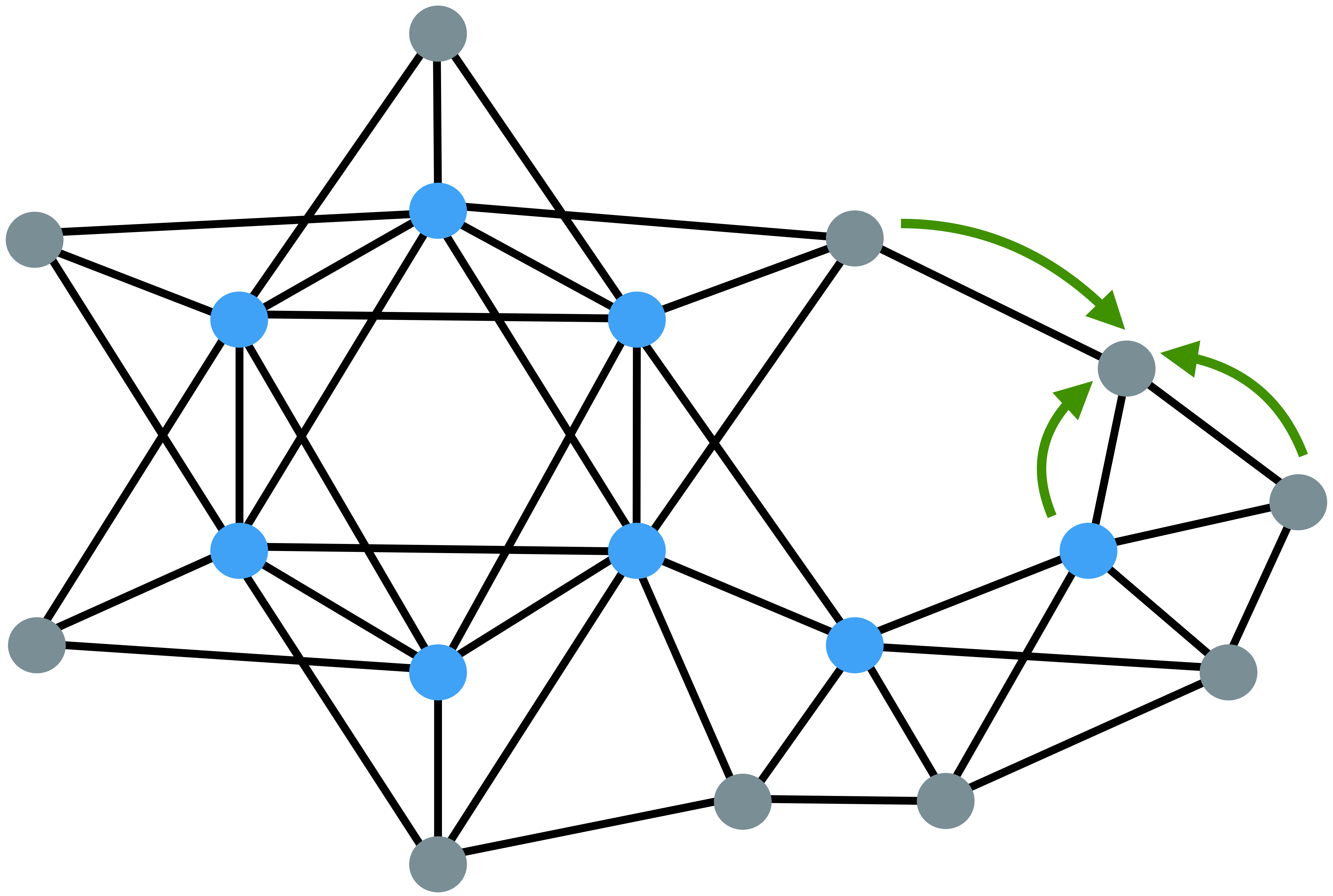
Practical Introduction to Neural Network Potentials Day 5:

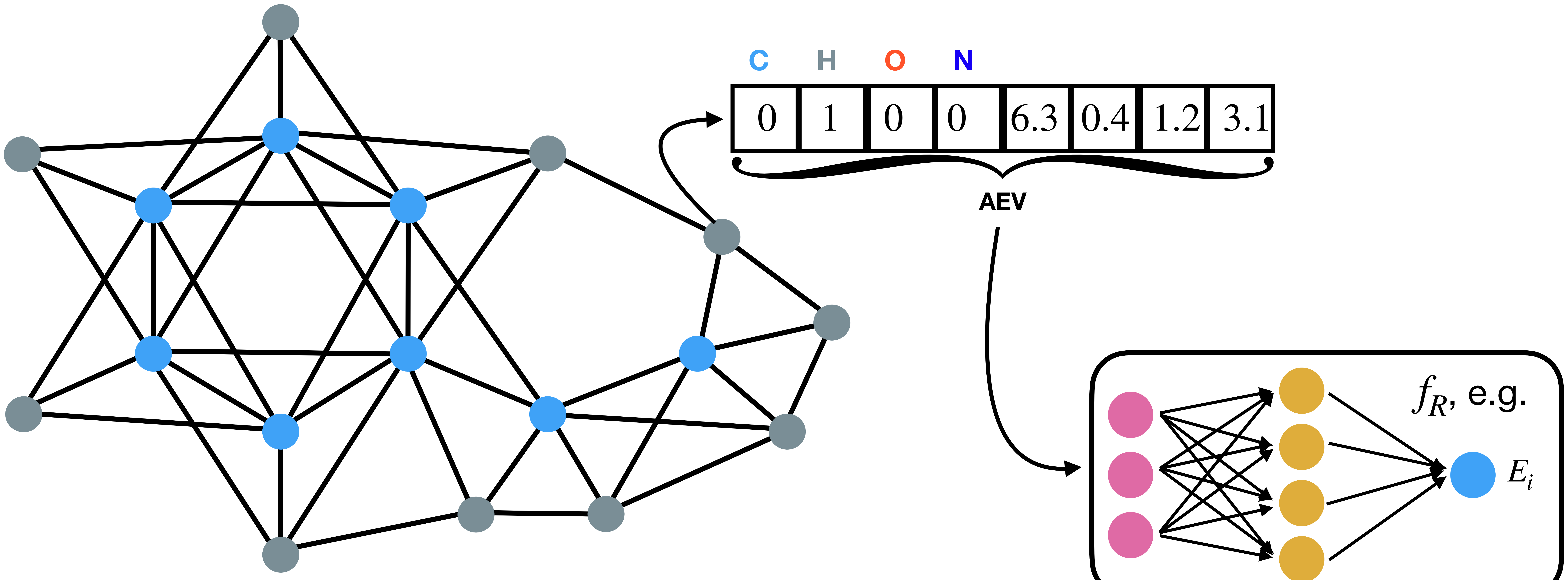
Equivariant message-passing neural networks

Review: Message-passing neural networks (MPNNs)



1. Initialize nodes $\overrightarrow{x}_i$ and edges $\overrightarrow{e}_{ij}$
2. Pass messages between all atoms $\overrightarrow{m}_{ij}$
3. Accumulate messages $\overrightarrow{M}_i$
4. Repeat
5. Readout

Review: Message-passing neural networks (MPNNs)



Review: Message-passing neural networks (MPNNs)

Open Access | Published: 09 January 2017

Quantum-chemical insights from deep tensor neural networks

Kris

Nat

49

Molecular graph convolutions: moving beyond fingerprints

Steven Kearnes[1]
Patrick Riley[2]

Convolutional Networks on Graphs for Learning Molecular Fingerprints

SchNet: A continuous-filter convolutional neural network for modeling quantum interactions

K. T. Schütt[1*], P.-J. Kindermans[1], H. E. Sauceda[2], S. Chmiela[1], A. Tkatchenko[3], K.-R. Müller[1,4,5†]
[1] Machine Learning Group, Technische Universität Berlin, Germany
[2] Theory Department, Fritz-Haber-Institut der Max-Planck-Gesellschaft, Berlin, Germany
[3] Physics and Materials Science Research Unit, University of Luxembourg, Luxembourg
[4] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[5] Dept. of Brain and Cognitive Engineering, Korea University, Seoul, South Korea
[*] kristof.schuett@tu-berlin.de [†] klaus-robert.mueller@tu-berlin.de

Neural Message Passing for Quantum Chemistry

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

Definition: Invariance

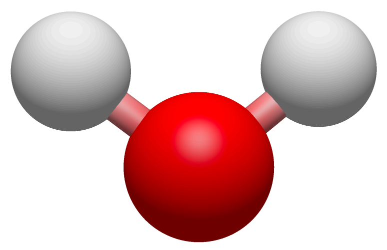
$$f : X \mapsto y$$

f is a function that maps a molecule (X) to a property (y)

Definition: Invariance

$$f : X \mapsto y$$

f is a function that maps a molecule (X) to a property (y)



