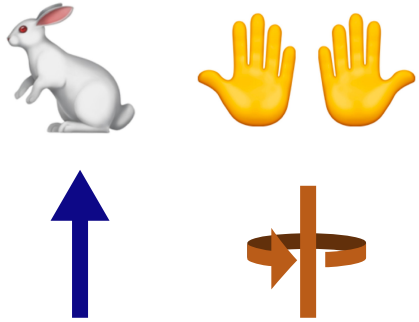
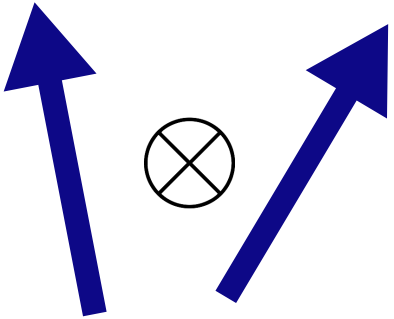
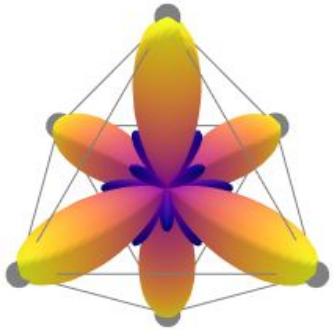
Harnessing Euclidean Symmetry-Equivariant Neural Networks for the Understanding and Design of Physical Systems







Tess Smidt

Atomic Architects

MIT EECS

rLe AT MIT RESEARCH LABORATORY OF ELECTRONICS AT MIT

Harnessing Euclidean Symmetry-Equivariant Neural Networks for the Understanding and Design of Physical Systems

Tess Smidt

Principal Investigator

Tricia O'Donnell

Administrative Assistant

Aria Mansouri Tehrani

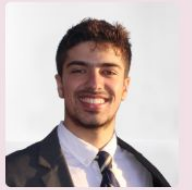
Postdoctoral Scholar

Rui Wang

Postdoctoral Scholar

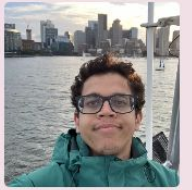
Elyssa Hofgard

PhD Student



Allan Costa

PhD Student



Mit Kotak

SM Student

Julia Balla

PhD Student

Yi-Lun Liao

PhD Student

Ameya Daigavane

PhD Student

Adriana Ladera

SM Student

YuQing Xie

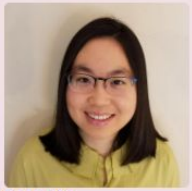
PhD Student

Giulianna Hashemi-Asasi

PhD Student

Max Aalto

PhD Student

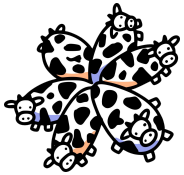


Song Kim

Undergraduate Student

Tuong Phung

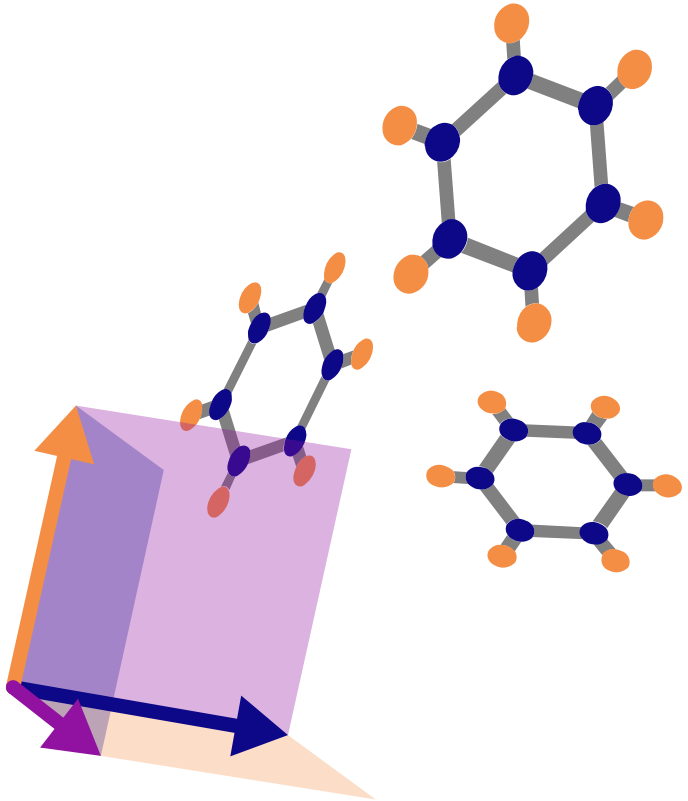
Undergraduate Student



Atomic Architects

RESEARCH LABORATORY OF ELECTRONICS AT MIT

How do we build neural networks that naturally handle data from physical systems?

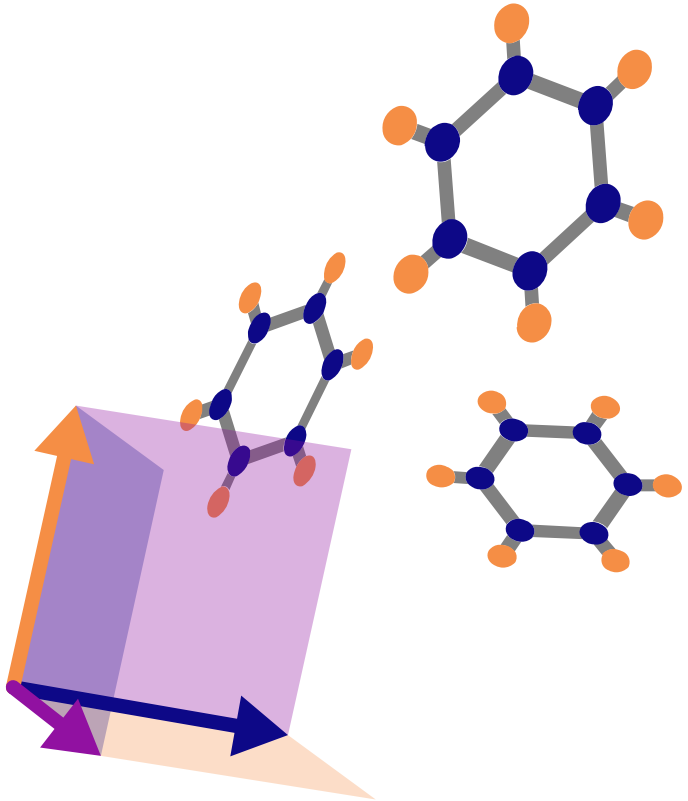


How do we build neural networks that naturally handle data from physical systems?

One option:

A minimal, yet powerful assumption

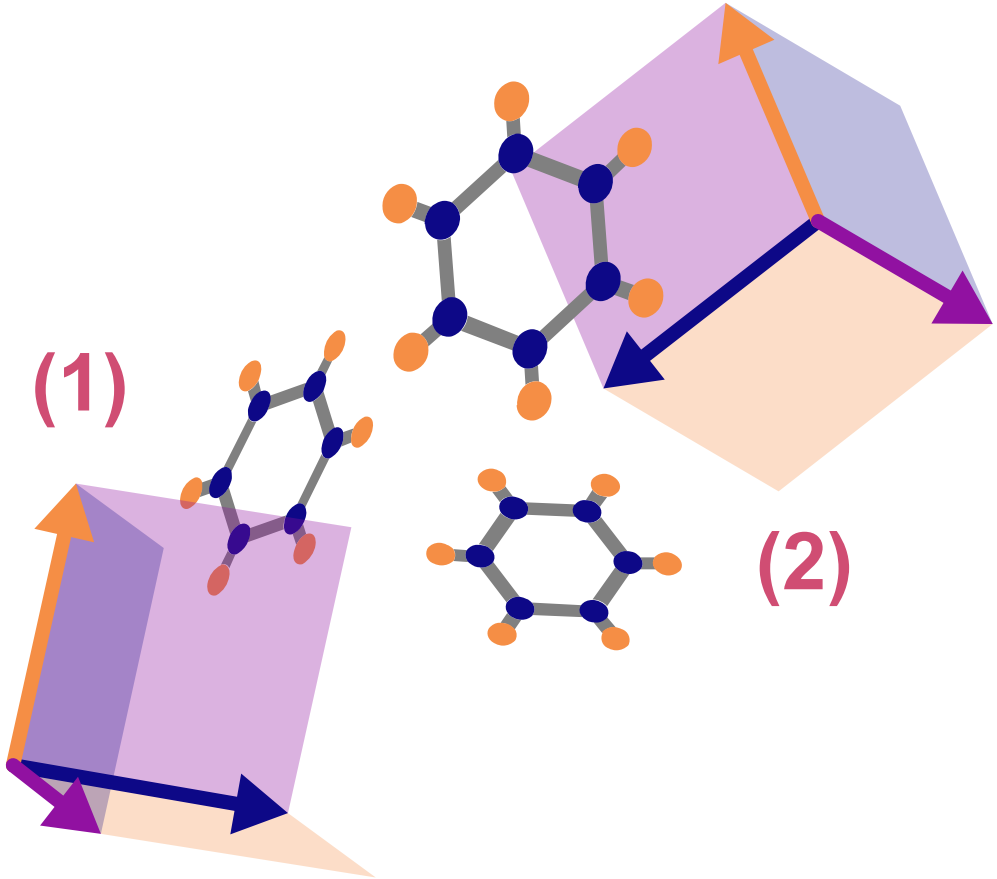
⇨ “Build-in” the *symmetry* of 3D space



To describe physical systems we use coordinate systems

(1) and (2) use different coordinate systems to describe the same physical system.

We can transform between coordinate systems using the symmetries of 3D Euclidean space (3D rotations, translations, and inversion)

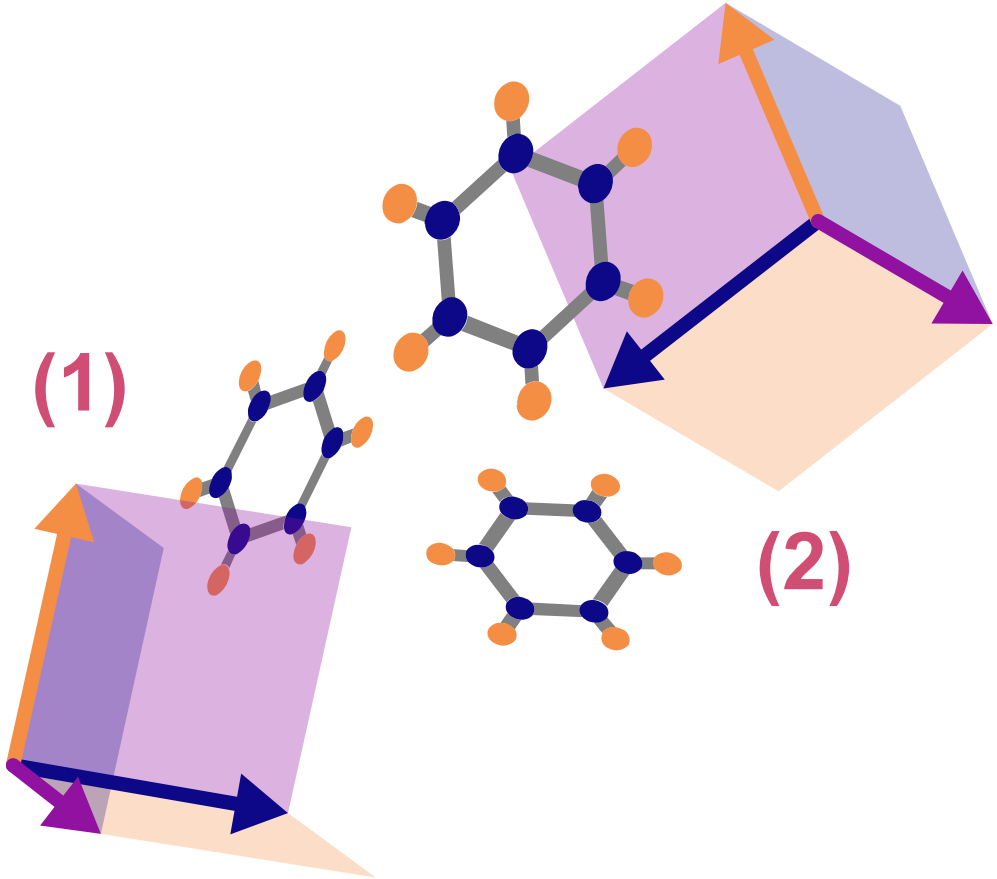


To describe physical systems we use coordinate systems

(1) and (2) use different coordinate systems to describe the same physical system.

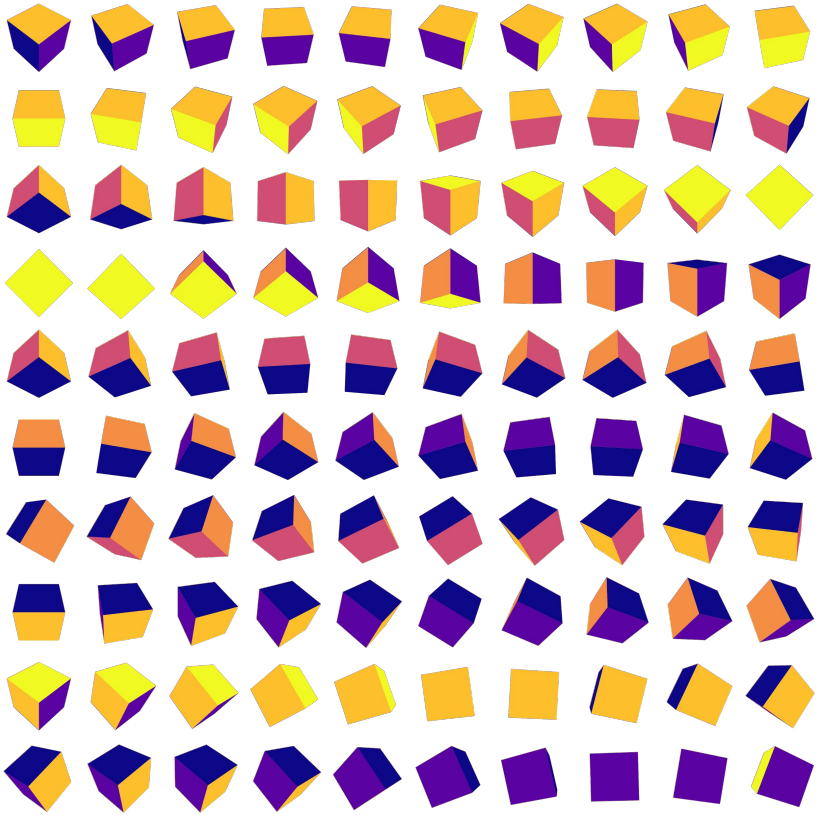
We can transform between coordinate systems using the symmetries of 3D Euclidean space (3D rotations, translations, and inversion)

Traditional machine learning see (1) and (2) as completely different!



Machine learning models not built to handle symmetry require data augmentation.
For 3D data, this is expensive, requiring ~500 fold augmentation.

training without rotational symmetry



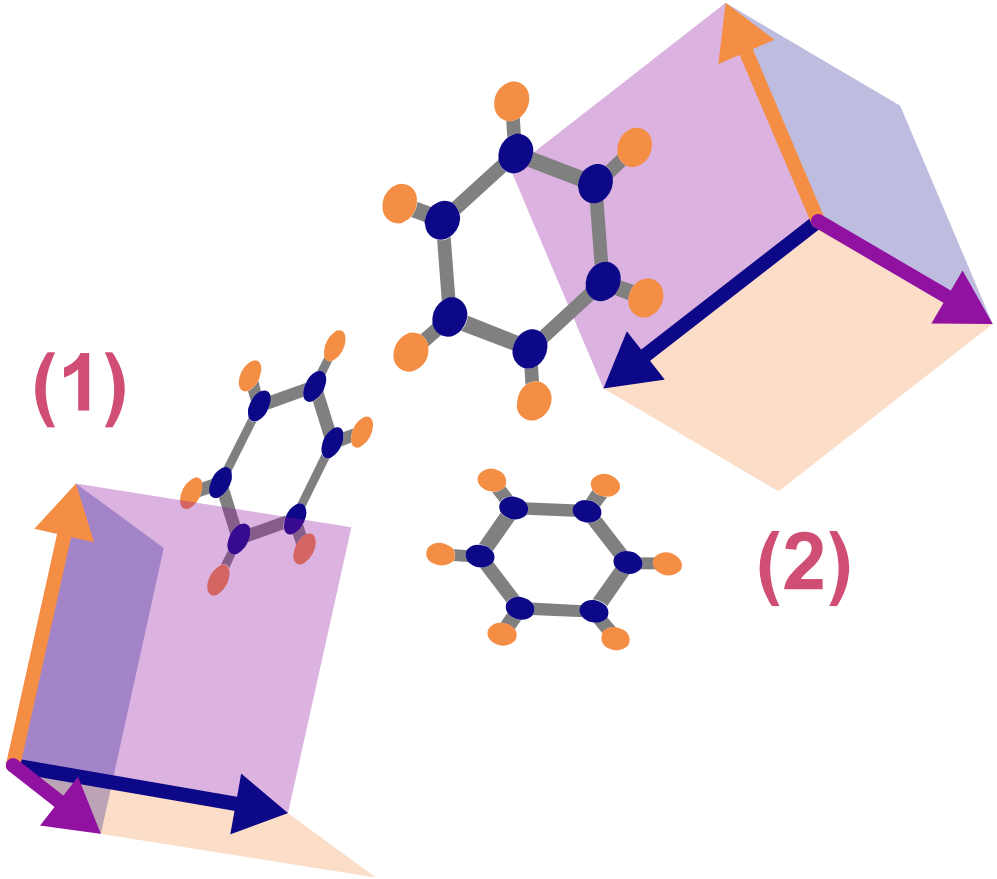
training with symmetry

To describe physical systems we use coordinate systems

(1) and (2) use different coordinate systems to describe the same physical system.

We can transform between coordinate systems using the symmetries of 3D Euclidean space (3D rotations, translations, and inversion)

Traditional machine learning see (1) and (2) as completely different!



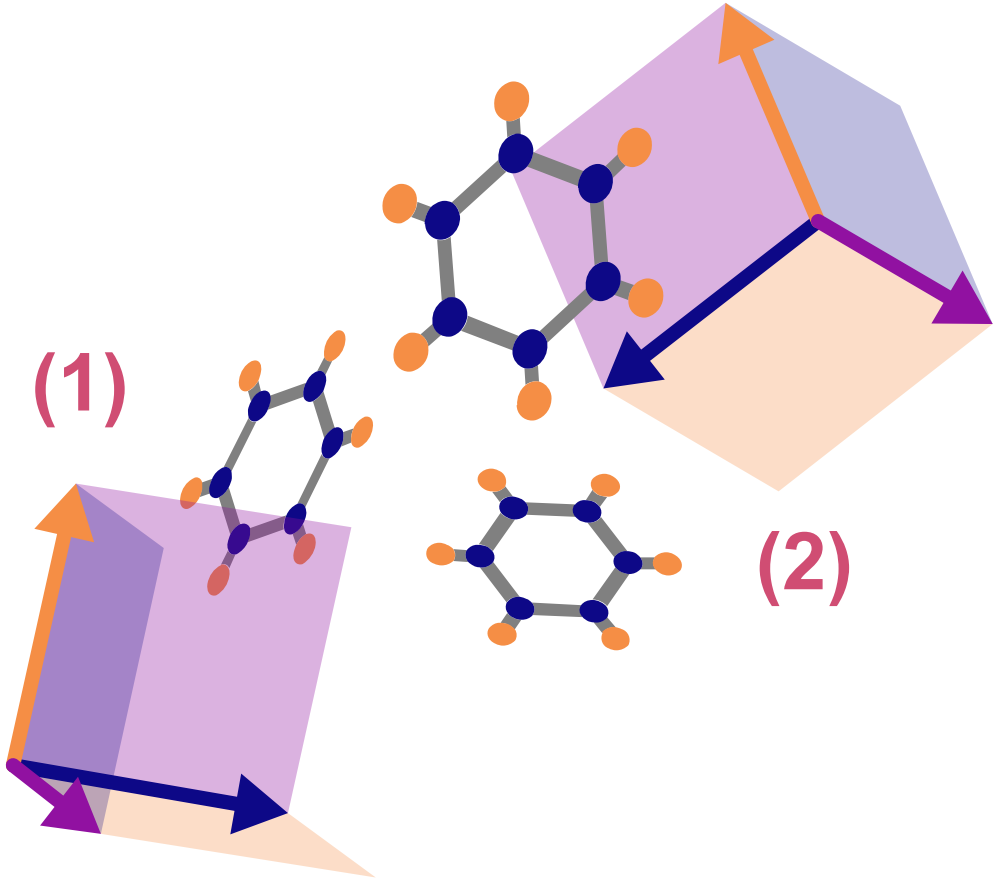
To describe physical systems we use coordinate systems

(1) and (2) use different coordinate systems to describe the same physical system.

We can transform between coordinate systems using the symmetries of 3D Euclidean space (3D rotations, translations, and inversion)

Traditional machine learning see (1) and (2) as completely different!

E(3) equivariant neural nets (E(3)NNs) see (1) and (2) as the same system described differently...



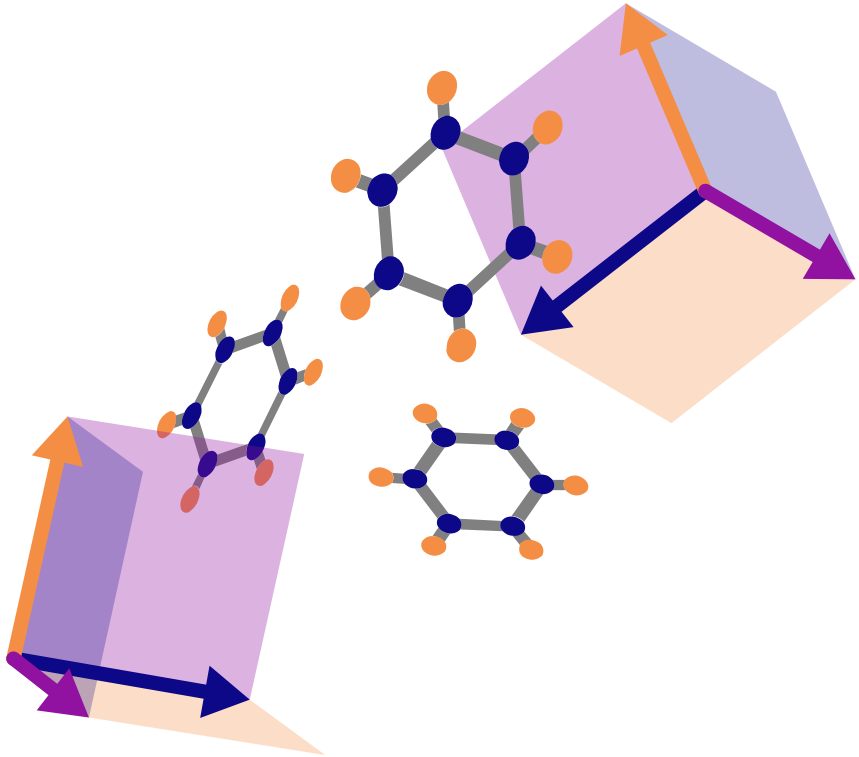
Symmetry emerges when different ways of representing something “mean” the same thing.

Representation can have symmetry, operations can preserve symmetry, and objects can have symmetry.

Euclidean symmetry, E(3):

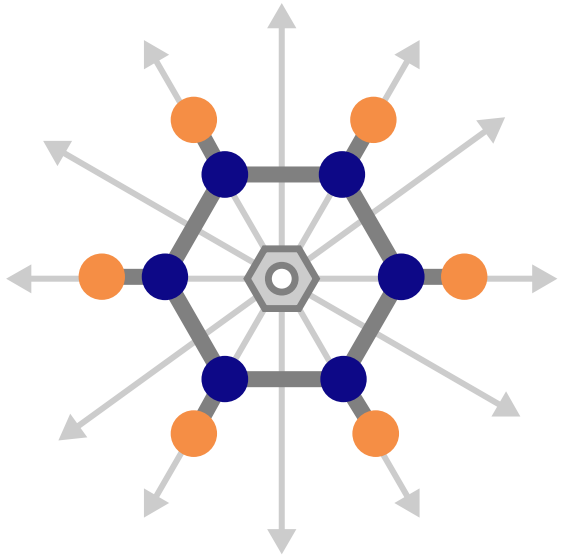
Symmetry of 3D space

The freedom to choose your coordinate system



Symmetry of geometric objects

Looks the same under specific rotations, translations, and inversion (includes mirrors).



$$D_{6h} = 6/mmm$$

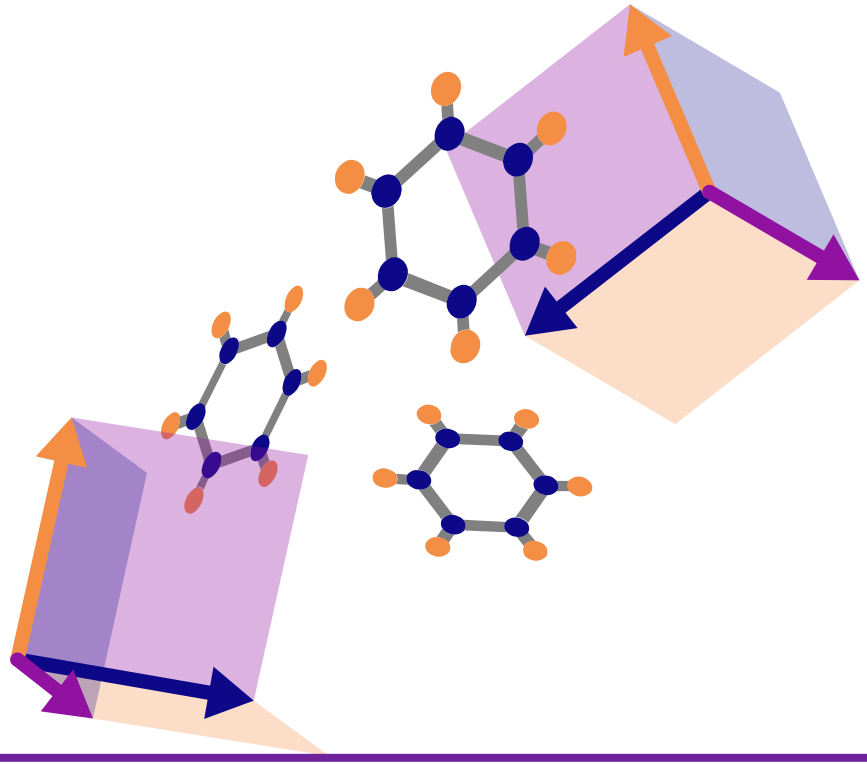
Symmetry emerges when different ways of representing something "mean" the same thing.

