arxiv.org/abs/2312.07511

A Hitchhiker's Guide to Geometric GNNs for 3D atomic systems

Alexandre Duval*, Simon V. Mathis*, Chaitanya K. Joshi*,

Victor Schmidt*, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen,
Pietro Liò, Yoshua Bengio, Michael Bronstein

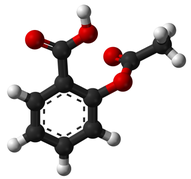
Objective of this talk

Present the content of this guide
but most importantly **trigger discussions and exchanges**
about the state of the field and research directions !

Executive Summary

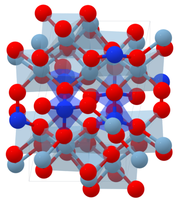
A taxonomy of Geometric GNN architectures

Categorised by intermediate features within layers



Small molecules

Biomolecules



Materials

Geometric GNNs

2018 2019 2020 2021 2022 2023

Invariant GNNs

SchNet
CGCNN

MEGNet

DimeNet

GemNet
SphereNet
Inv. Point Attention

GearNet
ComENet

...

Cartesian Equivariant GNNs

GVP-GNN

PaiNN
E(n)-GNN

Eq.Transformer
ClofNet
SO3krates

TensorNet

Spherical Equivariant GNNs

Tensor Field Network

Cormorant

SE(3)-Transformer

NequIP
SEGNN

MACE
Allegro
Equiformer

eSCN

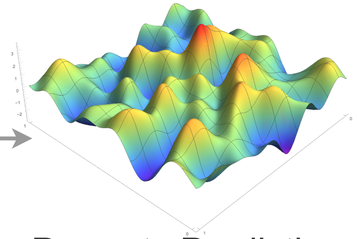
Unconstrained GNNs

ForceNet

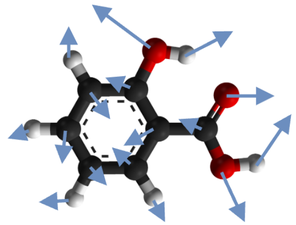
Spherical Channel Network

FAENet,
PET

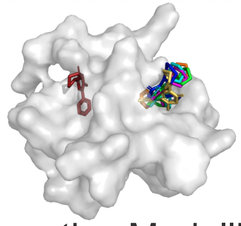
Applications



Property Prediction



Dynamics Simulation

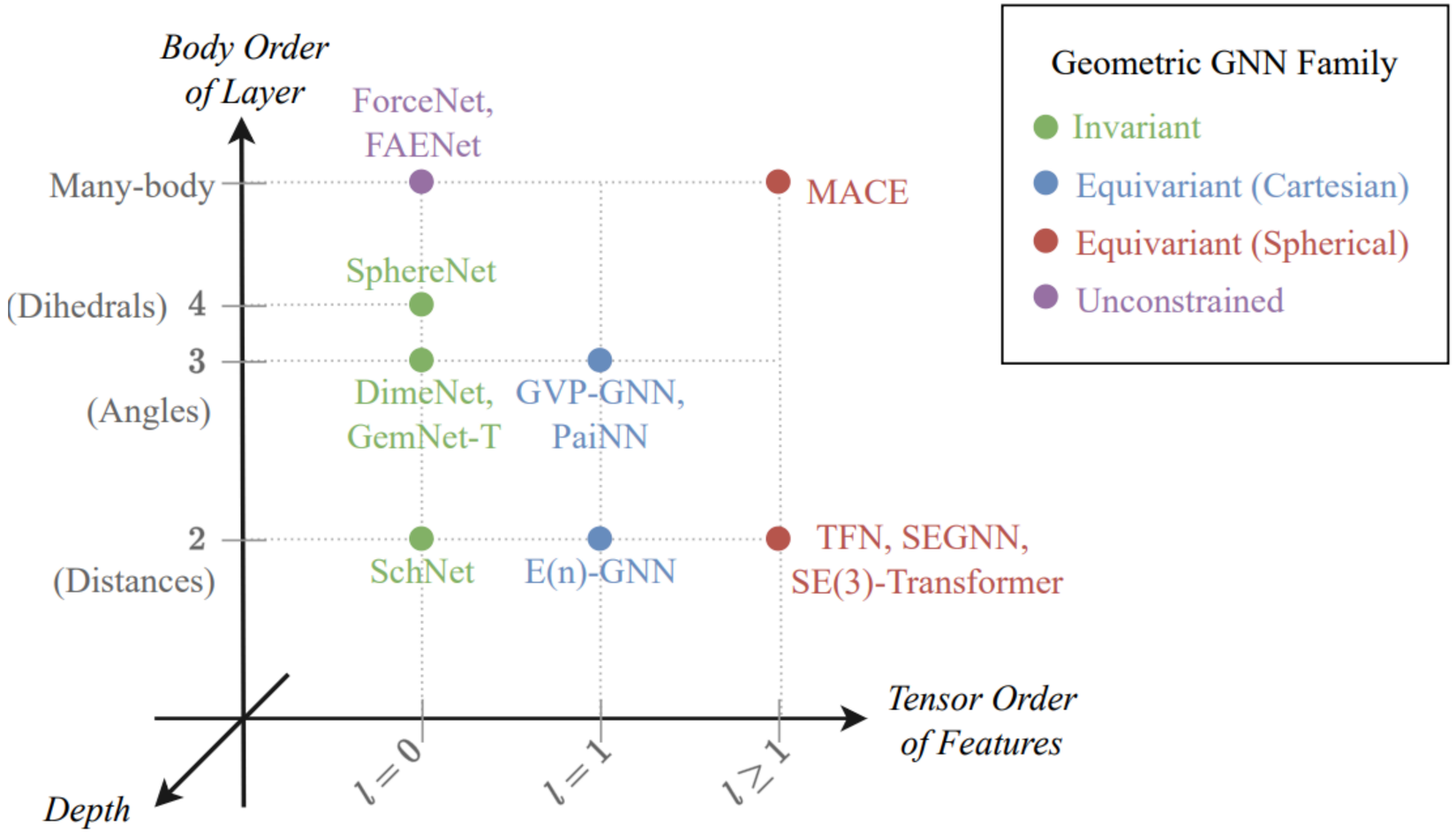


Generative Modelling

Structure Prediction

Understanding Geometric GNNs

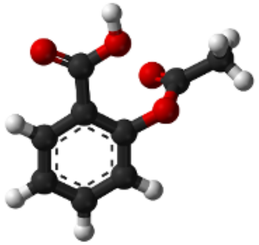
The “turnable knobs” where innovation may happen



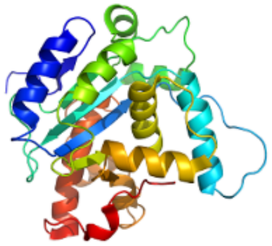
Joshi, Bodnar, Mathis, Cohen, Liò. On the Expressive Power of Geometric GNNs. ICML, 2023.

Why care about Geometric GNNs?

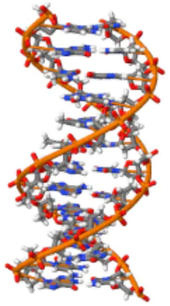
Systems with geometric & relational structure



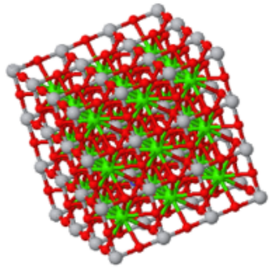
Small Molecules



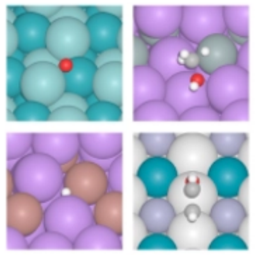
Proteins



DNA/RNA



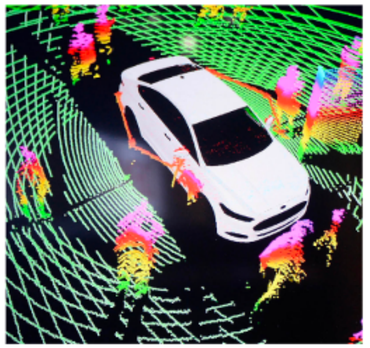
Inorganic Crystals



Catalysis Systems

Transportation & Logistics

Robotic Navigation

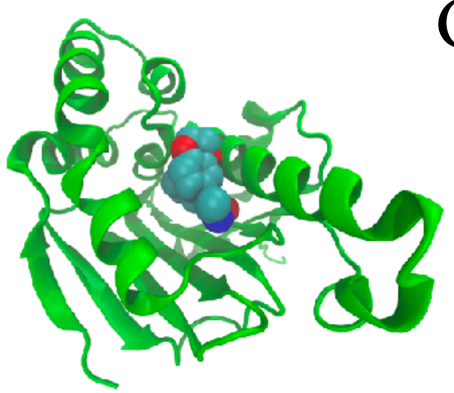


3D Computer Vision

Geometric Graph Neural Networks

Fundamental tool for machine learning on 3D graphs

Eg. High throughput virtual screening



Geometric Graph → Geometric GNN → Prediction

- Functional properties?
- Ligand binding affinity?
- Ligand efficacy?

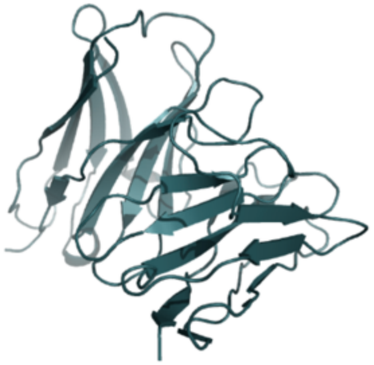
[1] Gligorijević et al., DeepFRI, Nature Communications, 2021.

[2] Zhang et al., GearNet, ICLR, 2023.

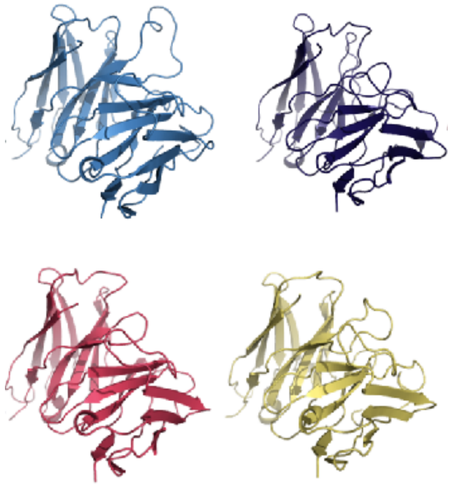
Geometric Graph Neural Networks

Embedder or denoiser within 3D generative models

Eg. Protein generation and inverse design



Geometric Graph → Geometric GNN → Generative Model → New Geometric Graphs



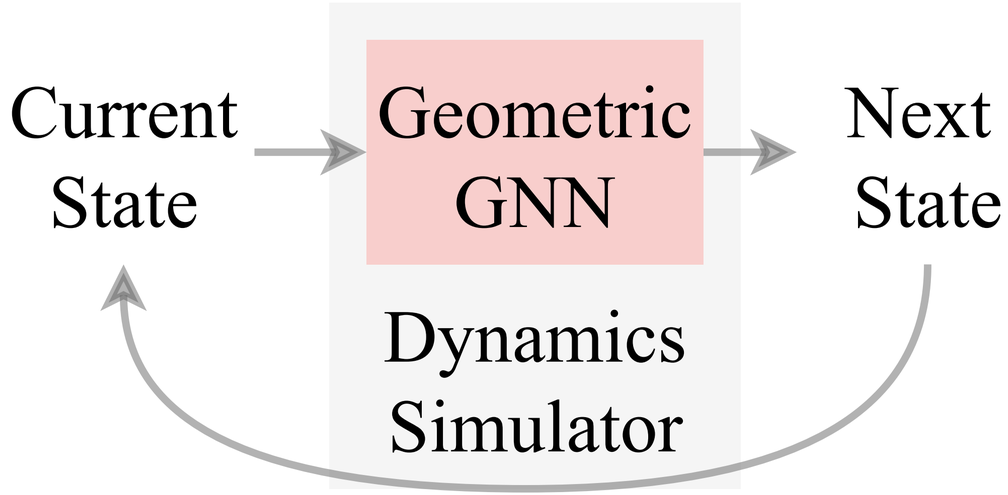
[1] Dauparas et al., ProteinMPNN, Science, 2022.

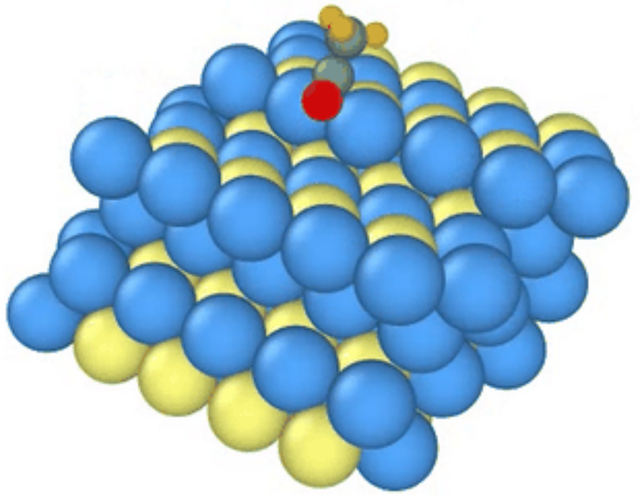
[2] Watson, Juergens et al., RFdiffusion, Nature, 2023.

Geometric Graph Neural Networks

Learning to simulating molecular dynamics

Eg. Catalyst-adsorbate interaction





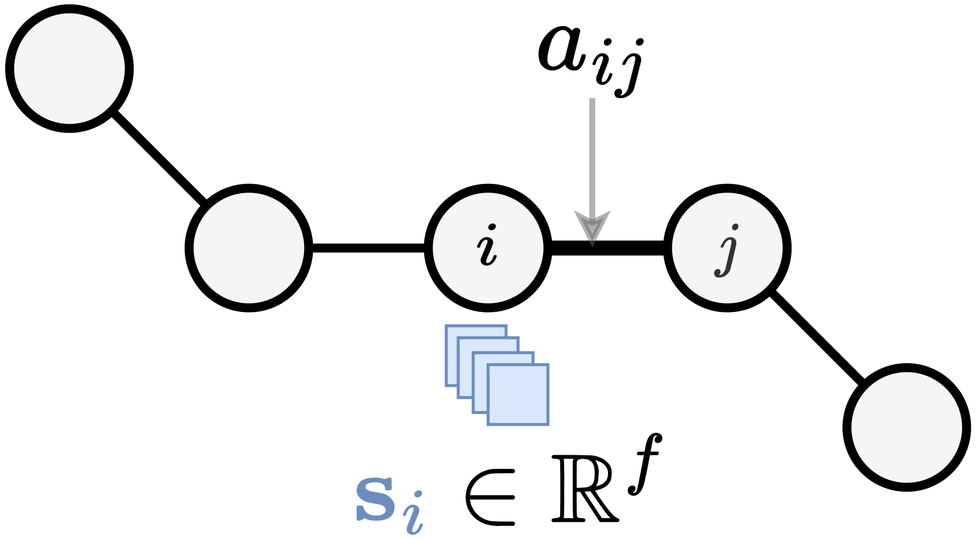
[1] Batzner et al., NequIP, Nature Communications, 2022.

[2] Batatia et al., MACE, NeurIPS 2022.

From GNNs to Geometric GNNs

Normal graphs

A graph is a set of nodes connected by edges



E.g. atom type

$$\mathcal{G} = (\boldsymbol{A}, \boldsymbol{S})$$

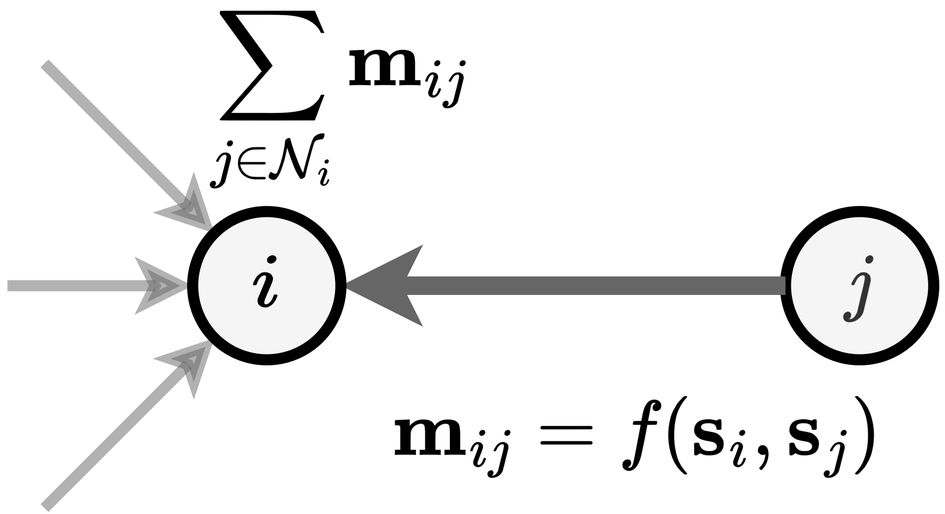
Scalar features $\in \mathbb{R}^{n \times f}$

$n \times n$ adjacency matrix

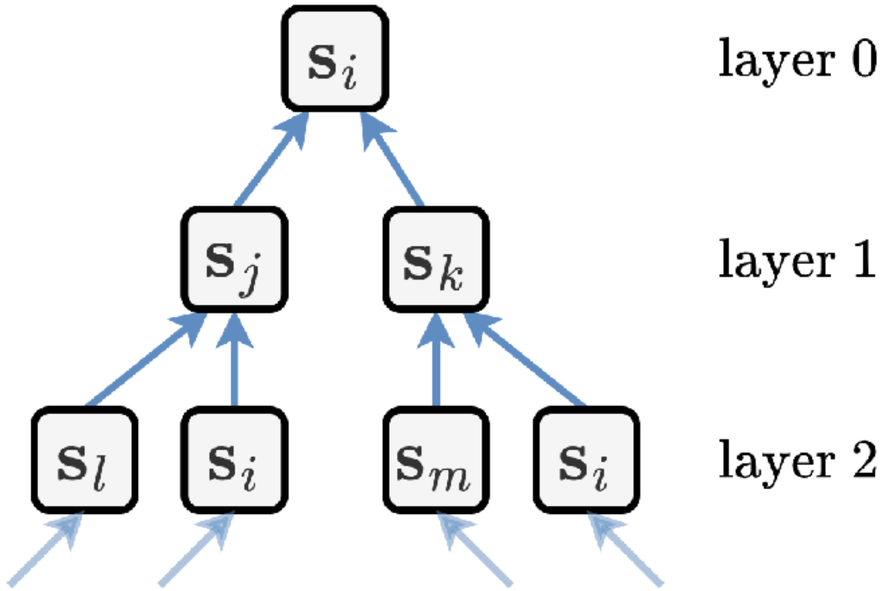
Note: f is the dimension or number of scalar feature channels, so $\mathbf{S}$ is a list.