$\mapsto$ **151.12 Har.**

Definition: Invariance

$$f : X \mapsto y$$

f is a function that maps a molecule (X) to a property (y)

$$\mathscr{T} : X \mapsto X'$$

$\mathscr{T}$ is a function that transforms a molecules (X) to some other molecule (X')

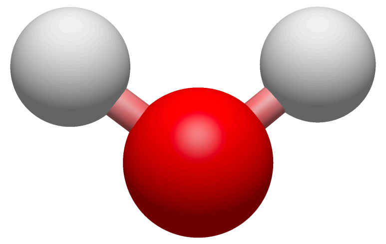
Definition: Invariance

$$f : X \mapsto y$$

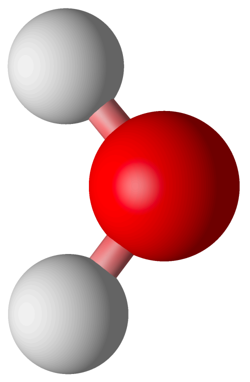
f is a function that maps a molecule (X) to a property (y)

$$\mathscr{T} : X \mapsto X'$$

$\mathscr{T}$ is a function that transforms a molecules (X) to some other molecule (X')



$\overset{\mathscr{T}}{\mapsto}$
(90° rotation)



Definition: Invariance

$$f : X \mapsto y$$

f is a function that maps a molecule (X) to a property (y)

$$\mathscr{T} : X \mapsto X'$$

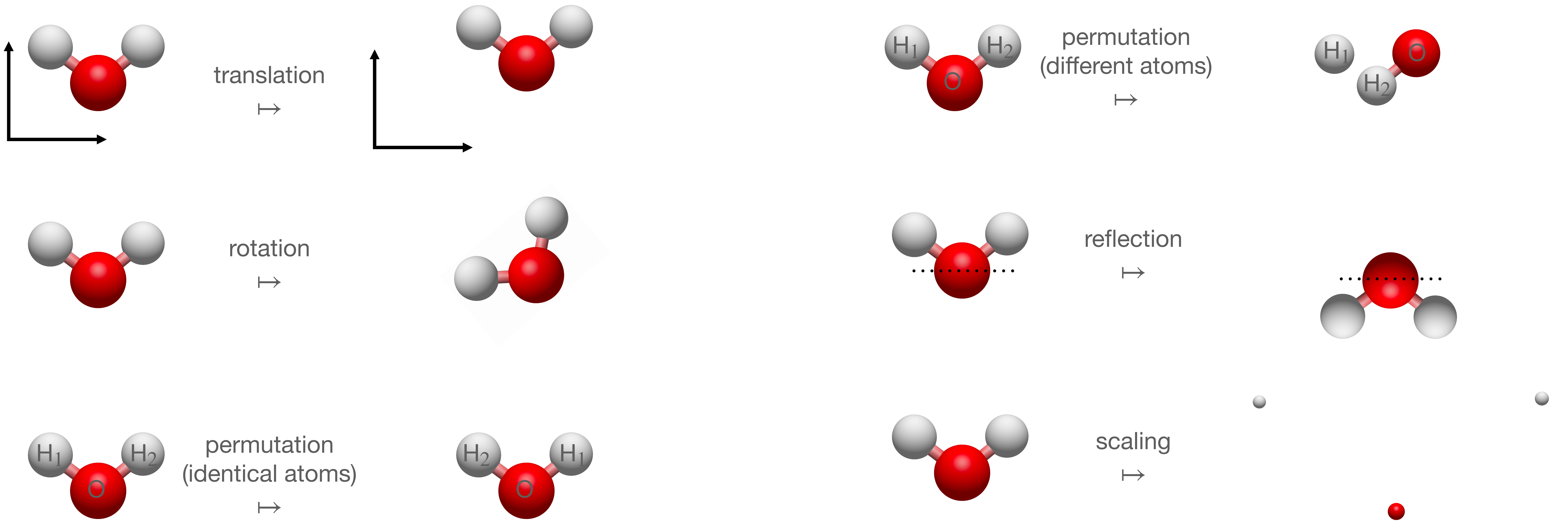
$\mathscr{T}$ is a function that transforms a molecules (X) to some other molecule (X')

f is **invariant** to $\mathscr{T}$ if the following is true for all X:

$$f(X) \equiv f(\mathscr{T}X)$$

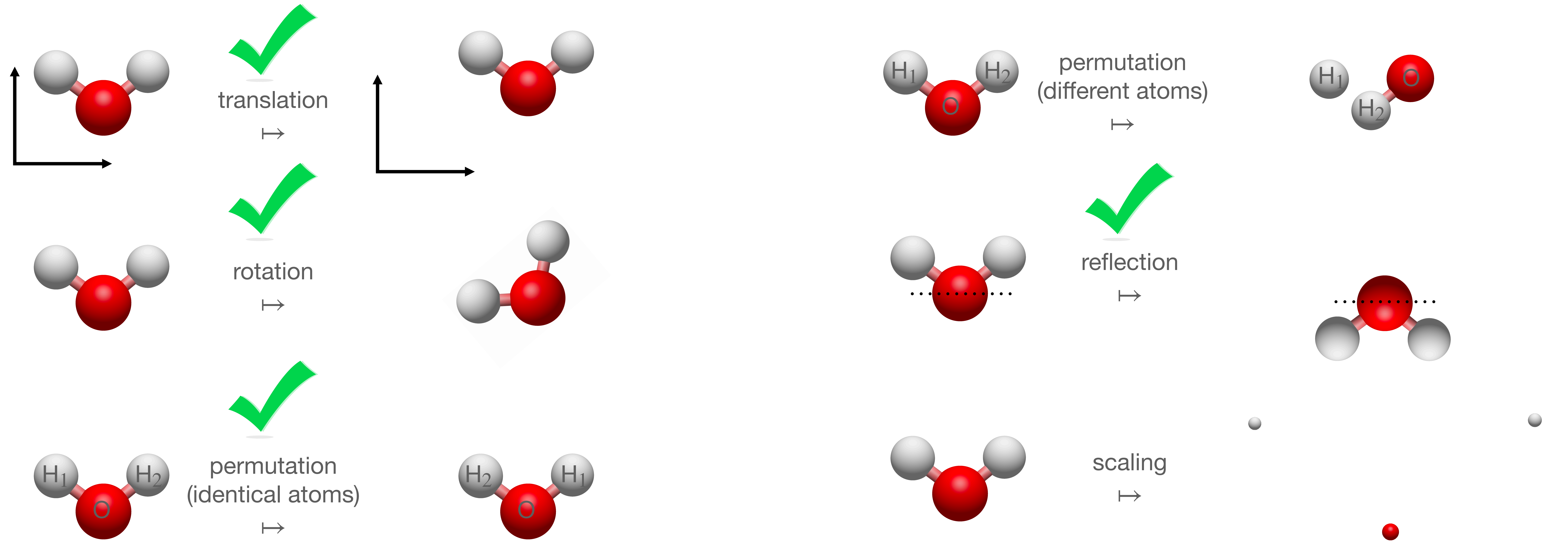
Examples of Invariance

The molecular energy (f) is invariant to which of the following transformations $(\mathscr{T})$?



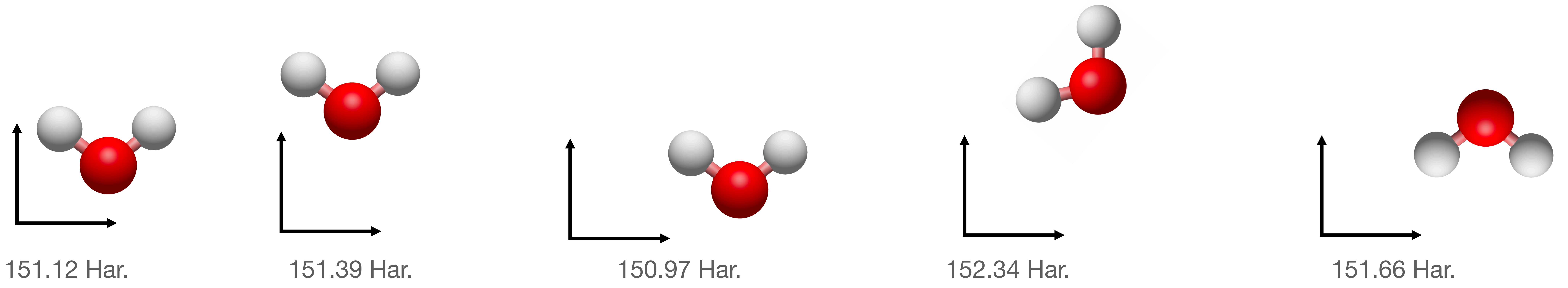
Examples of Invariance

The molecular energy (f) is invariant to which of the following transformations $(\mathscr{T})$?



