This leads to a flexible framework that enables localized, arrows, and deformable convolutions in G-CNNs by means of respectively localized, sparse and non uniform B spline expansions. The impact and potential of our approach is studied on two benchmark datasets: cancer detection in histopathology slides in which rotation equivariance plays a key role and facial. landmark localization in which scale equivariance is important. In both cases, G-CNN architectures outperform their classical 2D counterparts and the added value of atteus and localized group convolutions is studied in detail.

1 Introduction

arXiv:1909.12057v4

Group convolutional neural networks (G-CNNs) are a class of neural networks that are equipped with the geometry of groups. This enables them to profit from the structure and symmetries in signal data such as images (Cohen & Welling, [2016). A key feature of G-CNNs is that they are equivariant with respect to transformations described by the group, i.e., they guarantee predictable behavior under such transformations and are insensitive to both local and global transformations on the input data. Classical CNNs are a special case of G-CNNs that are equivariant to translations and, in contrast to unconstrained NNs, they make advantage of (and preserve) the basic structure of signal data throughout the network (LeCun et al.) [1990). By considering larger groups (i.e. considering not just translation equivariance) additional geometric structure can be utilized in order to improve performance and data efficiency (see G-CNN literature in Sec. [2]).

Part of the success of G-CNNs can be attributed to the lifting of feature maps to higher dimensional objects that are generated by matching kernels under a range of poses (transformations in the group). This leads to a discotanglement with respect to the pose and together with the group structure this enables a flexible way of learning high level representations in terms of low-level activated neurons observed in specific configurations, which we conceptually illustrate in Fig. [I] From a neuro-psychological viewpoint, this resembles a hierarchical composition from low- to high-level features akin to the recognition-by-components model by [Biedenman] (1987), a viewpoint which is also adopted in work on capsule networks [Hinton et al., [2011; Sapour et al., [2017). In particular in (Lenssen et al., [2018) the group theoretical connection is made explicit with equivariant capsules that provide a sparse index/value representation of feature maps on groups (Gens & Domingos. [2014).

ares to obtain \mathbb{R}^{C_h}

nessages on geometry

¹Bekkers, E. J. (2019, September). B-Spline CNNs on Lie groups. ICLR ²Finzi, M., Stanton, S., Izmailov, P., & Wilson, A. G. (2020, November). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. ICML

2021

arXiv:1909.12057v4

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in \mathbb{R}^{C_v}$
- edges $e_{ii} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ii} \in \mathbb{R}^{C_e}$

Special case (Lie group convolutions 1,2 , X = G):

Messages (linear transformations based on kernel)

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, g_j^{-1} g_i)$$
$$= k(\text{Log}(g_i^{-1} g_j))\mathbf{f}_j$$

Aggregate + node updates (convolution + activation fn)

$$\mathbf{f}'_{i} = \phi_{f}(\sum_{j \in \mathcal{N}(i)} k(\operatorname{Log}(g_{i}^{-1}g_{j}))\mathbf{f}_{j})$$

$$(k \star_{G} f)(g_{i})$$

Published as a conference paper at ICLR 2020.

B-SPLINE CNNS ON LIE GROUPS

Erik J. Bekkers

Centre for Analysis and Scientific Computing Department of Applied Mathematics and Computer Sc Findhoven University of Technology, Eindhoven, the b e.j.bekkers9tue.nl

ABSTRAC

Group convolutional neural networks (G-CNN cal CNNs by equipping them with the geom in the success of G-CNNs is the lifting of fe disentangled representations in which data char geometric data-augmentations are made obsole geometric transformations (equivariance) is a rently, however, the practical implementation discrete groups (that leave the grid intact) or o rotations (that enable the use of Fourier theory itations and propose a modular framework for G-CNNs for arbitrary Lie groups. In our app Lie groups is used to expand convolution kern that is defined on the Lie algebra. This leads to localized, atrous, and deformable convolutions tively localized, sparse and non uniform B sp potential of our approach is studied on two ber in histopathology slides in which rotation equiv landmark localization in which scale equivarian CNN architectures outperform their classical 21 of atteus and localized group convolutions is si