Normal Graph Neural Networks

Message passing updates node features using local aggregation



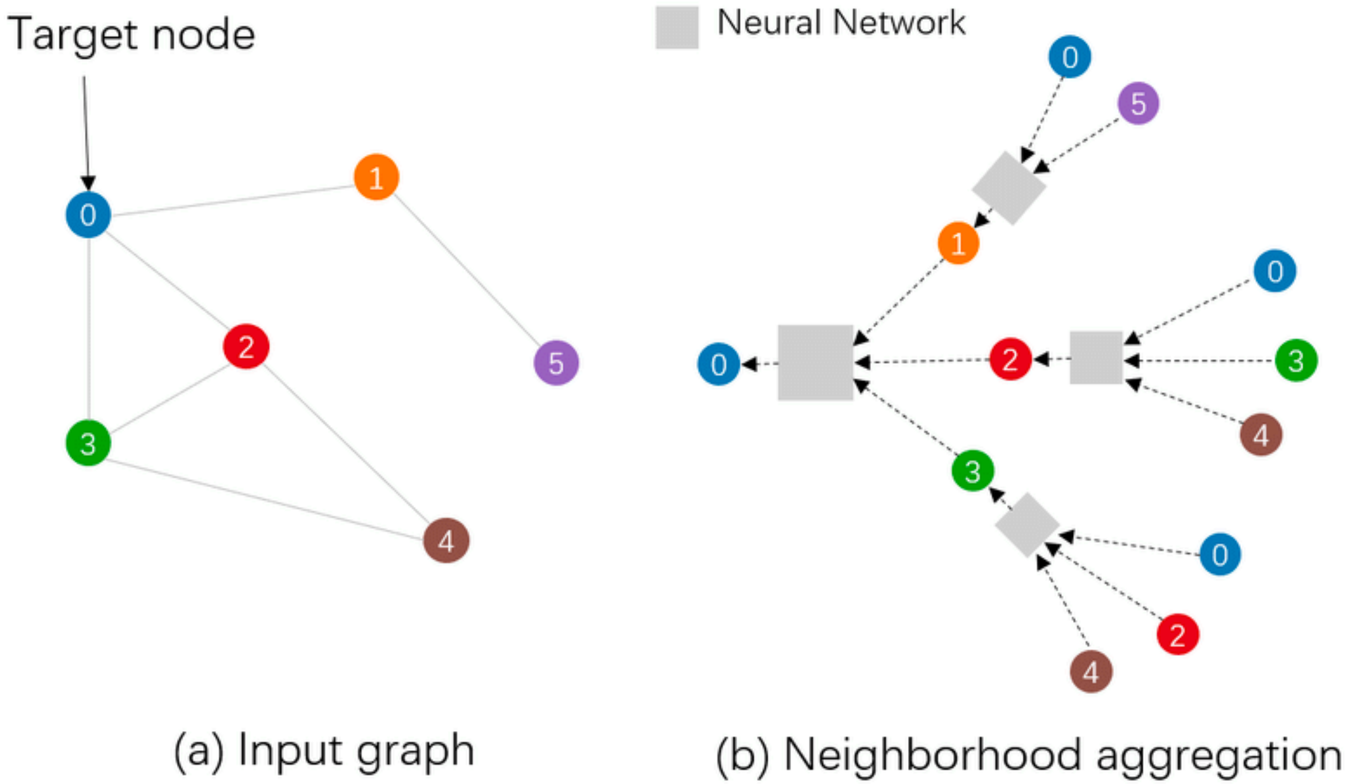
$$m_i^{(t)} := \text{AGG}\left(\{\!\{(s_i^{(t)}, s_j^{(t)}) \mid j \in \mathcal{N}_i\}\!\}\right),$$
$$s_i^{(t+1)} := \text{UPD}\left(s_i^{(t)}, m_i^{(t)}\right),$$



Computation tree:
Message passing gathers & propagates features beyond local neighbourhoods.

Normal Graph Neural Networks

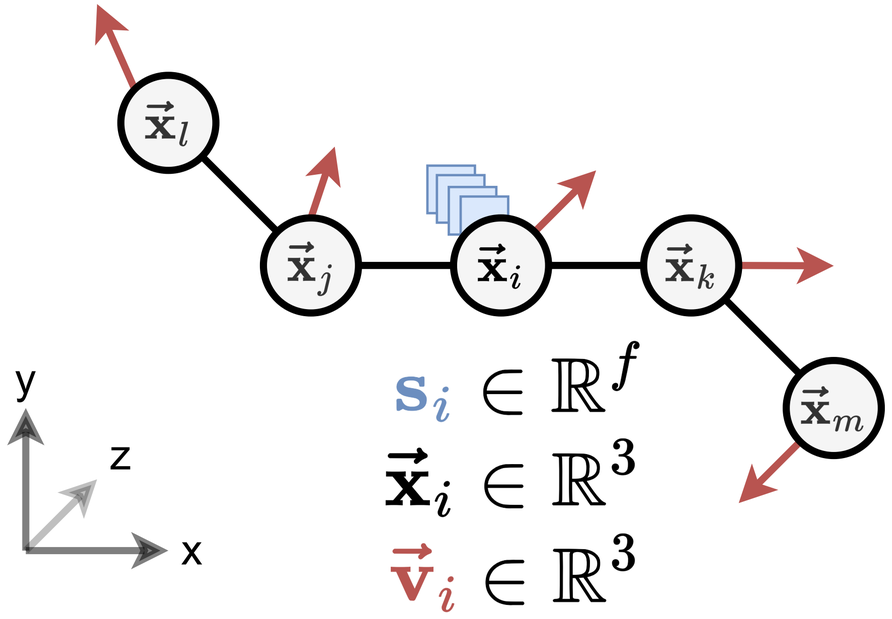
Learn how to propagate information along the graph



Geometric graphs

Each node is:

- **embedded in Euclidean space** *e.g.* atoms in 3D
- **decorated with geometric attributes** *s.a.* velocity



$$\mathcal{G} = (\boldsymbol{A}, \boldsymbol{S}, \vec{\boldsymbol{X}}, \vec{\boldsymbol{V}})$$

Adjacency
$\in \mathbb{R}^{n \times n}$

Scalar features
$\in \mathbb{R}^{n \times a}$

Geometric features*
$\in \mathbb{R}^{n \times b \times d}$

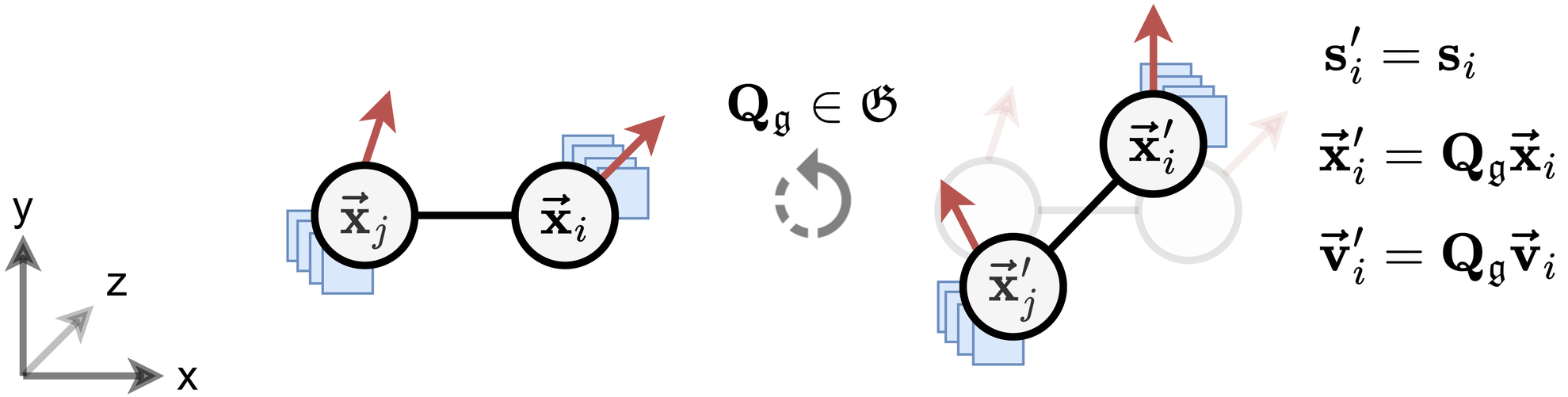
Coordinates
$\in \mathbb{R}^{n \times d}$

* We work with a single vector feature per node, but our setup generalises to multiple vector features and higher-order tensors.

Physical symmetries

Geometric attributes transform with Euclidean transformations of the system

Rotations & **Reflections** $Q_{\mathfrak{g}} \in \mathfrak{G}$ act on only vectors $\vec{V}$ and coordinates $\vec{X}$:



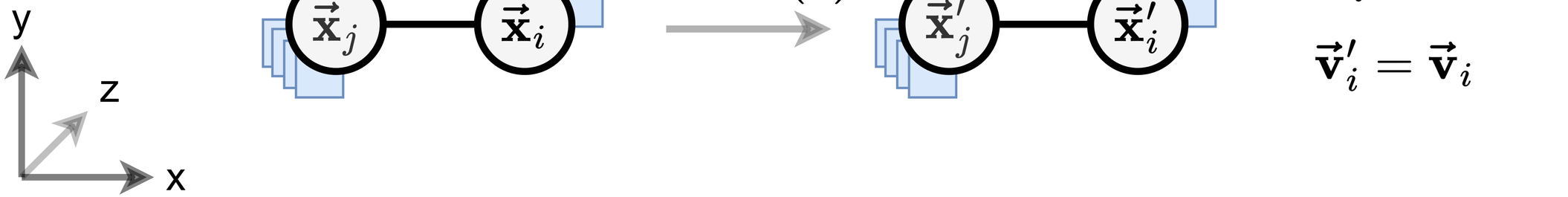
Scalar features remain unchanged $\rightarrow$ **invariant**.

* We use $\mathfrak{G}$ to denote rotations SO(d) or rotations and reflections O(d)

Physical symmetries

Geometric attributes transform with Euclidean transformations of the system

Translations $\vec{t} \in T(d)$ act on only the coordinates $\vec{X}$:



$$\mathbf{s}'_i = \mathbf{s}_i$$

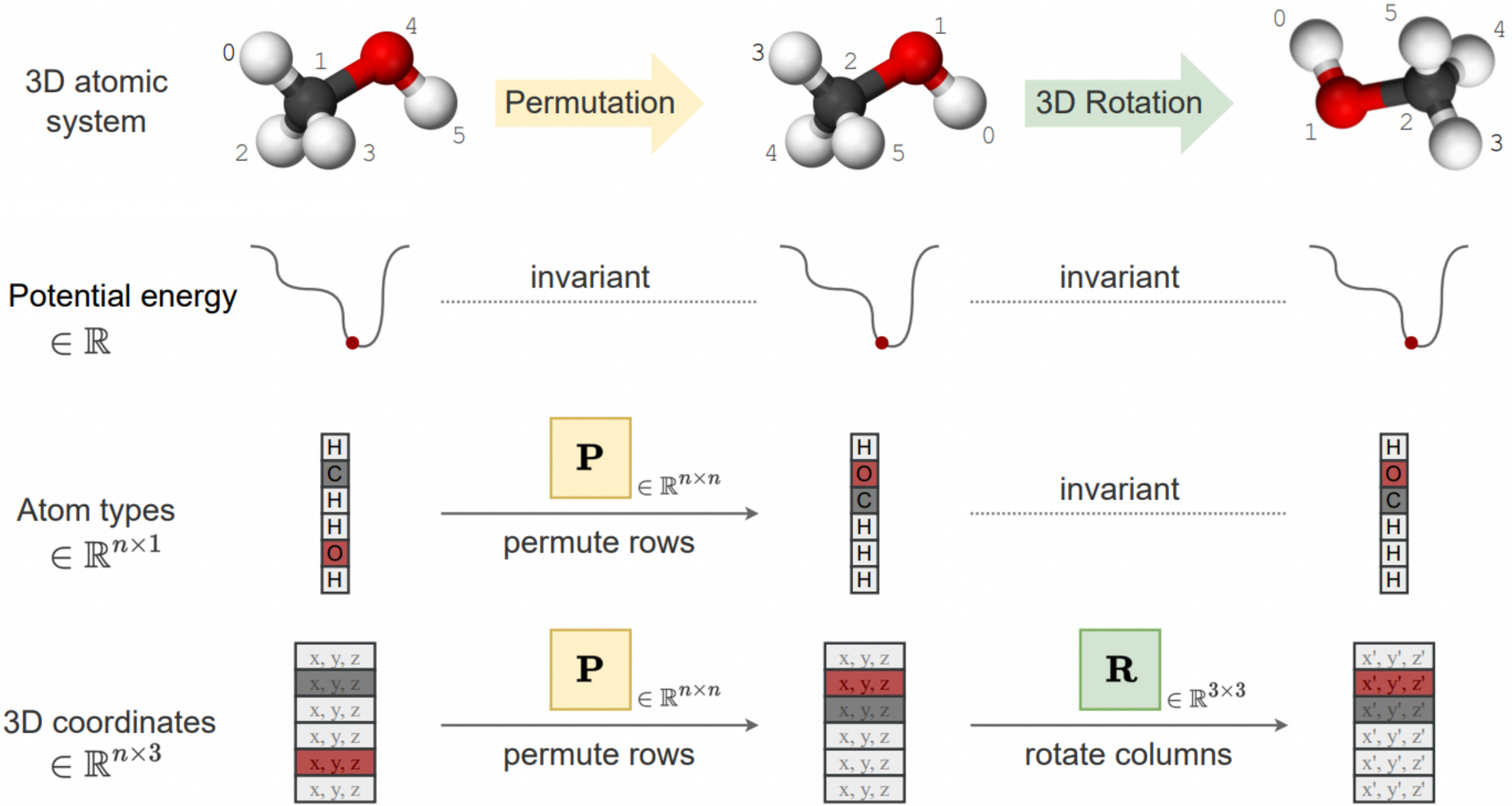
$$\vec{\mathbf{x}}'_i = \vec{\mathbf{x}}_i + \vec{\mathbf{t}}$$

$$\vec{\mathbf{v}}'_i = \vec{\mathbf{v}}_i$$

Scalar and vector features remain unchanged $\rightarrow$ **invariant**.

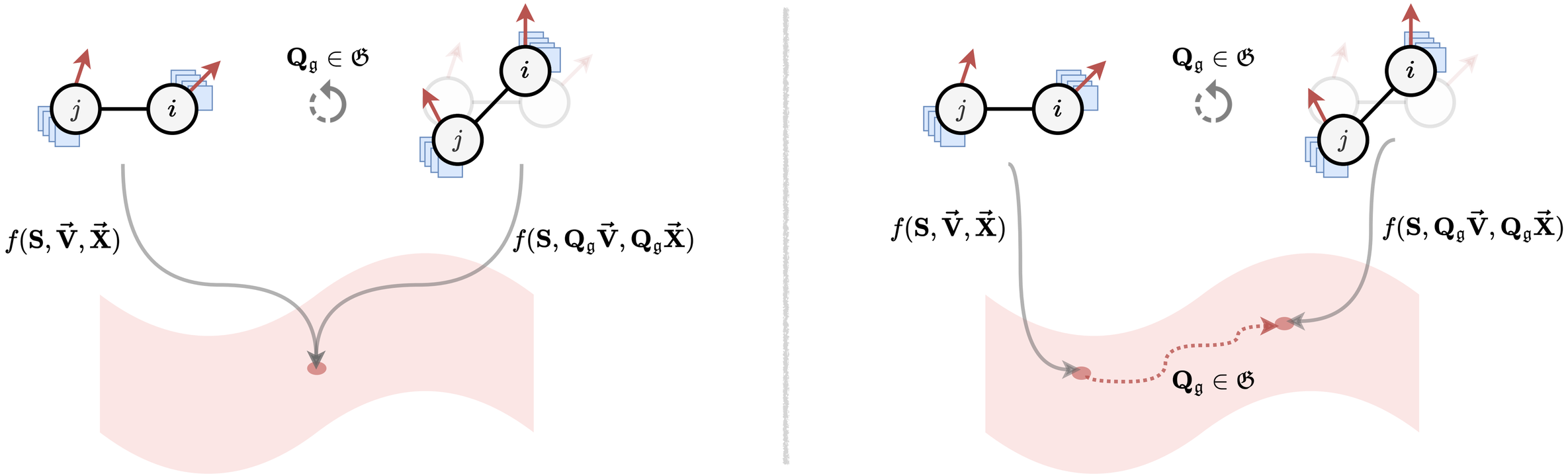
Why building physics into GNNs?