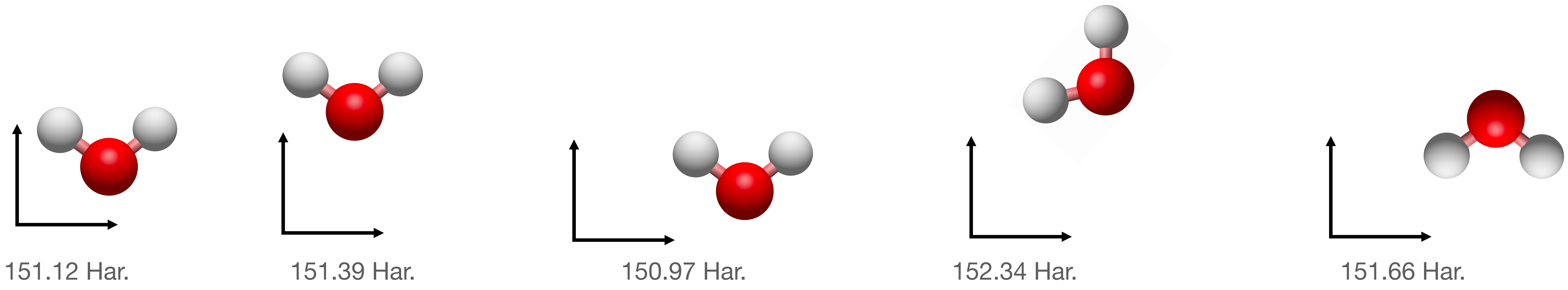
Why Care about Invariance?

Models that ignore physical invariances perform poorly and train inefficiently:



Why Care about Invariance?

Models that ignore physical invariances perform poorly and train inefficiently:



Models that have unphysical invariances are wrong, no matter how much data:

$$\hat{y} = -0.6n_H - 38.1n_C - 54.8n_N - 75.2n_O - 99.9n_F$$

How do NN Potentials Enforce Invariance?

Molecular Coordinates → **Atomic Environment Vectors** → **Atomic Energies**

Molecular Coordinates	Atomic Environment Vectors	Atomic Energies
Completely variant (to translation, rotation, etc.)	Invariant (to translation, rotation, etc.)	Invariant (to translation, rotation, etc.)

Invariance is enforced at the level of the atomic feature vector.

How do NN Potentials Enforce Invariance?

Molecular Coordinates	→	Atomic Environment Vectors	→	Atomic Energies
Completely variant (to translation, rotation, etc.)		Invariant (to translation, rotation, etc.)		Invariant (to translation, rotation, etc.)

Invariance is enforced at the level of the atomic feature vector.
This could be a problem for two reasons:

1) **We may want to model a variant property with a NN (e.g. dipole)**
   Invariant AEVs can't do this.

How do NN Potentials Enforce Invariance?

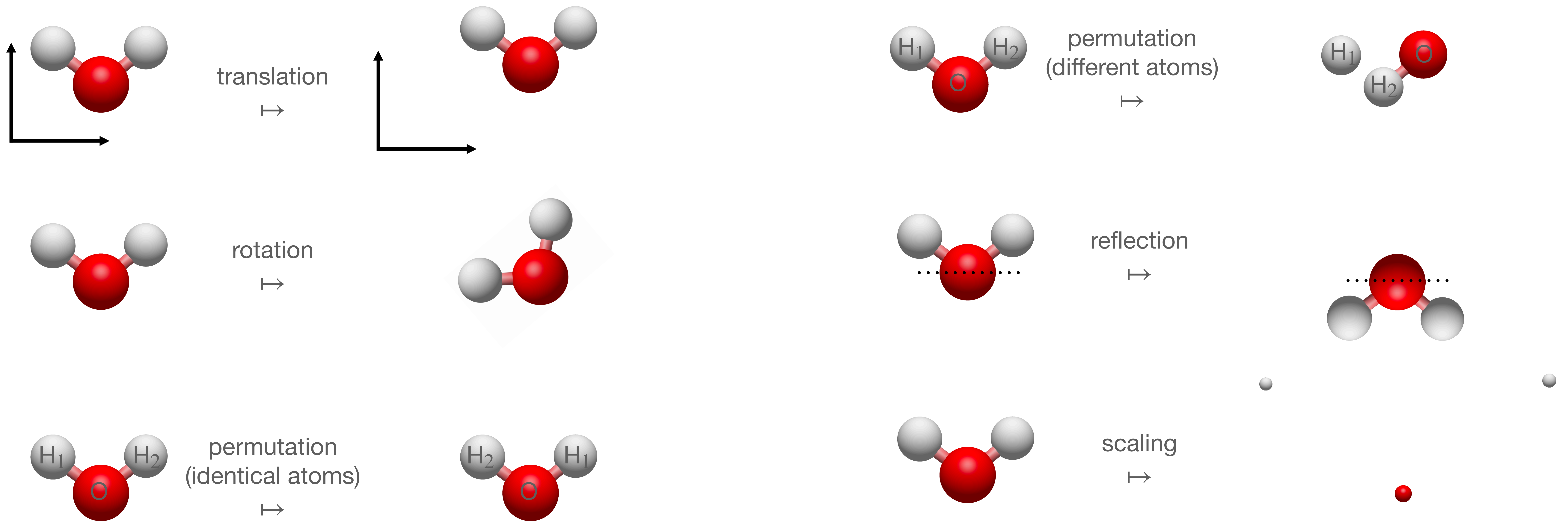
Molecular Coordinates	→	Atomic Environment Vectors	→	Atomic Energies
Completely variant (to translation, rotation, etc.)		Invariant (to translation, rotation, etc.)		Invariant (to translation, rotation, etc.)

Invariance is enforced at the level of the atomic feature vector.
This could be a problem for two reasons:

1) **We may want to model a variant property with a NN (e.g. dipole)**
   Invariant AEVs can't do this.

2) **Invariant AEVs contain less info than the original (variant) coordinates.**
   We want to include this info (w/o losing invariance of prediction)

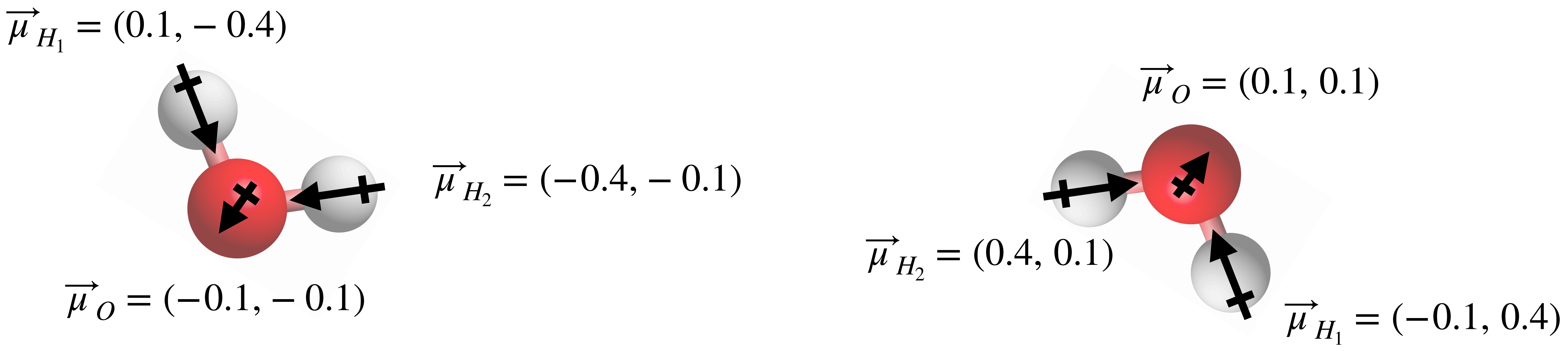
Is the molecular dipole $\overrightarrow{\mu} = (\mu_x, \mu_y, \mu_z)$ invariant or otherwise variant to each of the following transformations?



$\overrightarrow{\mu} = (\mu_x, \mu_y, \mu_z)$ varies with rotation and reflection, unlike energy.