Representation can have symmetry, operations can preserve symmetry, and objects can have symmetry.

Euclidean symmetry, E(3):

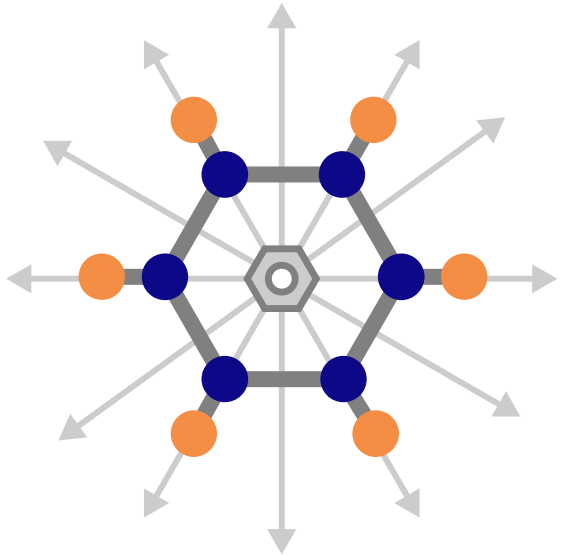
Symmetry of 3D space

The freedom to choose your coordinate system



Symmetry of geometric objects

Looks the same under specific rotations, translations, and inversion (includes mirrors).

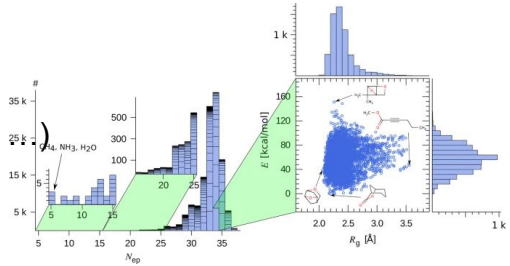


$$D_{6h} = 6/mmm$$

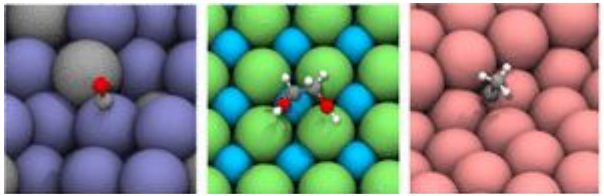
Euclidean symmetry-equivariant networks are state-of-the-art on many atomistic ML tasks.

Some (non-exhaustive) examples include…

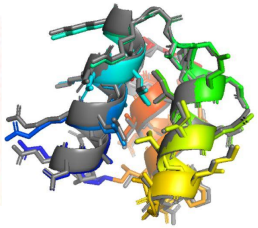
QM9 (PaiNN, TorchMD-NET, …)



Open Catalysis 2020 Dataset (eSCN, Equifomer, …)

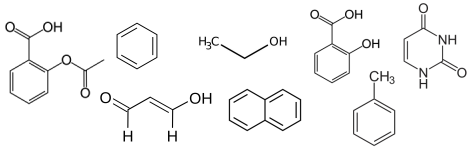


Protein folding (Equifold, …)

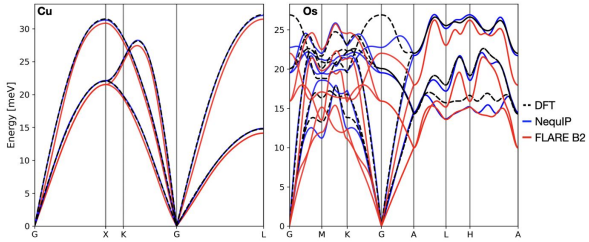


RNA structure scoring (ARES)

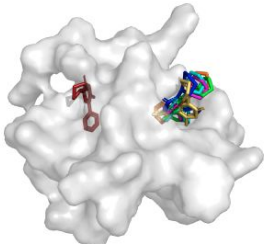
MD17 (Nequip, Allegro, TorchMD-NET, Equiformer, …)



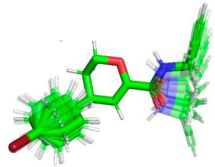
Transition metal MD + phonons (Nequip)



Ligand Docking (DiffDock, …)

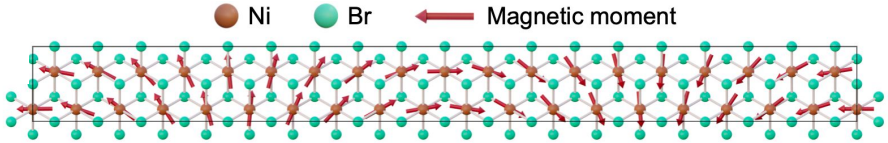


Conformer Generation (Torsional diffusion, …)

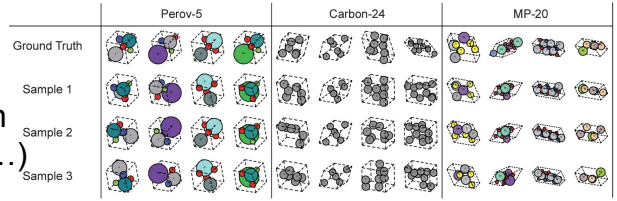


Large biomolecules (Allegro, …)

Magnetic structure (xDeepH, …)



Crystal Generation (CDVAE, …)



Euclidean symmetry-equivariant networks are state-of-the-art on many atomistic ML tasks.

Some (non-exhaustive) examples include…

QM9 (PaiNN, TorchMD-NET, …)

Open Catalysis 2020 Dataset (eSCN, Equifomer, …)

Protein folding (Equifold, …)

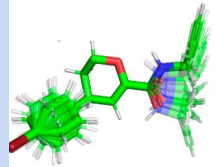
RNA structure scoring (ARES)

MD17 (Nequip, Allegro, TorchMD-NET, Equiformer, …)

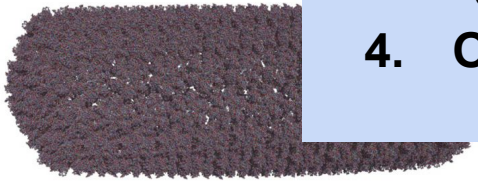
This talk…

1. **Properties of E(3)NNs**
2. **What's different about Euclidean neural nets?**
   - **Data types, interactions, handling geometry**
3. **What these properties have enabled us to do.**
   - **Expressive features and functions**
   - **Manipulate and generate geometry**
   - **Uncover missing information**
4. **Open questions**

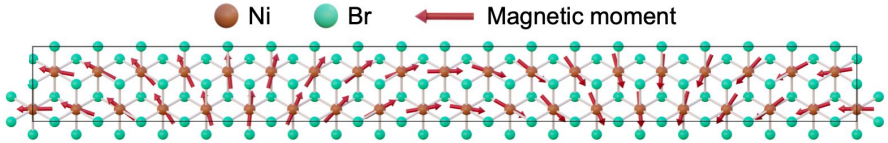
nformer Generation
rsional diffusion, …)



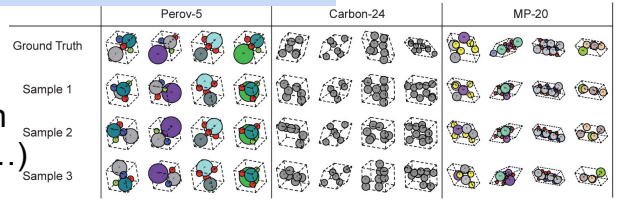
Large biomolecules (



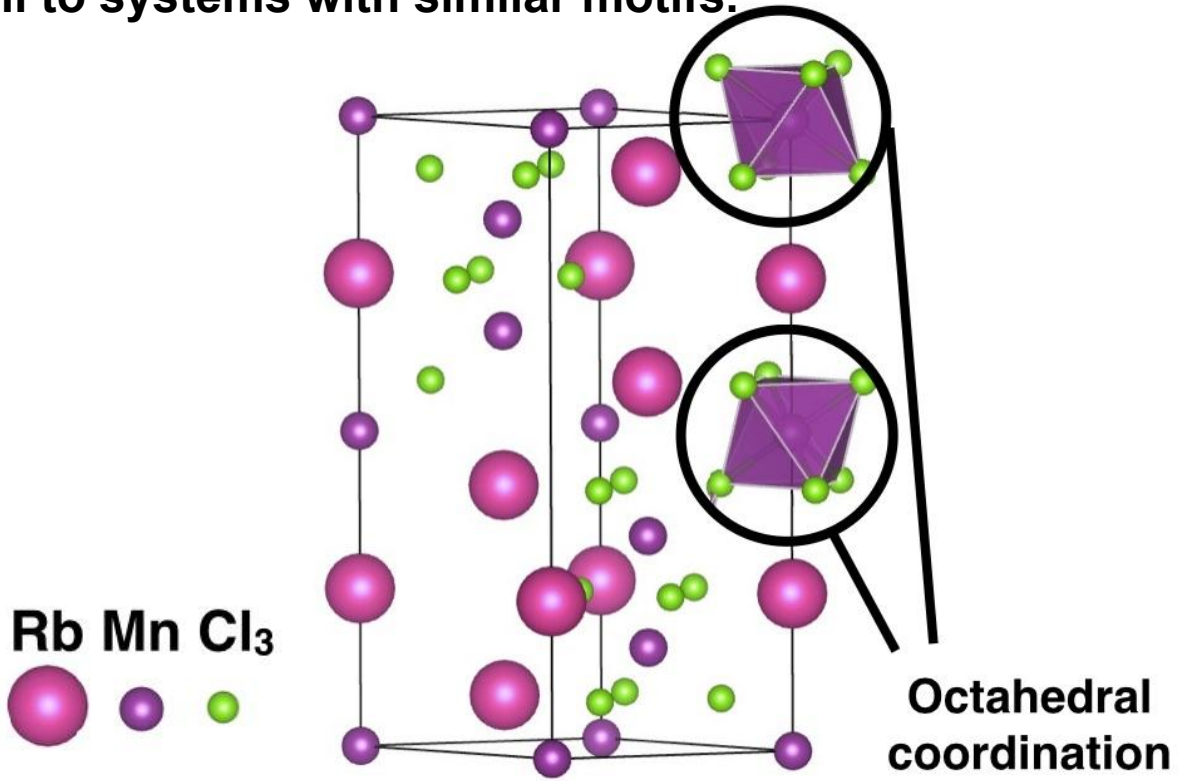
Magnetic structure (xDeepH, …)

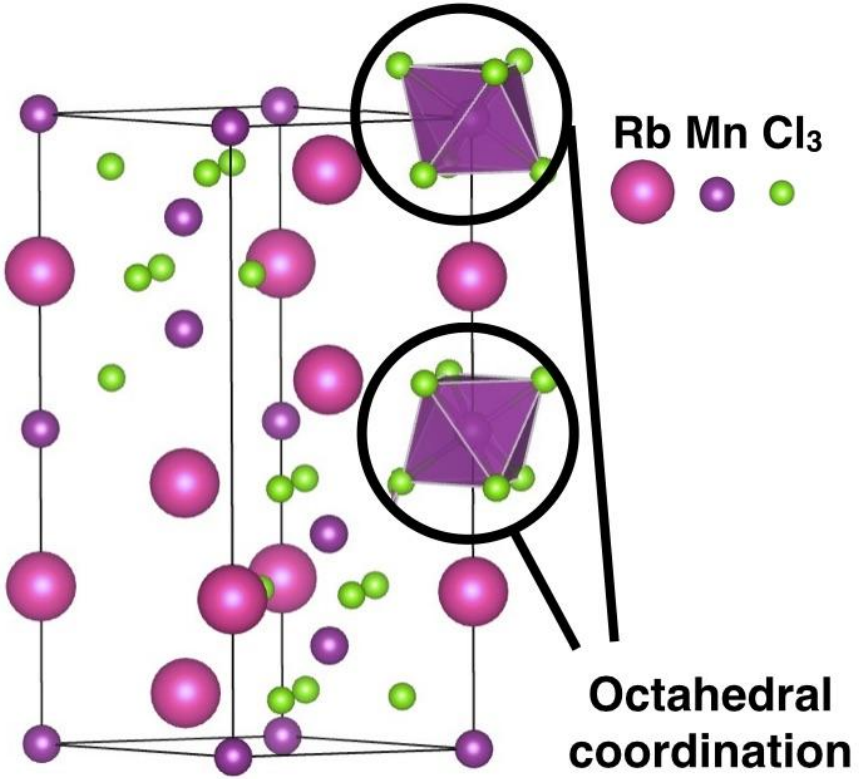


Crystal Generation (CDVAE, …)



E(3)NNs can recognize equivalent recurring geometric patterns that appear in different locations and orientations (from seeing only one example) **and generalize well to systems with similar motifs.**

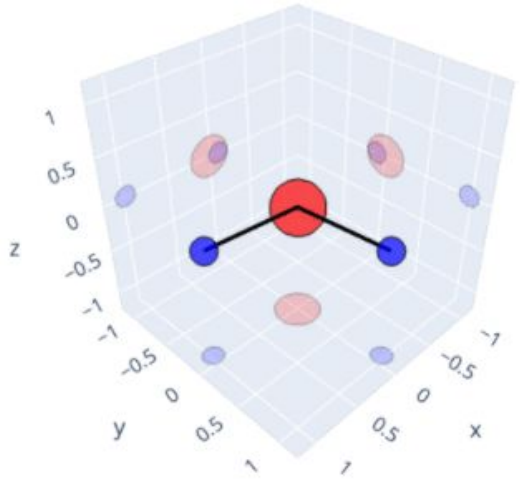



Rb Mn Cl_3
Octahedral coordination

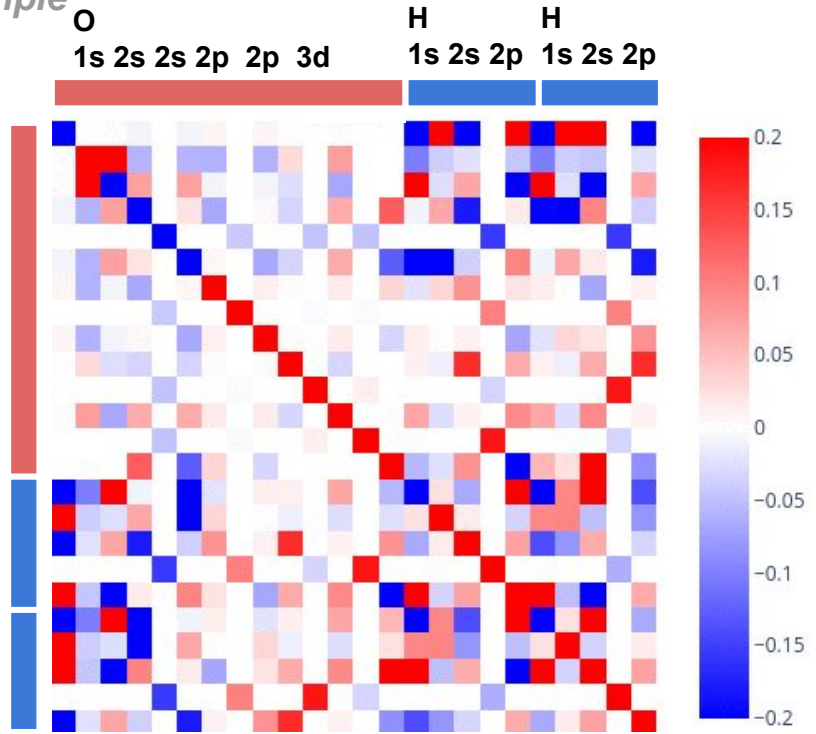
E(3)NNs will automatically (without training) transform complex geometric objects correctly, e.g. the electronic Hamiltonian of this water molecule. **The same system under rotation still "means" the same thing.**

An Euclidean neural network trained on one example of water, can predict properties in any rotation.

$R\vec{v}$

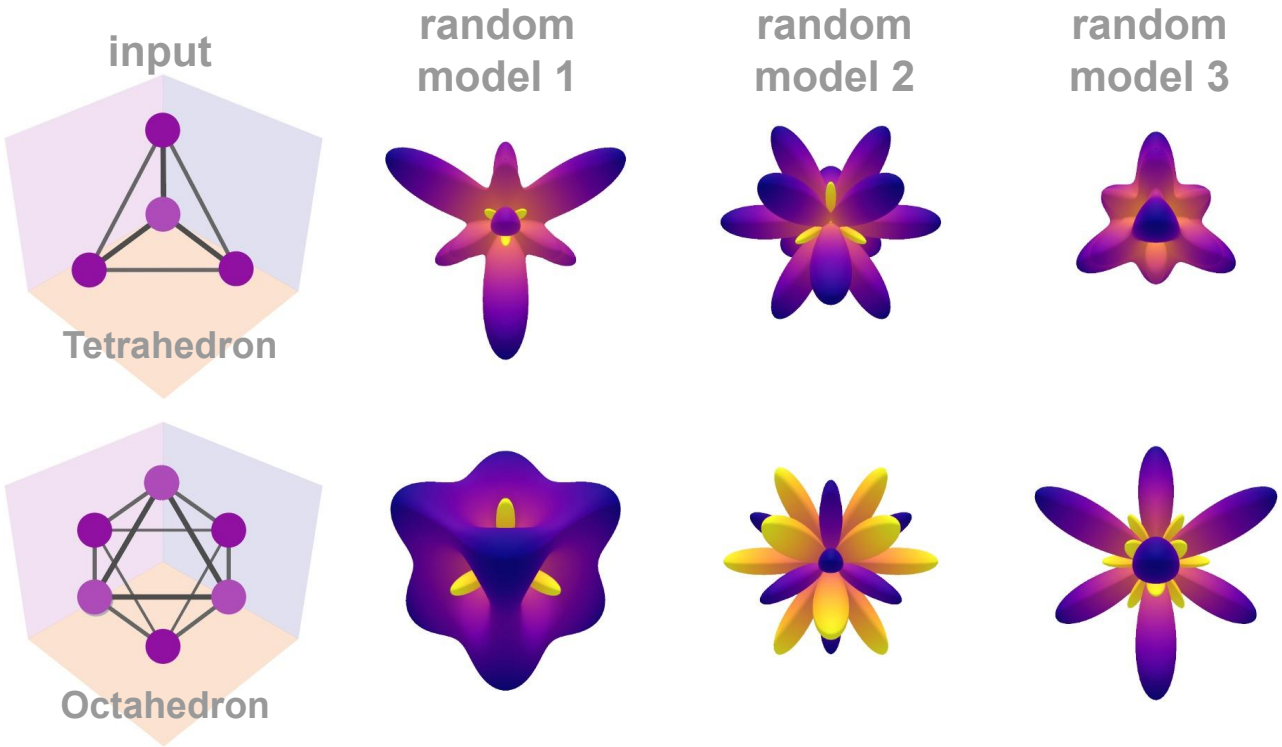


RHR^T

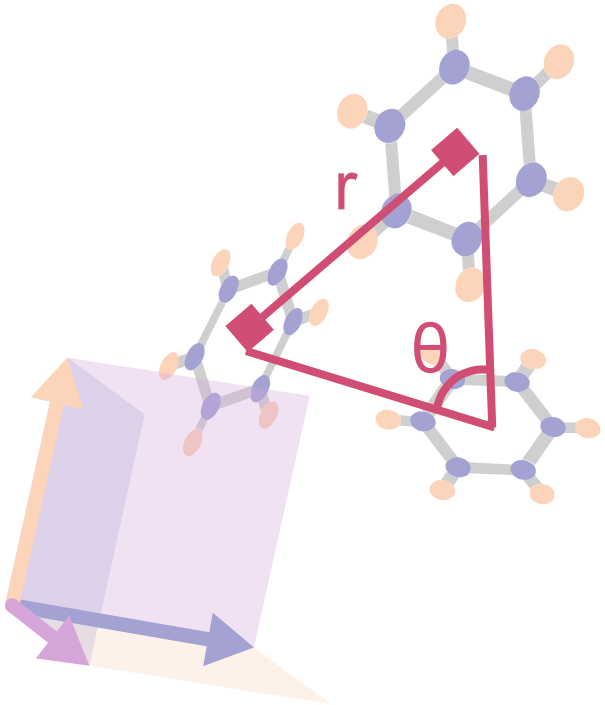


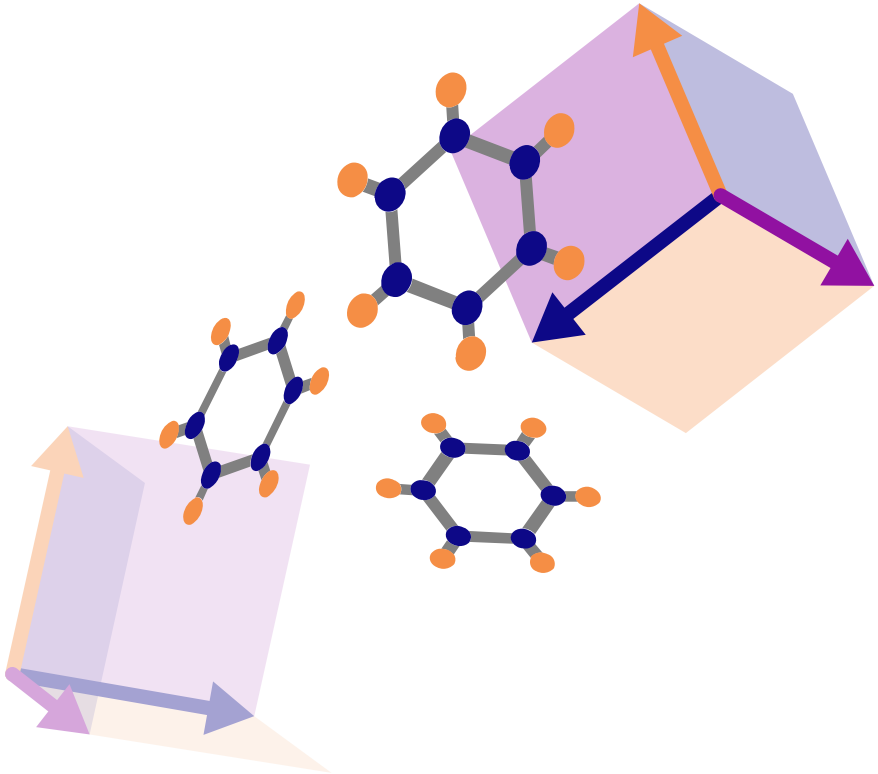
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.

Curie's principle (1894): "When effects show certain asymmetry, this asymmetry must be found in the causes that gave rise to them."

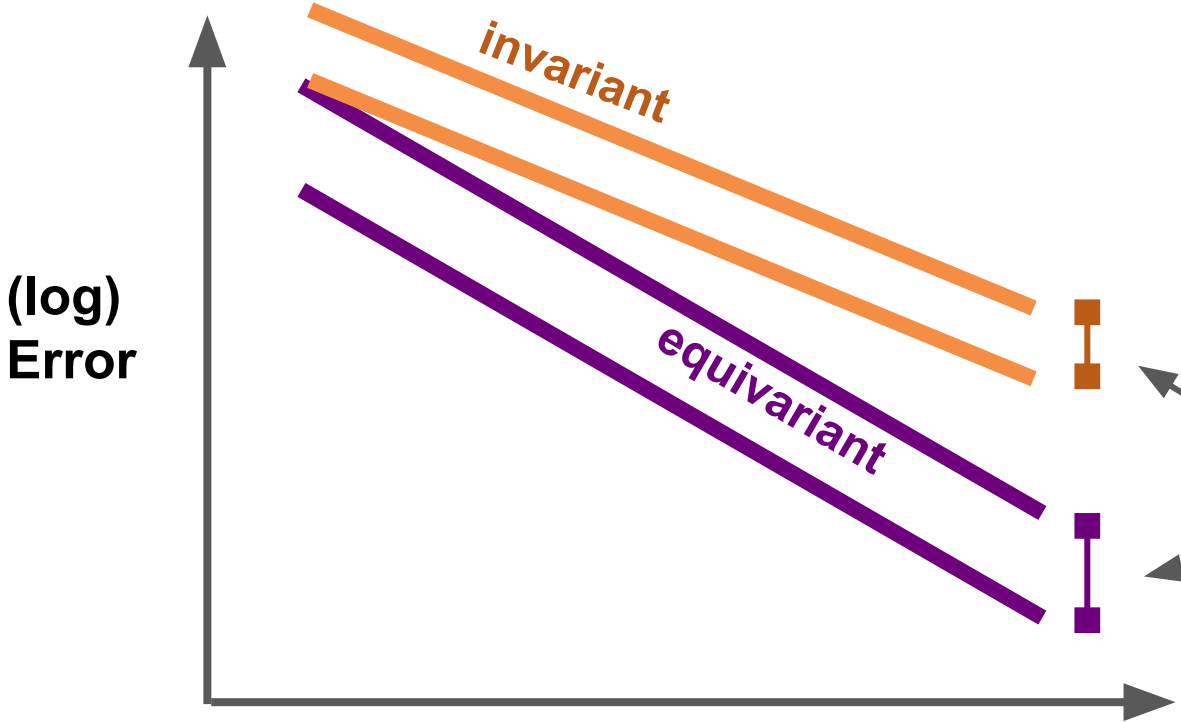


Invariant models pre-compute invariant features and throw away the coordinate system.
Equivariant models keep the coordinate system
AND if the coordinate system changes, the outputs change accordingly.
I'll focus on equivariant models.