Geometric GNNs should account for physical symmetries



Building blocks of Geometric GNNs

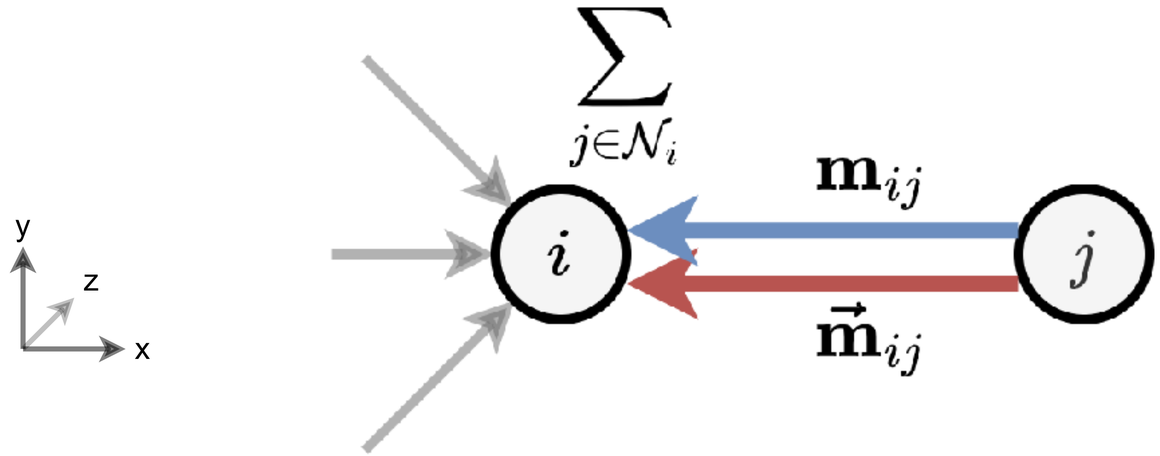
- **Scalar features** must be updated in an invariant manner.
- **Vector features** must be updated in an equivariant manner.



Invariant functions vs. Equivariant functions

Geometric message passing

- **update scalar and** (optionally) **vector features**
- **aggregate and update functions which retain transformation semantics**



$$\boldsymbol{m}_i^{(t)}, \vec{\boldsymbol{m}}_i^{(t)} := \quad \text{(Aggregate)}$$

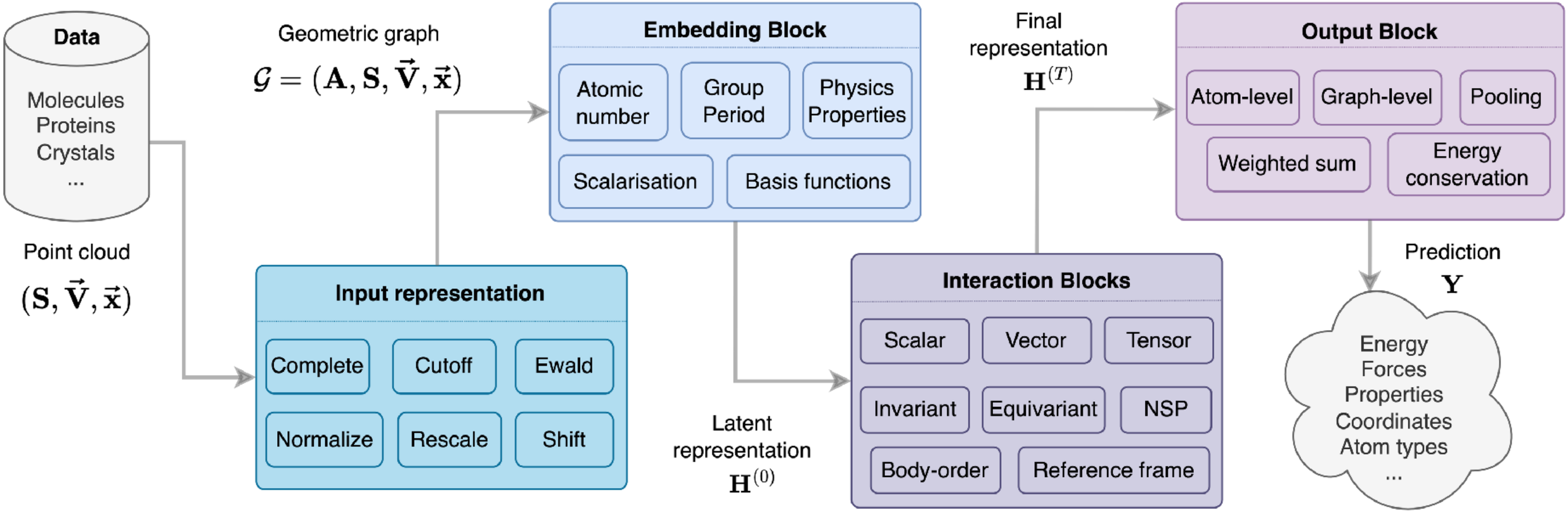
$$\boldsymbol{s}_i^{(t+1)}, \vec{\boldsymbol{v}}_i^{(t+1)} := \quad \text{(Update)}$$

This talk: studying how these functions are defined.

Modelling pipeline

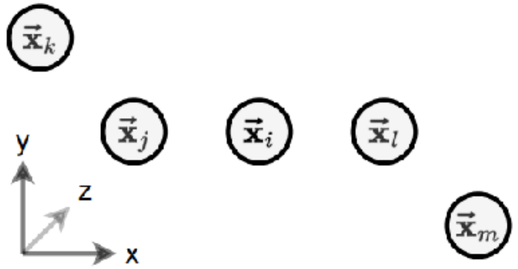
Pipeline

for prediction tasks on 3D atomic systems

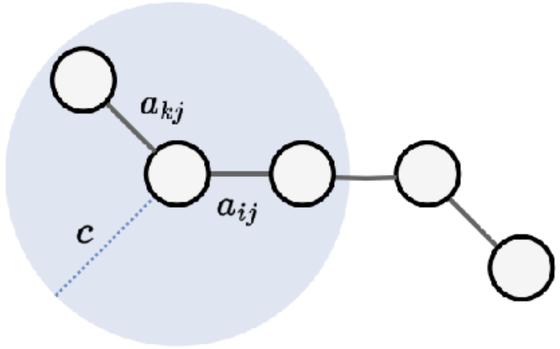


Input representation

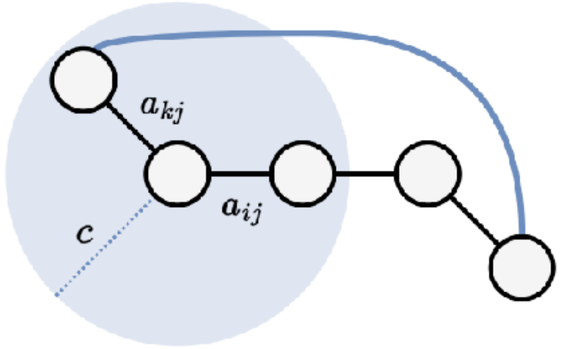
Various heuristics for constructing edges



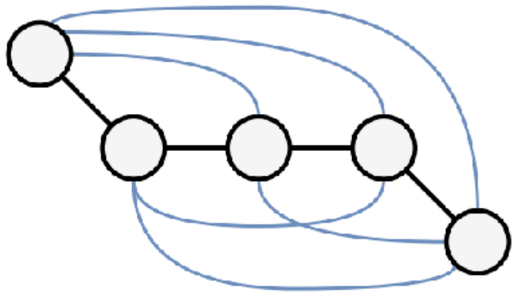
3D point cloud



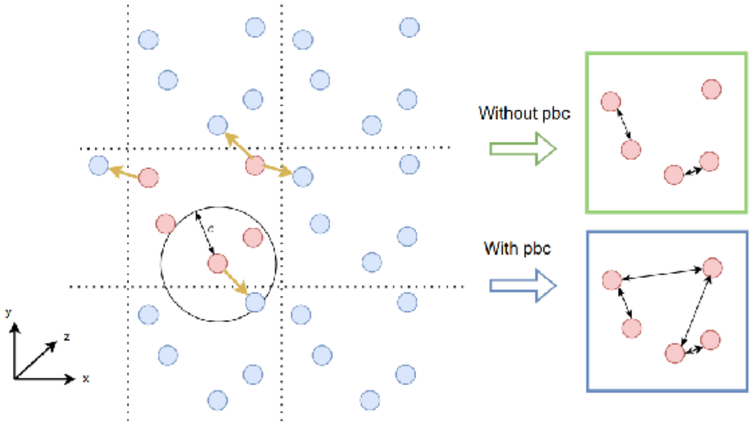
Smoothed cutoff graph

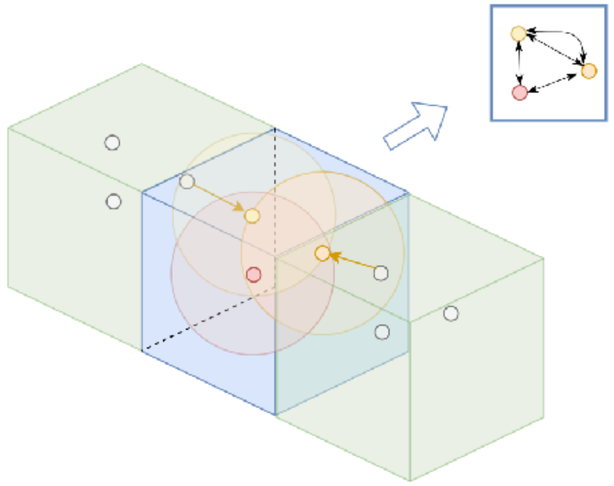


Long-range connections



Complete graph

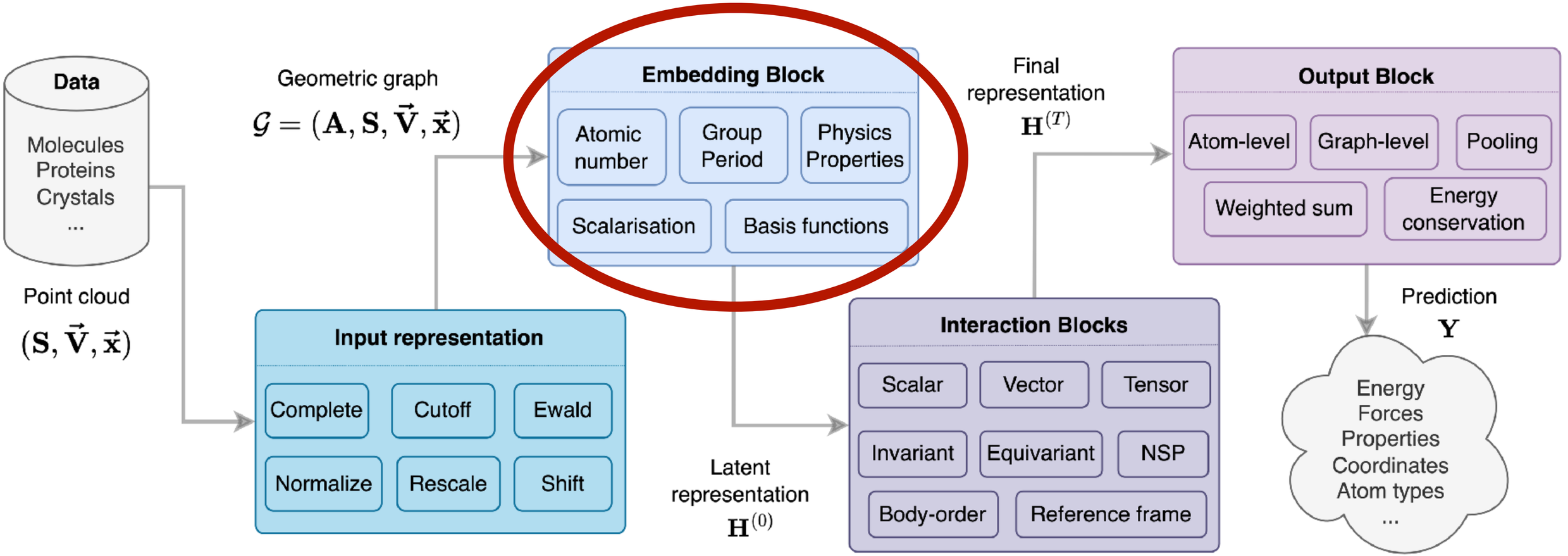




Periodic boundary conditions

Embedding Block

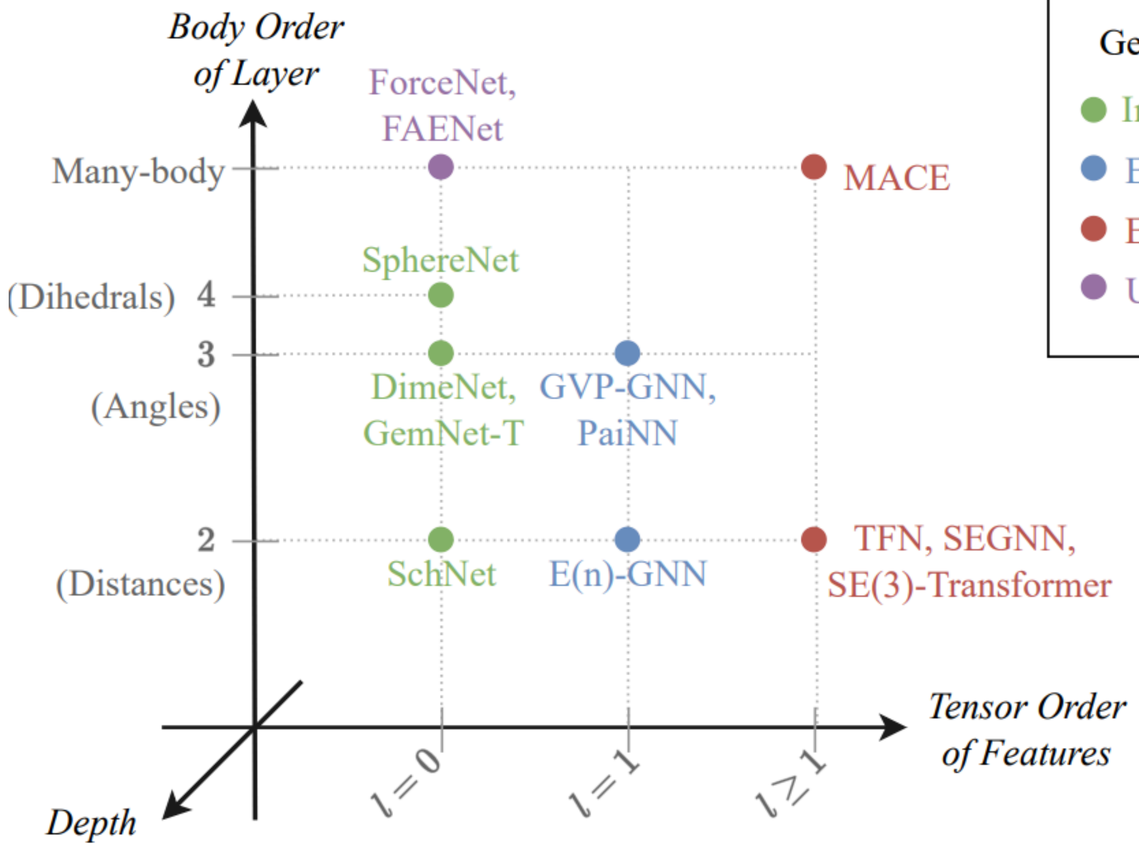
How to encode the geometric graph ?



[1] Duval et al., PhAST, JMLR, 2023.

Interaction Block

How to learn efficient geometric and relational features ?