Atomic Dipole Variance

The dipole vector norm is **invariant** to rotation/reflection, but the dipole direction isn't



This specific type of directional variance is called **equivariance**

Definition: Equivariance

$$f : X \mapsto y$$

f is a function that maps a molecule (X) to a property (y)

$$\mathcal{T} : X \mapsto X'$$

$\mathcal{T}$ is a function that transforms a molecules (X) to some other molecule (X')

f is **equivariant** to $\mathcal{T}$ if the following is true for all X:

$$\mathcal{T}(f(X)) \equiv f(\mathcal{T}X)$$

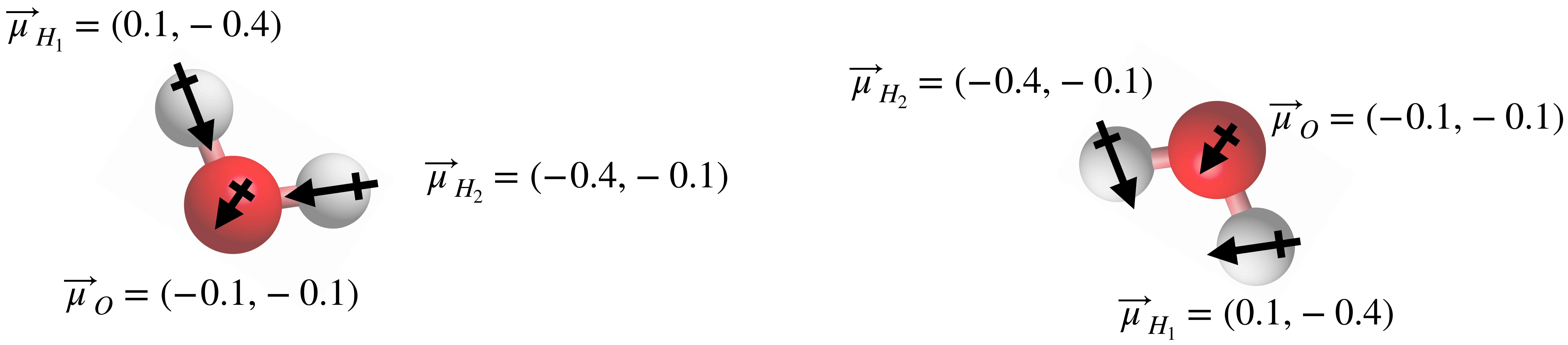
Invariance

$$f(X) \equiv f(\mathscr{T}X)$$

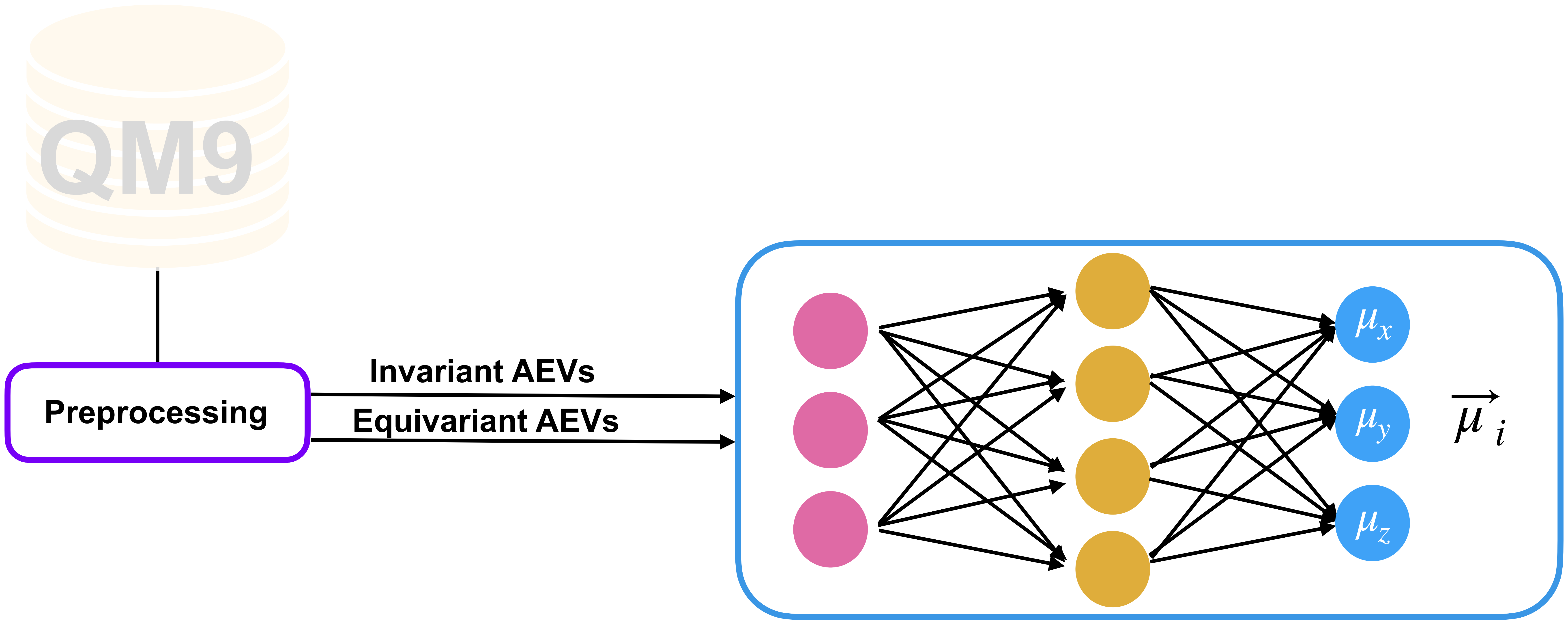
Equivariance

$$\mathscr{T}\left(f(X)\right) \equiv f(\mathscr{T}X)$$