1 Introduction

Group convolutional neural networks (G-CNNs) are a with the geometry of groups. This enables them to p signal data such as images (Cohen & Welling, 2016). equivariant with respect to transformations described b behavior under such transformations and are insensitive the input data. Classical CNNs are a special case of G-C in contrast to unconstrained NNs, they make advantage of data throughout the network (LeCun et al., 1990). By not just translation equivariance) additional geometric s performance and data efficiency (see G-CNN literature)

Part of the success of G-CNNs can be attributed to t sional objects that are generated by matching kernels t the group). This leads to a disentanglement with resp. structure this enables a flexible way of learning high lev vated neurons observed in specific configurations, which a neuro-psychological viewpoint, this resembles a hiera features akin to the recognition-by-components model I also adopted in work on capsule networks (Hinton et al (Lenssen et al., 2018) the group theoretical connection is provide a sparse index/value representation of feature m

Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data

Abstract

The translation equivariance of convolutional layers enables convolutional neural networks to generalize well on image problems. While translation equivariance provides a powerful inductive bias for images, we often additionally desire equivariance to other transformations, such as rotations, especially for non-image data. We propose a general method to construct a convolutional layer that is equivariant to transformations from any specified Lie group with a surjective exponential map. Incorporating equivariance to a new group requires implementing only the group exponential and logarithm maps, enabling rapid prototyping. Showcasing the simplicity and generality of our method, we apply the same model architecture to images, ball-and-stick molecular data, and Hamiltonian dynamical systems. For Hamiltonian systems, the equivariance of our models is especially impactful, leading to exact conservation of linear and angular momentum.

1. Introduction

Symmetry pervades the natural world. The same law of gravitation governs a game of catch, the orbits of our planets, and the formation of galaxies. It is precisely because of the order of the universe that we can hope to understand it. Once we started to understand the symmetries inherent in physical laws, we could predict behavior in galaxies billions of light-years away by studying our own local region of time and space. For statistical models to achieve their full potential, it is essential to incorporate our knowledge of naturally occurring symmetries into the design of algorithms and architectures. An example of this principle is the translation equivariance of convolutional layers in neural networks (LeCun et al., 1995); when an input (e.g. an

¹New York University. Correspondence to: Marc Pinzi <maf820@nyu.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the au-

Samuel Stanton 1 Pavel Izmailov 1 Andrew Gordon Wilson 1

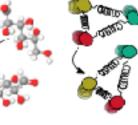




Figure I. Many modalities of spatial data de not lie on a grid, but still possess important symmetries. We propose a single model to learn from continuous spatial data that can be specialized to respect a given continuous symmetry group.

image) is translated, the output of a convolutional layer is

Group theory provides a mechanism to reason about symmetry and equivariance. Convolutional layers are equivariant to translations, and are a special case of group convolution. A group convolution is a general linear transformation equivariant to a given group, used in group equivariant convolutional networks (Cohen and Welling, 2016a).

In this paper, we develop a general framework for equiv ariant models on arbitrary continuous (spatial) data represented as coordinates and values $\{(x_i,f_i)\}_{i=1}^N$. Spatial data is a broad category, including ball-and-stick representations of molecules, the coordinates of a dynamical system, and images (shown in Figure 1). When the inputs or group elements lie on a grid (e.g., image data) one can simply enumerate the values of the convolutional kernel at each group element. But in order to extend to continuous data, we define the convolutional kernel as a continuous function on the group parameterized by a neural network.