Geometric GNN Family

- Invariant
- Equivariant (Cartesian)
- Equivariant (Spherical)
- Unconstrained

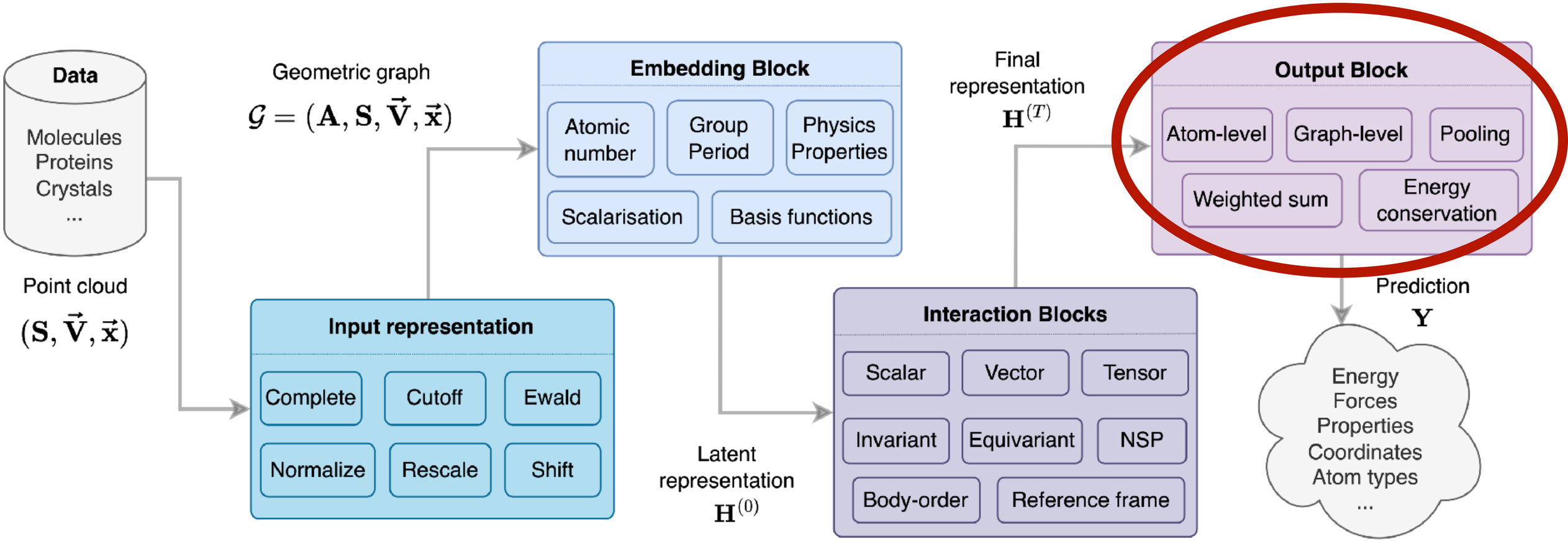
Devise efficient and expressive approaches for encoding and processing geometric graphs while respecting physical symmetries. This can be seen as an *inductive bias.*

$$\boldsymbol{s}_i^{(t+1)}, \vec{\mathbf{v}}_i^{(t+1)} := \underline{\textsc{Upd}}\left(\left(\boldsymbol{s}_i^{(t)}, \vec{\mathbf{v}}_i^{(t)}\right), \bigoplus_{j \in \mathcal{N}_i} \underline{\textsc{Msg}}\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{s}_j^{(t)}, \vec{\mathbf{v}}_i^{(t)}, \vec{\mathbf{v}}_j^{(t)}, \vec{x}_{ij}\right)\right),$$

$$\boldsymbol{s}_i^{(t+1)} := \underline{\textsc{Upd}}\left(\boldsymbol{s}_i^{(t)}, \bigoplus_{j \in \mathcal{N}_i} \left(\underline{\textsc{Msg}}(\boldsymbol{s}_i^{(t)}, \boldsymbol{s}_j^{(t)}, \vec{x}_{ij})\right)\right).$$

Output Block

How to compute predictions from final representations ?



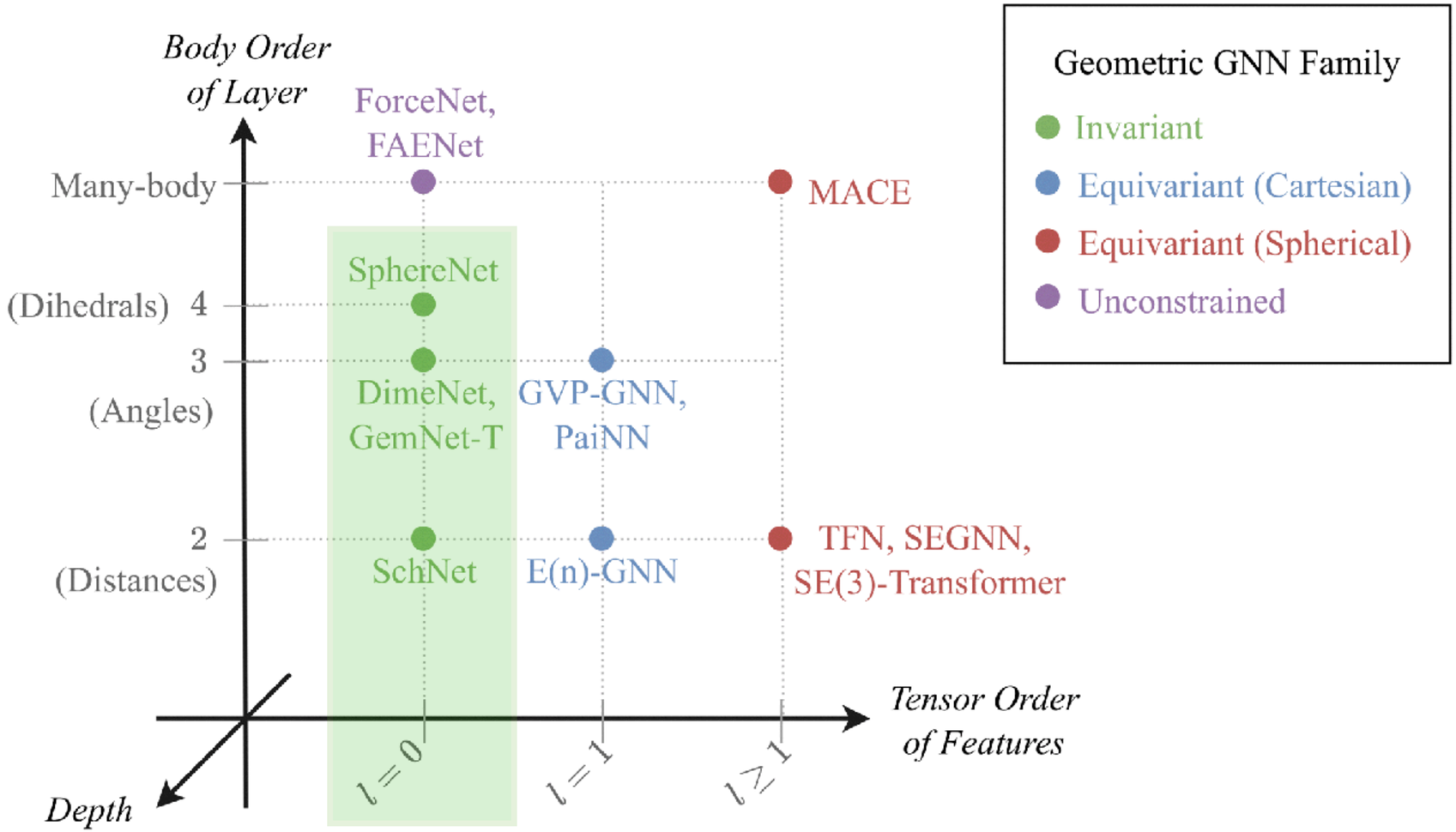
[1] Duval et al., PhAST, JMLR, 2023.

Invariant Geometric GNNs

𝔊-invariant Geometric GNNs

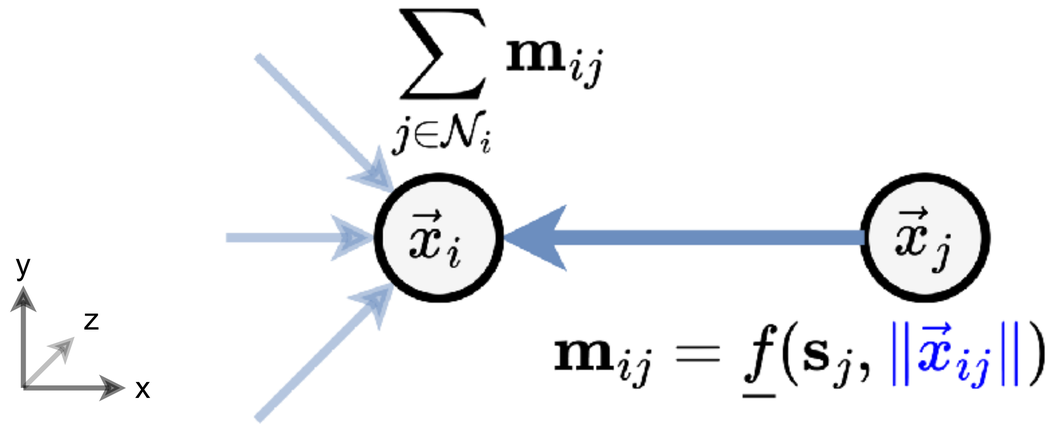
Update latent representations by scalarising local geometric information

Key design choice:
Body order of scalarisation



Distance-based GNNs

SchNet[1] uses relative distances $\|\vec{x}_i - \vec{x}_j\|$ to scalarise local geometry



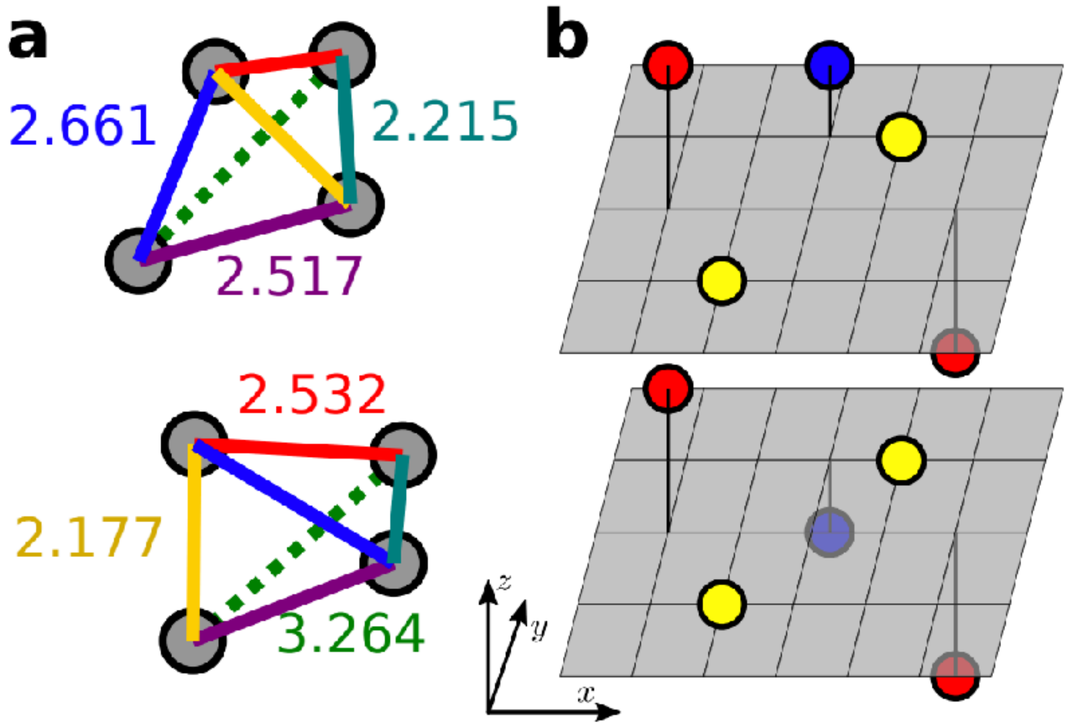
Relative distances are invariant to global rotations & translations.

$$\boldsymbol{s}_i^{(t+1)} := \boldsymbol{s}_i^{(t)} + \sum_{j \in \mathcal{N}_i} f_1\left(\boldsymbol{s}_j^{(t)},\ \|\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j\|\right)$$

[1] Schütt et al., SchNet, Journal of Chemical Physics, 2018.
[2] Xie and Grossman, CGCNN, Phys. Rev. Letters, 2018.
[3] Li et al., IROS, 2020. Similar architecture for multi-agent robotics.
[4] Sanchez-Gonzalez et al., ICML, 2020. Similar architecture for physical simulation

Distinguish geometric graphs

Can you tell these two local neighbourhoods apart using the **unordered set of distances** only?[1]

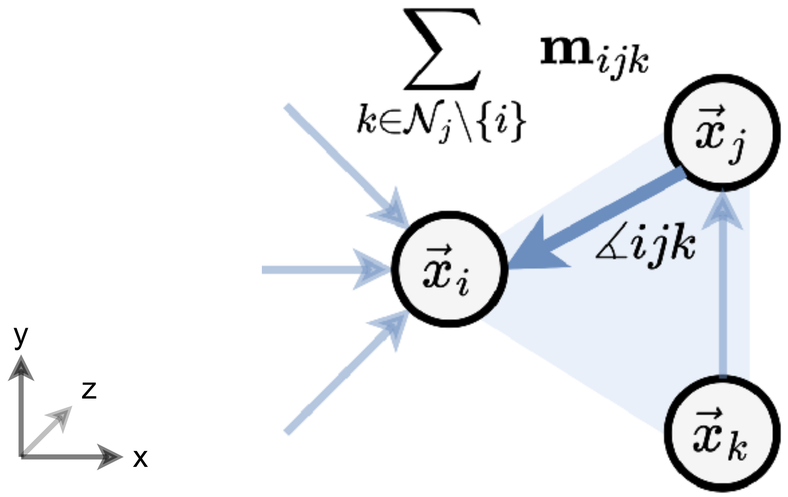


Ideally, you want to distinguish between these two graphs, but SchNet is not capable of doing so because atom pairwise distances are identical.

[1] Bartok et al., Phys. Rev. B, 2013. [2] Pozdynakov et al., Incompleteness of atomic structural representations, Phys. Rev. Letters, 2020.

Going beyond distances

DimeNet uses distances AND angles $\measuredangle ijk = \vec{x}_{ij} \cdot \vec{x}_{jk}$ among triplets



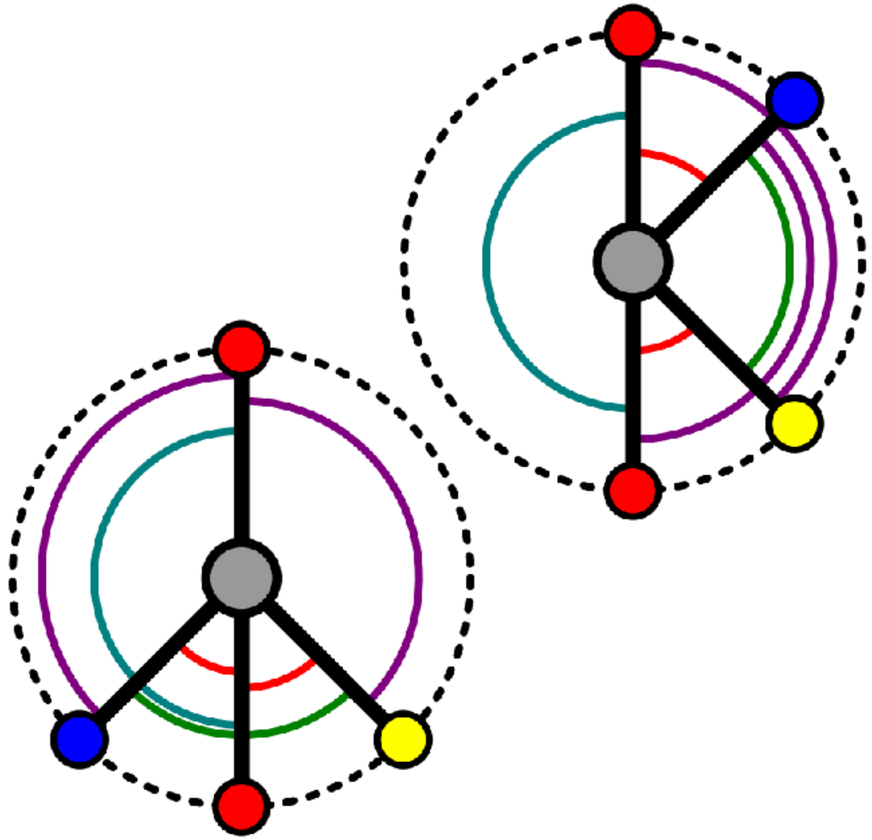
$$\mathbf{m}_{ijk} = f(\mathbf{s}_j, \mathbf{s}_k, \|\vec{x}_{ij}\|, \measuredangle ijk)$$

Body order of scalarisation:
number of nodes involved in computing local invariant scalars.

$$\boldsymbol{s}_i^{(t+1)} := \sum_{j \in \mathcal{N}_i} \underline{f_1}\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{s}_j^{(t)}, d_{ij}, \sum_{k \in \mathcal{N}_j \setminus \{i\}} \underline{f_2}\left(\boldsymbol{s}_j^{(t)}, \boldsymbol{s}_k^{(t)}, d_{ij}, d_{jk} \measuredangle ijk\right)\right)$$

Distinguish geometric graphs

Can you tell these two local neighbourhoods apart using the **unordered set of distances** and **angles**, only?[1]



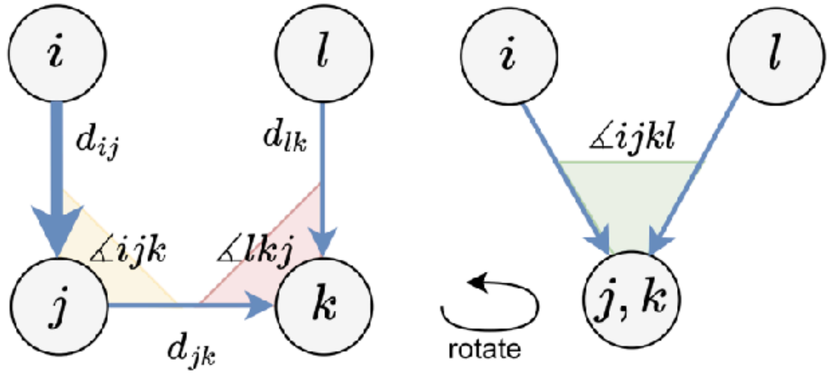
Relevant for local scalarisation in geometric GNNs — the ideal aggregator would distinguish all neighbourhoods.

[1] Pozdnyakov et al., Incompleteness of atomic structural representations, Phys. Rev. Letters, 2020.