Problem: Our models are *rotationally invariant*

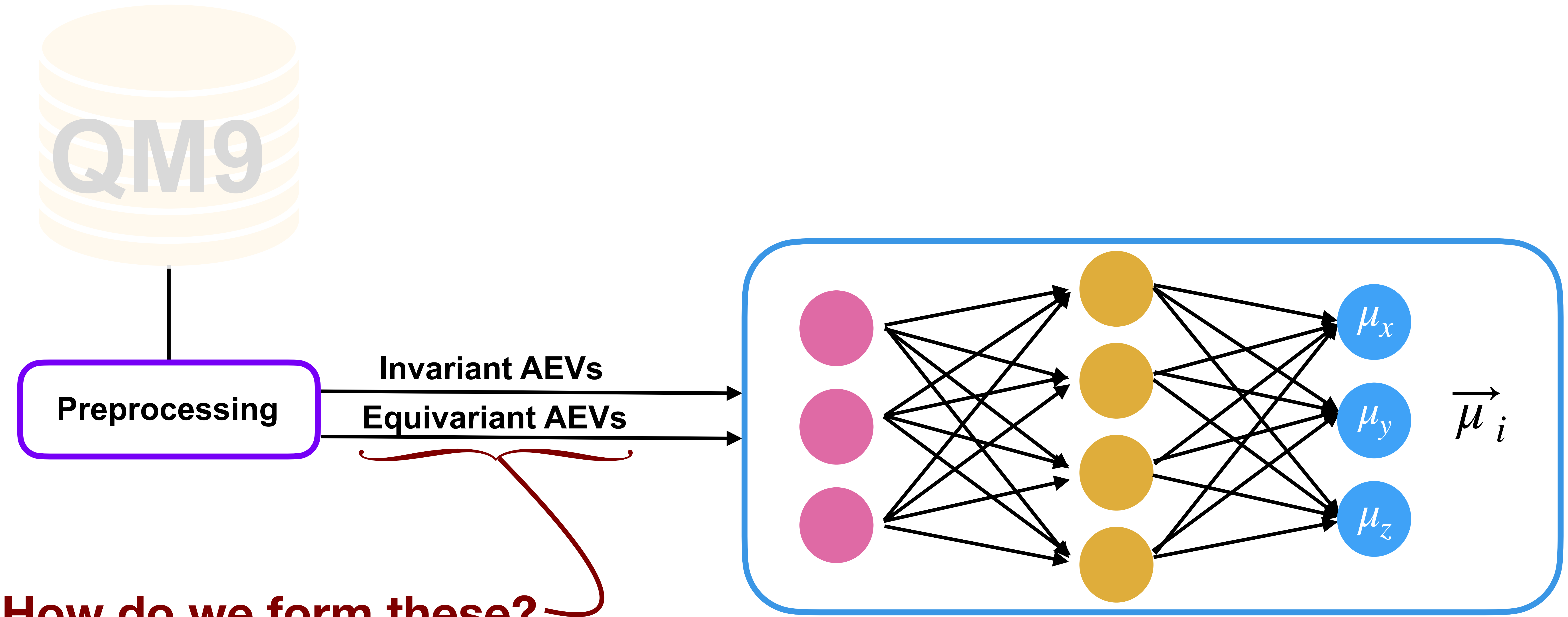


Predicting properties with rotational equivariance



$$\mathcal{R}(\overrightarrow{NN}(X)) = \overrightarrow{NN}(\mathcal{R}X)$$

Predicting properties with rotational equivariance

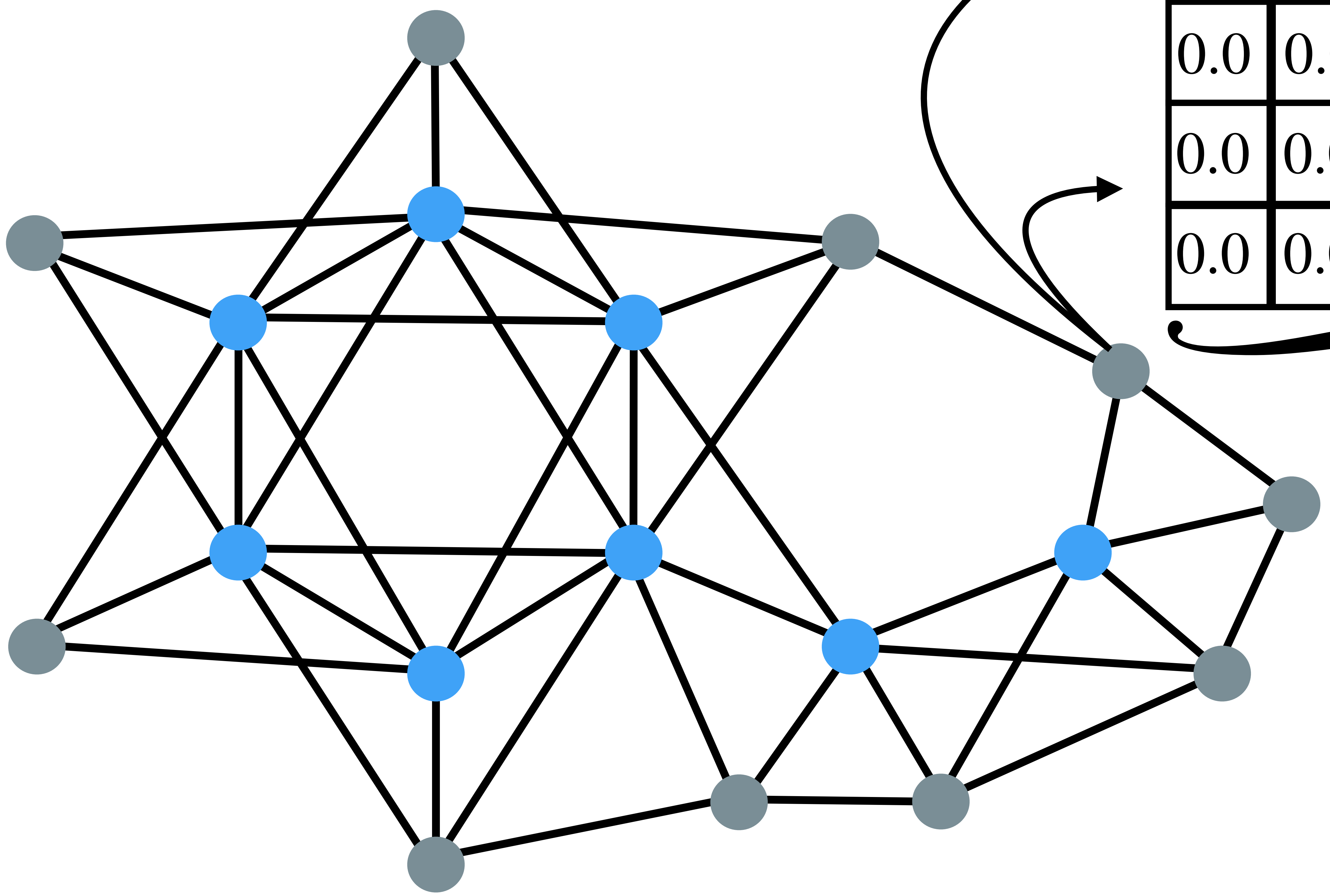


1. How do we form these?
2. How do we enforce this?

$$\mathscr{R}(\overrightarrow{NN}(X)) = \overrightarrow{NN}(\mathscr{R}X)$$

1. How do we form rotationally equivariant features?

Initialize



Invariant AEV

C	H	O	N				
0	1	0	0	0.0	0.0	0.0	0.0

$\vec{h}_i \in \mathbb{R}^{n_f}$

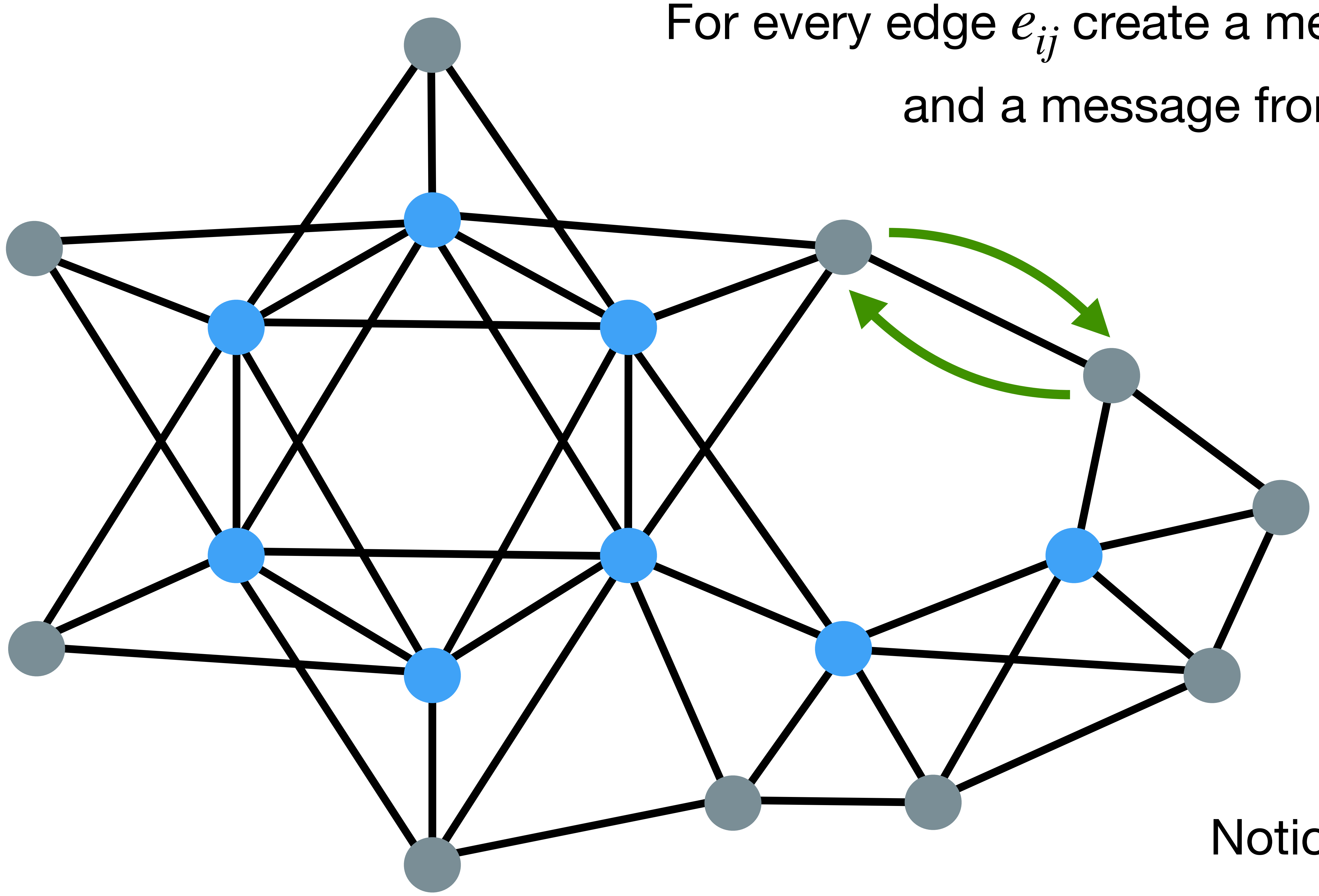
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Equivariant AEV