We consider the large class of continuous groups known as Lie groups. In most cases, Lie groups can be parameterized in terms of a vector space of infinitesimal generators (the Lie algebra) via the logarithm and exponential maps. Many use ful transformations are Lie groups, including translations, rotations, and scalings. We propose LieConv, a convolutional layer that can be made equivariant to a given Lie. group by defining exp and log maps. We demonstrate the

¹Bekkers, E. J. (2019, September). B-Spline CNNs on Lie groups. ICLR ²Finzi, M., Stanton, S., Izmailov, P., & Wilson, A. G. (2020, November). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. ICML

Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Steerable group convolutions¹, $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\hat{\mathbf{m}}_{ij} = \hat{\phi}_m(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

$$= \hat{\mathbf{f}}_j \otimes_{cg}^{\hat{\mathbf{w}}(\|\mathbf{x}_j - \mathbf{x}_i\|)} Y(\mathbf{x}_j - \mathbf{x}_i)$$

Aggregate + node updates (convolution + activation fn)

$$\hat{\mathbf{f}}_i' = \phi_f(\sum_{j \in \mathcal{N}(i)} \hat{k}(\mathbf{x}_j - \mathbf{x}_i) \hat{\mathbf{f}}_j)$$

$$(\hat{k} \star \hat{f})(\mathbf{x}_i)$$
Lecture 2

Published as a conference paper at ICLR 2022.

GEOMETRIC AND PHYSICAL QUANTITIES IMPROVE E(3) EQUIVARIANT MESSAGE PASSING

Johannes Brandstetter*
University of Amsterdam
Johannes Kepler University Linz
brandstetter@ml.jku.at

Rob Hesselink* University of Amsterdam r.d.hesselink@uva.nl

Elise van der Pol UvA-Bosch DeltaLab University of Amsterdam e.e.vanderpol@uva.nl

Mar 2022

26

[cs.LG]

arXiv:2110.02905v3

Erik J Bekkers University of Amsterdam e.j.beccers@uva.nl Max Welling UvA-Bosch DeltaLab University of Amsterdam m. welling@uva.nl

ABSTRACT

Including covariant information, such as position, force, velocity or spin is important in many tasks in computational physics and chemistry. We introduce Steerable E(3) Equivariant Graph Neural Networks (SEGNNs) that generalise equivariant graph networks, such that node and edge attributes are not restricted to invariant scalars, but can contain covariant information, such as vectors or tensors. This model, composed of steerable MLPs, is able to incorporate geometric and physical information in both the message and update functions. Through the definition of steerable node attributes, the MLPs provide a new class of activation functions for general use with steerable feature fields. We discuss ours and related work through the lens of equivariant non-linear convolutions, which further allows us to pin-point the successful components of SEGNNs: non-linear message aggregation improves upon classic linear (steerable) point convolutions; steerable messages improve upon recent equivariant graph networks that send invariant messages. We demonstrate the effectiveness of our method on several tasks in computational physics and chemistry and provide extensive ablation studies.

1 INTRODUCTION

The success of Convolutional Neural Networks (CNNs) (LeCun et al., 1998) [2015] Schmidhuber, [2015] Krizhevsky et al. [2012] is a key factor for the rise of deep learning, attributed to their capability of exploiting translation symmetries, hereby introducing a strong inductive bias. Recent work has shown that designing CNNs to exploit additional symmetries via group convolutions has even further increased their performance Cohen & Welling, 2016; 2017; Worrall et al., 2017; Cohen et al. 2018; Kondor & Trivedi, 2018; Weiler et al., 2018; Bekkers et al., 2018; Bekkers 2019; Weiler & Cesa, 2019). Graph neural networks (GNNs) and CNNs are closely related to each other via their aggregation of local information. More precisely, CNNs can be formulated as message passing layers (Gilmer et al., 2017) based on a sum aggregation of messages that are obtained by relative position-dependent linear transformations of neighbouring node features. The power of message passing layers is, however, that node features are transformed and propagated in a highly *non-linear* manner. Equivariant GNNs have been proposed before as either PointConv-type (Wu et al., 2019; Kristof et al., [2017] implementations of steerable (Thomas et al., [2018] Anderson et al., [2019] Fuchs. et al., 2020) or regular group convolutions (Finzi et al., 2020). The most important component in these methods are the convolution layers. Although powerful, such layers only (pseudo¹) linearly transform the graphs and non-linearity is only obtained via point-wise activations.

¹Methods such as SE(3)-transformers (Fuchs et al., 2020) and Cormorant (Anderson et al., 2019) include an input-dependent attention component that augments the convolutions.



Graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$

- nodes $v_i \in \mathcal{V}$ with node feature $\mathbf{f}_i \in \mathbb{R}^{C_v}$ and position $x_i \in X$
- edges $e_{ij} \in \mathcal{E}$ with edge attribute $\mathbf{a}_{ij} \in \mathbb{R}^{C_e}$

Special case (Steerable group convolutions¹, $X = \mathbb{R}^d$):

Messages (linear transformations based on kernel)

$$\hat{\mathbf{m}}_{ij} = \hat{\phi}_m(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

$$= \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

Aggregate + node updates (convolution + activation fn)

$$\hat{\mathbf{f}}_i' = \phi_f(\sum_{j \in \mathcal{N}(i)} \hat{k}(\mathbf{x}_j - \mathbf{x}_i) \hat{\mathbf{f}}_j)$$

$$\hat{k} \times \hat{f}(\mathbf{x}_i)$$

$$\hat{k} \times \hat{f}(\mathbf{x}_i)$$
Lecture 2

Published as a conference paper at ICLR 2022.

GEOMETRIC AND PHYSICAL QUANTITIES IMPROVE E(3) EQUIVARIANT MESSAGE PASSING

Johannes Brandstetter*
University of Amsterdam
Johannes Kepler University Linz
brandstetter@ml.jku.at

Rob Hesselink*
University of Amsterdam
r.d.hesselink@uva.nl

Elise van der Pol UvA-Bosch DeltaLab University of Amsterdam e.e.vanderpol@uva.nl

Mar 2022

26

[cs.LG]

arXiv:2110.02905v3

Erik J Bekkers University of Amsterdam e.j.beccers@uva.nl Max Welling UvA-Bosch DeltaLab University of Amsterdam m. welling@uva.nl

ABSTRACT

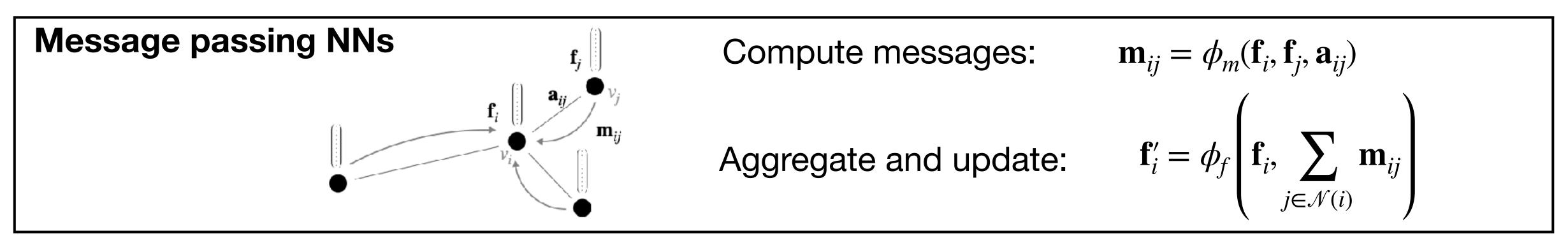
Including covariant information, such as position, force, velocity or spin is important in many tasks in computational physics and chemistry. We introduce Steerable E(3) Equivariant Graph Neural Networks (SEGNNs) that generalise equivariant graph networks, such that node and edge attributes are not restricted to invariant scalars, but can contain covariant information, such as vectors or tensors. This model, composed of steerable MLPs, is able to incorporate geometric and physical information in both the message and update functions. Through the definition of steerable node attributes, the MLPs provide a new class of activation functions for general use with steerable feature fields. We discuss ours and related work through the lens of equivariant non-linear convolutions, which further allows us to pin-point the successful components of SEGNNs: non-linear message aggregation improves upon classic linear (steerable) point convolutions; steerable messages improve upon recent equivariant graph networks that send invariant messages. We demonstrate the effectiveness of our method on several tasks in computational physics and chemistry and provide extensive ablation studies.