Moving to higher body order terms

GemNet[1] also uses torsion angles among quadruplets

$$\mathbf{m}_{ijkl} = f(\mathbf{s}_l, \mathbf{s}_k, d_{ij}, d_{jk}, d_{lk}, \measuredangle ijk, \measuredangle lkj, \measuredangle ijkl)$$



$$\mathbf{m}_{ij} = f(\mathbf{s}_i, \mathbf{s}_j, d_{ij}, \sum_{k,d} \mathbf{m}_{ijkl})$$

Computing higher body order scalar quantities beyond pairwise interactions improves **expressiveness** and lead to more accurate but more **computationally expensive** models.

$$\boldsymbol{s}_i^{(t+1)} := \sum_{j \in \mathcal{N}_i} \underline{f_1}\left(\boldsymbol{s}_i^{(t)}, \boldsymbol{s}_j^{(t)}, d_{ij}, \sum_{\substack{k \in \mathcal{N}_j \setminus \{i\}, \\ l \in \mathcal{N}_k \setminus \{i,j\}}} \underline{f_2}(\boldsymbol{s}_k, \boldsymbol{s}_l, d_{kl}, d_{ij}, d_{jk}, \measuredangle ijk, \measuredangle jkl, \measuredangle ijkl) \right).$$

[1] Gasteiger et al., GemNet, 2022

[2] Wang et al., ComENet, 2022

Expressivity

GemNet is theoretically universal, but not its final architecture

GemNet requires a complete graph and a certain discretisation scheme to be universal.

So far, the precise body order of scalars at which all geometric graphs can be uniquely identified remains an open question.

Opinion

The GemNet paper includes a theoretical section, in which it is stated that GNNs with directed edge embeddings and two-hop message passing can universally approximate predictions that are invariant to translation, and equivariant to permutation and rotation. This statement needs careful reading. It is important to note that the universality claim requires conditions like an infinite cut-off (i.e. a fully connected graph) and appropriate discretization, as it builds upon a previous proof by Dym and Maron [2020] which showed that Tensor Field Networks are universal when operating on full graphs and using infinite tensor rank for equivariant features. As highlighted in the paper, the choice of discretization scheme can affect the universality of the approximation, and depending on the discretization scheme the resulting mesh might not provide a universal approximation guarantee. How to relax these two requirements and construct sufficient geometric conditions for universality is still an open research question and emphasized in Section 5.9 in [Gasteiger, 2023].

In particular, this means that while the theoretical model in the GemNet paper can be universal, the practical final architecture is not. The 4-body message passing in GemNet-Q sacrifices universality guarantees by operating on a discretization of representations in the directions of each atom's neighbours. Additionally, GemNet-T, the more efficient version of GemNet, performs 3-body message passing similar to DimeNet on radial cutoff graphs, which is not universal due to known counterexamples [Pozdnyakov et al., 2020]. The fact that the universality proof does not necessarily carry over to the final GemNet architecture was also emphasised by the authors of GemNet in this thread.

In summary, in the GemNet paper it is key to distinguish between the *theoretical model*, which can be universal, and the *final architecture*, which is not. We highlight this point here to avoid a misconception in the community that invariant architectures operating on distances, angles, and torsions angles are guaranteed to be universal or complete. Developing a universal geometric GNN in the general case, for sparse graphs and using finite tensor rank, remains an open question which we discuss in Section 8.

In a word...

Invariant GNNs pre-compute scalar quantities to capture geometric information, which is both a blessing & a curse:

- Simple usage of non-linearities on many-body scalars after pre-computation leads to great performance on some use-cases (e.g. GemNet on OC20)

- Rigidity and scalability of scalars pre-computation
- The accounting of higher-order tuples is expensive.
- Making invariant predictions may still require solving equivariant sub-tasks
- May lack generalisation capabilities (equivariant tasks, multi-domain)

Invariant GNNs constraint the geometric information that can be utilised, but not model operations

Equivariant Geometric GNNs

Equivariant Geometric GNNs

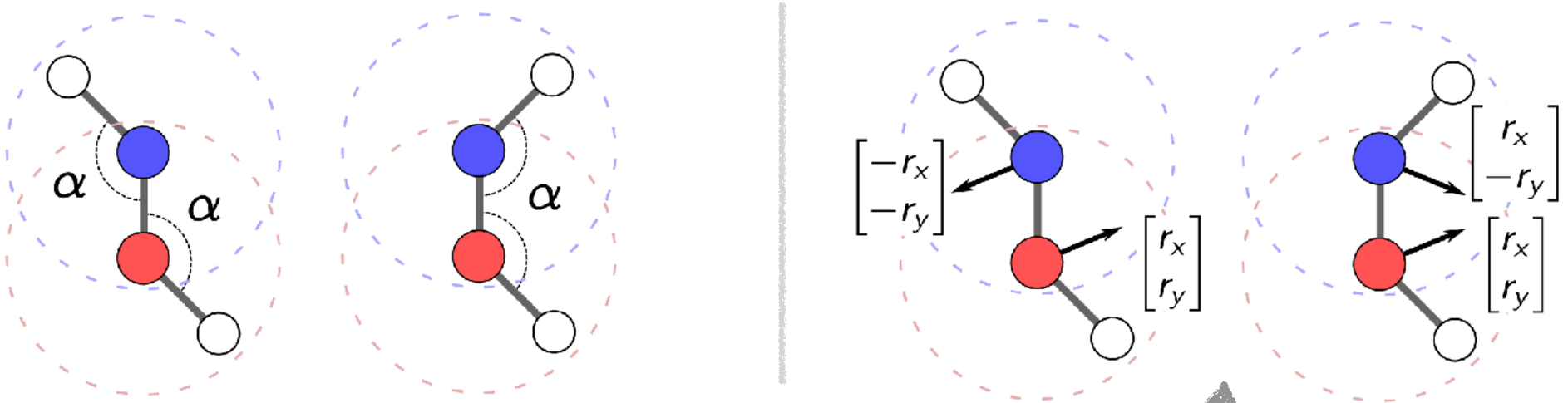
Why would you want to do that?

- Invariant GNNs **ensure** we will obtain invariant outputs by only working with **invariant features** within their layers.
  - —> Invariants are ‘fixed’ prior to message passing. Changing the number of message passing layers does not add new invariants. Also, ‘pre-computing’ invariants may be expensive.
- Equivariant GNNs build up invariants ‘on the go’ during message passing.
  - —> More layers of message passing can lead to more complex invariants being built up.
  - —> Furthermore, equivariant quantities remain available.

Motivation: **Discriminating geometric graphs**

What if all local neighbourhoods have identical invariant scalars?[1]

Pair of graphs cannot be discriminated using only scalars.

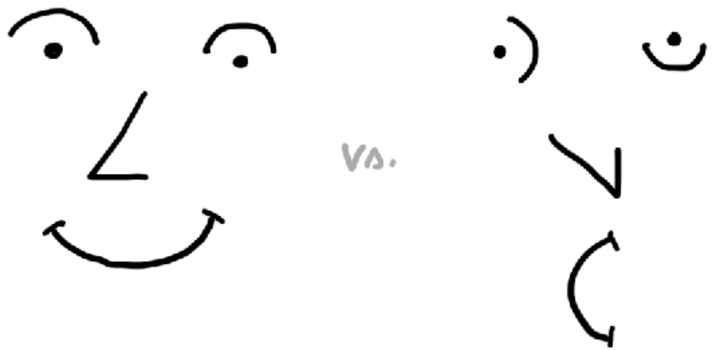


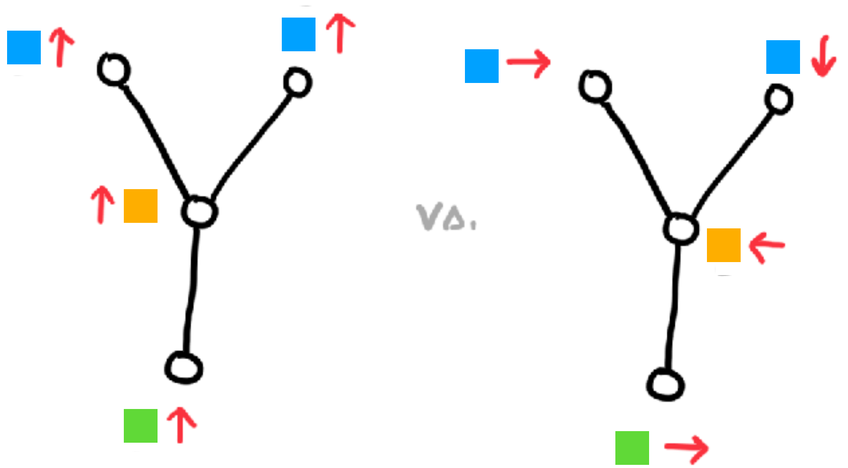
How to distinguish them?
Relative orientation of local neighbourhoods, *i.e.* geometric information.

[1] Schütt et al., PaiNN, ICML, 2021.

Intuition: **The Picasso Problem**

Making invariant predictions may still require solving equivariant sub-tasks





Relative orientation of eyes, nose, mouth is important
(**orientation of sub-graphs w.r.t. one another**),
not just their presence (**invariant quantities**)!

Equivariant Geometric GNNs

How do they look?

$$f(s_1, s_2, ..., s_n)$$

Invariant GNNs only operate on scalars. In internal message-passing, they only pass around scalar objects.

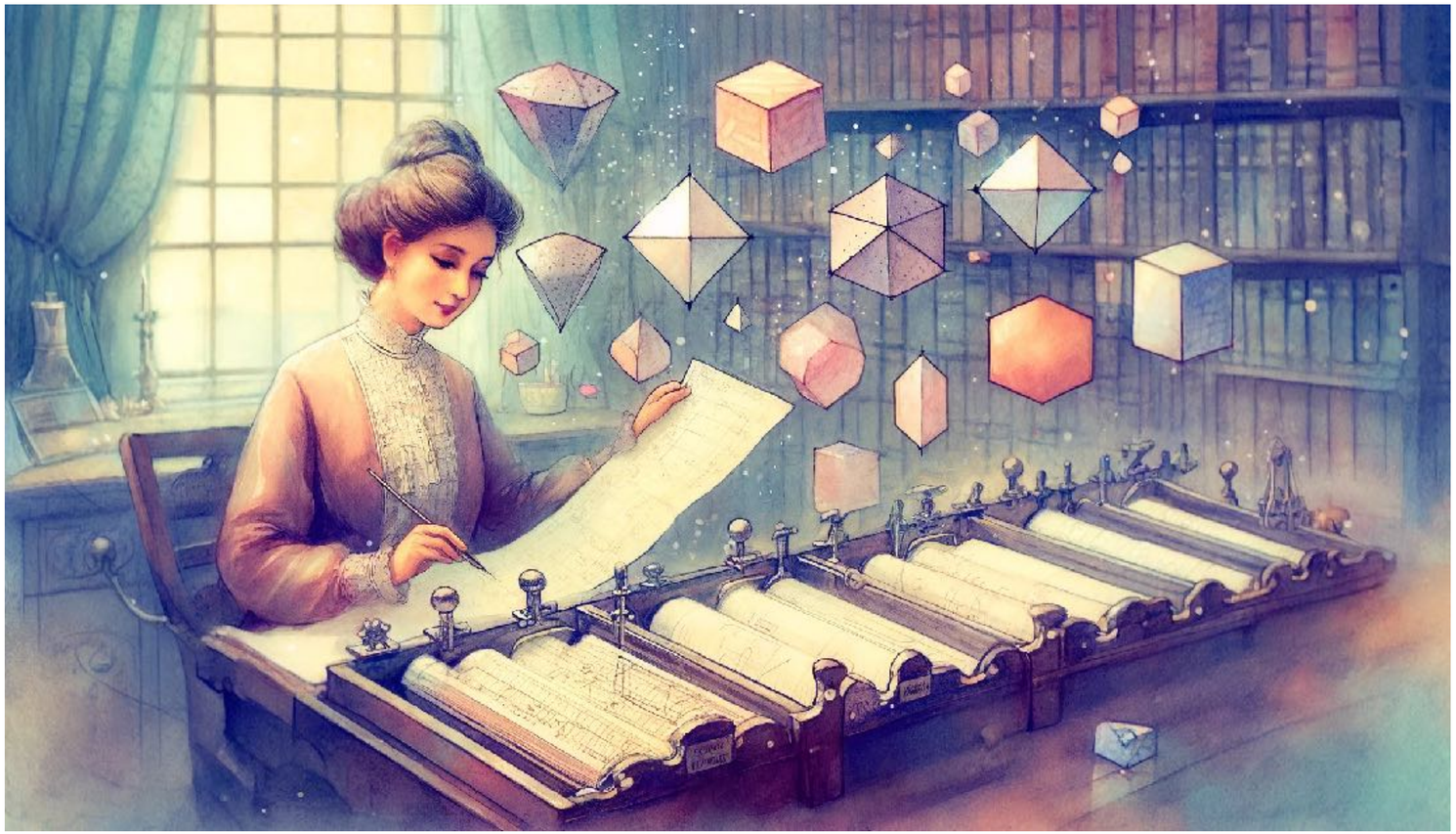
$$f(s_1, s_2, ...s_{n'}, \vec{v}_1, ..., \vec{v}_{m'})$$

Equivariant GNNs “keep” vectors (and other geometric objects) around. In internal message-passing, they pass around geometric objects.

If we predict an invariant quantity, we project the geometric information to scalars only at the end of all message-passing layers.

Equivariant Geometric GNNs

Key idea: Keep track of how different features “transform”.



An accountant of geometry.
Figure courtesy Dall-E.

$$f(s_1, s_2, ...s_{n'}, \vec{v}_1, ..., \vec{v}_{m'})$$

Because we keep around a mixed bag of information in which different objects transform differently (e.g. scalar vs vectors) we need keep track of what transforms how.

We need to become accountants of geometry.

Accounting rules for geometric objects

- **Type**: Each feature is associated with a type that tells us how it transforms under rotations and reflections:
  - e.g. scalar, vector, pseudo vector, …
- **Addition**: How do we add features of given types? What type do we get out?
- **Multiplication**: How do we multiply features of given types? What type do we get out?
- **Non-linear operations**: How

Example: “Scalar-vector” GNNs

We restrict ourselves to working with scalars and vectors only.

Scalar message

$$\mathbf{m}_i := \underline{f_1}(\mathbf{s}_i, \|\vec{\mathbf{v}}_\mathbf{i}\|) + \sum_{j \in \mathcal{N}_i} \underline{f_2}\left(\mathbf{s}_i, \mathbf{s}_j, \|\vec{x}_{ij}\|, \|\vec{\boldsymbol{v}}_j\|, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_j, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_i, \vec{\mathbf{v}}_i \cdot \vec{\mathbf{v}}_j\right)$$

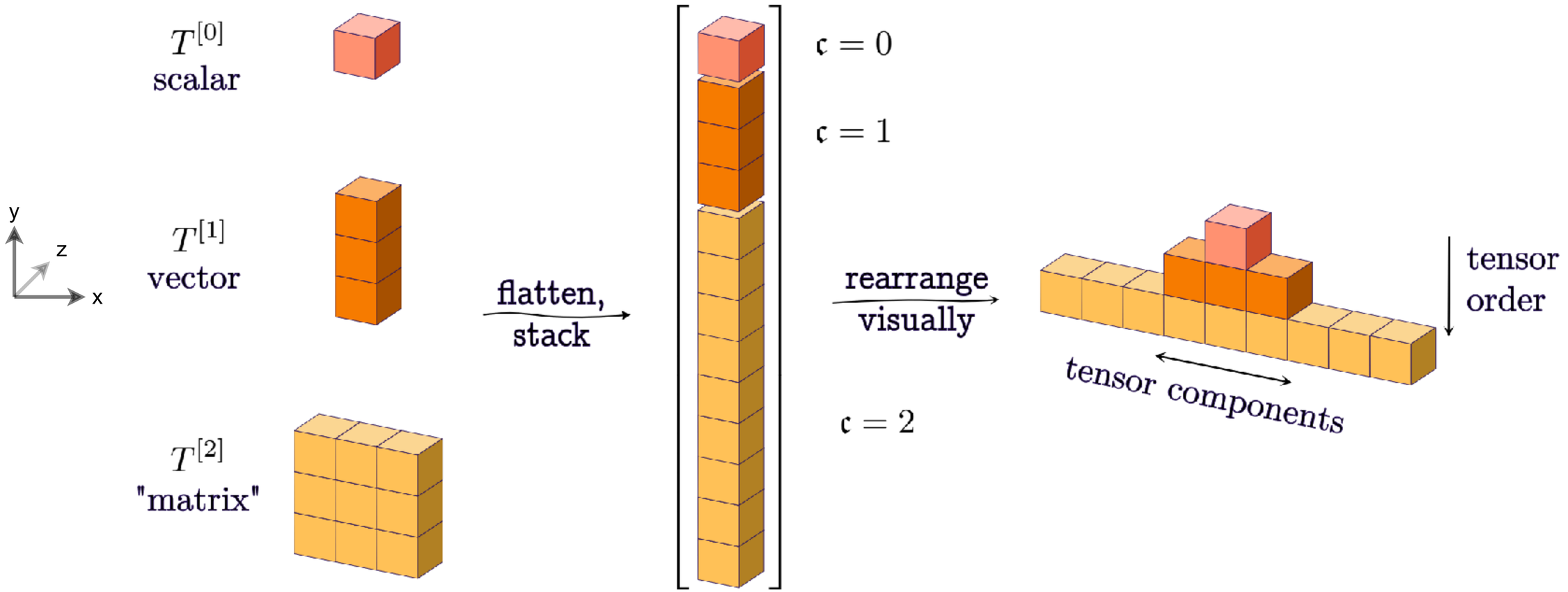
Vector message

$$\begin{aligned}\vec{\mathbf{m}}_i := \quad & \underline{f_3}(\mathbf{s}_i, \|\vec{\mathbf{v}}_\mathbf{i}\|) \odot \vec{\mathbf{v}}_i + \sum_{j \in \mathcal{N}_i} \underline{f_4}\left(\mathbf{s}_i, \mathbf{s}_j, \|\vec{x}_{ij}\|, \|\vec{\boldsymbol{v}}_j\|, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_j, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_i, \vec{\mathbf{v}}_i \cdot \vec{\mathbf{v}}_j\right) \odot \vec{\mathbf{v}}_j \\ & + \sum_{j \in \mathcal{N}_i} \underline{f_5}\left(\mathbf{s}_i, \mathbf{s}_j, \|\vec{x}_{ij}\|, \|\vec{\boldsymbol{v}}_j\|, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_j, \vec{x}_{ij} \cdot \vec{\mathbf{v}}_i, \vec{\mathbf{v}}_i \cdot \vec{\mathbf{v}}_j\right) \odot \vec{x}_{ij},\end{aligned}$$

What other geometric “types” are there?