Equivariant methods are more data-efficient than invariant methods

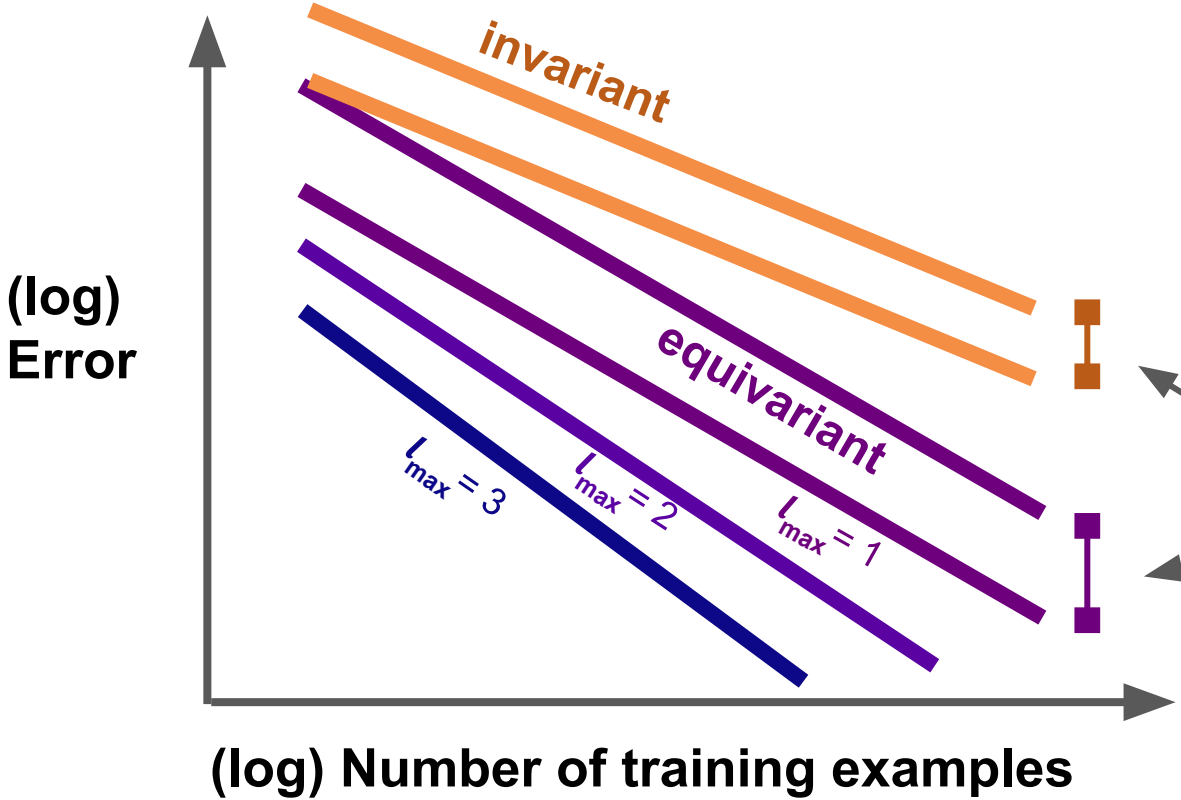


Power law
scaling exponent
β = slope
$\beta_{eq} > \beta_{inv}$

Architecture
and task
dependent
offset.

Refs:
[1, 2, 3, 4]

Equivariant methods are more data-efficient than **invariant** methods and (for some tasks) higher-order interactions improve accuracy.



Power law scaling exponent

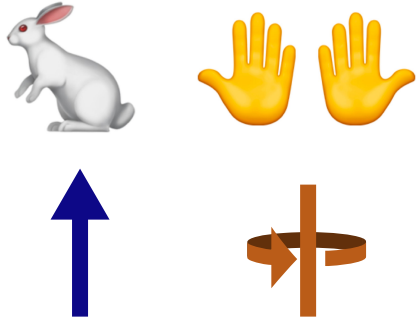
β = slope

$\beta_{eq} > \beta_{inv}$

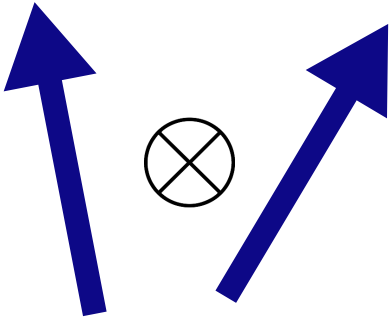
Architecture and task dependent offset.

Refs:
[1, 2, 3, 4]

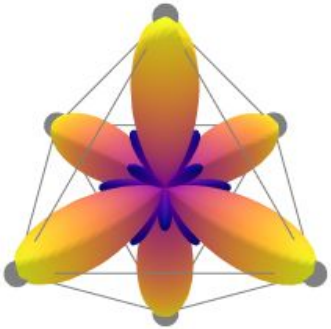
Euclidean neural networks differ from traditional neural networks in how you...



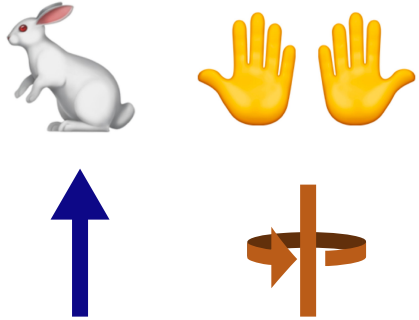
***Define how data transforms* (Irreps)**



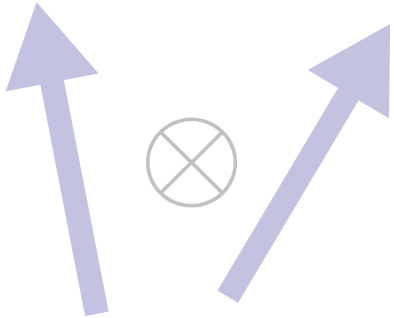
***Interact Data* (Tensor products)**



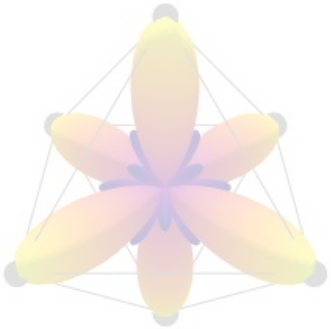
***Featurize Geometry* (Spherical Harmonics)**



***Define how data transforms* (Irreps)**



Interact Data
(Tensor products)



Featurize Geometry
(Spherical Harmonics)

<u>All data</u> acted on by O(3) can be broken up into simpler "data types" *(irreps)* defined by...

L — **angular frequency** (positive int)
rate of change under rotation

parity — **even** or **odd**
does not or does flip sign under inversion

Some examples include...

L=0

Even parity
(scalars)
Classification labels

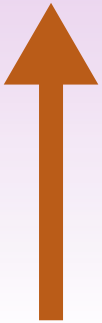
rabbit

Odd parity
(pseudoscalars)
Chirality or "handedness"

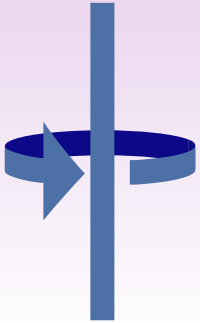
right hand

L=1

Odd parity
(vectors)
Coordinates



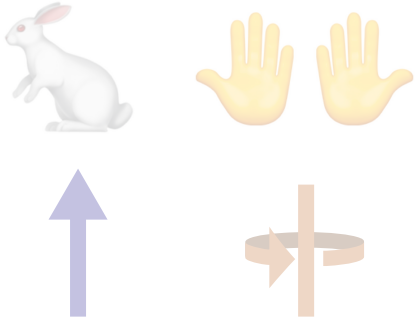
Even parity
(pseudovectors)
Rotation axes



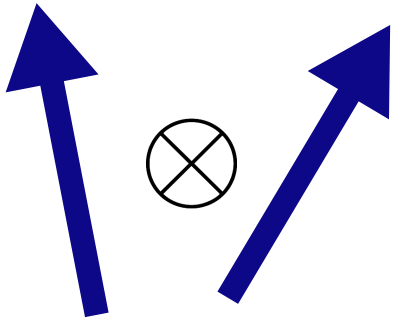
L=2

Even parity
Double-headed Ray

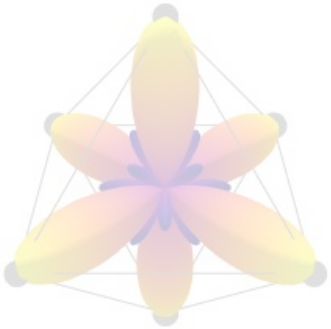
Odd Parity
Helix



Define how data transforms (Irreps)

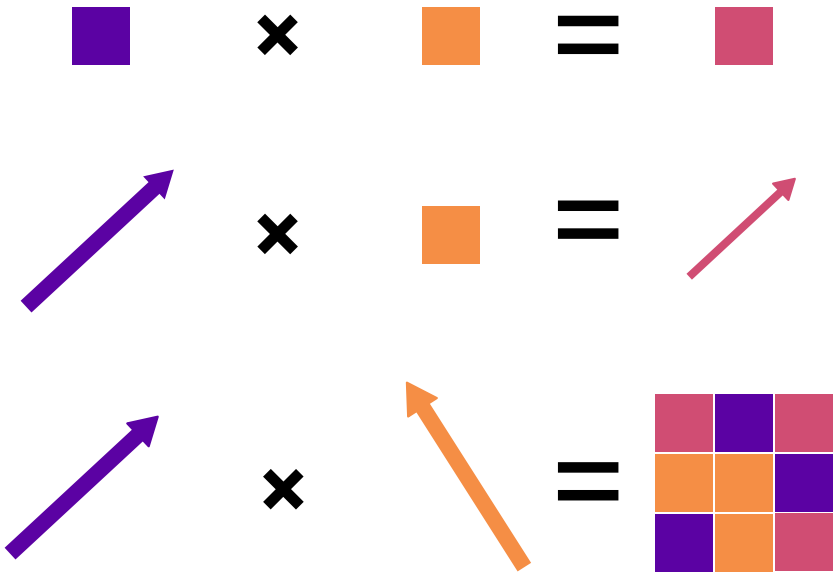


Interact Data
(Tensor products)



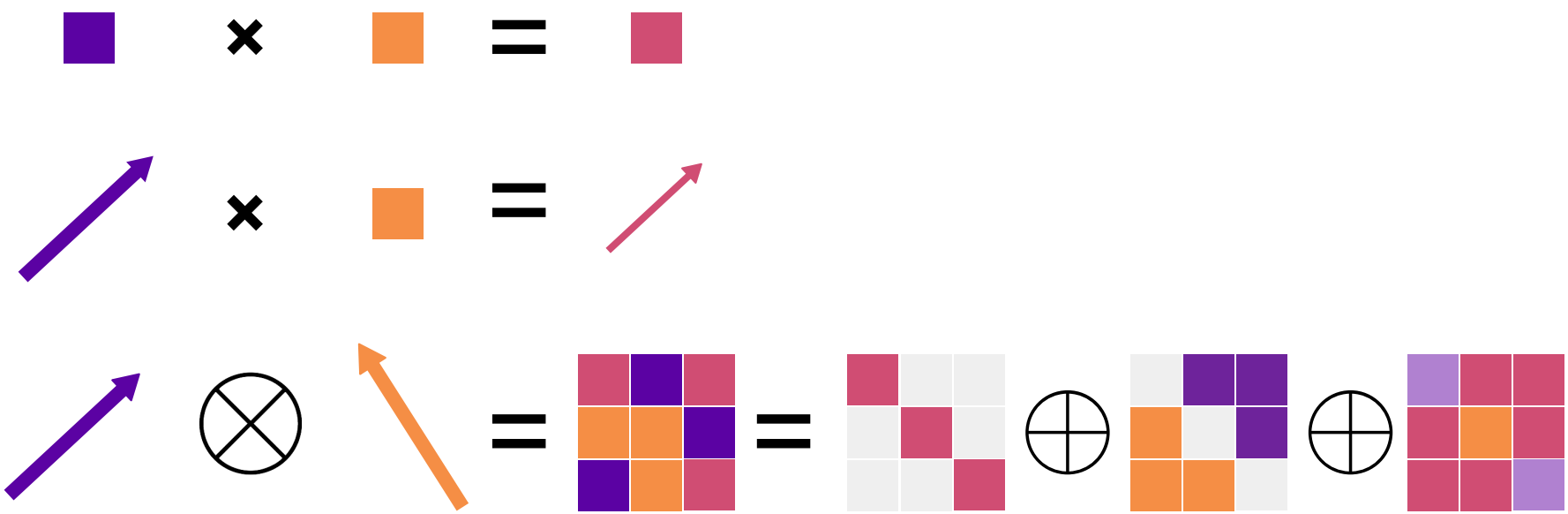
Featurize Geometry
(Spherical Harmonics)

What are all the ways in which we can multiply (interact) scalars and vectors?



Not a scalar or vector.
Does not have a single angular frequency.
Can be decomposed into "irreps".

What are all the ways in which we can multiply (interact) scalars and vectors?

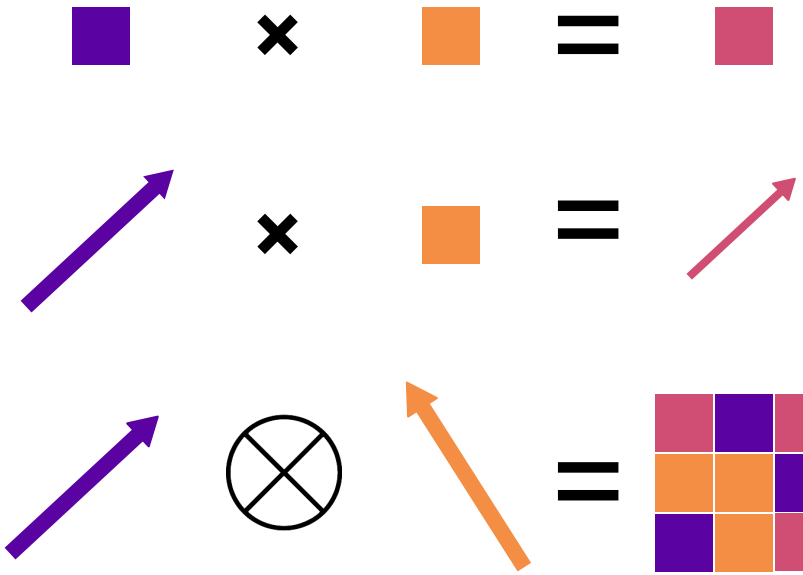


dot product
trace
invariant
L=0, parity=even
1 degree of freedom

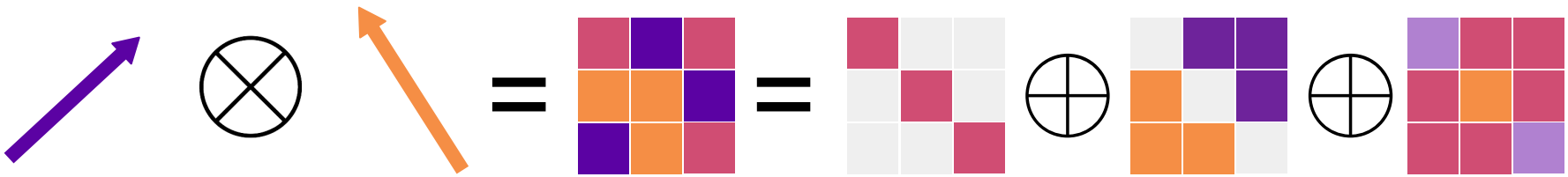
cross-product
antisymmetric
equivariant
L=1, parity=even
3 degrees of freedom

symmetric
traceless
equivariant
L=2, parity=even
5 degrees of freedom

What are all the ways in which we can multiply (interact) scalars and vectors?



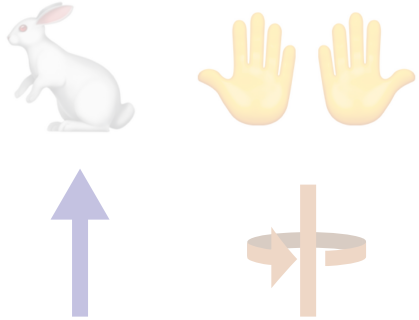
Even if you only give the model scalars and vectors, interacting vectors generates "higher-degree" irreps!
e.g. pseudovector and (L=2, parity=even)



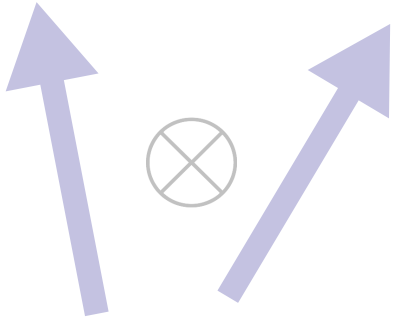
dot product
trace
invariant
L=0, parity=even
1 degree of freedom

cross-product
antisymmetric
equivariant
L=1, parity=even
3 degrees of freedom

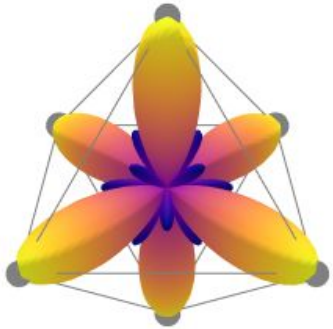
symmetric
traceless
equivariant
L=2, parity=even
5 degrees of freedom



Define how data transforms (Irreps)

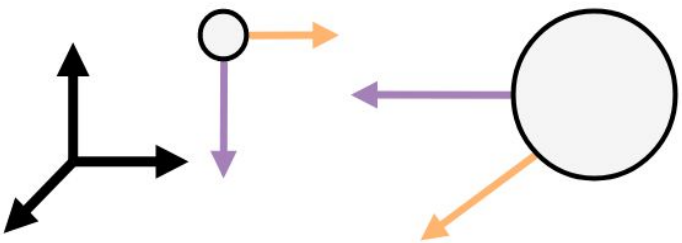


Interact Data (Tensor products)



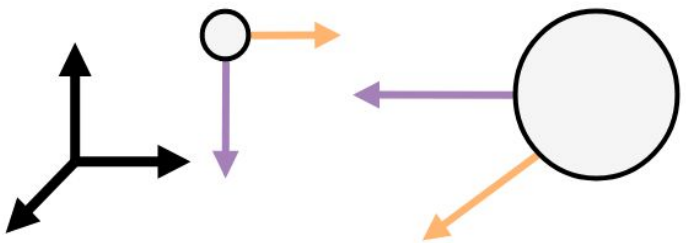
Featurize Geometry **(Spherical Harmonics)**

The input to E(3)NNs is (typically) geometry and (geometric tensor) features on that geometry.



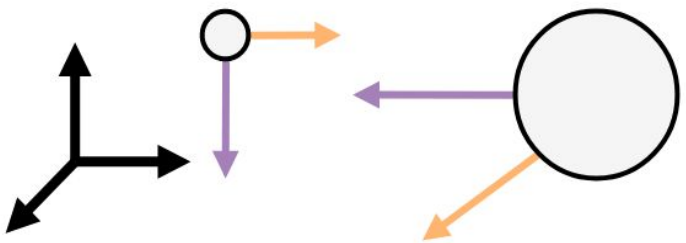
```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
...
```

The input to E(3)NNs is (typically) geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity (as irreps).



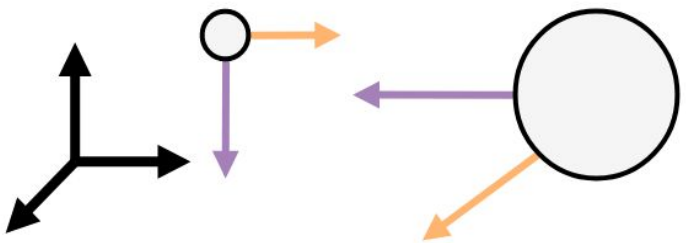
```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even
vector = e3nn.o3.Irrep("1o")  # L=1, odd
irreps = 1 * scalar + 1 * vector + 1 * vector
```

The input to E(3)NNs is (typically) geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity (as irreps).



```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even
vector = e3nn.o3.Irrep("1o")  # L=1, odd
irreps = 1 * scalar + 1 * vector + 1 * vector
```

The input to E(3)NNs is (typically) geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity (as irreps).

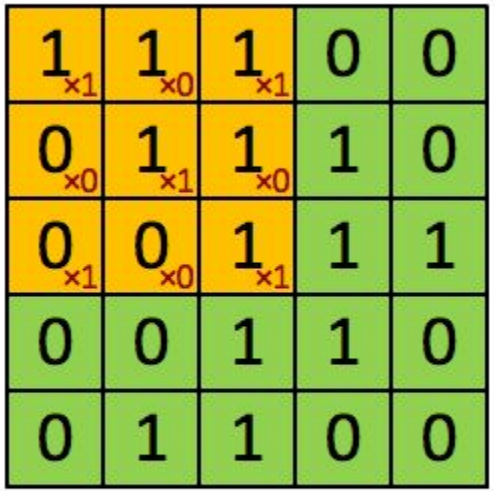


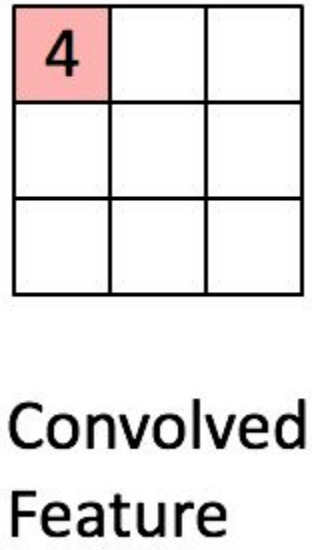
```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even
vector = e3nn.o3.Irrep("1o")  # L=1, odd
irreps = 1 * scalar + 1 * vector + 1 * vector
```

What do we do with the geometry?
Typically, we use convolution operations in the network to interact geometry with features.

A quick recap on convolutions: A filter is convolved with an image to produce a new image.

(Technically… cross-correlated)





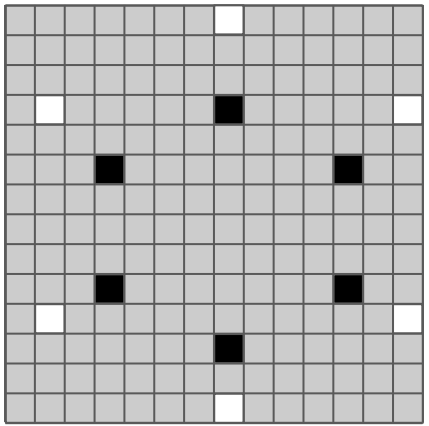
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

In E(3)NNs we can use convolutions to interact point geometry with features on points.

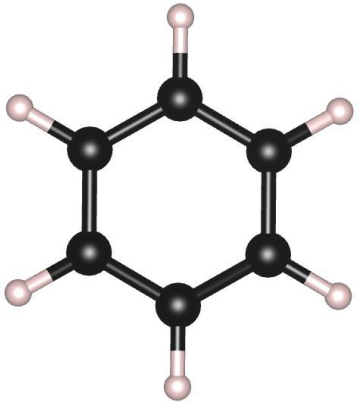
We can operate any geometric data: voxels, meshes, points, etc.

For atoms, we use points.

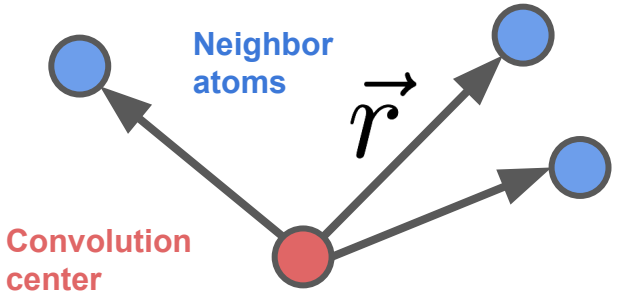
Images of atomic systems are sparse and imprecise.



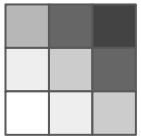
vs.