$\vec{\mathbf{x}}_i \in \mathbb{R}^{3 \times n_f}$

1. How do we form rotationally equivariant features?

Pass messages



For every edge e_{ij} create a message from i to j, $\overrightarrow{m}_{ij}$

and a message from j to i, $\overrightarrow{m}_{ji}$

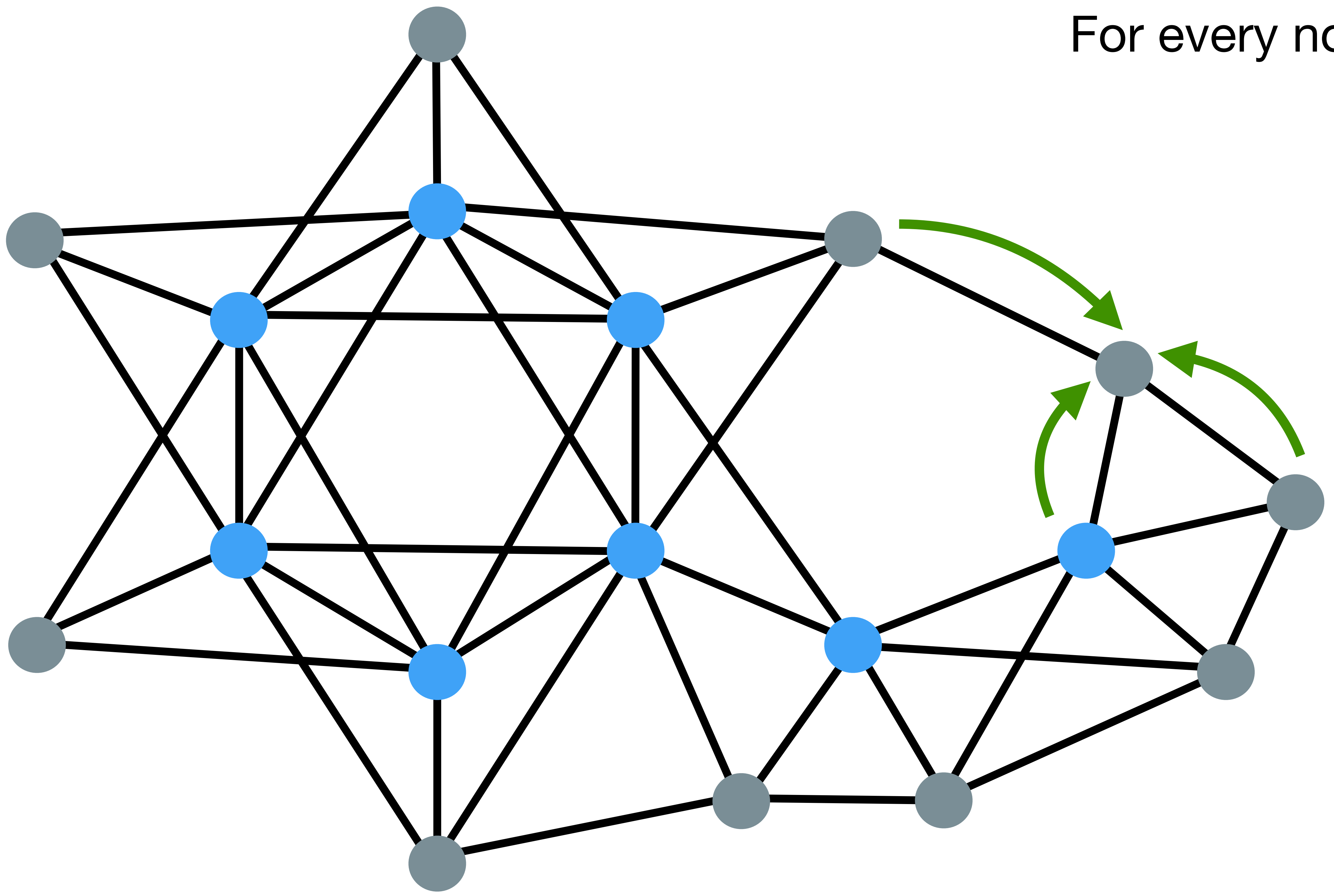
$$\overrightarrow{m}_{ij} = f_m(\overrightarrow{h}_i, \overrightarrow{h}_j, \overrightarrow{e}_{ij}, \underbrace{||\overrightarrow{\mathbf{x}}_i - \overrightarrow{\mathbf{x}}_j||^2}_{\text{Notice this term is invariant to rotations}})$$

Notice this term is **invariant** to rotations

1. How do we form rotationally equivariant features?

Update equivariant features

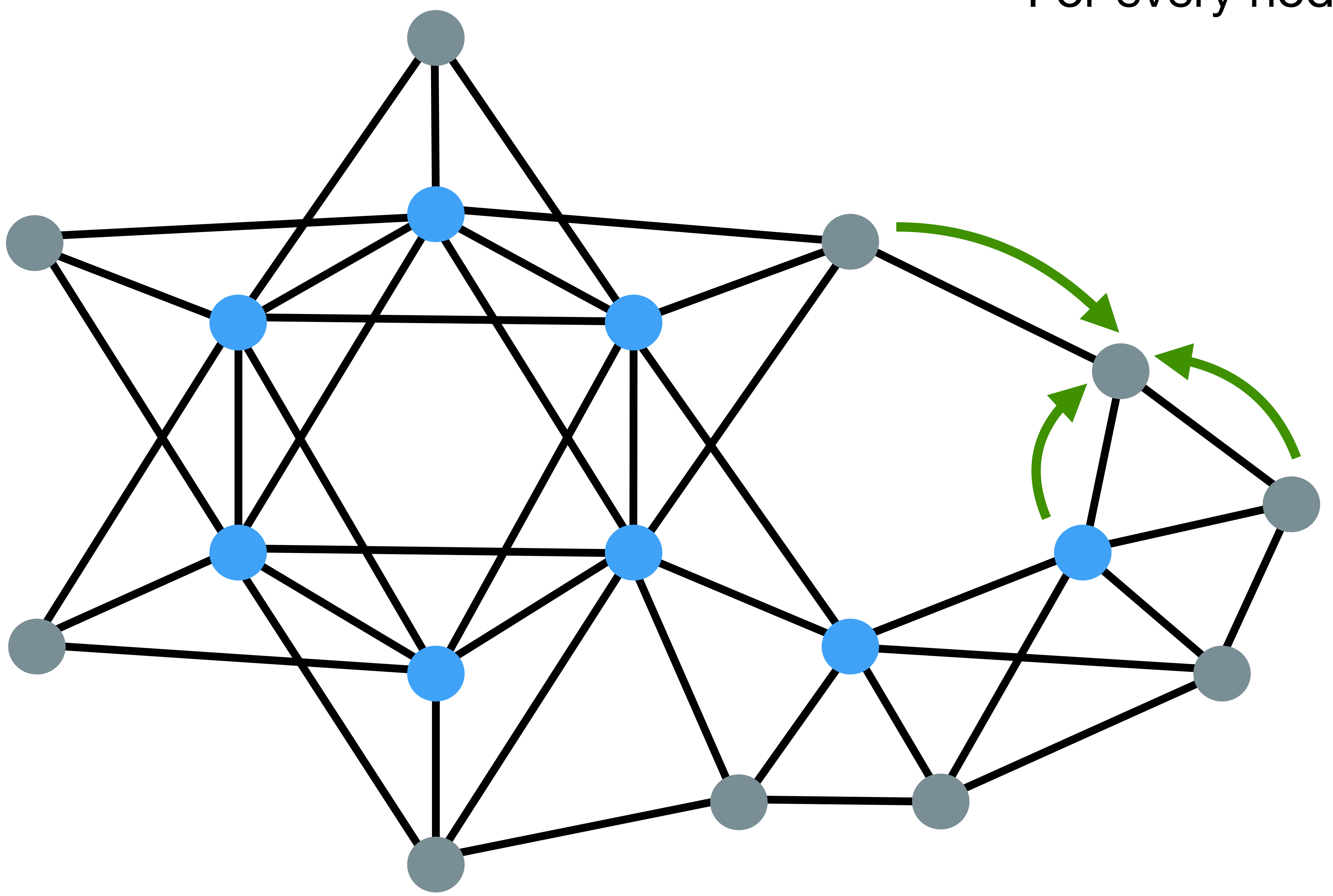
For every node, update $\vec{\mathbf{x}}_i$ preserving rotational equivariance



$$\vec{\mathbf{x}}_i \mathrel{+}= \frac{1}{N_j} \sum_{j \in \mathcal{N}(i)} \underbrace{(\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j)}_{\text{Notice this term is equivariant to rotations}} \underbrace{f_{\mathbf{x}}(\vec{m}_{ij})}_{\text{Another free function}}$$

1. How do we form rotationally equivariant features?

Update invariant features



For every node, accumulate all messages $\{\vec{m}_{ji}\}$ into a bundle $\vec{M}_i$