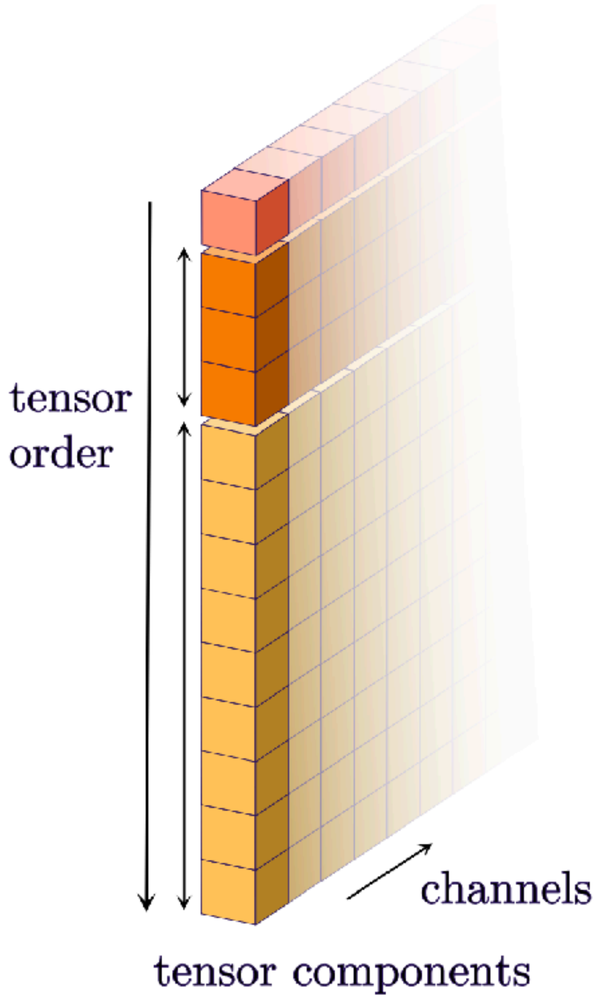
What are tensors-type features?

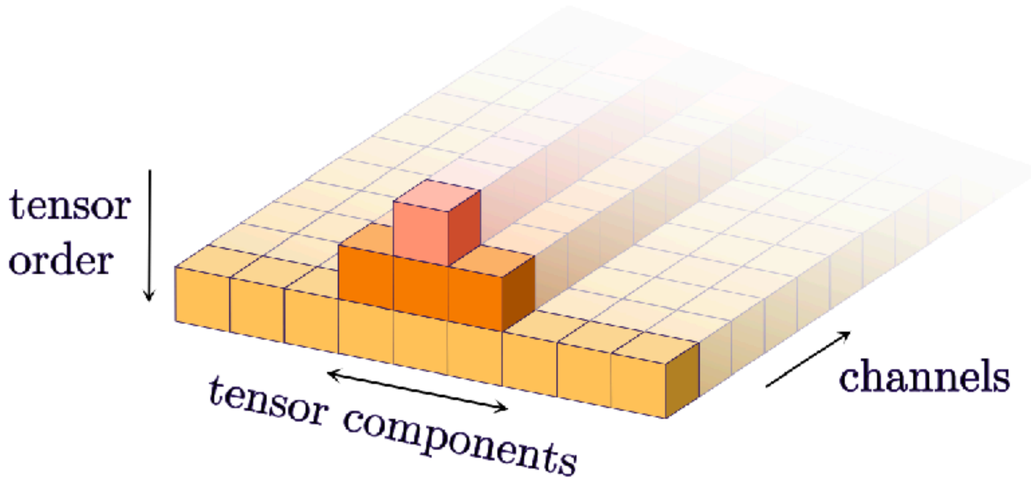
Example: Cartesian tensors



What are tensors-type features?

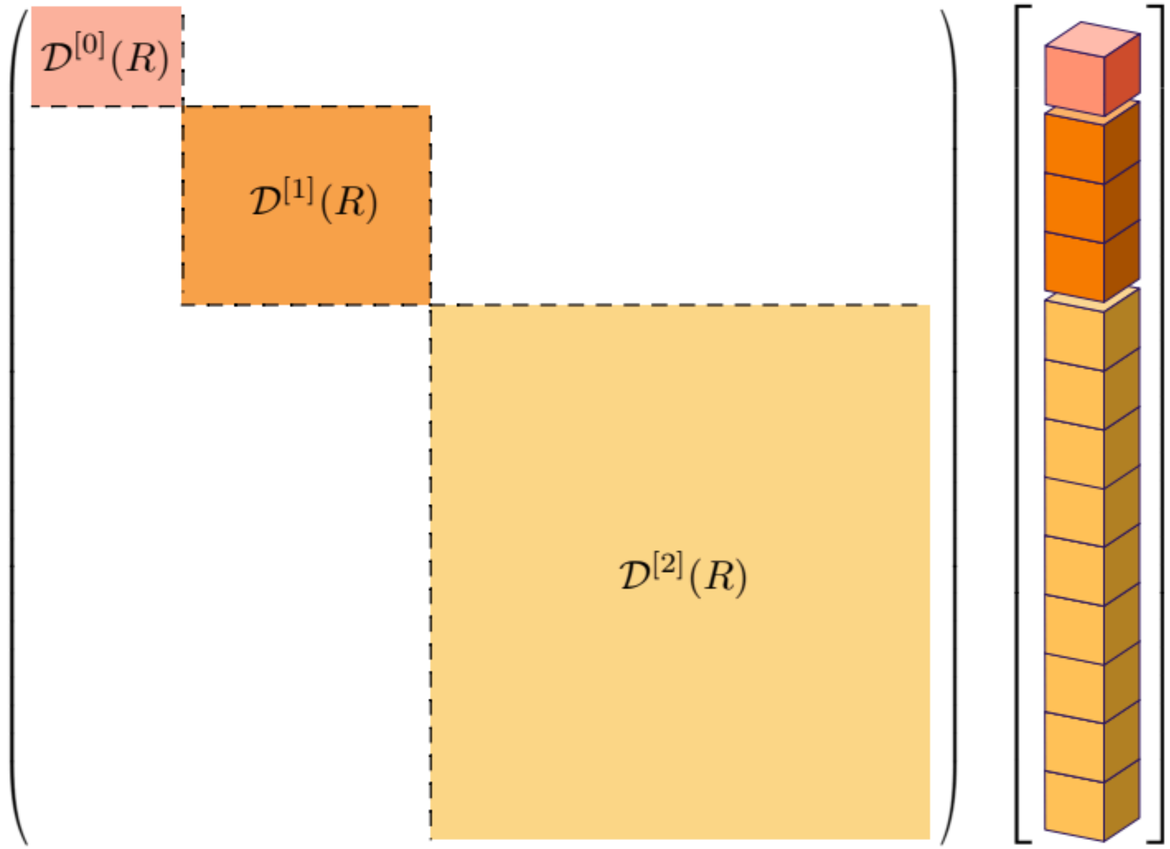
Example: Cartesian tensors

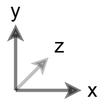




What are tensors-type features?

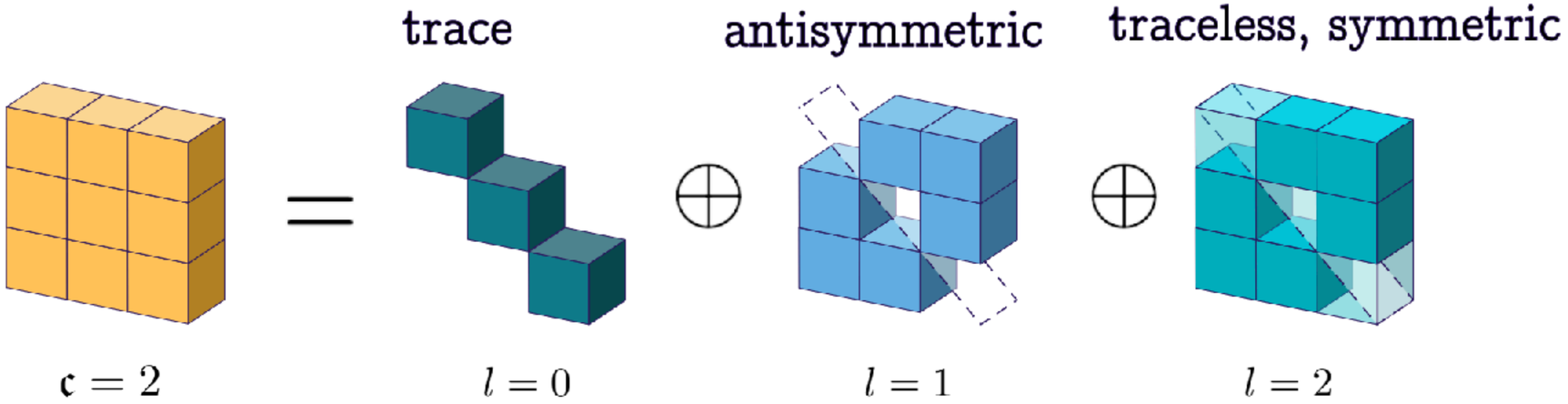
Example: Cartesian tensors

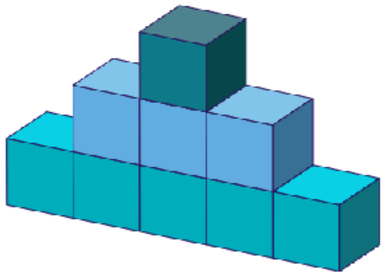




From Cartesian to Spherical tensors

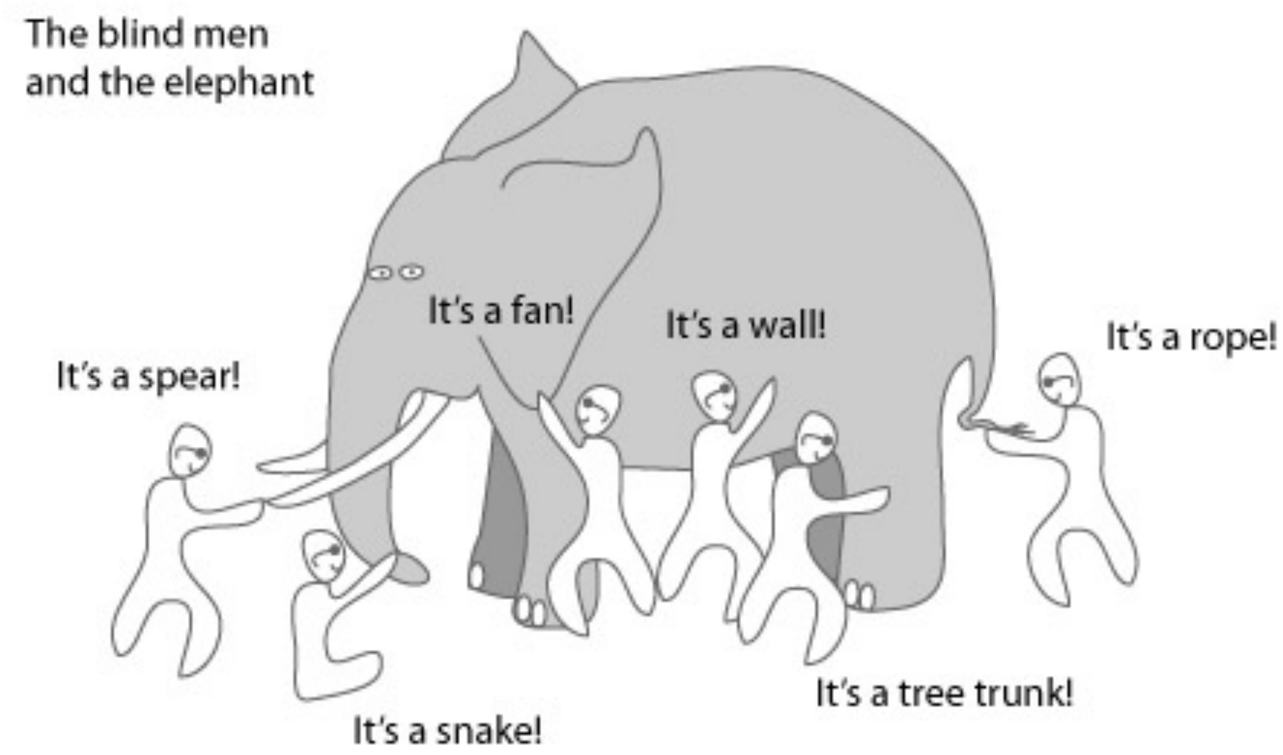
Example: rank-2 Cartesian tensor into Spherical tensors





Why spherical harmonics are confusing?

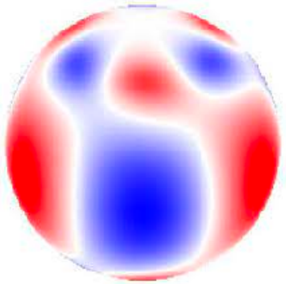
Metaphor of the blind men and the elephant.

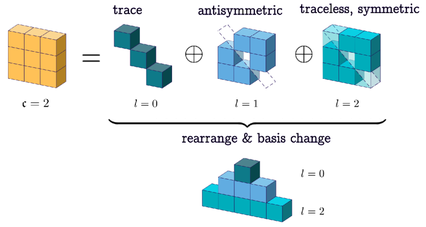


Why spherical harmonics are confusing?

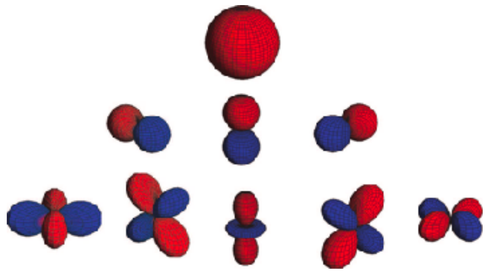
Similarly there are many different perspectives on the spherical harmonics.

Calculus:
Spherical harmonics are a complete, orthogonal basis for functions on the sphere in 3D. We can therefore do "harmonic analysis" on the sphere.





Representation theory:
Spherical harmonics are a basis of the irreducible representations of SO(3). We can decompose any representation of SO(3) into its irreducible components.



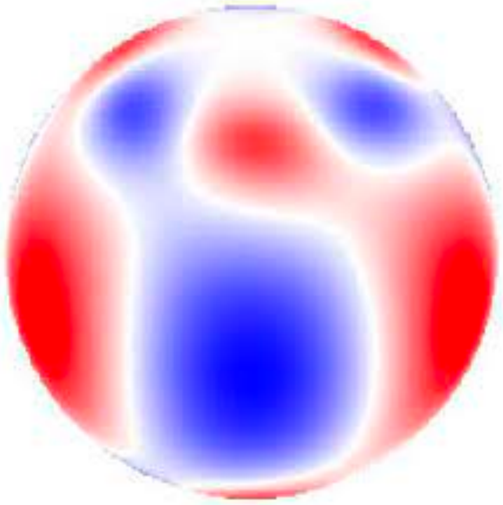
$$1, \quad x, \quad y, \quad xy, \quad x^2 - y^2$$

Physics:
Spherical harmonics are related to the shapes of the orbitals of the hydrogen atom.

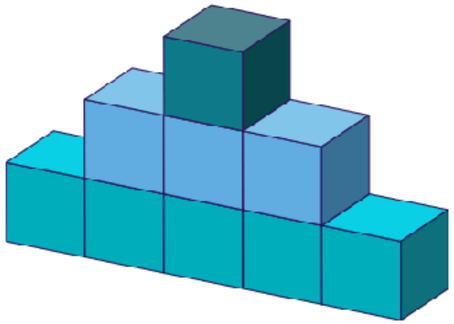
Algebra:
Spherical harmonics are instances of harmonic polynomials, which are eigenfunctions of the Laplace operator and a subclass of symmetric Polynomials.