We use continuous convolutions with atoms as convolution centers.



filter



filter function

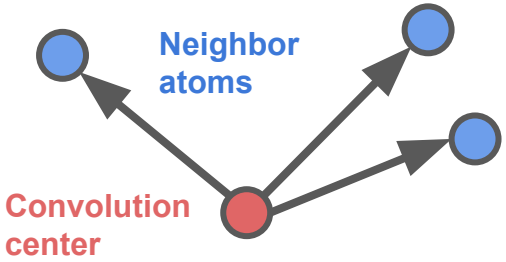
$$W(\vec{r})$$

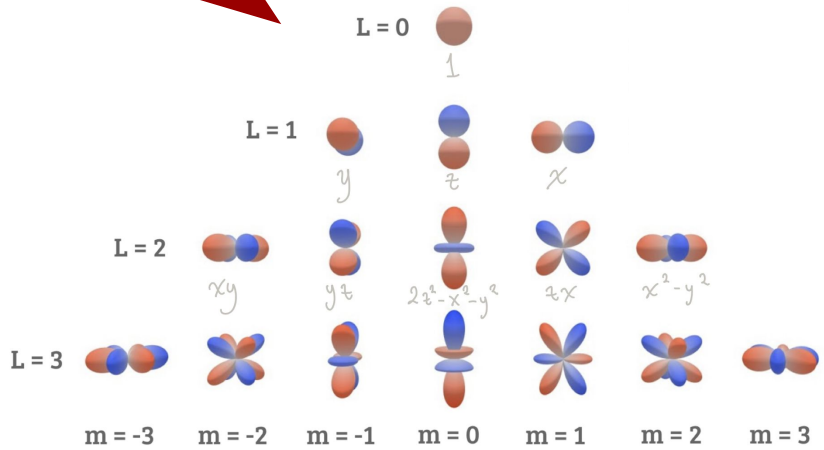
O(3) equivariant convolutional filters are based on learned radial functions and spherical harmonics…

Filter is indexed by channel c, degree l, and order m.

To "interact" filter with node features, you use tensor products.

$$W_{clm}(\vec{r}) = R_{cl}(|\vec{r}|)Y_{lm}(\hat{r})$$

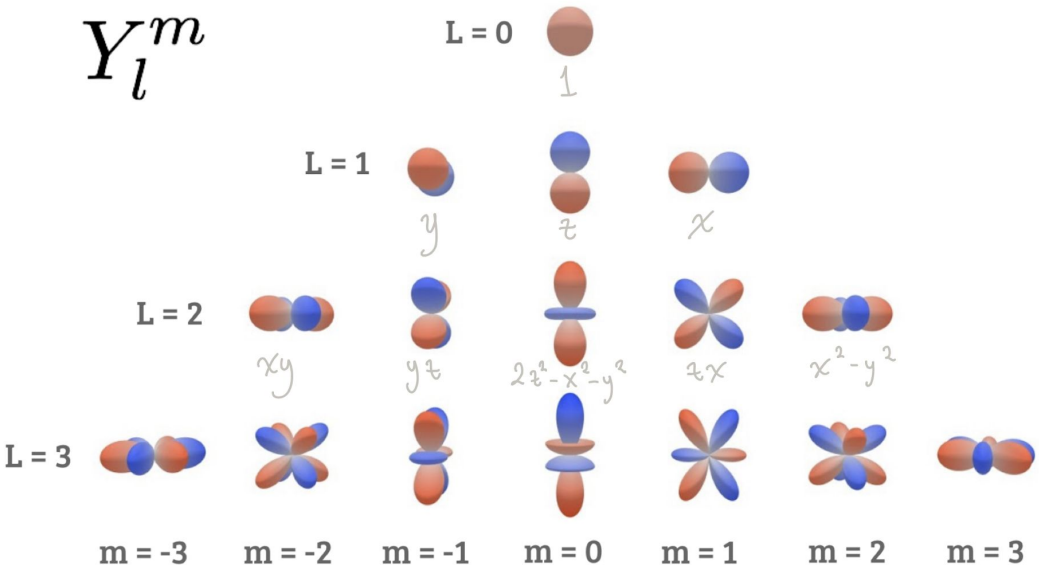




Integrate / sum over space
(e.g. sum over nearest neighbors in case of local filter).

The spherical harmonics are the simplest functions that transform as irreps of O(3).
The L spherical harmonics take in a unit vector and give 2L + 1 coefficients that transform as the (L, parity = $(-1)^L$) irrep.

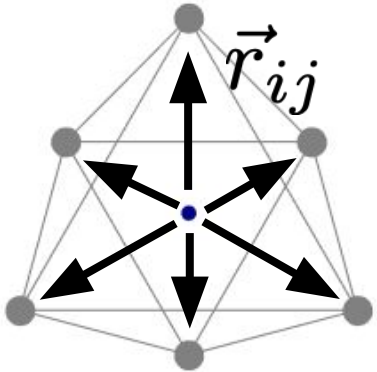
Spherical harmonics




```
from e3nn import o3

sphharm_0 = o3.Irrep("0e")

sphharm_1 = o3.Irrep("1o")

sphharm_2 = o3.Irrep("2e")

sphharm_3 = o3.Irrep("3o")
```

The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

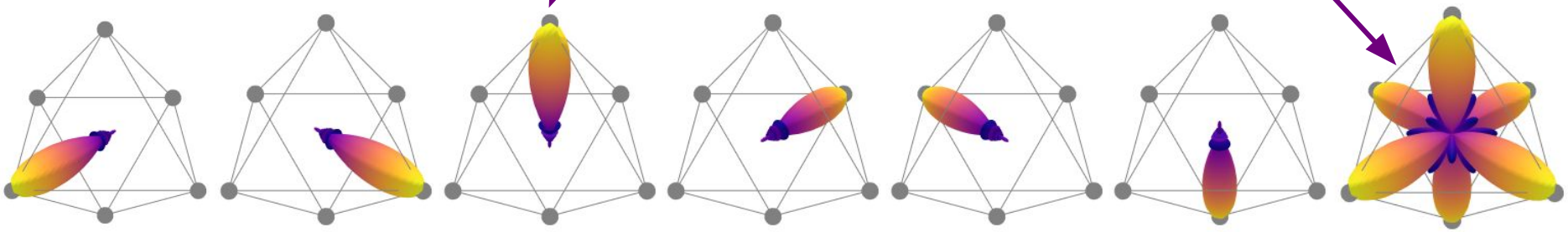
Spherical harmonic projections



"Coefficients"

$$\sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$

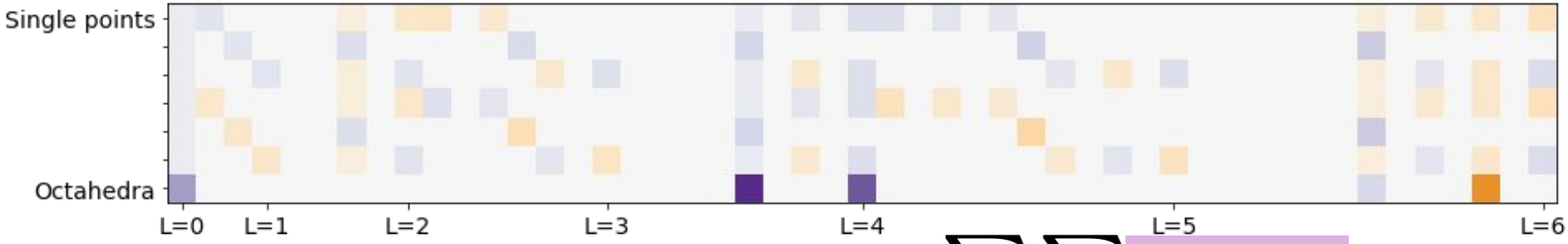
$$\sum_{j} \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$



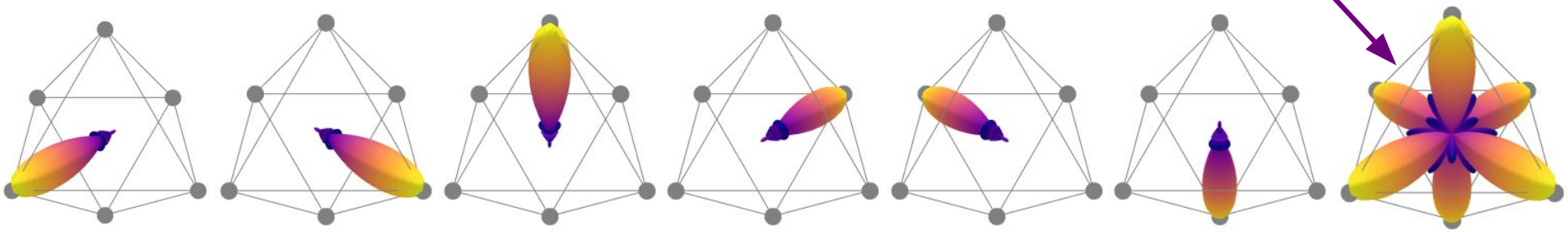
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections

Coefficients attached to each spherical harmonic



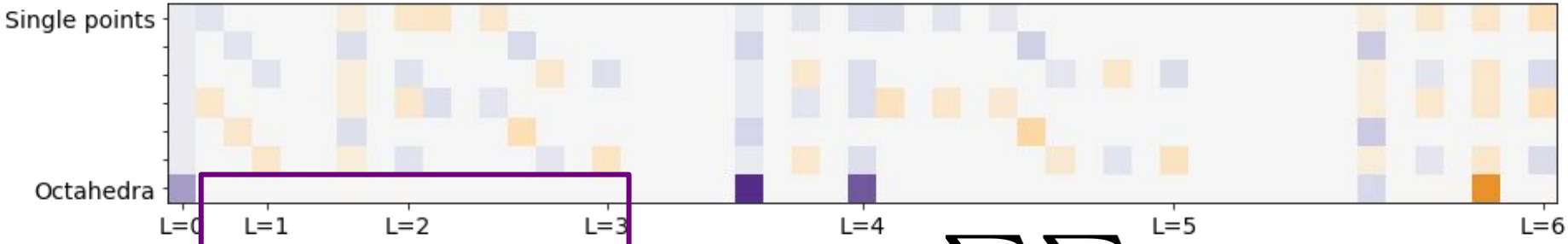
$$\sum_j \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$



The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

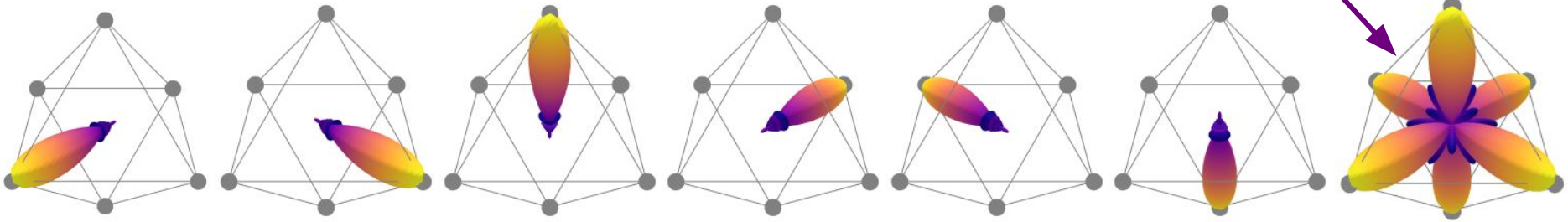
Spherical harmonic projections

Coefficients attached to each spherical harmonic



Typical signature of high symmetry objects: cancelation of terms.

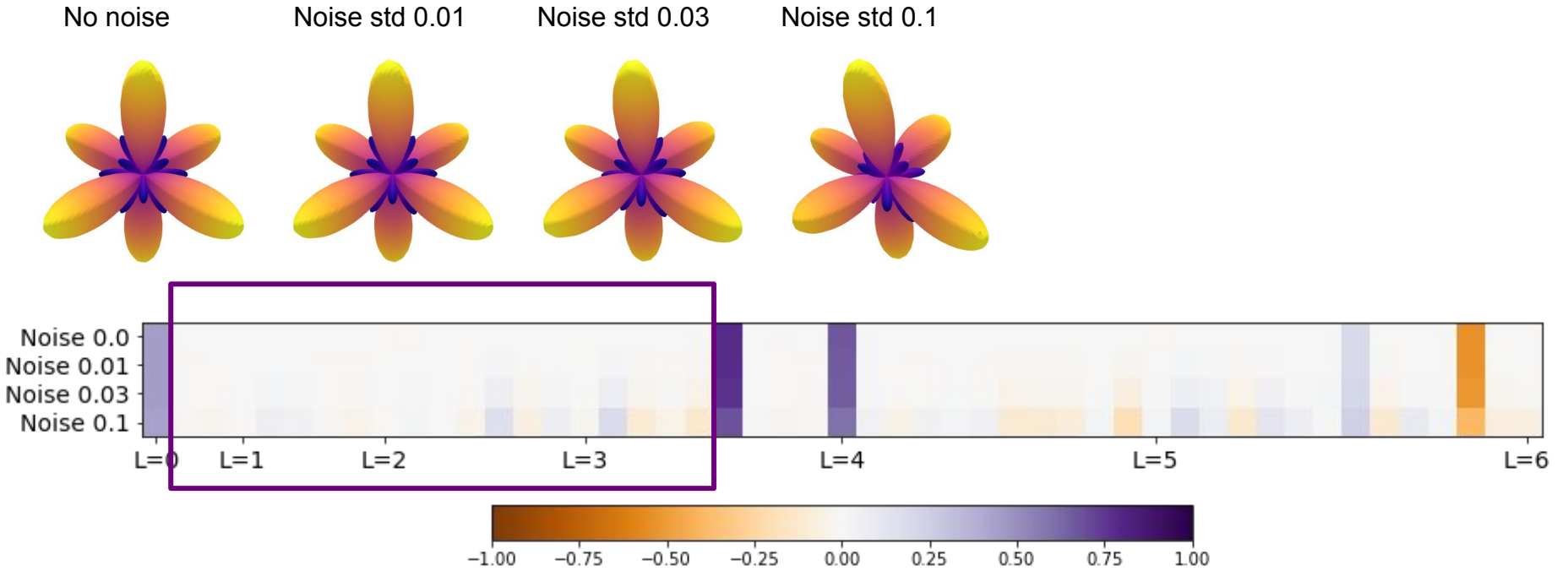
$$\sum_j \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$



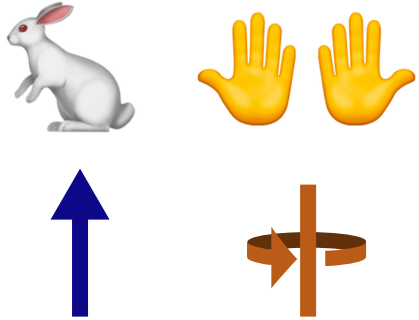
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections

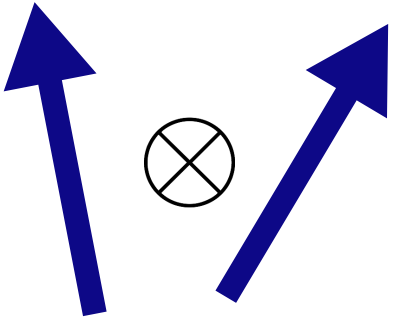
These projections are robust to noise. Symmetry is NOT binary - you still get cancellations.



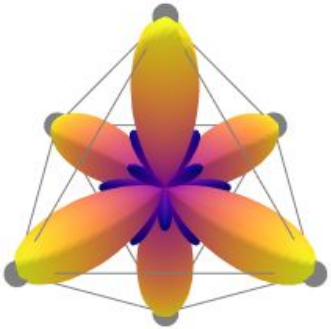
What do these differences give us?



***Define how data transforms* (Irreps)**



***Interact Data* (Tensor products)**



***Featurize Geometry* (Spherical Harmonics)**

We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

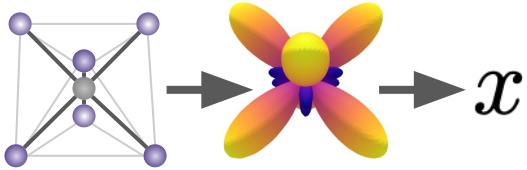
Tuong Phung

Elyssa Hofgard

Aria Mansouri Tehrani

We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

Example: *Bispectra*



Take invariants (scalars) from…

$$x \otimes x \otimes x$$

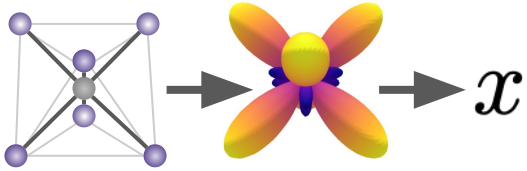
Tuong Phung

Elyssa Hofgard

Aria Mansouri Tehrani

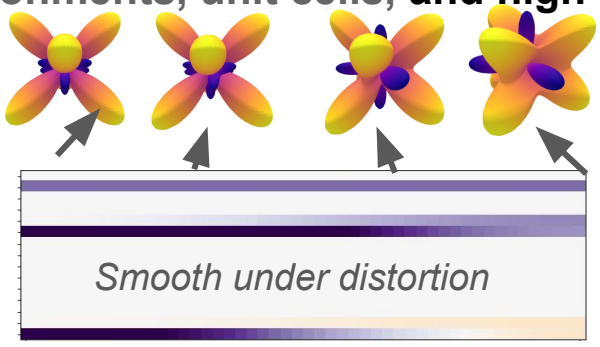
We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

Example: *Bispectra*



Take invariants (scalars) from…

$$x \otimes x \otimes x \rightarrow$$



Octahedron

Trigonal Prism

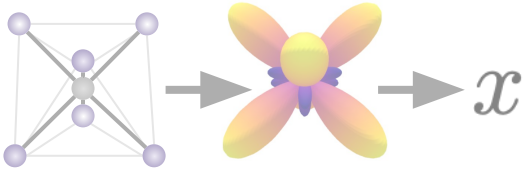
Tuong Phung

Elyssa Hofgard

Aria Mansouri Tehrani

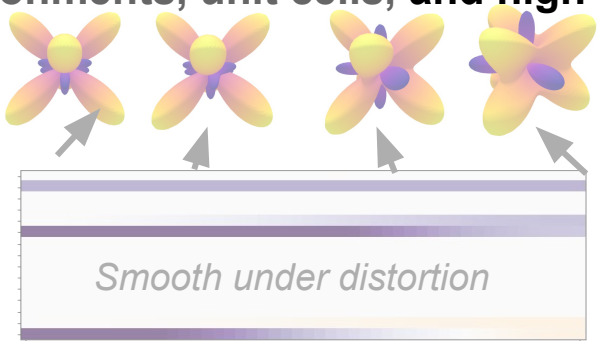
We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

Example: *Bispectra*



Take invariants (scalars) from…

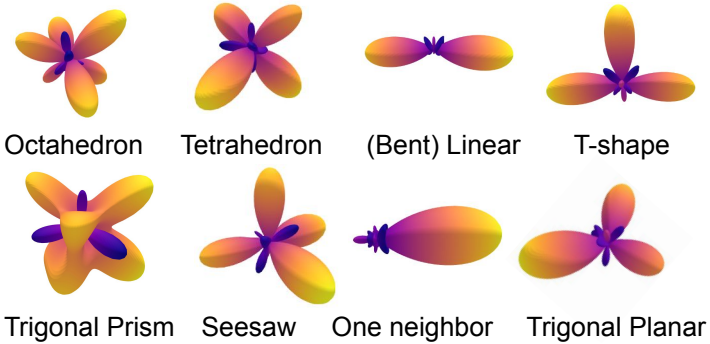
$$x \otimes x \otimes x \rightarrow$$



Octahedron

Trigonal Prism

Clustering of bispectra from materials data and *inverting back to geometry* yields common local environments



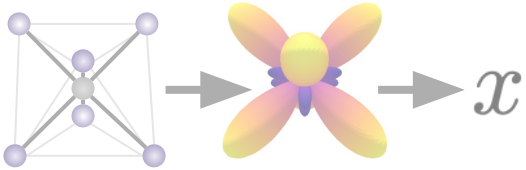
Tuong Phung

Elyssa Hofgard

Aria Mansouri Tehrani

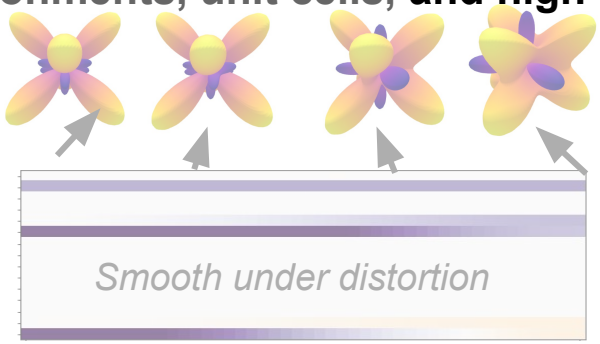
We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

Example: *Bispectra*



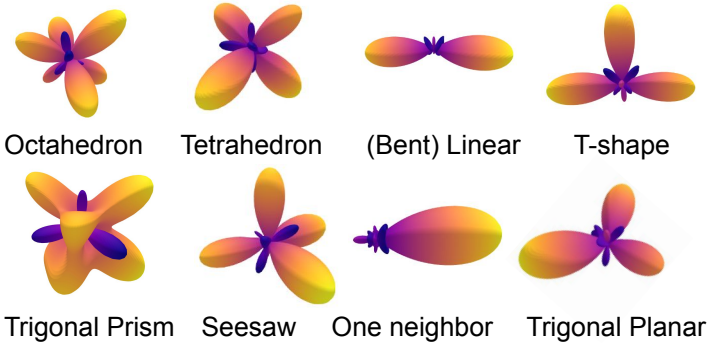
Take invariants (scalars) from…

$$x \otimes x \otimes x \rightarrow$$



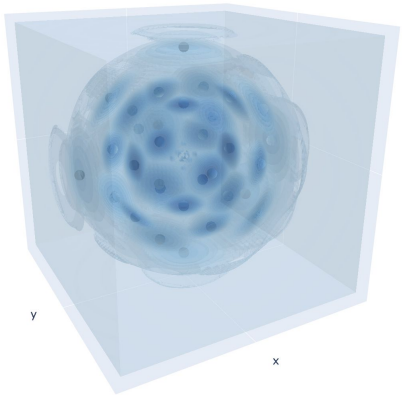
Octahedron Trigonal Prism

Clustering of bispectra from materials data and *inverting back to geometry* yields common local environments



Bispectra (with radial functions) can be used as a convention free descriptor of lattice geometry.

Reciprocal lattice points up to k_{max}



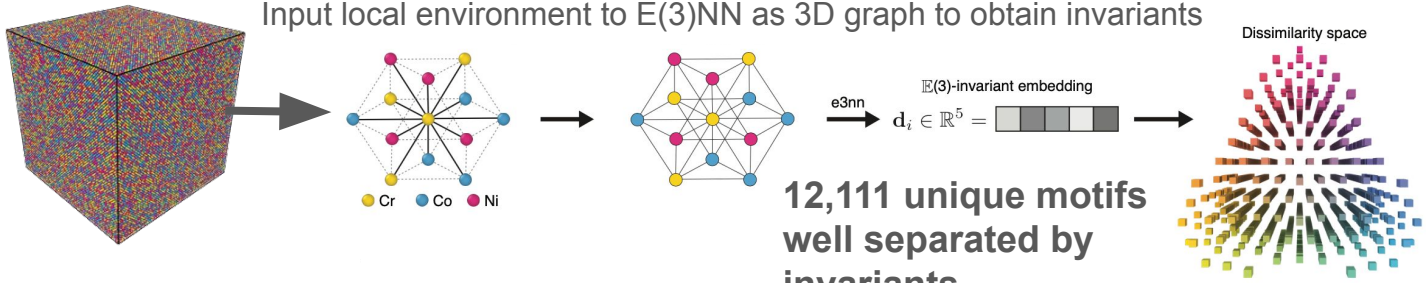
Tuong Phung

Elyssa Hofgard

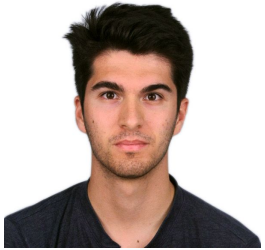
Aria Mansouri Tehrani

We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

Input local environment to E(3)NN as 3D graph to obtain invariants



12,111 unique motifs well separated by invariants



Killian Sheriff

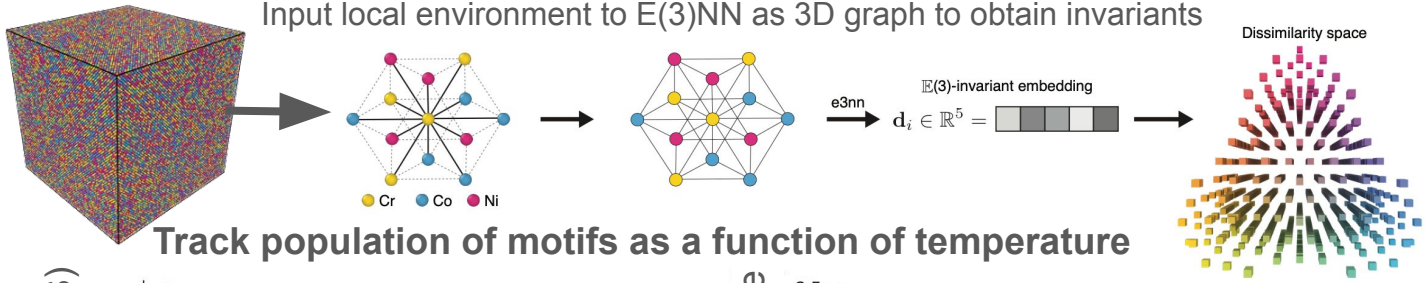
Yifan (Henry) Cao

Rodrigo Freitas

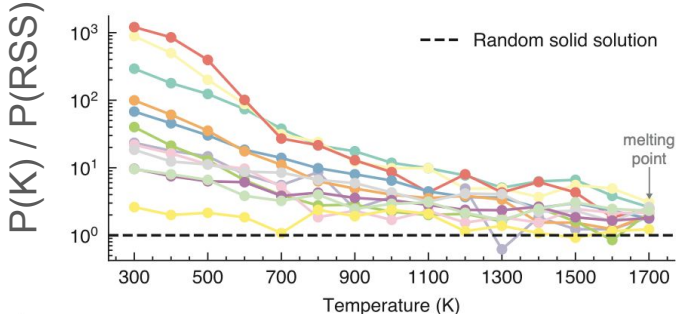
arXiv:2311.01545

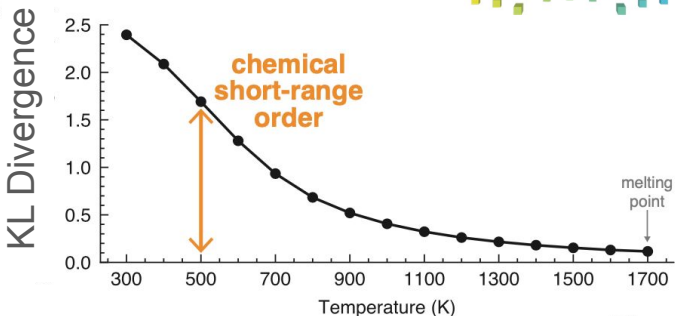
We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.