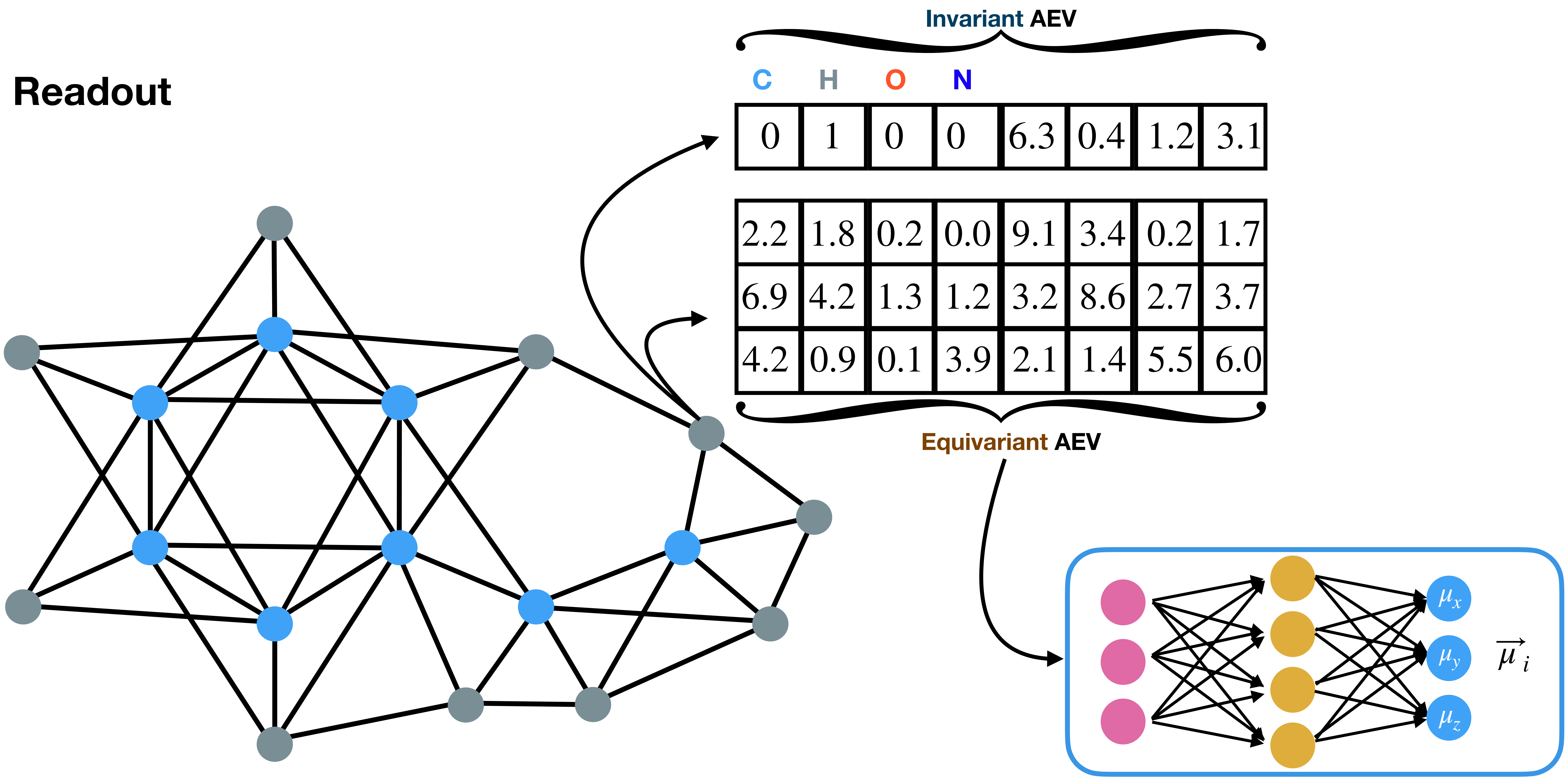
$$\vec{M}_i = \sum_j \vec{m}_{ji}$$

Update node states $\vec{h}_i$ based on message bundle $\vec{M}_i$

$$\vec{h}_i = f_U(\vec{h}_i, \vec{M}_i)$$

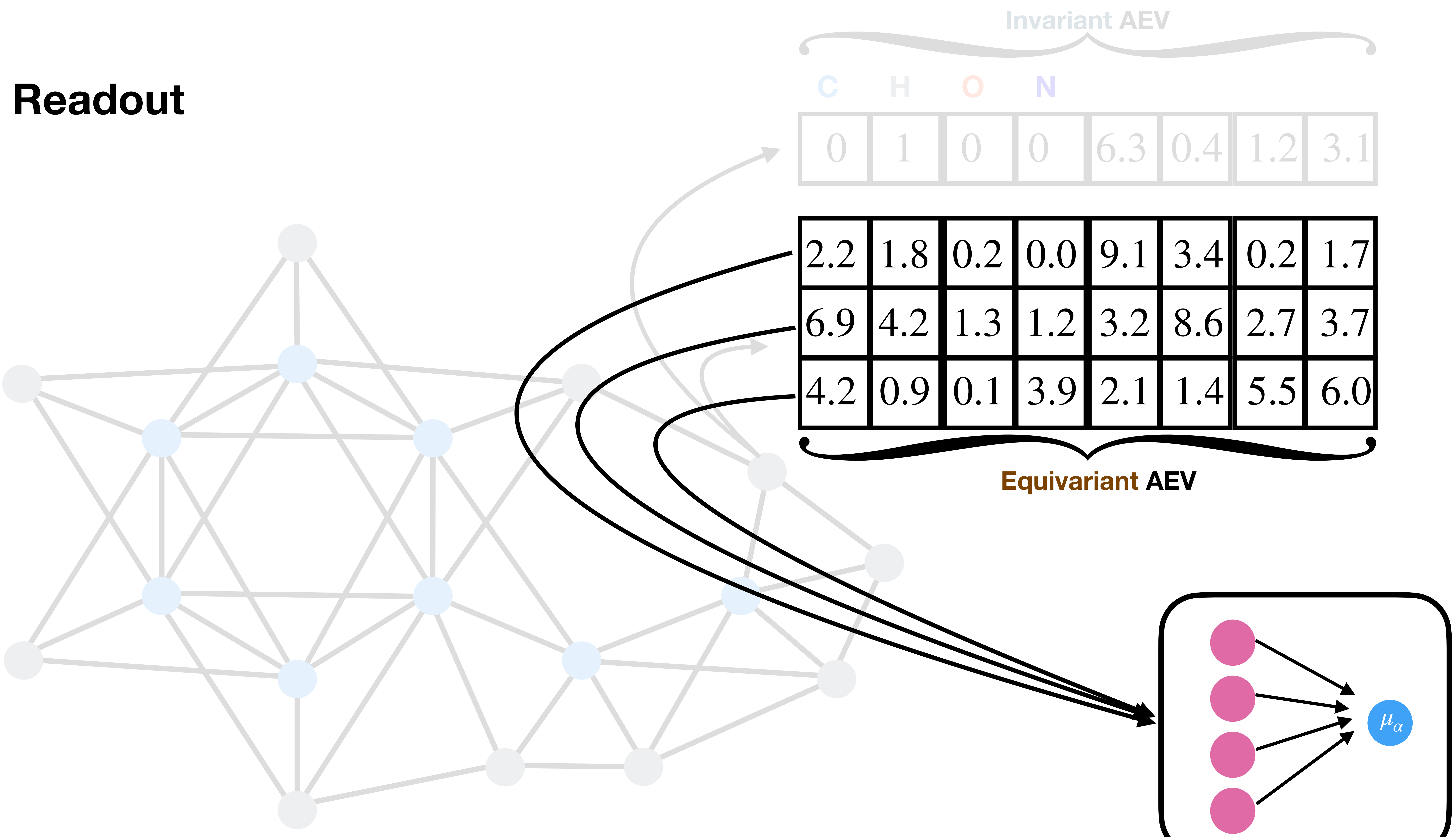
1. How do we form rotationally equivariant features?