1 INTRODUCTION

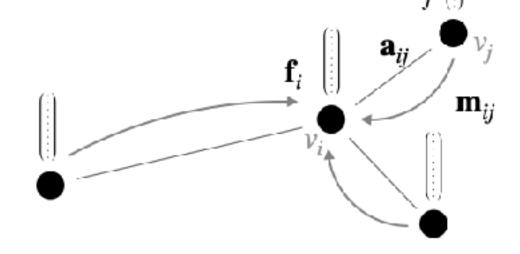
The success of Convolutional Neural Networks (CNNs) (LeCun et al., 1998) [2015] Schmidhuber, [2015] Krizhevsky et al. [2012] is a key factor for the rise of deep learning, attributed to their capability of exploiting translation symmetries, hereby introducing a strong inductive bias. Recent work has shown that designing CNNs to exploit additional symmetries via group convolutions has even further increased their performance Cohen & Welling, 2016; 2017; Worrall et al., 2017; Cohen et al. 2018; Kondor & Trivedi, 2018; Weiler et al., 2018; Bekkers et al., 2018; Bekkers 2019; Weiler & Cesa, 2019). Graph neural networks (GNNs) and CNNs are closely related to each other via their aggregation of local information. More precisely, CNNs can be formulated as message passing layers (Gilmer et al., 2017) based on a sum aggregation of messages that are obtained by relative position-dependent linear transformations of neighbouring node features. The power of message passing layers is, however, that node features are transformed and propagated in a highly *non-linear* manner. Equivariant GNNs have been proposed before as either PointConv-type (Wu et al., 2019; Kristof et al., [2017] implementations of steerable (Thomas et al., [2018] Anderson et al., [2019] Fuchs. et al., 2020) or regular group convolutions (Finzi et al., 2020). The most important component in these methods are the convolution layers. Although powerful, such layers only (pseudo¹) linearly transform the graphs and non-linearity is only obtained via point-wise activations.

¹Methods such as SE(3)-transformers (Fuchs et al., 2020) and Cormorant (Anderson et al., 2019) include an input-dependent attention component that augments the convolutions.





Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

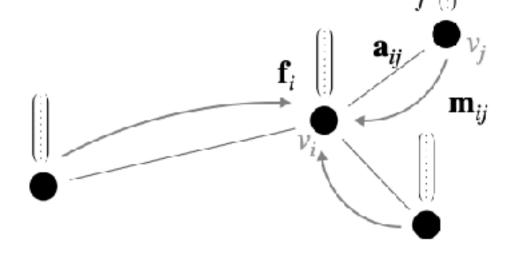
Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1}g_j)\mathbf{f}_j$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$
$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1}g_i)\mathbf{f}_j$$

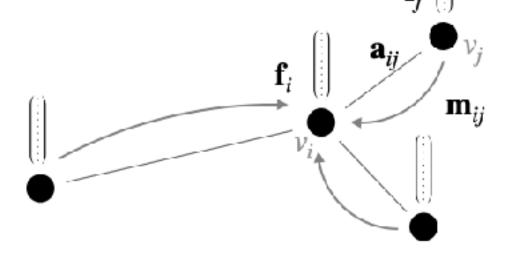
Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$
$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1}g_j)\mathbf{f}_j$$

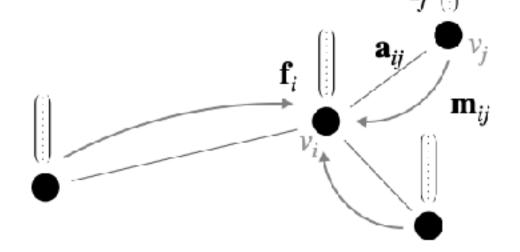
Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$
$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1}g_j)\mathbf{f}_j$$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

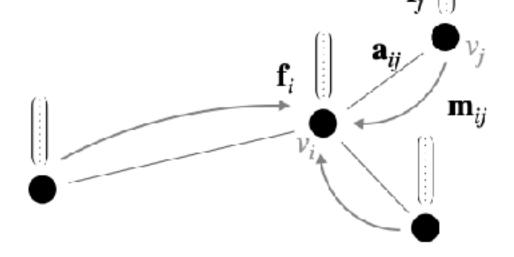
$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, ||\mathbf{x}_j - \mathbf{x}_i||)$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$
$$\mathbf{m}_{ij} = \mathbf{W}(g_i^{-1}g_i)\mathbf{f}_j$$

Steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, ||\mathbf{x}_j - \mathbf{x}_i||)$$

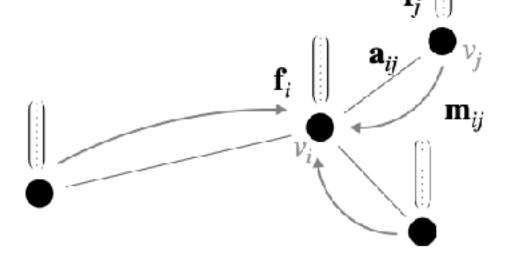
Equivariant (Steerable) Message Passing NNs (Lecture 3)

$$\hat{\mathbf{m}}_{ij} = \widehat{\mathrm{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\mathsf{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(n)}(...(\sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(1)}\hat{\mathbf{h}}_i))))$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$



Linear convolution $W(g_i^{-1}g_j)\mathbf{f}_i$

steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, ||\mathbf{x}_j - \mathbf{x}_i||)$$

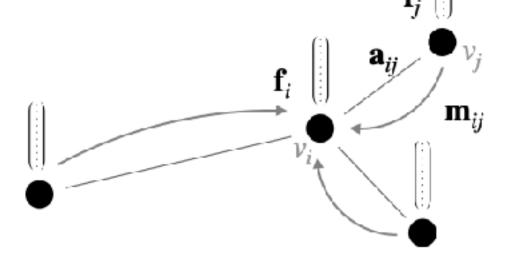
Equivariant (Steerable) Message Passing NNs (Lecture 3)

$$\hat{\mathbf{m}}_{ij} = \widehat{\mathrm{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\mathsf{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(n)}(...(\sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(1)}\hat{\mathbf{h}}_i))))$$

Message passing NNs



Compute messages:

$$\mathbf{m}_{ij} = \phi_m(\mathbf{f}_i, \mathbf{f}_j, \mathbf{a}_{ij})$$

Aggregate and update:

$$\mathbf{f}'_i = \phi_f \left(\mathbf{f}_i, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \right)$$

Classic point convolutions

(Lecture 1.7: regular g-convs on homogeneous spaces)

$$\mathbf{m}_{ij} = \mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)\mathbf{f}_j$$

Linear convolution $W(g_i^{-1}g_i)\mathbf{f}_i$

steerable G-CNNs

(Lecture 2: steerable g-convs)

$$\mathbf{m}_{ij} = \mathbf{W}_{\hat{\mathbf{a}}_{ij}}(\|\mathbf{x}_j - \mathbf{x}_i\|)\hat{\mathbf{f}}_j$$

$$:= \hat{\mathbf{f}}_j \otimes_{cg}^{\mathbf{W}(\|\mathbf{x}_j - \mathbf{x}_i\|)} \hat{\mathbf{a}}_{ij}$$

Invariant Message Passing NNs

(Lecture 3)

$$\mathbf{m}_{ij} = \text{MLP}(\mathbf{f}_i, \mathbf{f}_j, ||\mathbf{x}_j - \mathbf{x}_i||)$$

Non-linear "convolution"

Equivariant (Steerable) Message Passing

(Lecture 3)

$$\hat{\mathbf{m}}_{ij} = \widehat{\mathrm{MLP}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \mathbf{x}_j - \mathbf{x}_i)$$

With steerable MLP:

$$\widehat{\mathsf{MLP}}_{\hat{\mathbf{a}}_{ij}}(\hat{\mathbf{f}}_i, \hat{\mathbf{f}}_j, \|\mathbf{x}_j - \mathbf{x}_i\|) := \sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(n)}(...(\sigma(\mathbf{W}_{\hat{\mathbf{a}}_{ij}}^{(1)}\hat{\mathbf{h}}_i))))$$