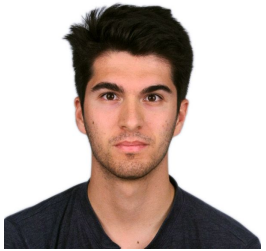
Input local environment to E(3)NN as 3D graph to obtain invariants



Track population of motifs as a function of temperature







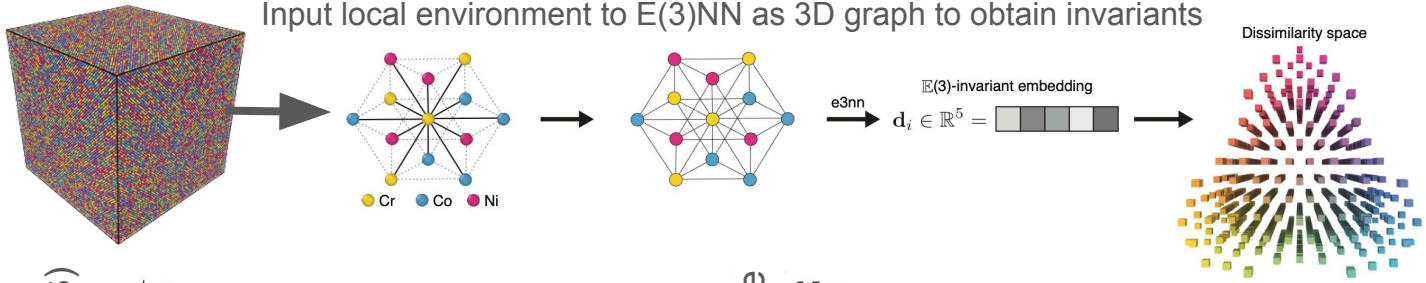
Killian Sheriff

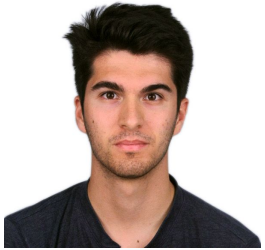
Yifan (Henry) Cao

Rodrigo Freitas

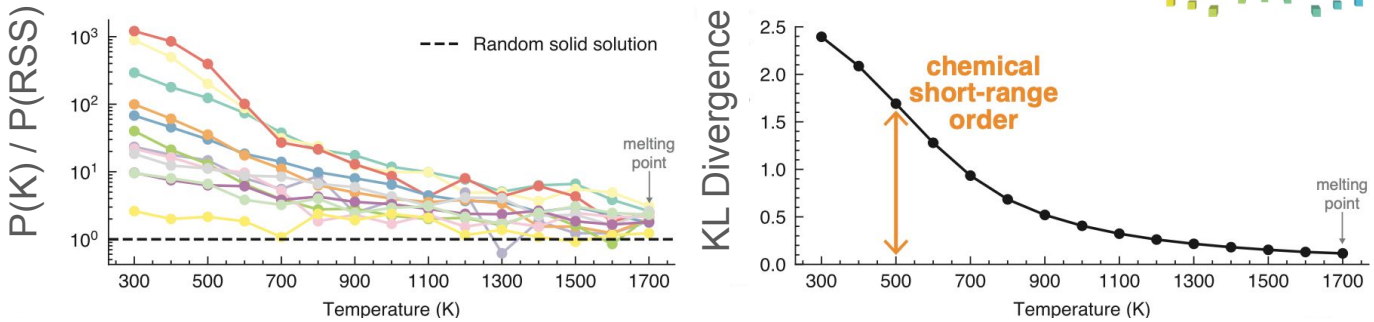
arXiv:2311.01545

We've used the tensor interactions that E(3)NNs are built from to explore trends in local environments, unit cells, and high entropy alloys.



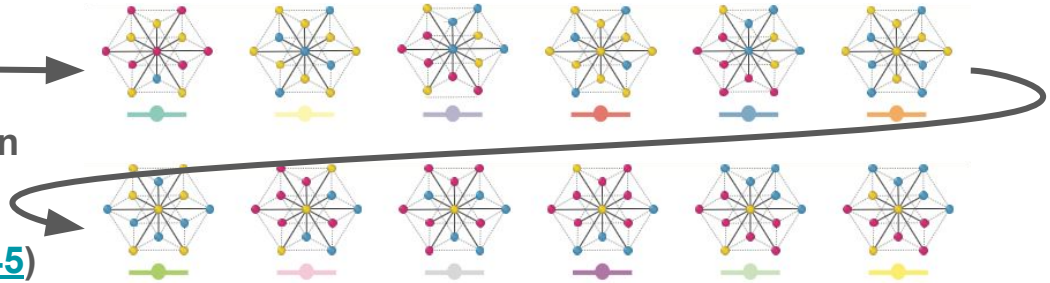


Killian Sheriff



Yifan (Henry) Cao

Identify local env. with increasing lattice distortion

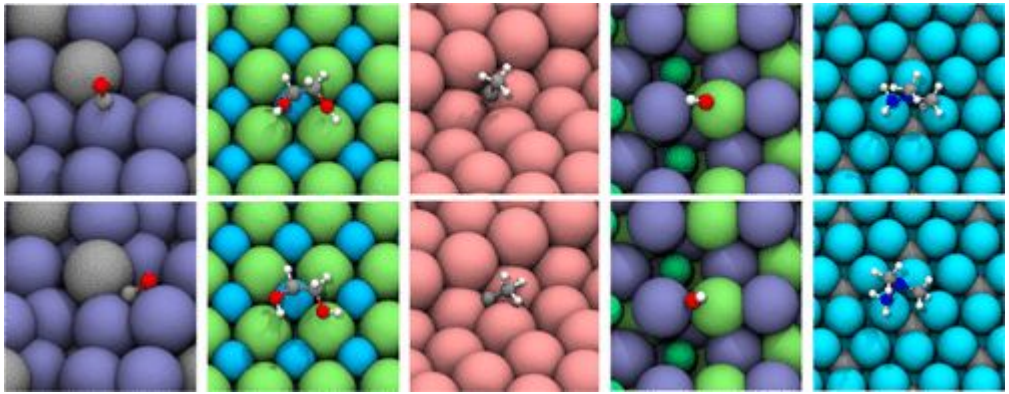


Rodrigo Freitas

(arXiv:2311.01545)

We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and E(3)NNs are state-of-art for OC20, OC22 and ODAC23.

Open Catalysis 2020 Dataset (examples)



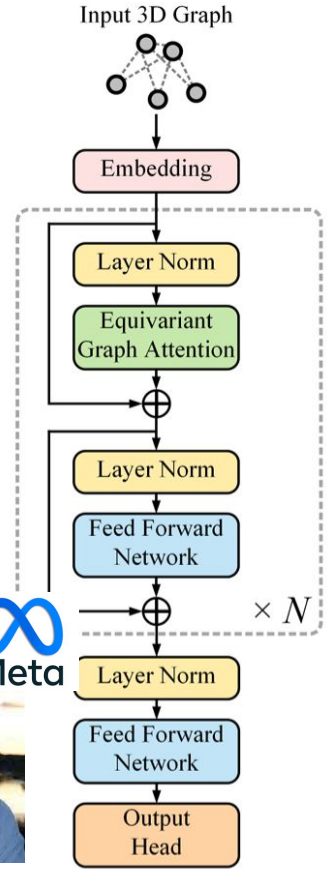
Predict energy, forces of given configurations and relaxed structures.

Graph attention built from tensor products of irrep features

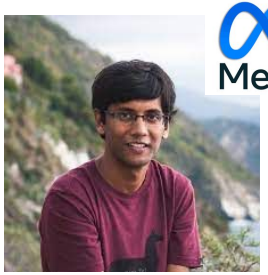
EquiformerV2
ICLR 2024
(arXiv:2306.12059**)**

Equiformer:
Equivariant graph attention trans**former**
ICLR 2023
(arXiv:2206.11990**)**

First equivariant transformer to be state-of-art on multiple atomistic benchmarks (QM9, MD17, OC20).




Yi-Lun Liao


Abhishek Das

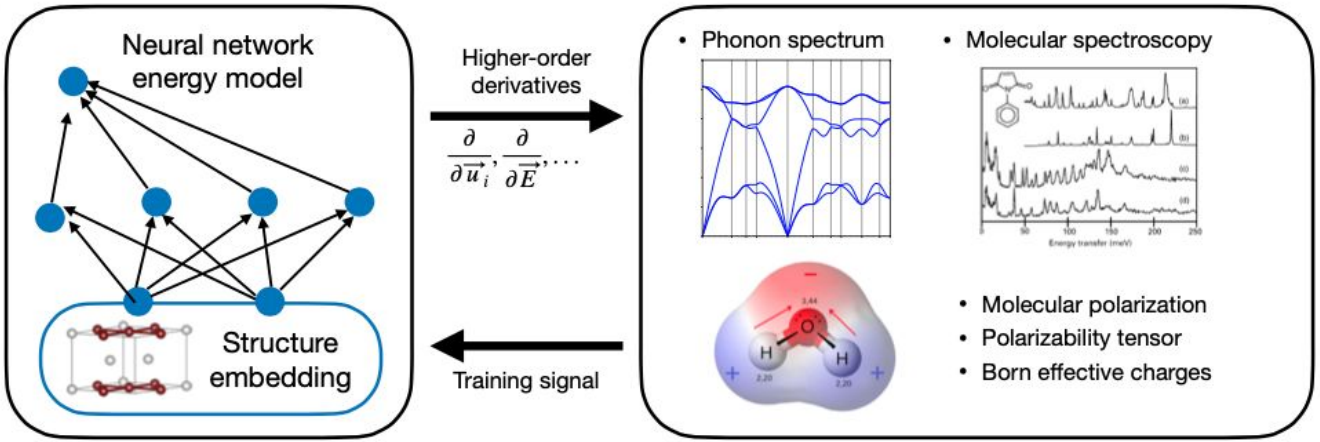

Brandon Wood

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax.

(Ultimately paving the way to training directly on spectra...)

(arXiv:2403.11347)



Shiang Fang

Mario Geiger

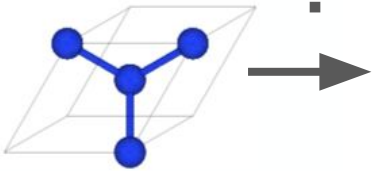
Joseph Checkelsky

Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

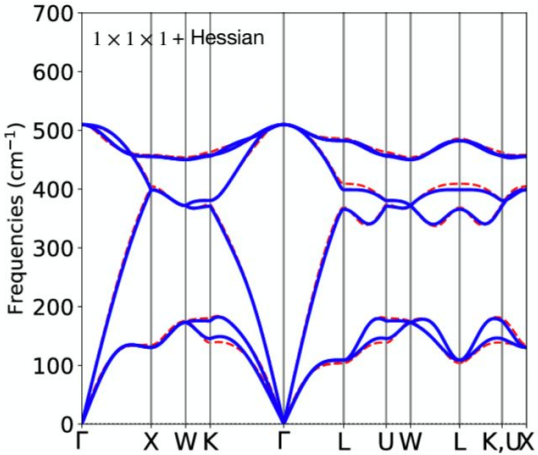
We've used E(3)NNs to build data-efficient and scalable models of physical processes.

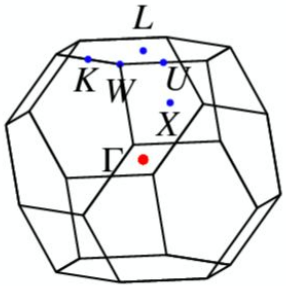
Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax. (arXiv:2403.11347)

How to go from structure to phonons in a way that is straightforward to do with deep learning frameworks and preserves physics formalism?

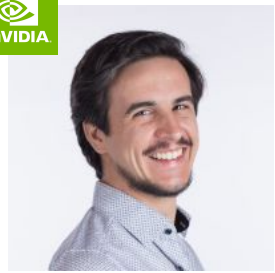


?





Shiang Fang



Mario Geiger

Joseph Checkelsky

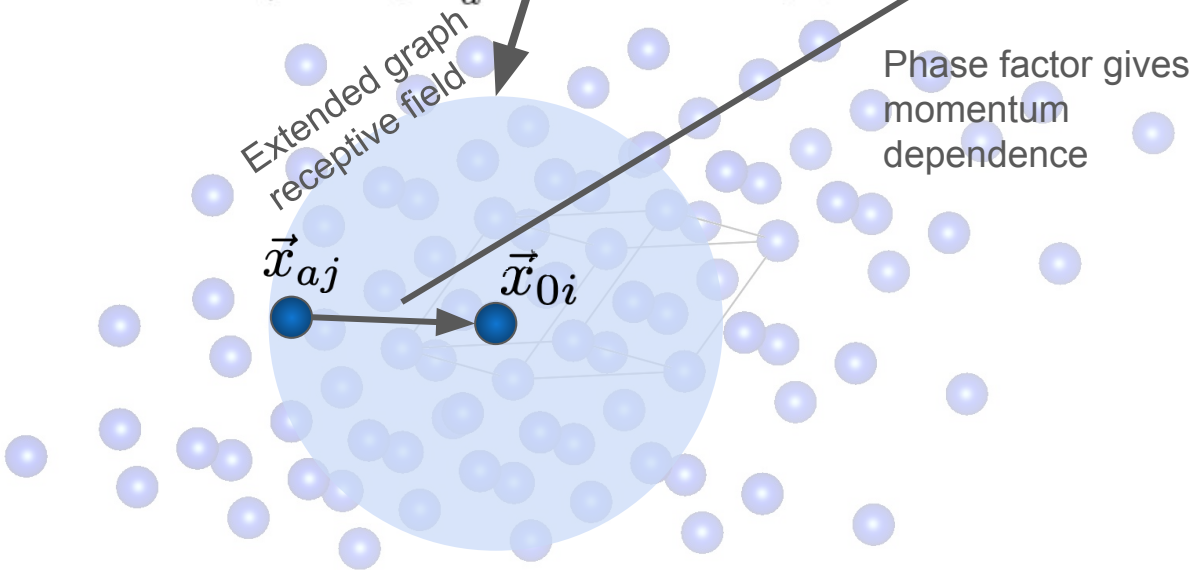
Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax. (arXiv:2403.11347)

Phonons from the Dynamical matrix built from force constants.

$$D_{\alpha\beta}(ij,\vec{q}) = \frac{1}{\sqrt{m_i m_j}} \sum_a \frac{\partial^2 E}{\partial(\vec{x}_{0i})_\alpha \partial(\vec{x}_{aj})_\beta} e^{i\vec{q}\cdot(\vec{x}_{aj}-\vec{x}_{0i})}$$



Shiang Fang

Mario Geiger

Joseph Checkelsky

Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax.
(arXiv:2403.11347)

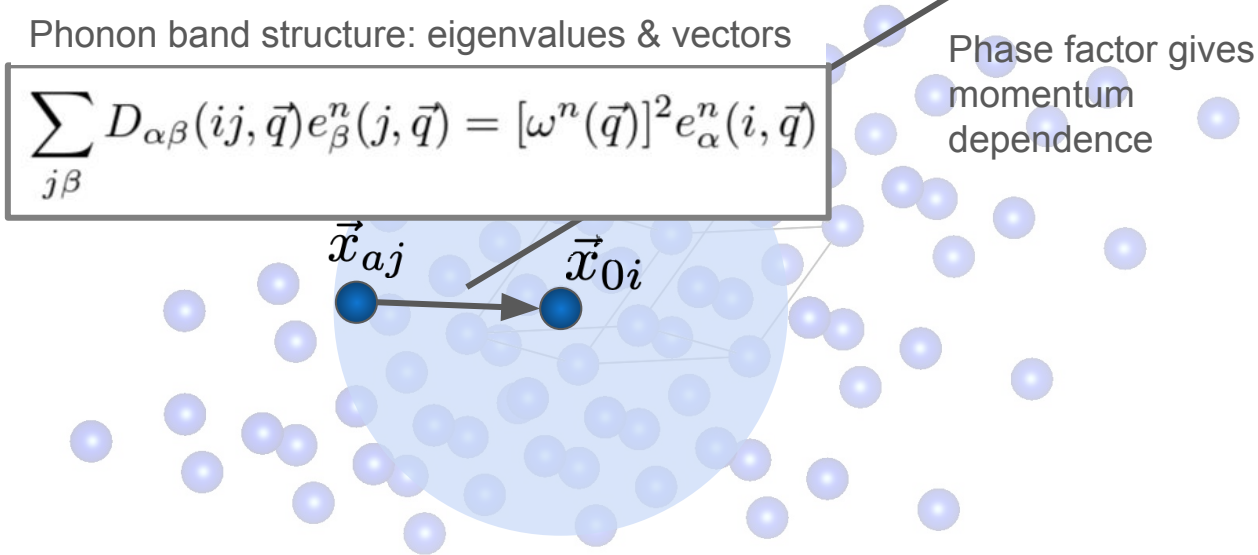
Phonons from the Dynamical matrix built from force constants.

$$D_{\alpha\beta}(ij,\vec{q}) = \frac{1}{\sqrt{m_i m_j}} \sum_a \frac{\partial^2 E}{\partial(\vec{x}_{0i})_\alpha \partial(\vec{x}_{aj})_\beta} e^{i\vec{q}\cdot(\vec{x}_{aj}-\vec{x}_{0i})}$$

Phonon band structure: eigenvalues & vectors

$$\sum_{j\beta} D_{\alpha\beta}(ij,\vec{q}) e^n_\beta(j,\vec{q}) = [\omega^n(\vec{q})]^2 e^n_\alpha(i,\vec{q})$$

Phase factor gives momentum dependence



Shiang Fang

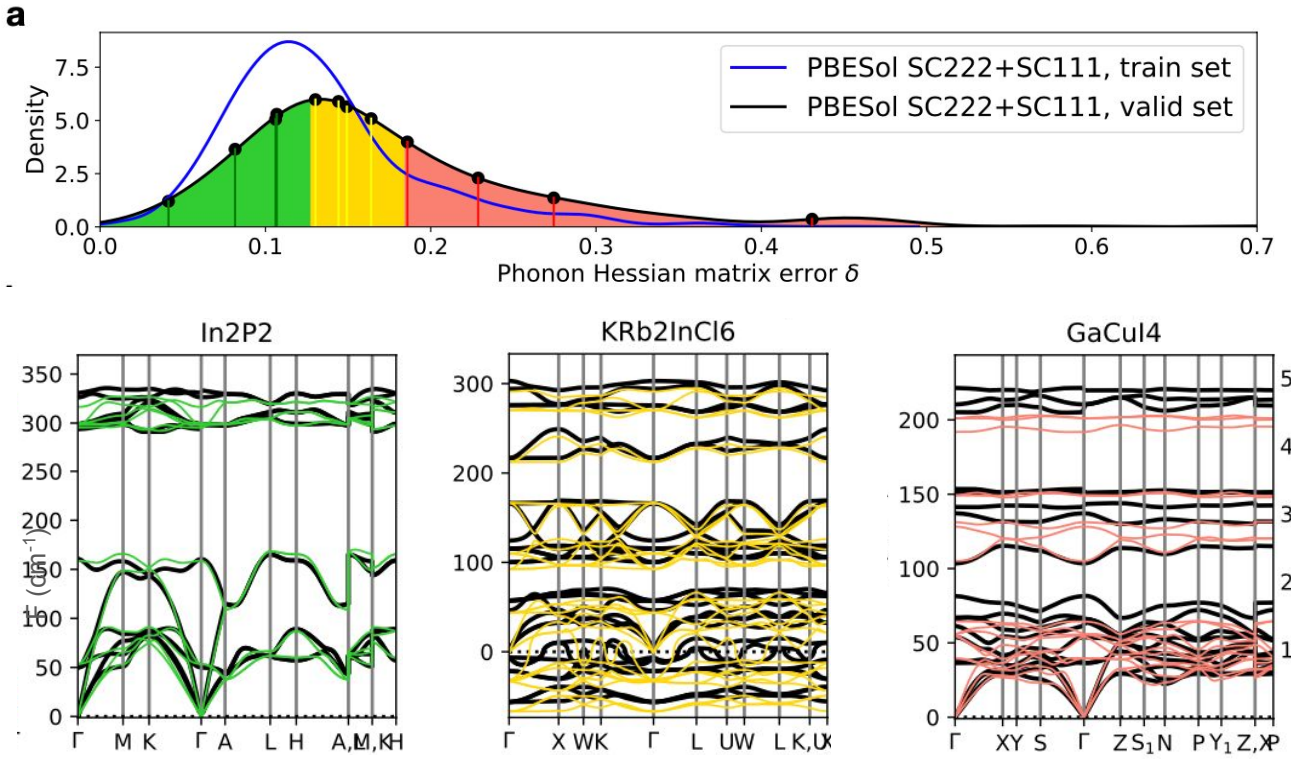
Mario Geiger

Joseph Checkelsky

Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax. (arXiv:2403.11347)



Shiang Fang

Mario Geiger

Joseph Checkelsky

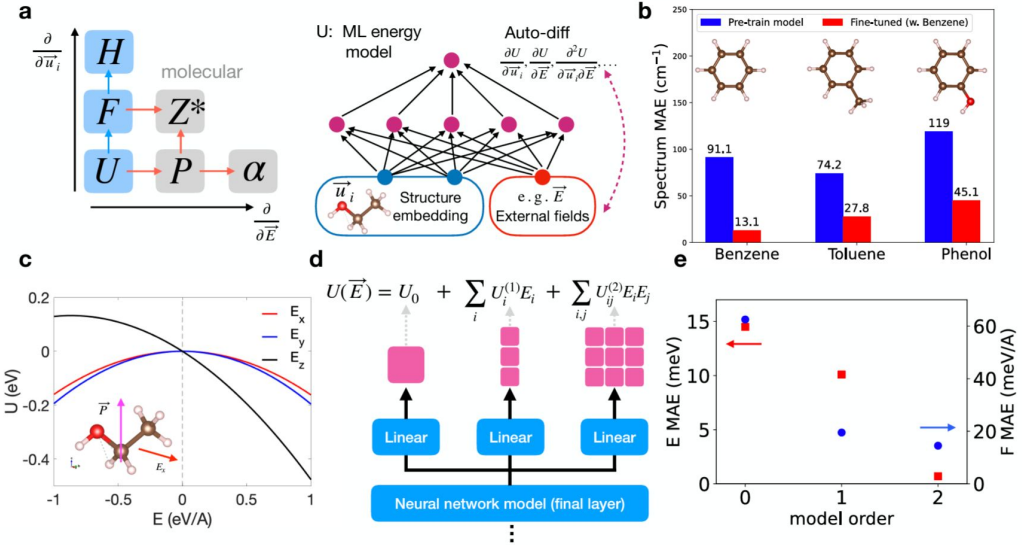
Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax. (arXiv:2403.11347)

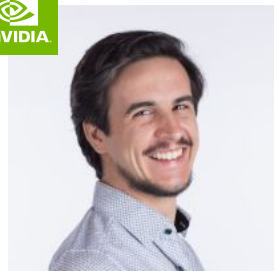
Full access to eigenmodes

- ⇨ Determine symmetry of states (e.g. IR vs. Raman active)
- ⇨ Compare directly to experiment (e.g. spectroscopies, neutron)
- ⇨ Backpropogate through energy model
- ⇨ Generalize to other derivatives (e.g. E field)



Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

Shiang Fang



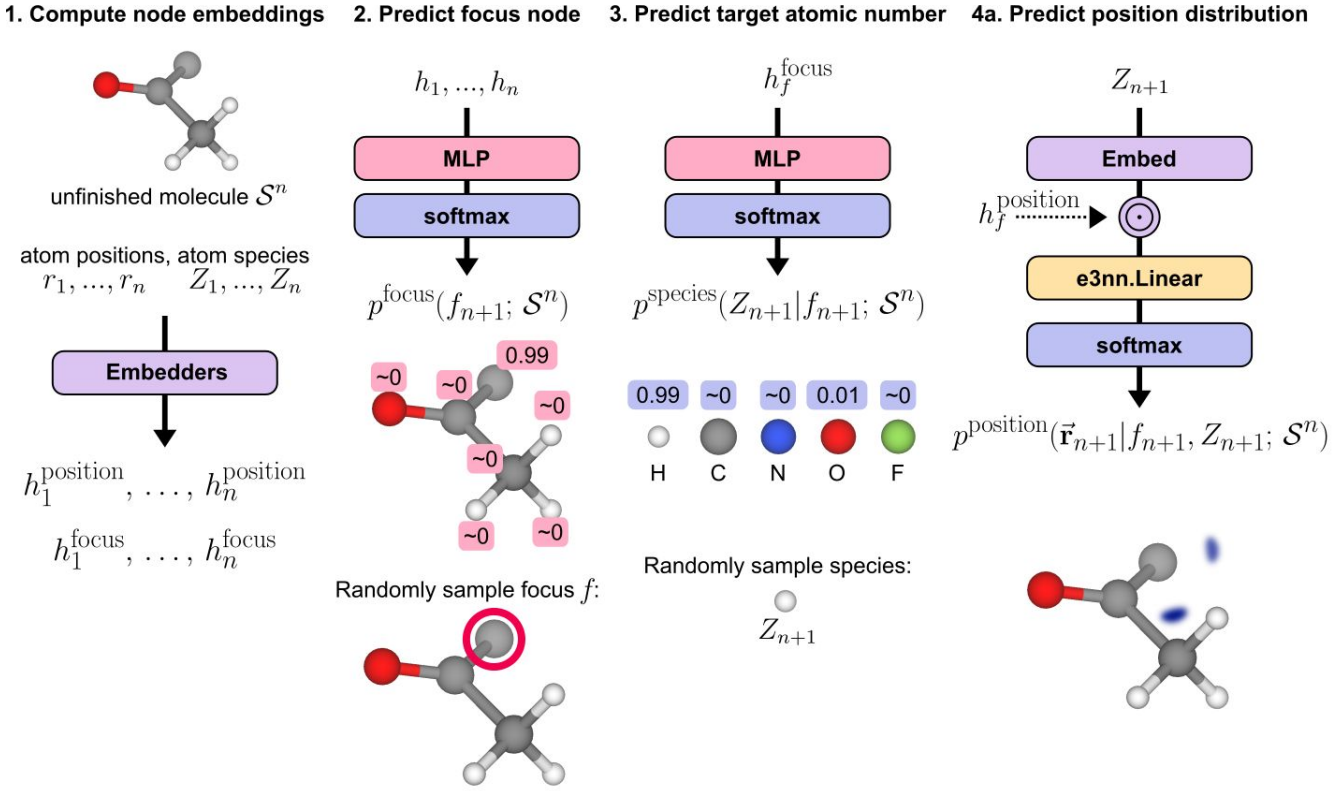
Mario Geiger

Joseph Checkelsky

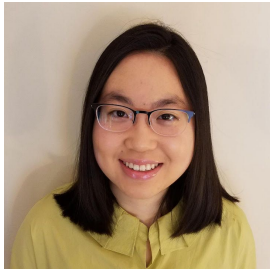
We've used E(3)NNs to build generative models of atomic systems.