Readout



2. How do we form enforce an equivariant readout?

Readout



$$\mu_\alpha = w \vec{\mathbf{x}}_\alpha + b$$

$$\alpha \in \{x, y, z\}$$

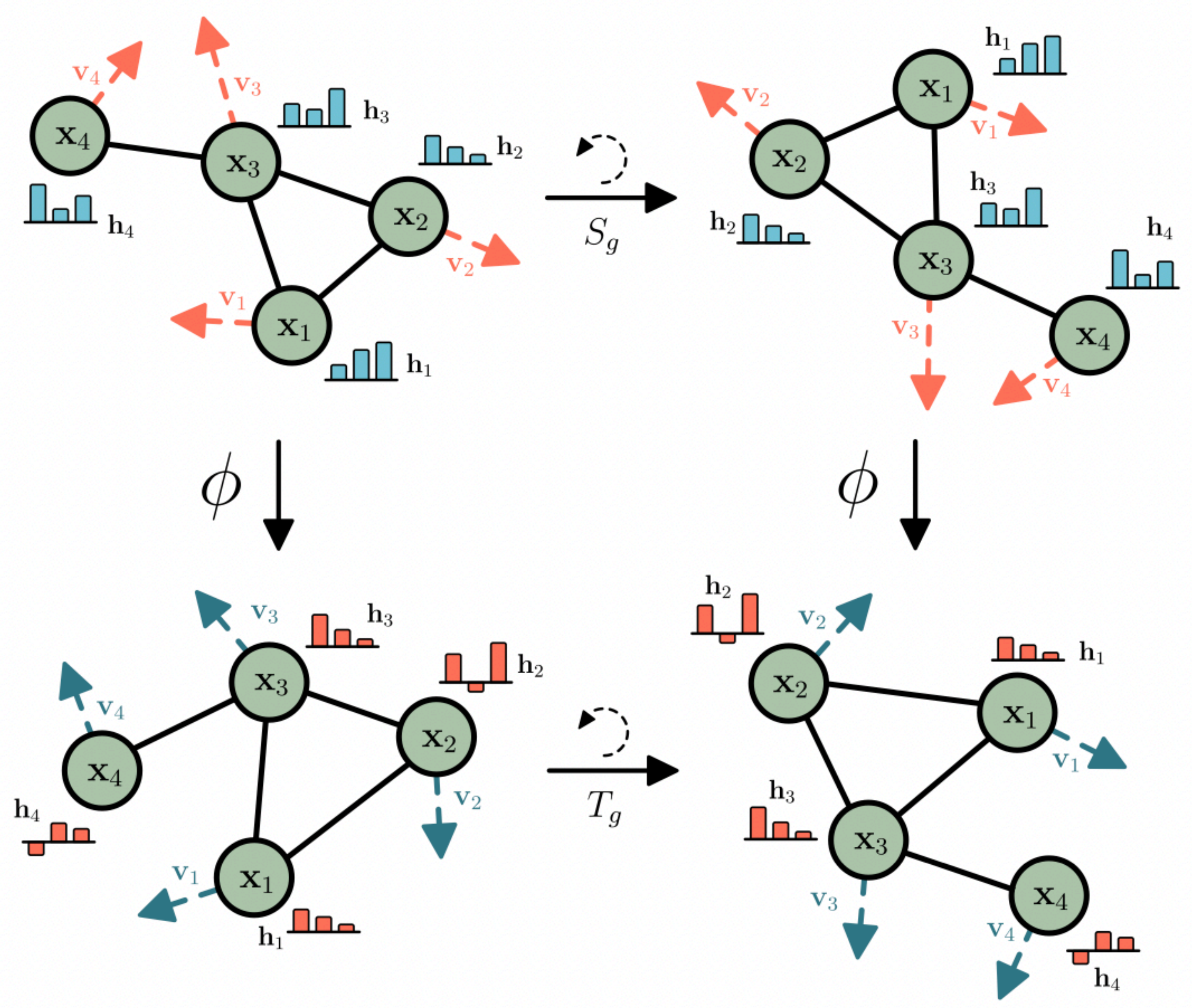
$$w \in \mathbb{R}^{n_f}$$

$$b \in \mathbb{R}^3$$

E(n) equivariance: The framework presented here

E(n) Equivariant Graph Neural Networks

Victor Garcia Satorras [1] Emiel Hoogeboom [1] Max Welling [1]



Generalizing equivariance

e3nn is a python library that implements general arbitrary equivariant convolutions

This is done with spherical harmonics and tensor products

e3nn

e3nn: a modular PyTorch framework for Euclidean neural networks

View My GitHub Profile

Welcome!
Getting Started
- How to use the Resources
- Installation

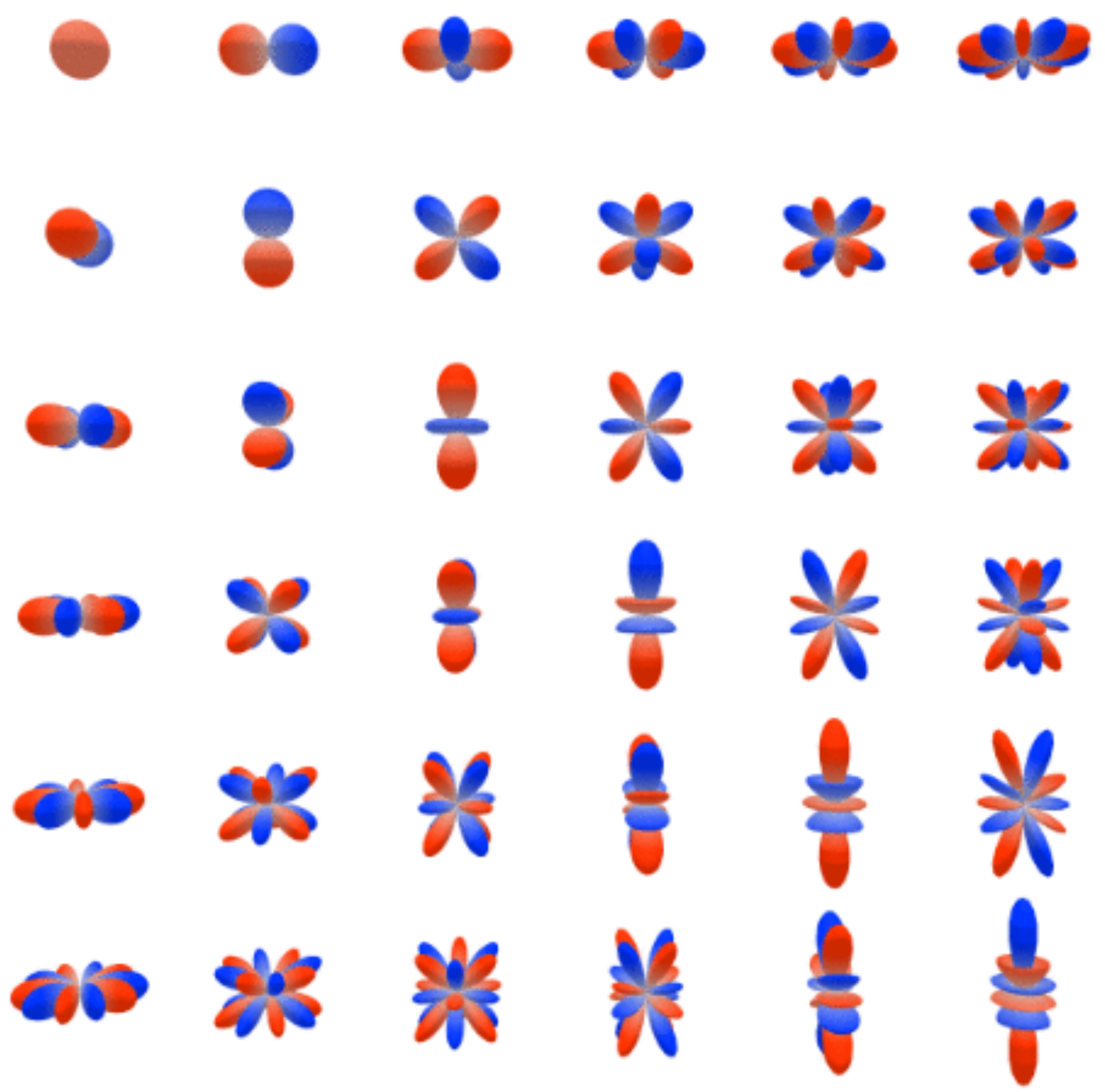
Help
Contributing
Resources
- Math that's good to know
- e3nn_tutorial
- e3nn_book
- Papers
- Previous Talks
- Poster
- Slack
- Recurring Meetings / Events

Calendar
e3nn Team

Welcome to e3nn!

This is the website for the e3nn repository
- https://github.com/e3nn/e3nn/
- Documentation

E(3) is the Euclidean group in dimension 3. That is the group of rotations, translations and mirror. e3nn is a pytorch library that aims to create **E**(**3**) equivariant **n**eural **n**etworks.



e3nn, https://e3nn.org

Equivariant NNs model invariant properties better than invariant NNs

(4.7 mEV is 0.108 kcal / mol)

Model	U_0	U	H	G
Schnet [25]	14	19	14	14
DimeNet++ [54]	6.3	6.3	6.5	7.6
Cormorant [23]	22	21	21	20
LieConv [55]	19	19	24	22
L1Net [56]	13.5	13.8	14.4	14.0
SphereNet [57]	6.3	7.3	6.4	8.0
EGNN [32]	11	12	12	12
ET [40]	6.2	6.3	6.5	7.6
NoisyNodes [58]	7.3	7.6	7.4	8.3
PaiNN [27]	5.9	5.7	6.0	7.4
Allegro, 1 layer	<u>5.7</u> (0.2)	<u>5.3</u>	<u>5.3</u>	<u>6.6</u>
Allegro, 3 layers	**4.7** (0.2)	**4.4**	**4.4**	**5.7**

TABLE III: Comparison of models on the QM9 data set, measured by the MAE in units of [meV]. Allegro outperforms all existing message passing and transformer-based models, in particular even with a single layer. Best methods in bold, second-best method underlined.

Allegro (2022), https://doi.org/10.48550/arXiv.2204.05249

Takeaways

NNs are powerful nonlinear regressors

Libraries like PyTorch make it easy to write and train complicated NNs

Training a NN can require significant hyperparameter tuning

Neural Networks aren’t a one-size-fits-all tool

Takeaways

NNs are powerful nonlinear regressors

Libraries like PyTorch make it easy to write and train complicated NNs

Training a NN can require significant hyperparameter tuning

Neural Networks aren’t a one-size-fits-all tool

A useful NN requires carefully considering the nature of the input/output

- **To a first approximation, energy is linear in the number of atoms**
- **Chemistry is local, so atomic environments can be too**
- ***Ad hoc* AEVs (symfuns) aren’t as expressive as NN-learned AEVs (MPNNs)**
- **Most (scalar) properties are invariant to Euclidian transforms**
- **Other (tensorial) properties are equivariant with these transforms**