From Cartesian to Spherical tensors

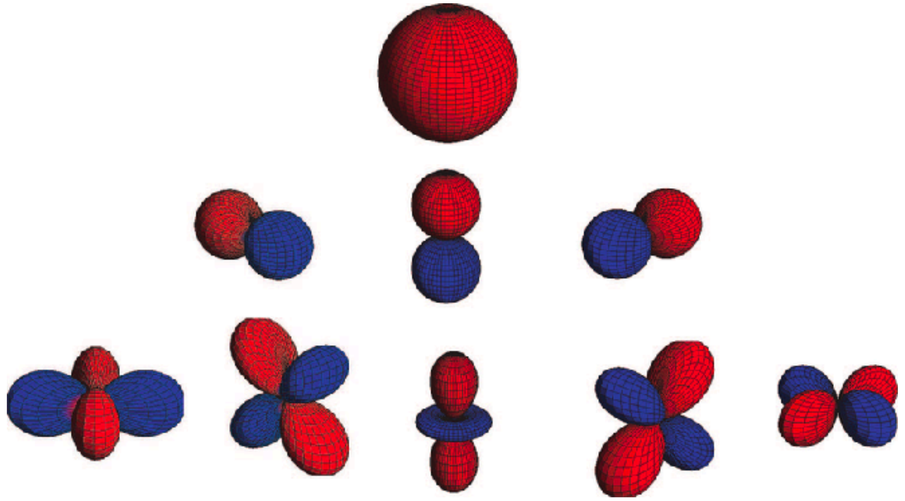
Spherical harmonics as the basis for spherical tensors



Function on a sphere



…as a spherical tensor



…and the underlying basis

Spherical harmonics are the *irreducible representations* of functions on a sphere, kind of like the simplest LEGO blocks into which all other functions can be decomposed.

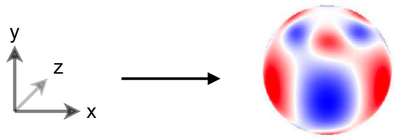
Tensor Field Networks [1]

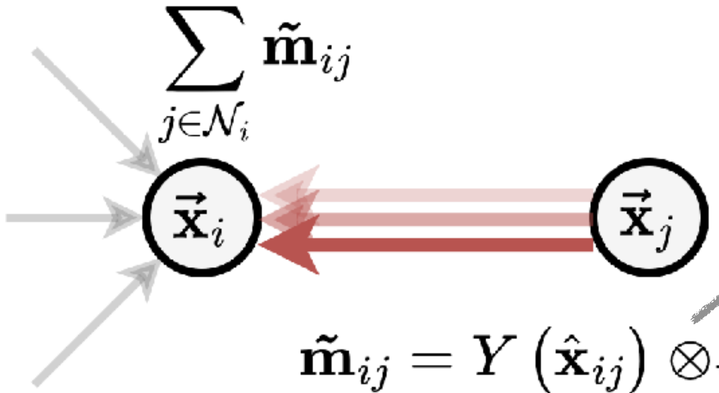
- **Higher order spherical tensors as node features** $\tilde{\boldsymbol{h}}_{i,l} \in \mathbb{R}^{2l+1 \times f},\ l = 0, \ldots, L$
- **…updated via tensor products** $\otimes_w$ **of neighbourhood features**
- **…with spherical harmonic expansion of displacement** $Y_l\left(\hat{\boldsymbol{x}}_{ij}\right) \in \mathbb{R}^{2l+1}$

Connection with cartesian basis:

$$\tilde{\boldsymbol{h}}_{i,l=0} \in \mathbb{R}^{1 \times f} \equiv \boldsymbol{s}_i$$

$$\tilde{\boldsymbol{h}}_{i,l=1} \in \mathbb{R}^{3 \times f} \equiv \vec{\boldsymbol{v}}_i$$

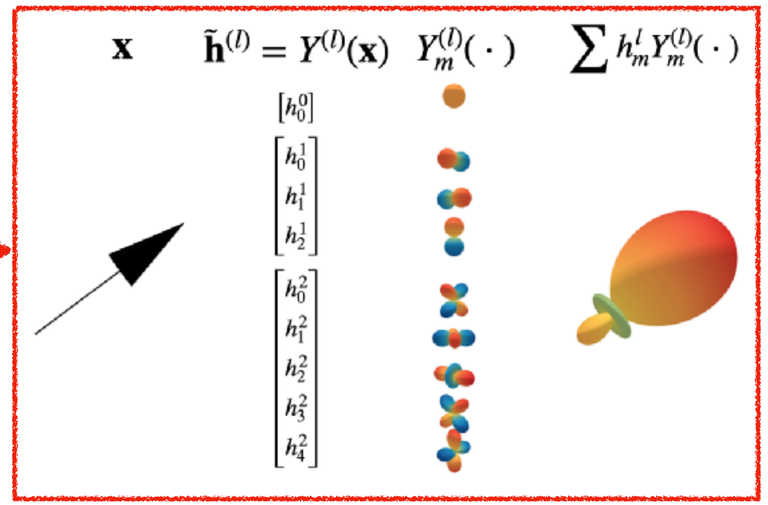




$$\tilde{\mathbf{m}}_{ij} = Y\left(\hat{\mathbf{x}}_{ij}\right) \otimes_{\mathbf{w}} \tilde{\mathbf{h}}_j^{(t)}$$

Tensor product weights:

$$\boldsymbol{w} = f_{\mathrm{RBF}}\left(\boldsymbol{s}_j, \|\vec{\boldsymbol{x}}_{ij}\|\right)$$

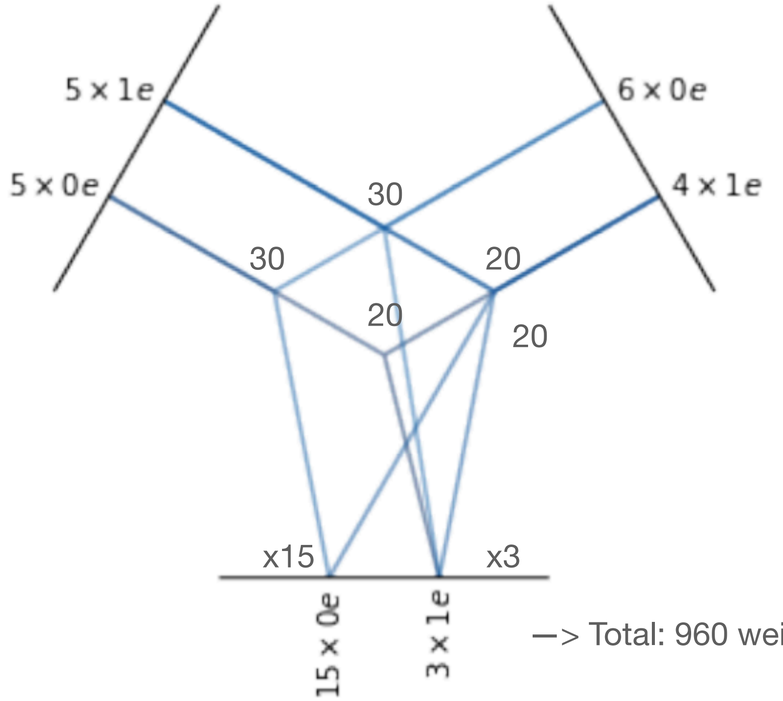


$$\tilde{\boldsymbol{h}}_i^{(t+1)} := \tilde{\boldsymbol{h}}_i^{(t)} + \sum_{j \in \mathcal{N}_i} Y\left(\hat{\boldsymbol{x}}_{ij}\right) \otimes_{\boldsymbol{w}} \tilde{\boldsymbol{h}}_j^{(t)},$$

[1] Thomas, Smidt, et al., TFN, 2018. [2] Brandstetter, Hesselink, et al., SEGNN, ICLR,2022. [3] Batatia et al., MACE, NeurIPS, 2022. [4] Geiger and Smidt, e3nn, 2022.

Parameterising the tensor product

$$\vec{S}^{(0:l_{\max})} \underline{\otimes} \vec{T}^{(0:l_{\max})} = \bigoplus_{l_3=0}^{l_{\max}} \left(\sum_{\text{Paths}(l_1,l_2 \to l_3)} \underline{w}_{l_1,l_2,l_3} \, \vec{S}^{(l_1)} \otimes \vec{T}^{(l_2)} \big|_{(l_3)} \right)$$



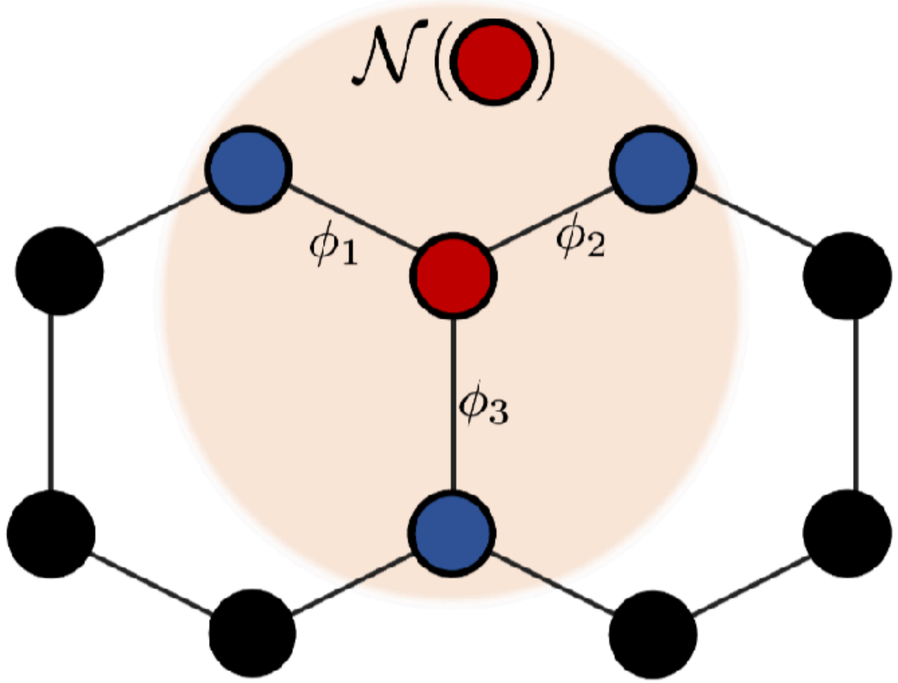
—> Total: 960 weights

$$\begin{bmatrix} \vec{Y}^{(0)} \\ \vec{Y}^{(1)} \end{bmatrix} \underline{\otimes} \begin{bmatrix} \vec{H}^{(0)} \\ \vec{H}^{(1)} \end{bmatrix} = \begin{bmatrix} \underline{w}_{0,0,0}(\vec{Y}^{(0)} \otimes \vec{H}^{(0)})\big|_{(0)} + \underline{w}_{1,1,0}(\vec{Y}^{(1)} \otimes \vec{H}^{(1)})\big|_{(0)} \\ \underline{w}_{0,1,1}(\vec{Y}^{(0)} \otimes \vec{H}^{(1)})\big|_{(1)} + \underline{w}_{1,0,1}(\vec{Y}^{(1)} \otimes \vec{H}^{(0)})\big|_{(1)} + \underline{w}_{1,1,1}(\vec{Y}^{(1)} \otimes \vec{H}^{(1)})\big|_{(1)} \end{bmatrix}$$

Figure credit: e3nn.org

MACE - Multi-Atomic Cluster Expansion [1]

Take tensor products of the aggregated message with itself to obtain **many-body features:**



$$A_{\bullet} = \sum_{j \in \mathcal{N}(\bullet)} \phi_j = (\phi_1 + \phi_2 + \phi_3)$$

$$\mathbf{B}_{\bullet} = (\phi_1 + \phi_2 + \phi_3)^2 = \phi_1^2 + \phi_2^2 + \phi_3^2 + \underbrace{2\phi_1\phi_2 + 2\phi_1\phi_3 + 2\phi_2\phi_3}_{\text{implicit 3-body terms}}$$

Figure credit:
Harry Shaw

[1] Batatia et al., MACE, NeurIPS, 2022.

In a word...

Equivariant GNNs allow the network to learn its set of invariants instead of pre-computing it, solving equivariant sub-tasks

- **Cartesian EGNNs** model atomic interactions in Cartesian coordinates and restrict the set of possible operations on geometric features to preserve equivariance. They update (and combine) both scalar and vector representations.
- **Spherical EGNNs** use spherical tensor components, which correspond to the irreducible representations of SO(3), as their feature type scalar and vector messages in parallel.

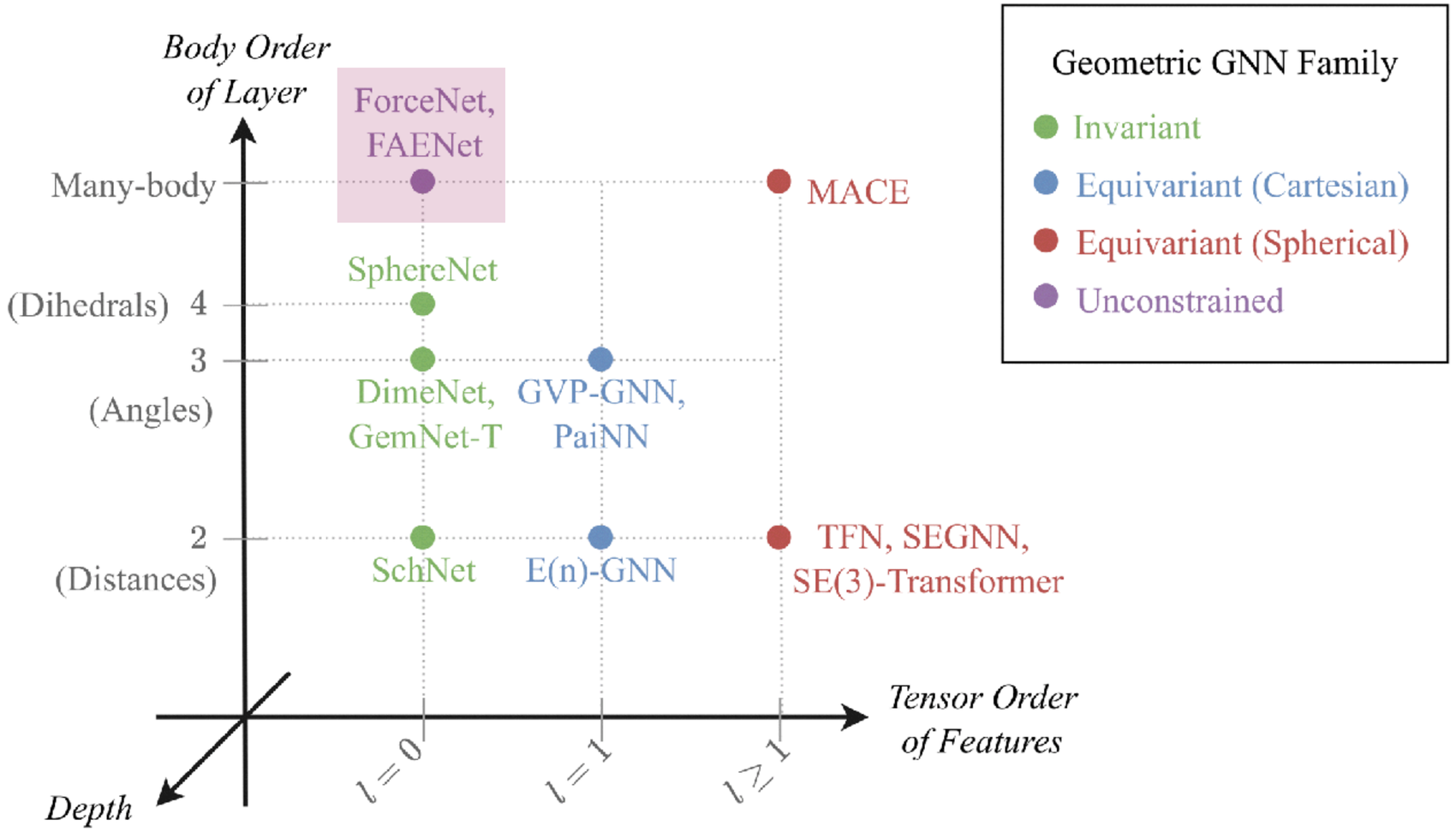
Equivariant GNNs perform diligent accounting of how each hidden feature in each layer has to transform to remain equivariant

Unconstrained GNNs

𝔊-Unconstrained GNNs

Do not "bake" symmetries into the model design, unlike previous methods

Different **canonicalisation** strategies to deal with these inductive biases



Motivation

Have greater model flexibility and traverse more diverse optimisation paths

Limitations of enforcing symmetries directly into the architecture:

- May overly regularise the model, restricting its set of possible operations and hindering its capacity to fully express the intricacies of the data
- Render its functioning complex and computationally expensive

Does enforcing Euclidean equivariance as an inductive bias truly offset a potential reduction in optimization diversity within constrained learning spaces ?

[1] A. Duval et al., FAENet, ICML, 2023. [2] Puny et al., Frame Averaging, ICML, 2022.