Symphony: Symmetry-equivariant Point-Centered Harmonics for Molecular Generation

ICLR 2024 (arXiv:2311.16199)



Ameya Daigavane

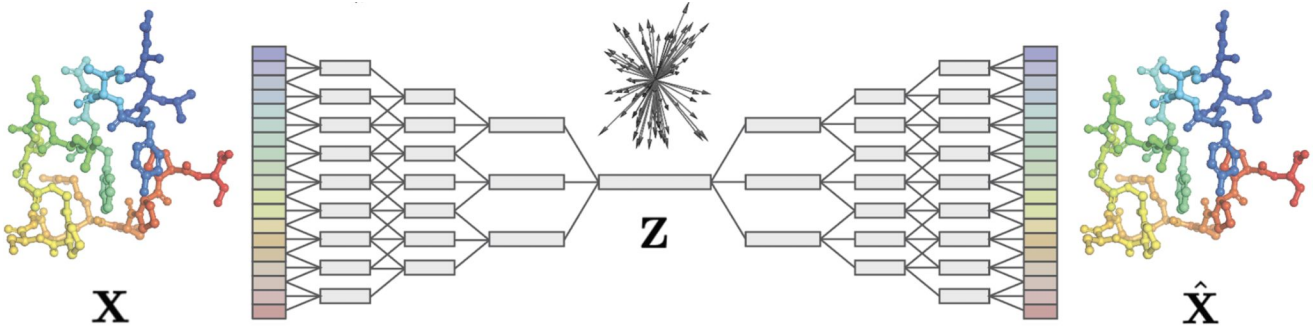


Song Kim

Mario Geiger

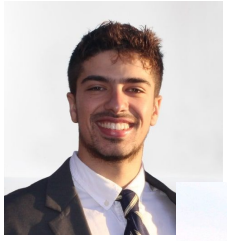
We've used E(3)NNs to build generative models of atomic systems.

Ophiuchus: Scalable Model of Protein Structures through Hierarchical Coarse-Graining SO(3)-equivariant Autoencoders (arXiv:2310.02508)

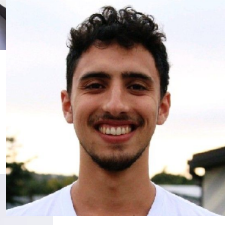


Iteratively coarsen along backbone.

Iteratively build up backbone.

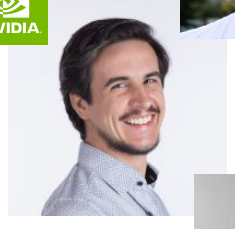


Allan Costa



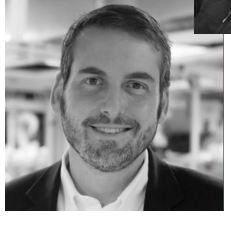
Ilan Mitnikov

NVIDIA



Mario Geiger

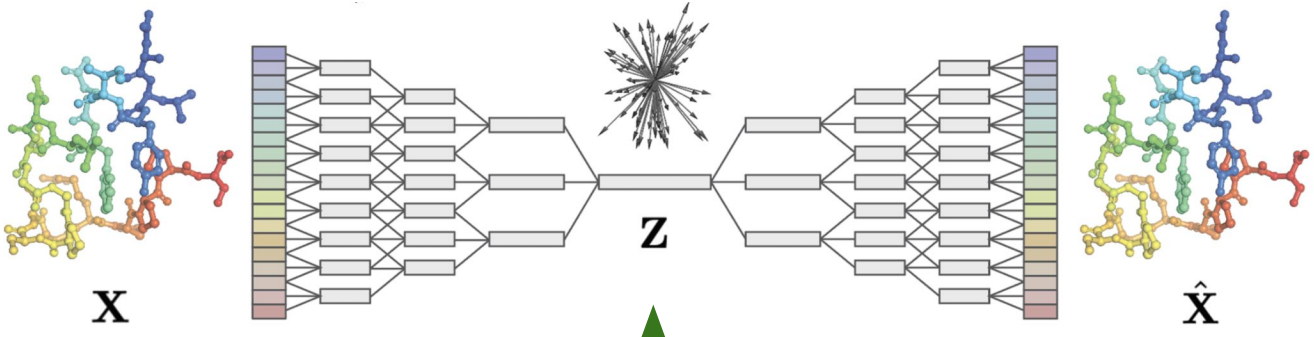
Manvitha Ponnapati



Joe Jacobson

We've used E(3)NNs to build generative models of atomic systems.

Ophiuchus: Scalable Model of Protein Structures through Hierarchical Coarse-Graining SO(3)-equivariant Autoencoders (arXiv:2310.02508)

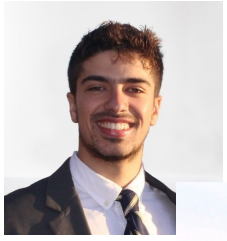


Iteratively coarsen along backbone.

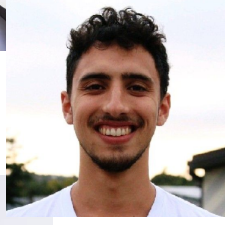
Latent has no geometry, just tensor features.

Can generate latent with separate diffusion model.

Iteratively build up backbone.



Allan Costa

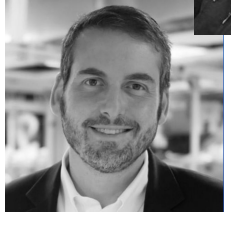


Ilan Mitnikov

NVIDIA

Mario Geiger

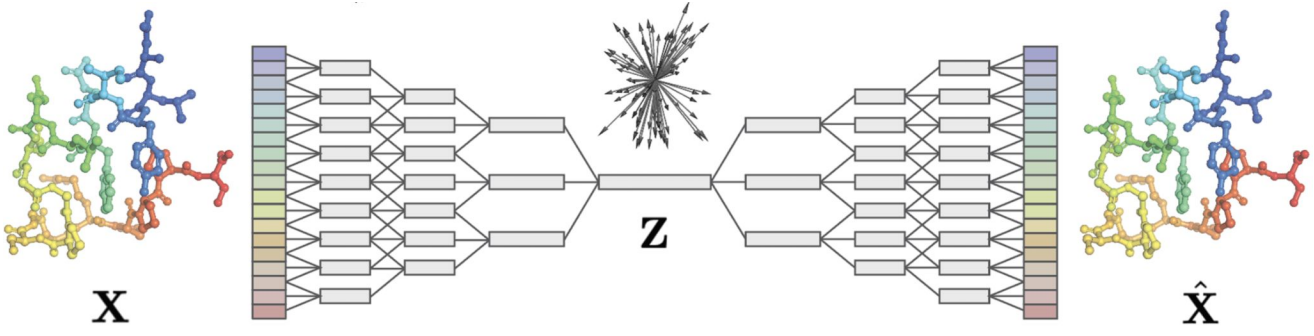
Manvitha Ponnapati



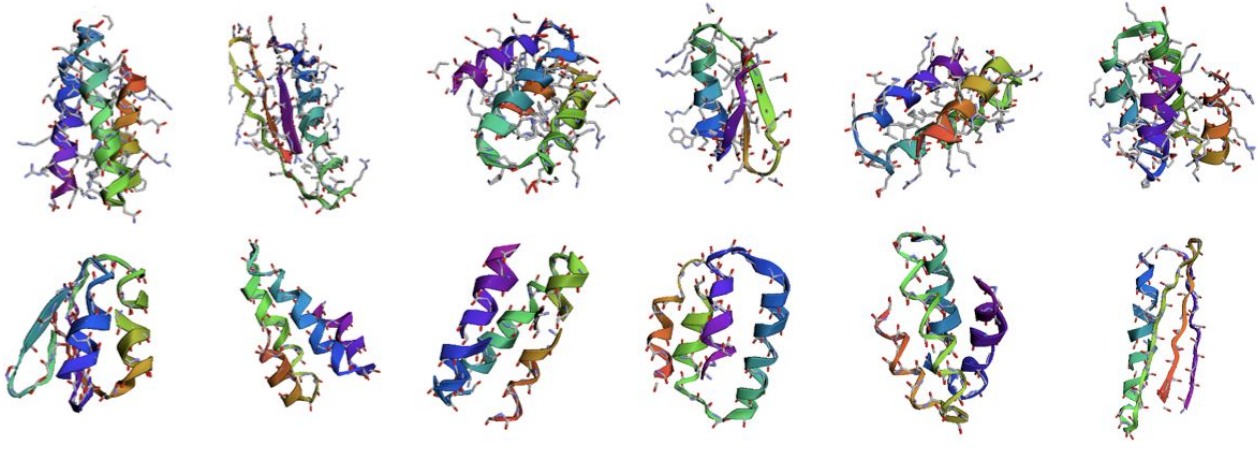
Joe Jacobson

We've used E(3)NNs to build generative models of atomic systems.

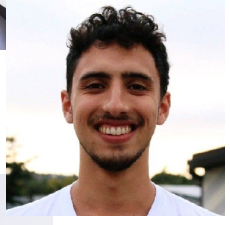
Ophiuchus: Scalable Model of Protein Structures through Hierarchical Coarse-Graining SO(3)-equivariant Autoencoders (arXiv:2310.02508)



Reconstructed all-atom proteins structures

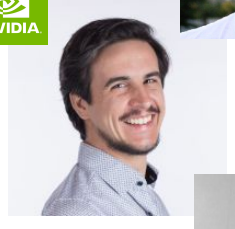


Allan Costa



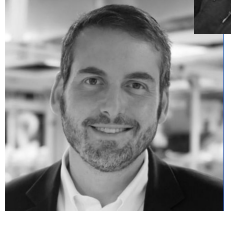
Ilan Mitnikov

NVIDIA



Mario Geiger

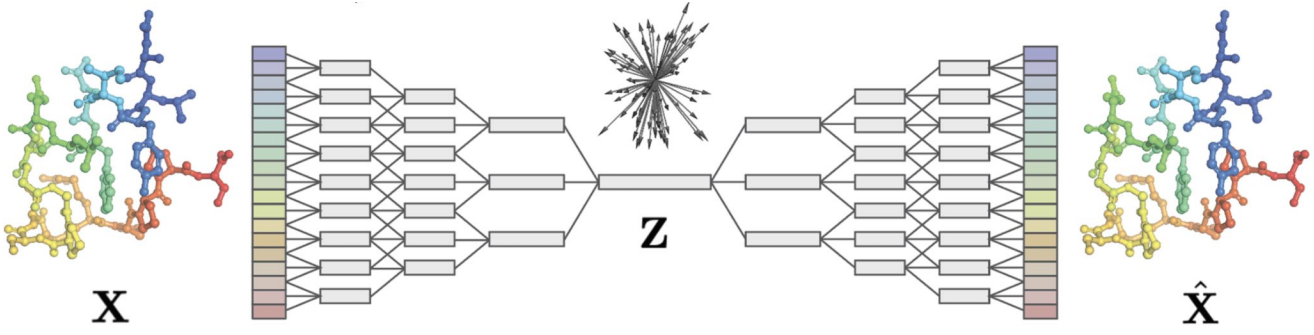
Manvitha Ponnapati



Joe Jacobson

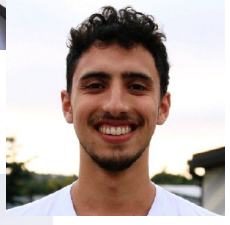
We've used E(3)NNs to build generative models of atomic systems.

Ophiuchus: Scalable Model of Protein Structures through Hierarchical Coarse-Graining SO(3)-equivariant Autoencoders (arXiv:2310.02508)



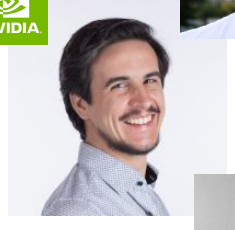
Latent interpolation between protein conformations are physical.

Allan Costa



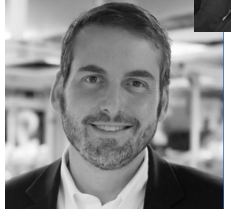
Ilan Mitnikov

NVIDIA



Mario Geiger

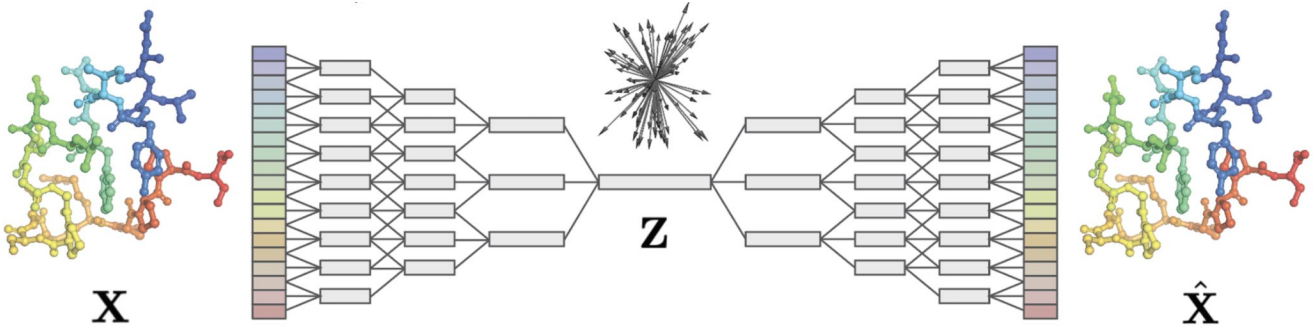
Manvitha Ponnapati



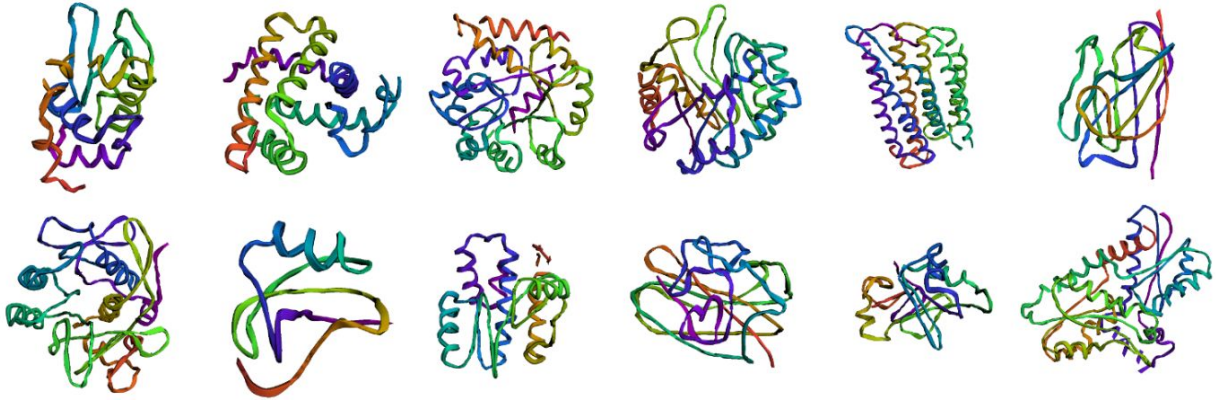
Joe Jacobson

We've used E(3)NNs to build generative models of atomic systems.

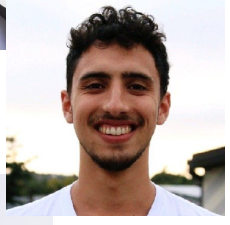
Ophiuchus: Scalable Model of Protein Structures through Hierarchical Coarse-Graining SO(3)-equivariant Autoencoders (arXiv:2310.02508)



Diffusion of random sampled latent to generate new backbones

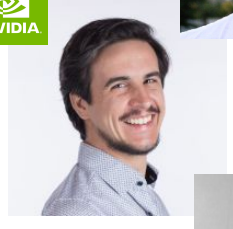


Allan Costa



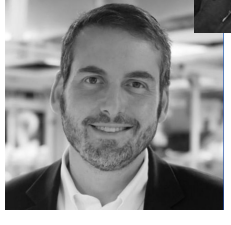
Ilan Mitnikov

NVIDIA



Mario Geiger

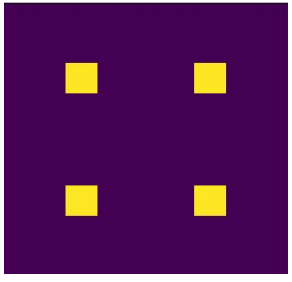
Manvitha Ponnapati



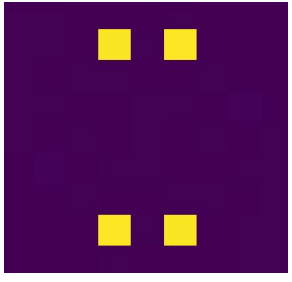
Joe Jacobson

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

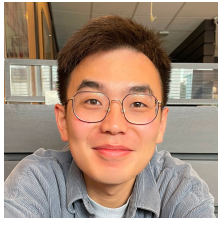
Input



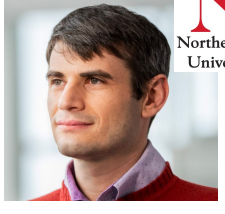
Target Output



Elyssa Hofgard



Rui (Ray) Wang



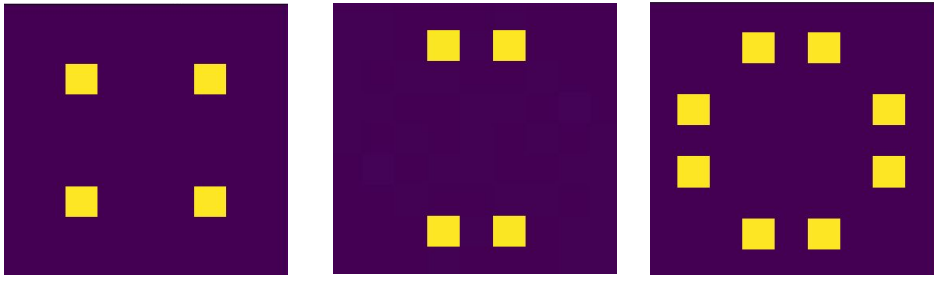
Northeastern University

Robin Walters

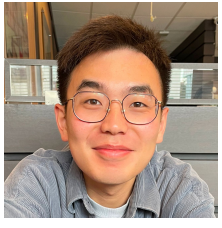
YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

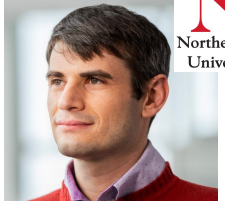
Input **Target Output** **Predicted Out.**



Elyssa Hofgard



Rui (Ray) Wang



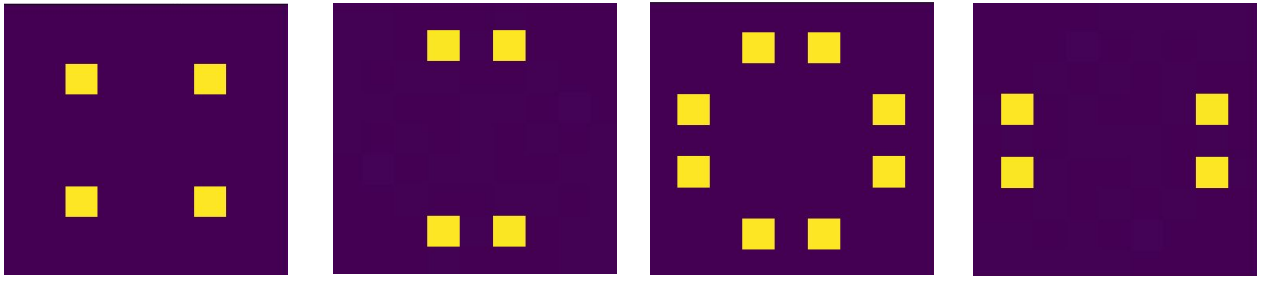
Northeastern University

Robin Walters

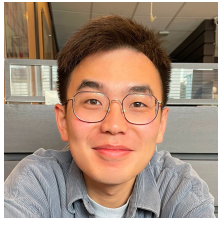
YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

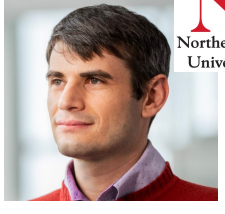
Input **Target Output** **Predicted Out.** **Implied Output**



Elyssa Hofgard



Rui (Ray) Wang

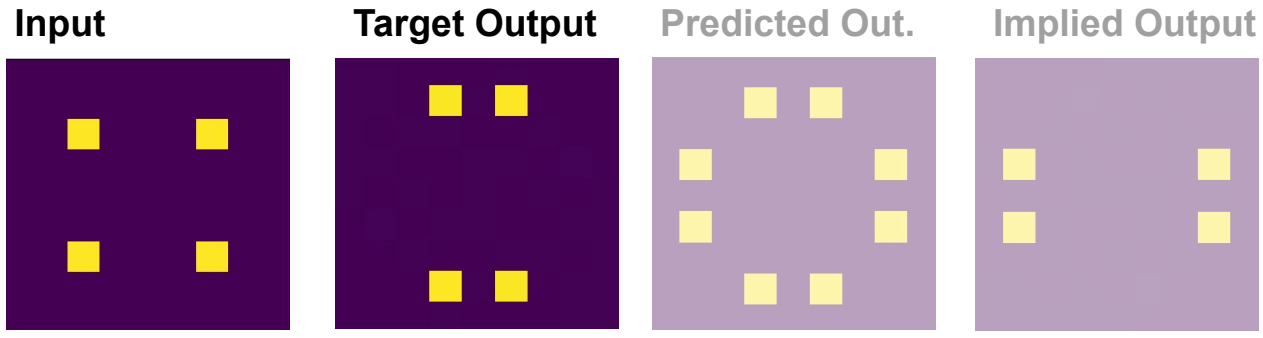


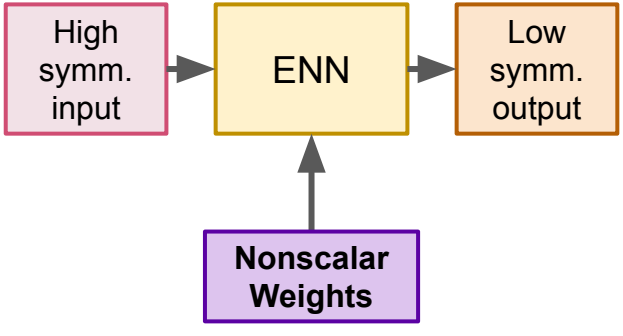
Northeastern University

Robin Walters

YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

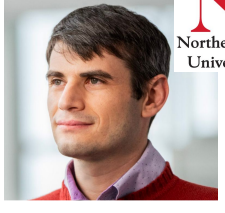




One solution, allow network to ***learn*** asymmetry.

Elyssa Hofgard

Rui (Ray) Wang

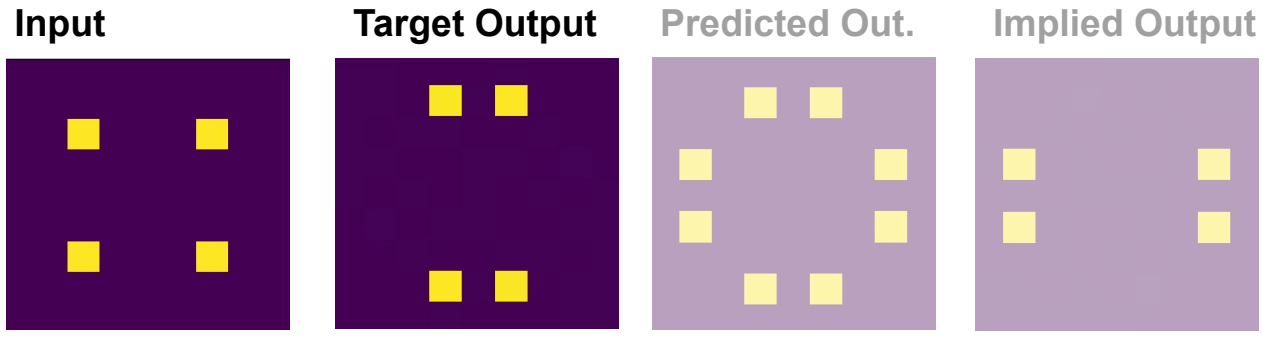


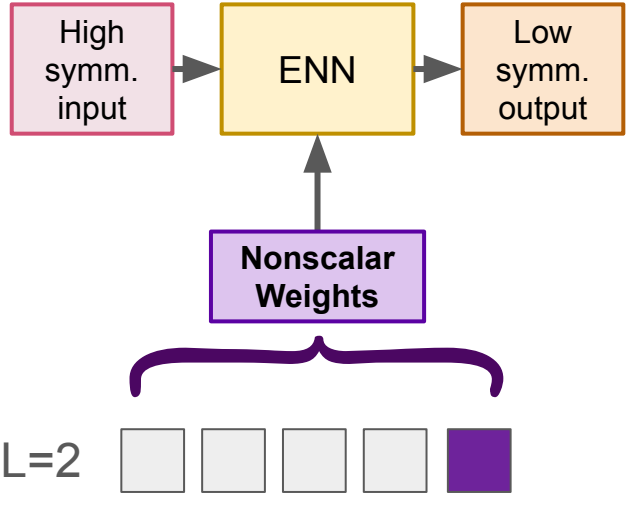
Northeastern University

Robin Walters

YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

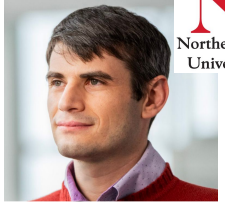




One solution, allow network to ***learn*** asymmetry.

Elyssa Hofgard

Rui (Ray) Wang

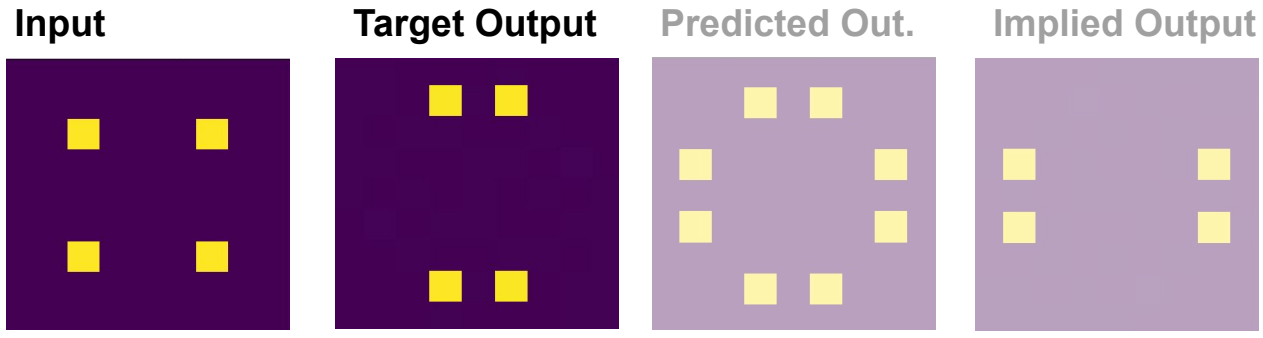


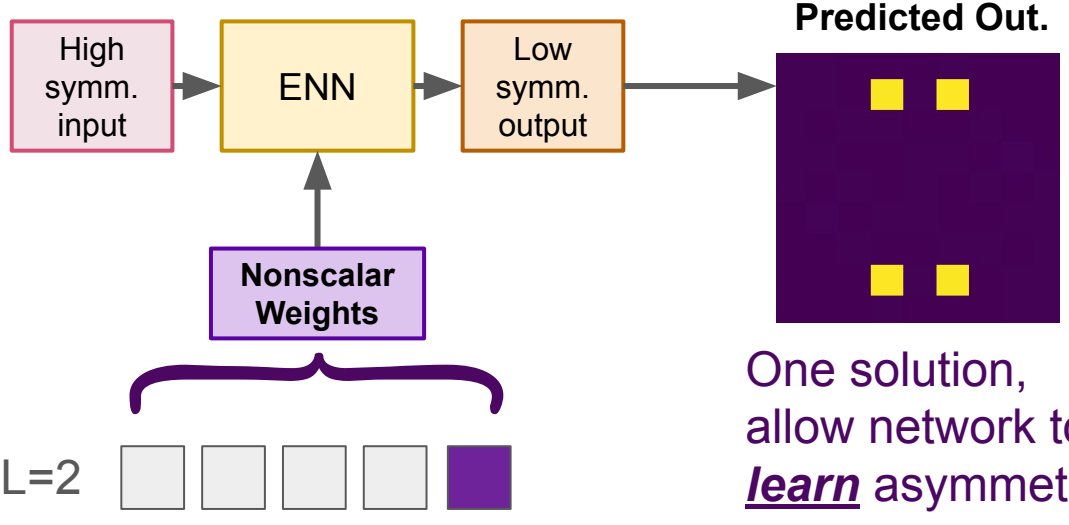
Northeastern University

Robin Walters

YuQing Xie

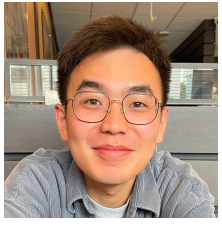
We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).



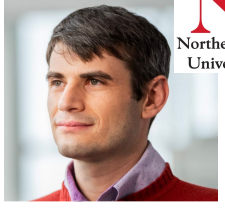


One solution, allow network to ***learn*** asymmetry.

Elyssa Hofgard



Rui (Ray) Wang

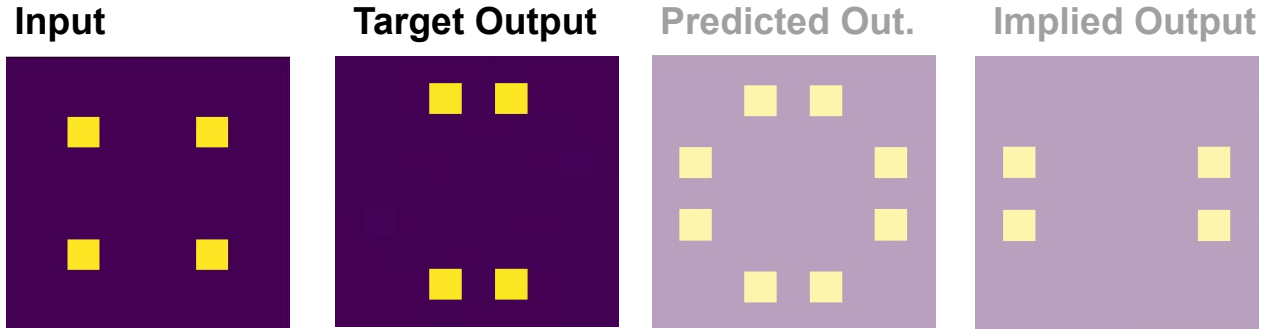


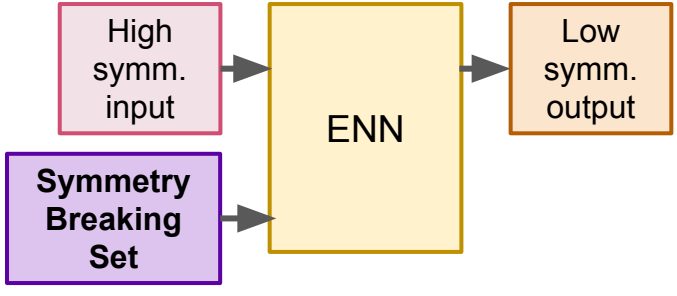
Northeastern University

Robin Walters

YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

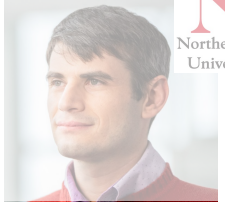




Another solution, build a ***symmetry breaking set*** of order parameters.

Elyssa Hofgard

Rui (Ray) Wang

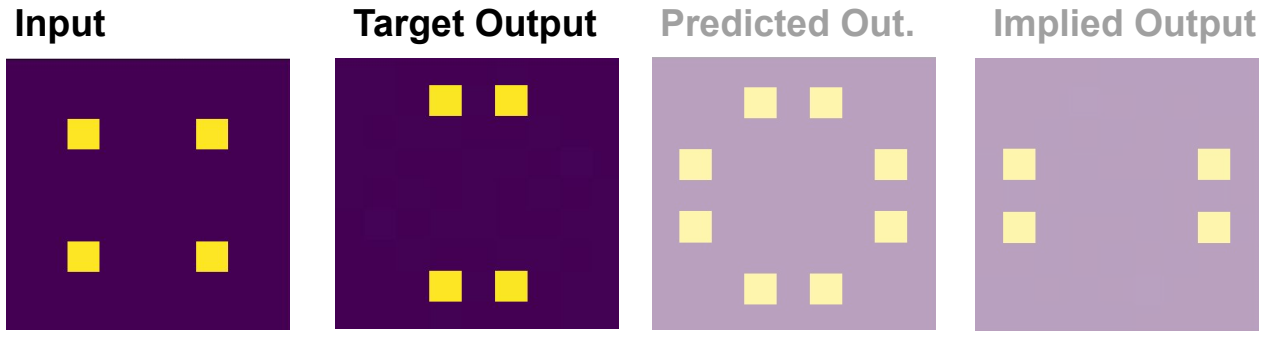


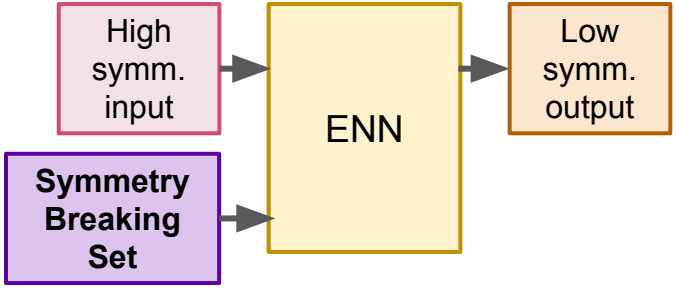
Northeastern University

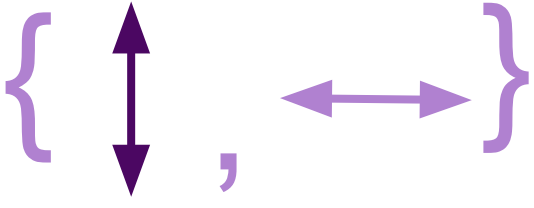
Robin Walters

YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).



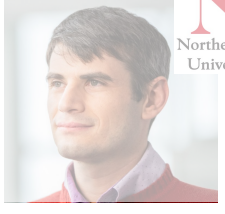




Another solution, build a ***symmetry breaking set*** of order parameters.

Elyssa Hofgard

Rui (Ray) Wang

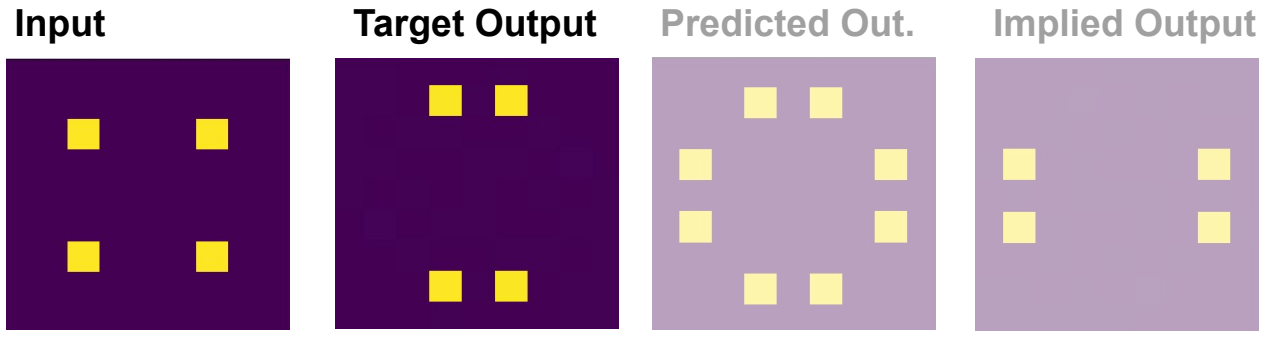


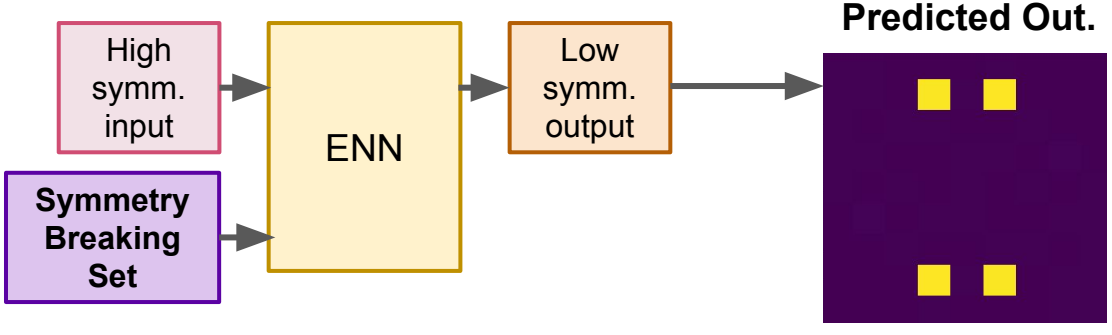
Northeastern University

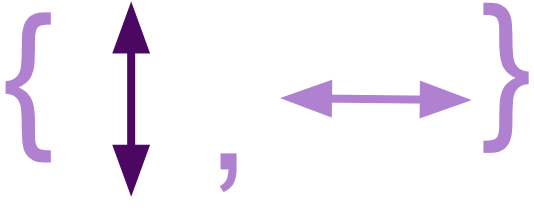
Robin Walters

YuQing Xie

We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).



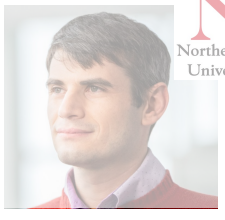




Another solution, build a ***symmetry breaking set*** of order parameters.

Elyssa Hofgard

Rui (Ray) Wang

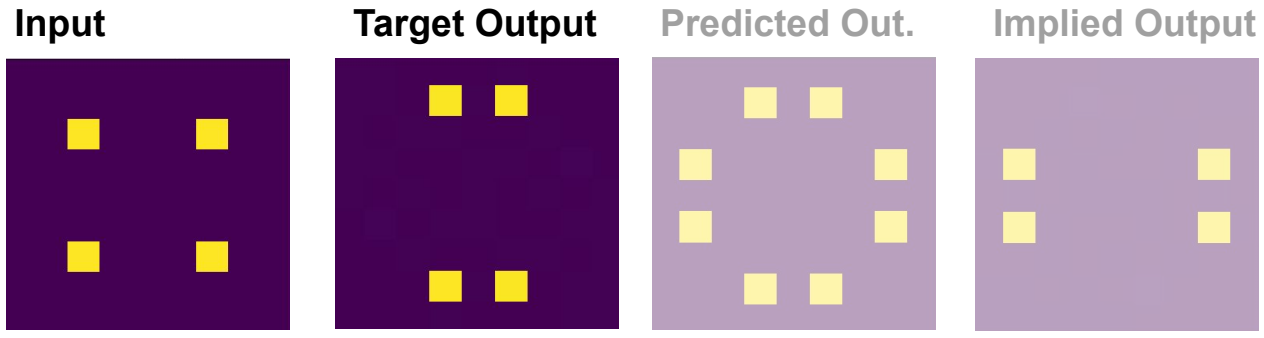


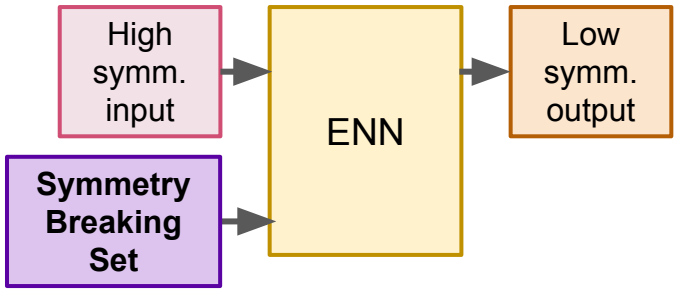
Northeastern University

Robin Walters

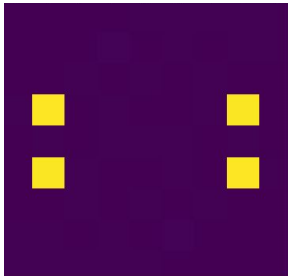
YuQing Xie

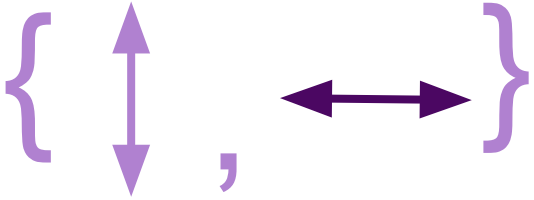
We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).





Predicted Out.





Another solution, build a ***symmetry breaking set*** of order parameters.

Elyssa Hofgard

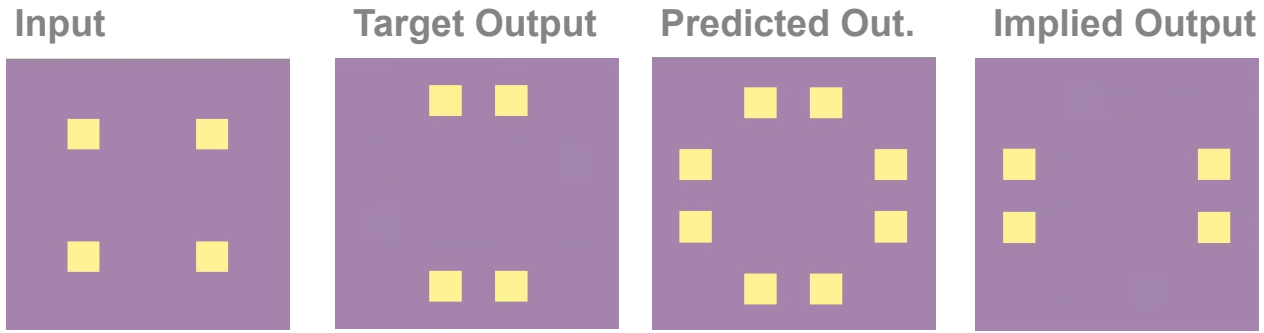
Rui (Ray) Wang

Northeastern University

Robin Walters

YuQing Xie

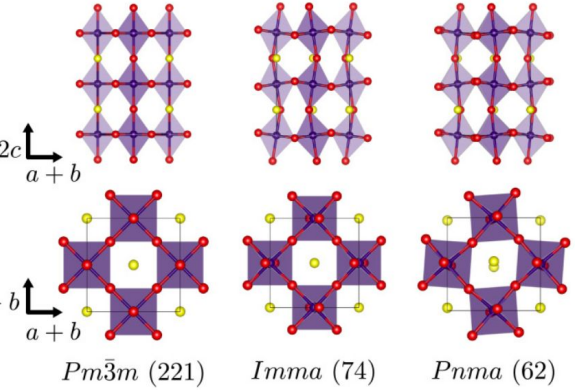
We've used E(3)NNs and E(2)NNs to uncover asymmetry (arXiv:2402.02681, ICML 2024) and handle spontaneous symmetry breaking (arXiv:2402.02681).

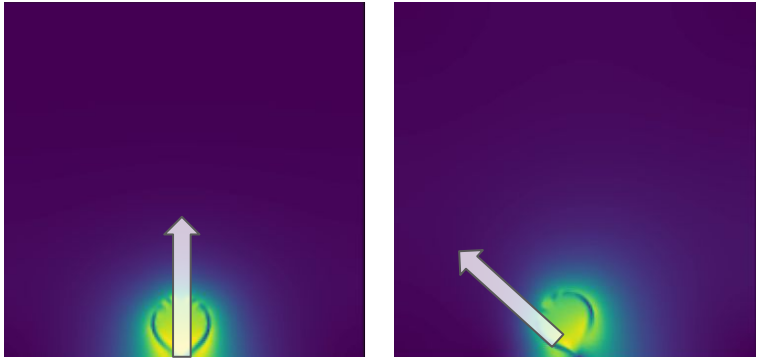


Elyssa Hofgard

Rui (Ray) Wang

Many provable guarantees about these approaches. Can be applied to diverse datasets, e.g. from structural phase transitions to fluid dynamics.





Northeastern University

Robin Walters

YuQing Xie

Open Questions

Equivariant nets can be slow. This is an engineering problem.

- **Tensor products have not been optimized in the same way as traditional operations (e.g. matmul). More equiv. models will put pressure to fix this.**

Equivariant generative models. Lots of activity: diffusion, autoregressive, etc.
But still many challenges:

- **Large, variable numbers (atomistic) point systems**
- **Handling different types of hierarchies in the same model (short, long, symmetry)**
- **Laying down patterns / motifs of points**

We still don't know the "best" way to compose building block / leverage properties.

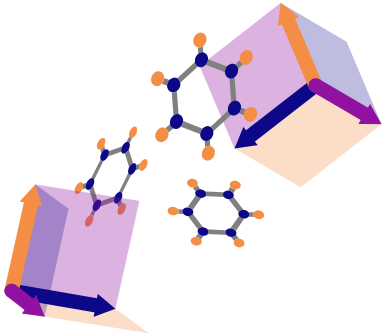
- **We are behind other fields (e.g. computer vision) in the number of things that have been "tried out". This is rapidly changing!**

Best training practices

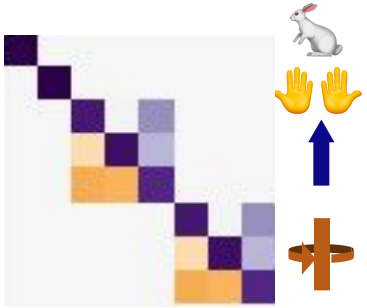
- **How to get good performance across diverse systems**
- **Any additional "tricks" for training generalizable models**

TAKEAWAYS

Euclidean neural networks are built with the powerful assumption that atomic systems exist in 3D Euclidean space.



E(3)NNs use specific data types to encode how data transforms and interacts. Higher order features can be created from simple vectors!



We've used E(3)NNs to understand and design atomic systems from molecules and proteins to crystals and alloys.

There still a lot to do!

- **"Best" building blocks**
- **Improved efficiency**
- **New generative capabilities**
- **"Best" training practices**
- **…**

Tess Smidt | tsmidt@mit.edu | e3nn.org | atomicarchitects.com

Thanks to the group…

Atomic Architects

MIT EECS

rle AT MIT RESEARCH LABORATORY OF ELECTRONICS AT MIT

Tess Smidt

Principal Investigator

Tricia O'Donnell

Administrative Assistant

Aria Mansouri Tehrani

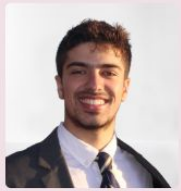
Postdoctoral Scholar

Rui Wang

Postdoctoral Scholar

Elyssa Hofgard

PhD Student



Allan Costa

PhD Student

Mit Kotak

SM Student

Julia Balla

PhD Student

Yi-Lun Liao

PhD Student

Ameya Daigavane

PhD Student

Adriana Ladera

SM Student

YuQing Xie

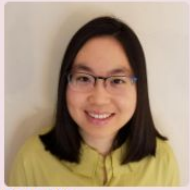
PhD Student

Giulianna Hashemi-Asasi

PhD Student

Max Aalto

PhD Student



Song Kim

Undergraduate Student

Tuong Phung

Undergraduate Student

Thanks to the group, our collaborators, …

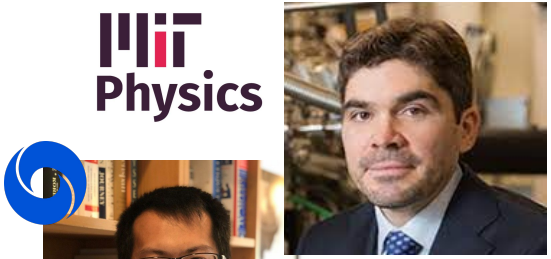
Abhishek Das

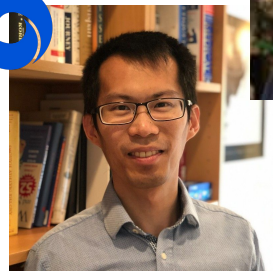
Brandon Wood

Robin Walters

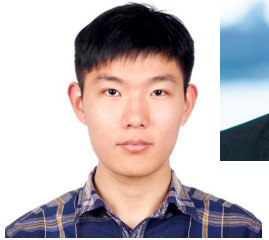
Joe Jacobson
Ilan Mitnikov
Manvitha Ponnapati

Mario Geiger





Joe Checkelsky
Shiang Fang



Rodrigo Freitas
Yifan (Henry) Cao
Killian Sheriff

Thanks to the group, our collaborators, and our funding!