FAENet[1]

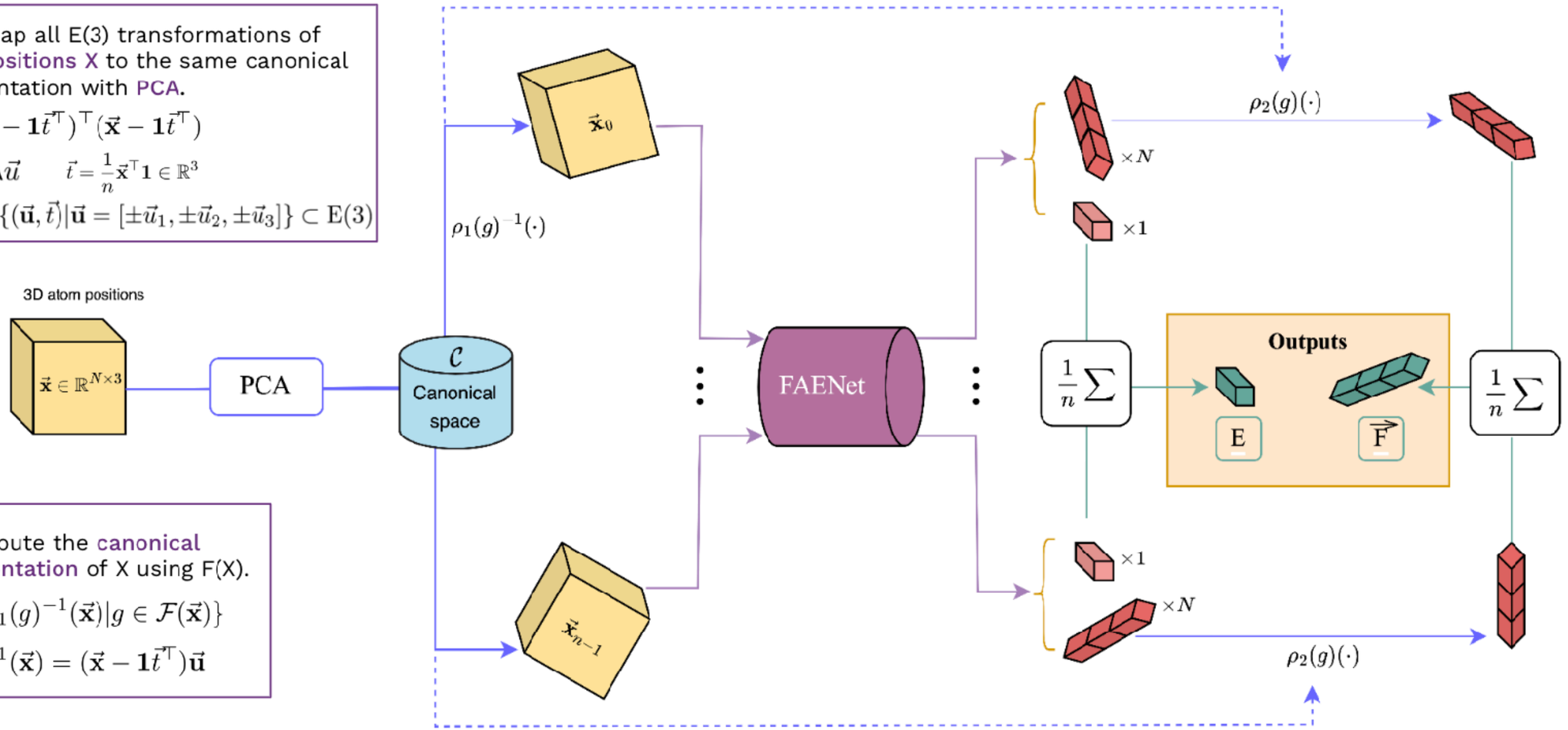
Outsource equivariance to the data using Frame Averaging[2]

1. We map all E(3) transformations of atom positions X to the same canonical representation with PCA.

$$\Sigma = (\vec{\mathbf{x}} - \mathbf{1}\vec{t}^{\top})^{\top}(\vec{\mathbf{x}} - \mathbf{1}\vec{t}^{\top})$$

$$\Sigma\vec{u} = \lambda\vec{u} \qquad \vec{t} = \frac{1}{n}\vec{\mathbf{x}}^{\top}\mathbf{1} \in \mathbb{R}^3$$

$$\mathcal{F}(\vec{\mathbf{x}}) = \{(\vec{\mathbf{u}}, \vec{t}) | \vec{\mathbf{u}} = [\pm\vec{u}_1, \pm\vec{u}_2, \pm\vec{u}_3]\} \subset \mathrm{E}(3)$$



2. Compute the canonical representation of X using F(X).

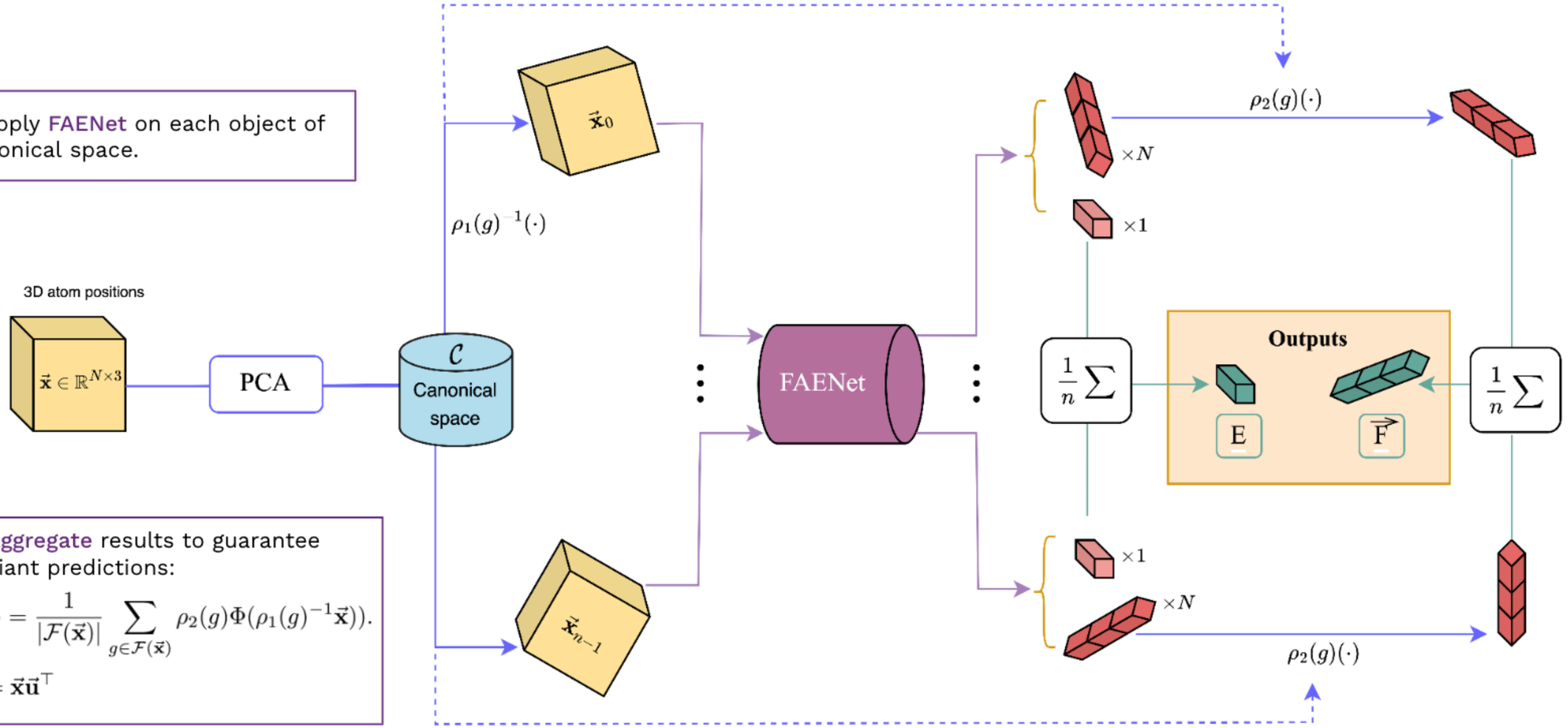
$$\mathcal{C} = \{\rho_1(g)^{-1}(\vec{\mathbf{x}}) | g \in \mathcal{F}(\vec{\mathbf{x}})\}$$

$$\rho_1(g)^{-1}(\vec{\mathbf{x}}) = (\vec{\mathbf{x}} - \mathbf{1}\vec{t}^{\top})\vec{\mathbf{u}}$$

FAENet

Equivariant GNN without any symmetry-preserving architectural constraints

3. We apply FAENet on each object of the canonical space.



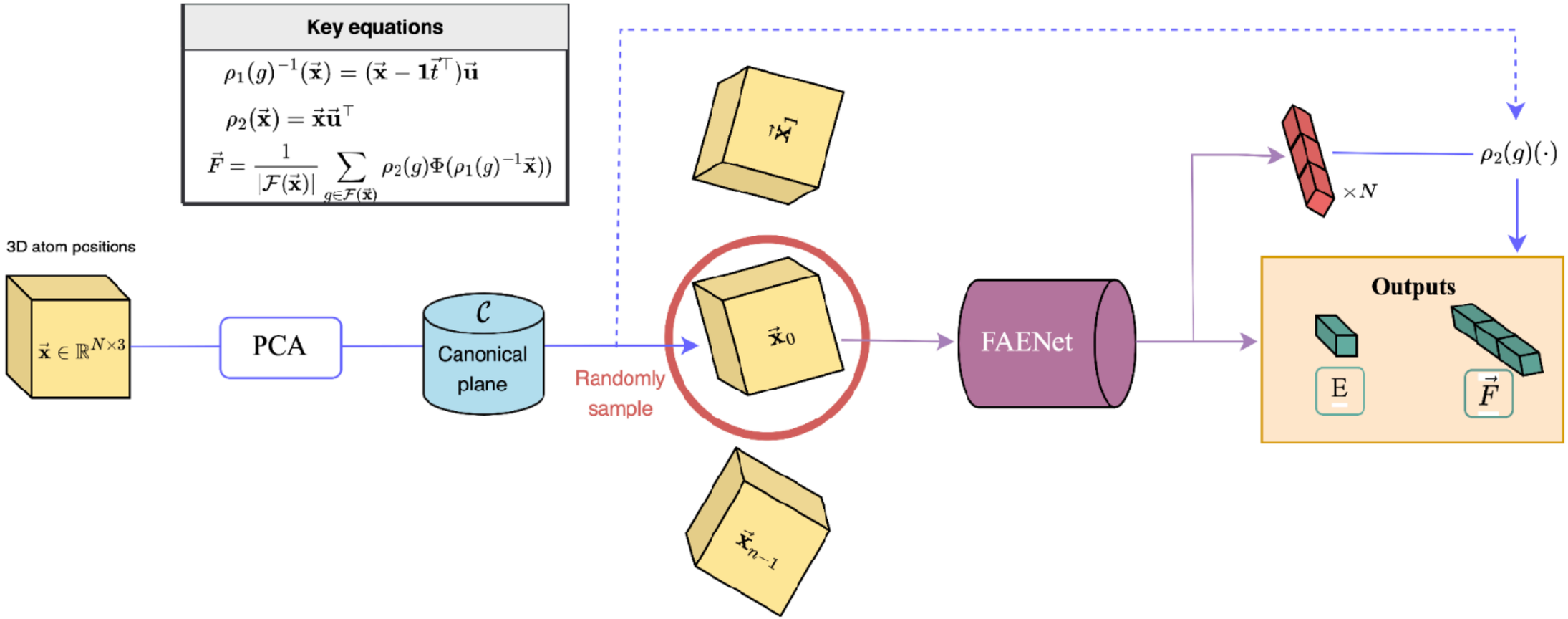
4. We aggregate results to guarantee equivariant predictions:

$$\langle \Phi \rangle_{\mathcal{F}}(\vec{\mathbf{x}}) = \frac{1}{|\mathcal{F}(\vec{\mathbf{x}})|} \sum_{g \in \mathcal{F}(\vec{\mathbf{x}})} \rho_2(g)\Phi(\rho_1(g)^{-1}\vec{\mathbf{x}})).$$

$$\rho_2(\vec{\mathbf{x}}) = \vec{\mathbf{x}}\vec{\mathbf{u}}^{\top}$$

FAENet

Stochastic Frame Averaging offers a good & fast approximation of equivariance



FAENet architecture

Can process geometric information without any design restrictions

1. Since we have a way to **outsource equivariance**, the model doesn't need to enforce it itself.

2. We can create a **light-weight** model whose capacity is only dedicated to predicting properties

3. We exploit **relative position vectors** directly, applying any non-linearity on them instead of relying on invariant / equivariant proxies.

$$\mathbf{e}_{ij} = \sigma\Big(\mathrm{MLP}\big(\vec{x}_{ij}||\mathrm{RBF}(d_{ij})\big)\Big).$$

$$\mathbf{f}_{ij}^{(l)} = \sigma(\mathrm{MLP}(\mathbf{e}_{ij}||\mathbf{s}_i^{(l)}||\mathbf{s}_j^{(l)}))$$

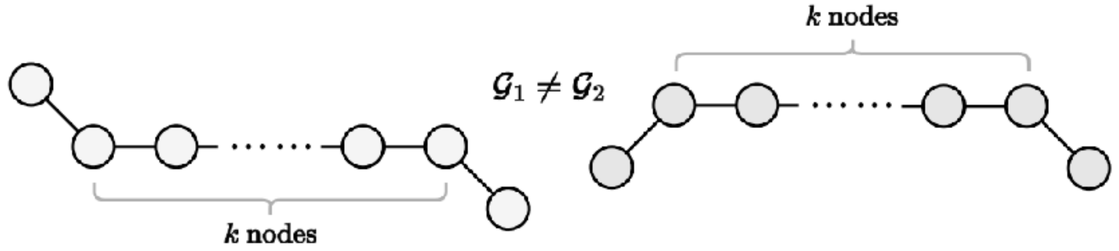
$$\mathbf{s}_i^{(l+1)} = \mathbf{s}_i^{(l)} + \mathrm{MLP}\left(\sum_{j\in\mathcal{N}_i} \mathbf{s}_j^{(l)} \odot \mathbf{f}_{ij}^{(l)}\right)$$

https://arxiv.org/pdf/2305.05577.pdf

FAENet expressive power

Distinguishes easily between any two graphs

Synthetic experiments by (Joshi, Mathis et al., 2023) to analyse the expressive power of geometric GNNs



	($k = 4$-chains)	Number of layers				
	GNN Layer	1	$\lfloor \frac{k}{2} \rfloor = 2$	$\lfloor \frac{k}{2} \rfloor + 1 = \mathbf{3}$	$\lfloor \frac{k}{2} \rfloor + 2$	$\lfloor \frac{k}{2} \rfloor + 3$
Inv.	IGWL	50%	50%	50%	50%	50%
	SchNet	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00
	DimeNet	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00	50.0 ± 0.00
Equiv.	GWL	50%	50%	**100%**	**100%**	**100%**
	E-GNN	50.0 ± 0.00	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	GVP-GNN	50.0 ± 0.00	50.0 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	TFN	50.0 ± 0.00	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	80.0 ± 24.5
	MACE	50.0 ± 0.00	50.0 ± 0.0	90.0 ± 20.0	90.0 ± 20.0	95.0 ± 15.0
	FAENet	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	FAENet-SFA	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**

Boom of unconstrained approaches

Data augmentation, soft constraints, canonicalisation function & local frames

- ForceNet (Hu et al., 2021): data augmentation as soft symmetry constraint
- SCN (Zitnick et al, 2022) relaxes the equivariant constraints "to enable more expressive non-linear transformations"

- SignNet (Lim et al., 2024): tackle the sign ambiguity issue of PCA by utilising a sign-equivariant network, allowing to use only 1 frame (similarly to SFA).
- (Dym et al. 2024) propose to weight frames to preserve continuity
- (Kaba et al., 2023) learns a shallow equivariant network to perform canonicalisation
- (Arnab et al., 2024) align the canonicalisation function with training data distribution

- (Pozdnyakov et al., 2024): defines local coordinate systems at each atom and averages over the predictions of a non-equivariant network for system (alternative to FA).

In a word...

Unconstrained GNNs tackle the interplay between the function space that a model can learn and the ease of optimisation of ML algorithms

- Unconstrained Geometric GNNs are an emerging and under-explored line of work
- They enable the use of a wider range of networks for 3D atomic systems, including simple, fast & expressive models.

Open questions:

- Should we rigorously enforce symmetries or learn / approximate them ?
- Is local equivariance desirable, as opposed to global equivariance ?

Unconstrained GNNs use canonicalisation functions, soft constraints or local frames to enforce (or approximate) symmetries, instead of model design

Future Directions

Some future research directions

Let's discuss them

1. To what extent should physics and symmetry be ‘baked in’ to Geometric GNNs? Is rotational equivariance too strong a constraint on GNN expressivity?

2. How to construct geometric graphs? Coarse-graining and hierarchical structures? Dynamics and flexibility?

3. How to scale up Geometric GNNs? Data-Architecture-Hardware? Foundation Models ?

Some future research directions

To what extent should physics and symmetries be 'baked in' to Geometric GNNs ?

1. Enforcing symmetries

   Invariant vs Equivariant vs Unconstrained GNNs ?
   Local vs Global symmetries
   Data Efficiency & Generalisation vs greater expressivity & efficiency

2. Energy conservation

   Scalability vs Simulation stability
   Quantify its importance

3. Deeper theoretical characterisation

   Ability to solve the geometric graph isomorphism problem
   Benefits of higher-order tensors ?

Some future research directions

How should we construct geometric graphs ?

1. Graph creation

   Which graph to create ? (complete, local cutoff, long range connections)
   Avoid over-squashing
   Coarse-graining

2. Temporal dynamics & conformational flexibility

   Looking beyond static structures

Some future research directions

How to scale up Geometric GNNs ?

1. Foundation models

   Universal potentials, across the whole range of atomic systems
   Are interactions the same for small molecules and crystals ?
   The place of LLMs in the field ?

2. Large scale datasets and infrastructures

   Release of public datasets
   DFT ground truth
   Computing resources

3. Discovery

   How do our computational methods relate to experiments ?
   Can we benefit real-world applications ?

Thank you for attending!

Keen to connect — please send us your feedback

alexduvalinho.github.io/
@ADuvalinho

Website: chaitjo.com
Twitter: @chaitjo

simonmathis.com
@SimMat20