DOE ICDI grant
DE-SC0022215

The NSF Institute for
Artificial Intelligence and
Fundamental Interactions

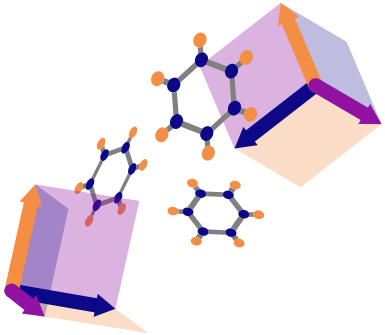
PHY 2019786

AFOSR
Young Investigator
Program

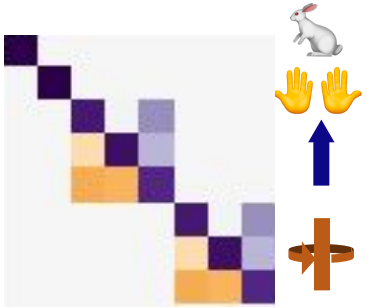
MIT **SuperUROP**

TAKEAWAYS

Euclidean neural networks are built with the powerful assumption that atomic systems exist in 3D Euclidean space.



E(3)NNs use specific data types to encode how data transforms and interacts. Higher order features can be created from simple vectors!



We've used E(3)NNs to understand and design atomic systems from molecules and proteins to crystals and alloys.

There still a lot to do!

- **"Best" building blocks**
- **Improved efficiency**
- **New generative capabilities**
- **"Best" training practices**
- **…**

Tess Smidt | tsmidt@mit.edu | e3nn.org | atomicarchitects.com

Calling in backup (slides)!

We've used E(3)NNs to build data-efficient and scalable models of physical processes.

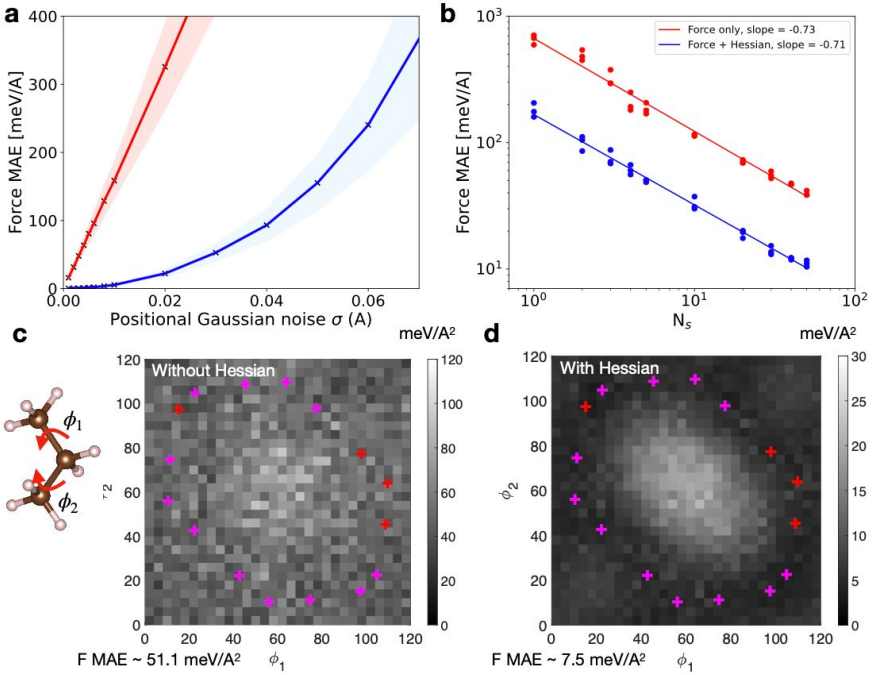
Predict phonons with symmetry guarantees using higher order derivatives with e3nn-jax.

(Ultimately paving the way to training directly on spectra...)

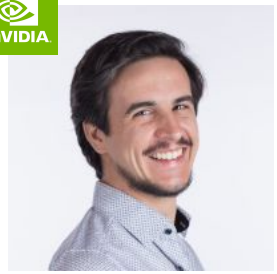
(arXiv:2403.11347)

Training with Hessian vs. Forces

Hessian improves stability but does not result in obvious scaling improvement.



Shiang Fang



Mario Geiger

Joseph Checkelsky

Phonon predictions with E(3)-equivariant graph neural networks, arXiv:2403.11347
NeurIPS 2023 - AI4Mat Workshop (https://openreview.net/forum?id=xxyHjer00Y).

Symphony

Test ↑	Symphony	EDM	G-SchNet	G-SphereNet
All Atoms Connected	99.92	99.88	99.87	**100.00**
Reasonable Bond Angles	99.56	**99.98**	99.88	97.59
Reasonable Bond Lengths	98.72	**100.00**	99.93	72.99
Aromatic Ring Flatness	**100.00**	**100.00**	99.95	99.85
Double Bond Flatness	**99.07**	98.58	97.96	95.99
Reasonable Internal Energy	**95.65**	94.88	95.04	36.07
No Internal Steric Clash	98.16	**99.79**	99.57	98.07

Symphony

MMD of Bond Lengths ↓	Symphony	EDM	G-SchNet	G-SphereNet
C-H: 1.0	0.0739	**0.0653**	0.3817	0.1334
C-C: 1.0	0.3254	**0.0956**	0.2530	1.0503
C-O: 1.0	0.2571	**0.0757**	0.5315	0.6082
C-N: 1.0	0.3086	**0.1755**	0.2999	0.4279
N-H: 1.0	**0.1032**	0.1137	0.5968	0.1660
C-O: 2.0	0.3033	**0.0668**	0.2628	2.0812
C-N: 1.5	0.3707	**0.1736**	0.5828	0.4949
O-H: 1.0	0.2872	0.1545	0.7899	**0.1307**
C-C: 1.5	0.4142	**0.1749**	0.2051	0.8574
C-N: 2.0	0.5938	**0.3237**	0.4194	2.1197

MMD of Bispectra ↓	Symphony	EDM	G-SchNet	G-SphereNet
C: C2,H2	0.2165	**0.1003**	0.4333	0.6210
C: C1,H3	0.2668	**0.0025**	0.0640	1.2004
C: C3,H1	**0.1111**	0.2254	0.2045	1.1209
C: C2,H1,O1	**0.1500**	0.2059	0.1732	0.8361
C: C1,H2,O1	0.3300	0.1082	**0.0954**	1.6772
O: C1,H1	0.0282	0.0056	0.0487	**0.0030**
C: C2,H1,N1	**0.1481**	0.1521	0.1967	1.3461
C: C2,H1	0.2525	**0.0468**	0.1788	0.2403
C: C1,H2,N1	0.3631	0.2728	**0.1610**	0.9171
N: C2,H1	**0.0953**	0.2339	0.2105	0.6141

Jensen-Shannon Divergence ↓	Symphony	EDM	G-SchNet	G-SphereNet
Atom Type Counts	0.0003	**0.0002**	0.0011	0.0026
Local Environment Counts	**0.0039**	0.0057	0.0150	0.1016

5 Conclusion

We have proposed Symphony, a new method to autoregressively generate 3D molecular geometries with spherical harmonic projections and higher-degree $E(3)$-equivariant features. We show promising results on molecular generation and completion, relative to existing autoregressive models. However, one drawback of our current formulation is that the discretization of our radial components is too coarse, so our bond length distributions are not as accurate as EDM or G-SchNet. This affects our validity when using lookup tables to assign bond orders as they are particularly sensitive to exact bond lengths. Further, Symphony incurs increased computational cost due to the use of tensor products to create higher degree $E(3)$-equivariant features. As a highlight, Symphony is trained on only ≈ 80 epochs, while G-SchNet and EDM are trained for 330 and 1100 epochs respectively. Further exploring the data efficiency of Symphony remains to be seen. In the future, we plan to explore normalizing flows to smoothly model the radial distribution without any discretization, and placing entire local environment motifs at once which would speed up generation.

Ophichus

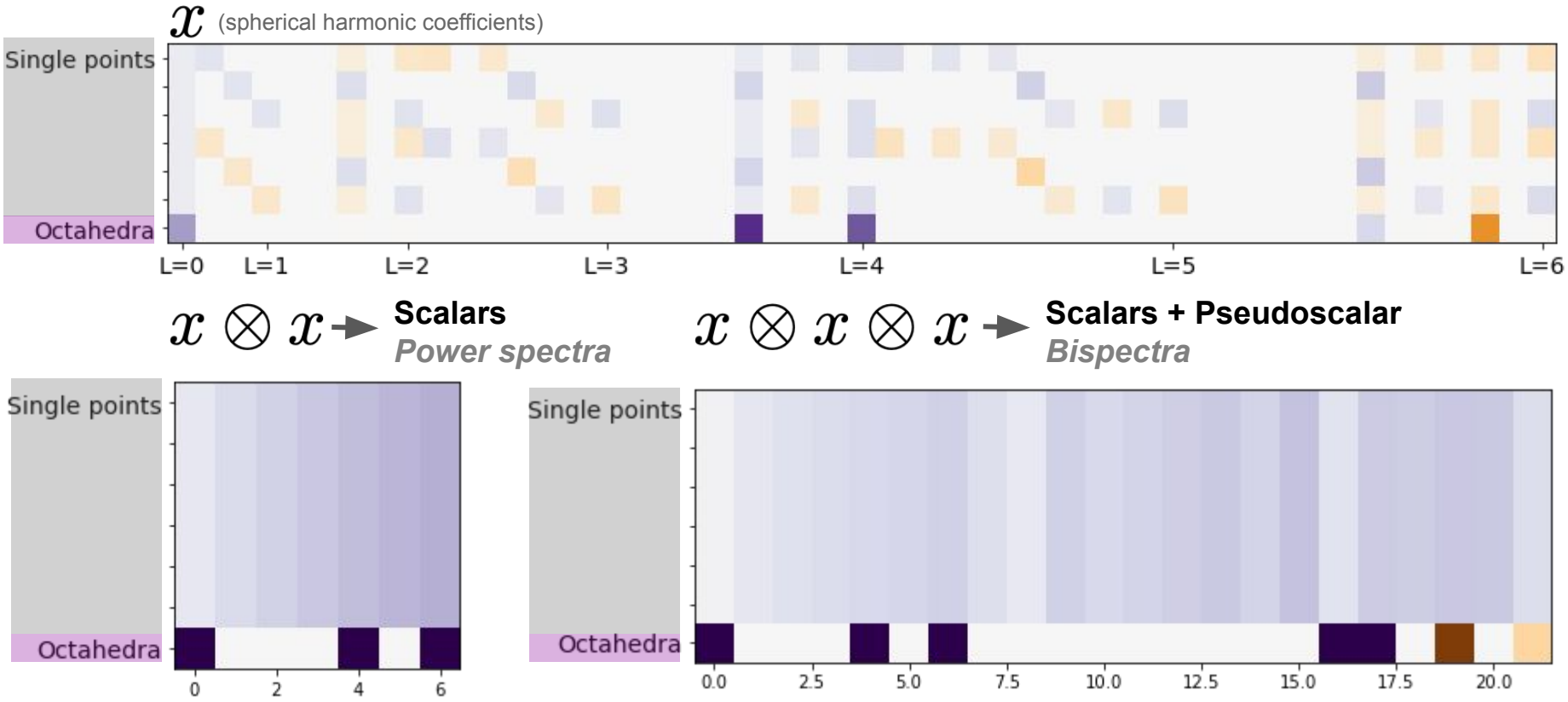
Table 3: Comparison to different diffusion models.

Model	**Dataset**	**Sampling Time** (s) ↓	**scRMSD** (< 2Å) ↑	**scTM** (>0.5) ↑	**Diversity** ↑
FrameDiff [Yim et al. (2023)]	PDB	8.6	0.17	0.81	0.42
RFDiffusion [Trippe et al. (2023)]	PDB + AlphaFold DB	50	0.79	0.99	0.64
Ophiuchus-64 All-Atom	MiniProtein	0.15	0.32	0.56	0.72
Ophiuchus-485 Backbone	PDB	0.46	0.18	0.36	0.39

Why “interact” these types of features?

Interactions lead to smooth, features with rich information.

Illustrative example: the bispectrum ***i.e. scalars and pseudoscalar from triple product***



The bispectrum of spherical harmonic coefficients up to $l_{max} = 4$ has 14 scalars + 1 pseudoscalar.

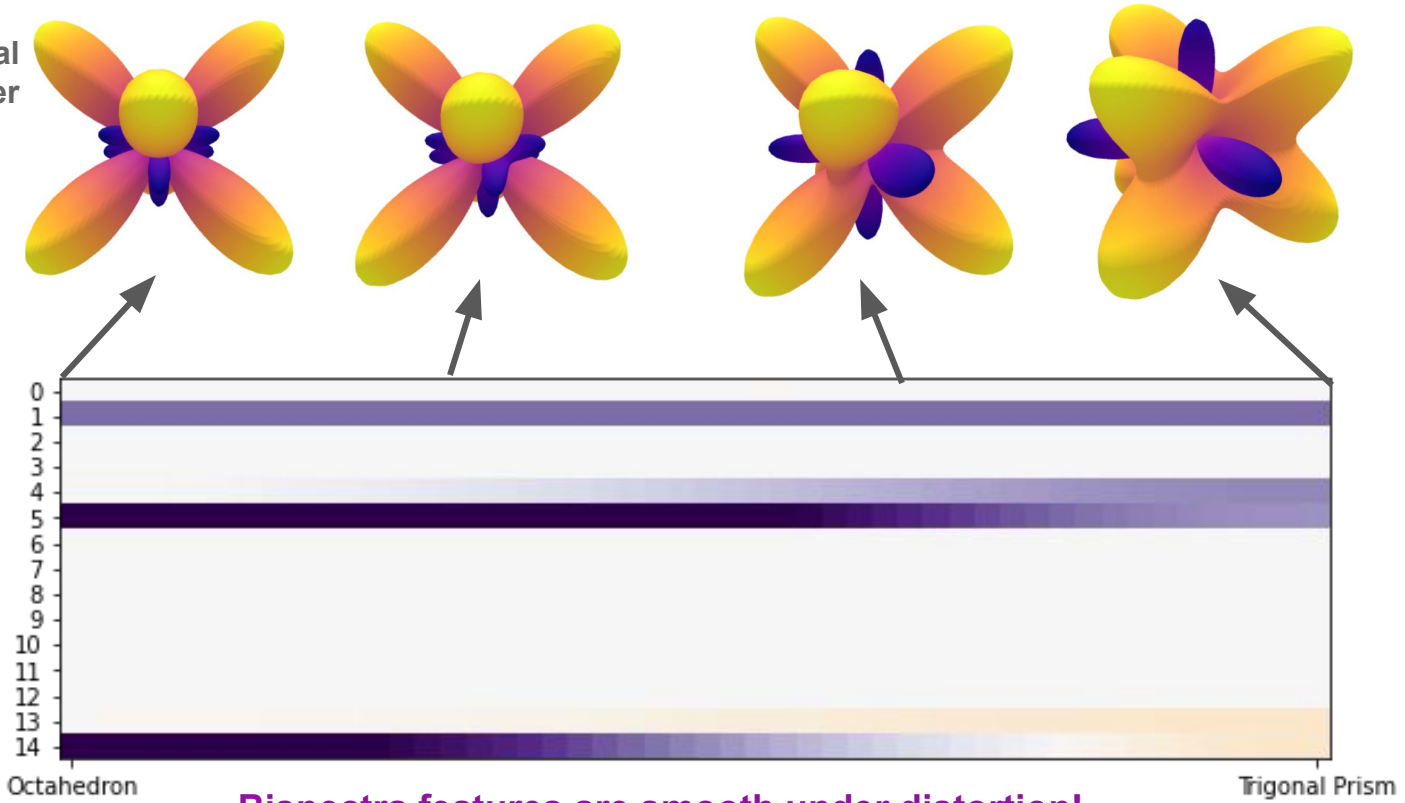
$x \otimes x \otimes x \rightarrow$ **Scalars + Pseudoscalar**

Bispectra

1. 0⊗0⊗0
2. 0⊗1⊗1
3. 0⊗2⊗2
4. 0⊗3⊗3
5. 0⊗4⊗4
6. 1⊗1⊗2
7. 1⊗2⊗3
8. 1⊗3⊗4
9. 2⊗2⊗2
10. 2⊗2⊗4
11. 2⊗3⊗3
12. 2⊗4⊗4
13. 3⊗3⊗4
14. 4⊗4⊗4

Proportional to the power spectra $x \otimes x$

+1 pseudoscalar
(parity of projection)



Bispectra features are smooth under distortion!

Okay, but what does this have to do with REAL materials?

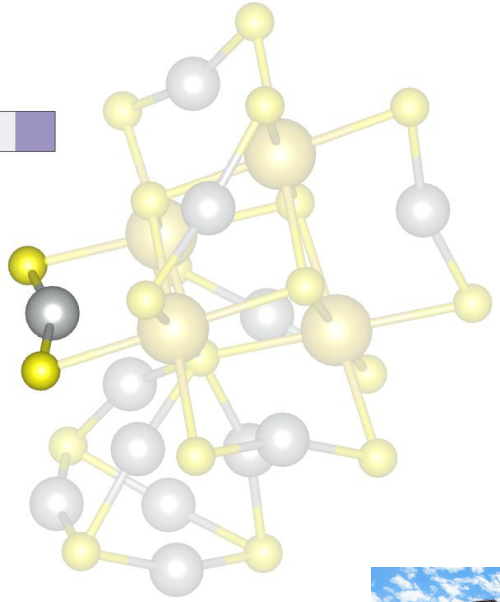
Ex: Clustering *local environments* with the bispectra

$$x \otimes x \otimes x \rightarrow$$ **Scalars + Pseudoscalar**

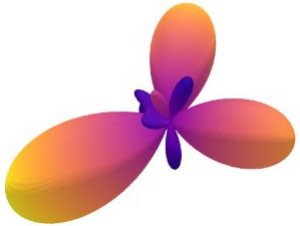
Bispectra

Database: Materials Project
Materials: Crystals with Ag-S bonds
 # Crystals: 669
Radial Cutoff: 3.1 Å
 # Environments: 3061
 # Clusters: 8

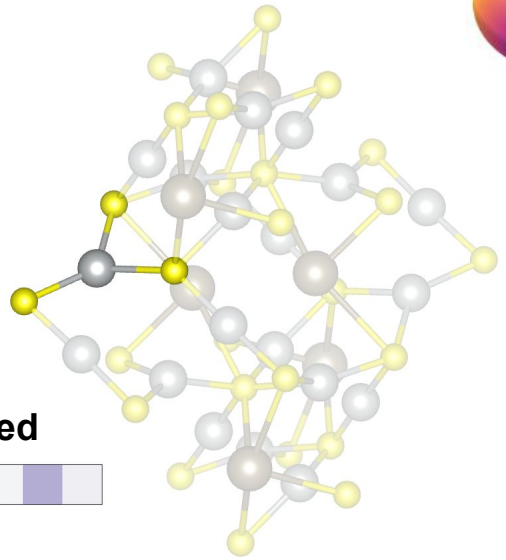
Cluster 2 centroid inverted



mp-16992: $NaAg_3S_2$



Cluster 7 centroid inverted



mp-4762: $TlAg_3S_2$

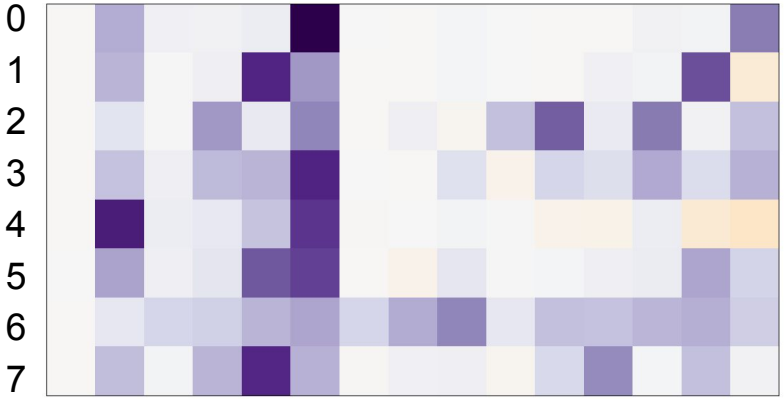
Tuong Phung

Okay, but what does this have to do with REAL materials?

Ex: Clustering *local environments* with the bispectra

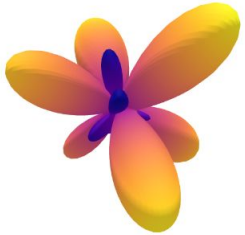
Database: Materials Project
Materials: Crystals with Ag-S bonds
 # Crystals: 669
Radial Cutoff: 3.1 Å
 # Environments: 3061
 # Clusters: 8

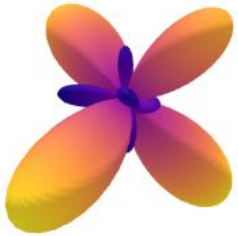
Cluster Centroid BiS ⇨ Inverted BiS ⇨



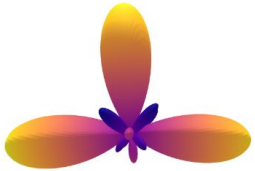
$x \otimes x \otimes x \rightarrow$ **Scalars + Pseudoscalar**

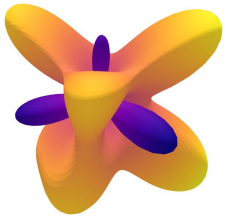
Bispectra

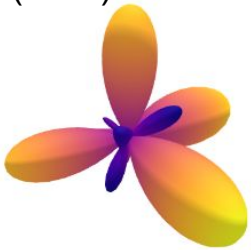

0: Octahedron


1: Tetrahedron

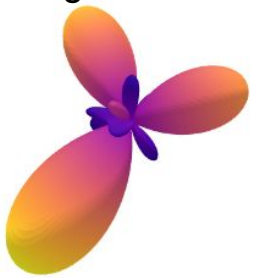

2: (Bent) Linear


3: T-shape


4: Trigonal Prism

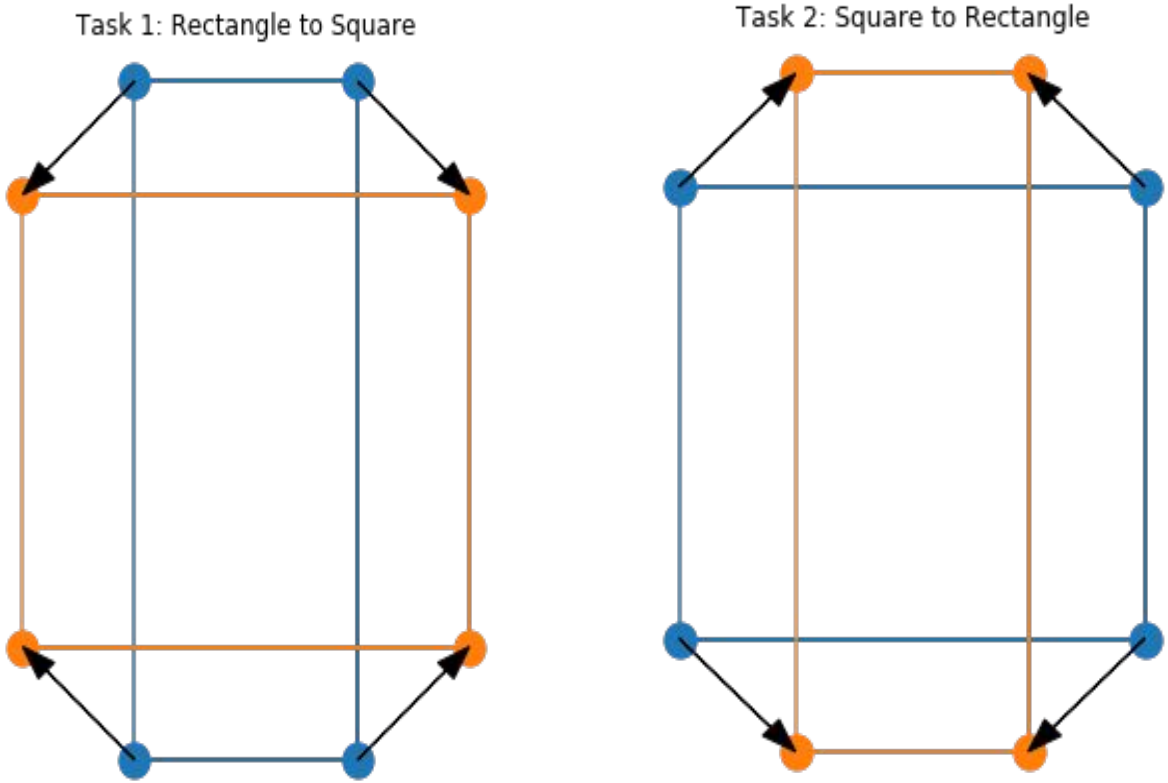

5: Seesaw


6: One (S) neighbor

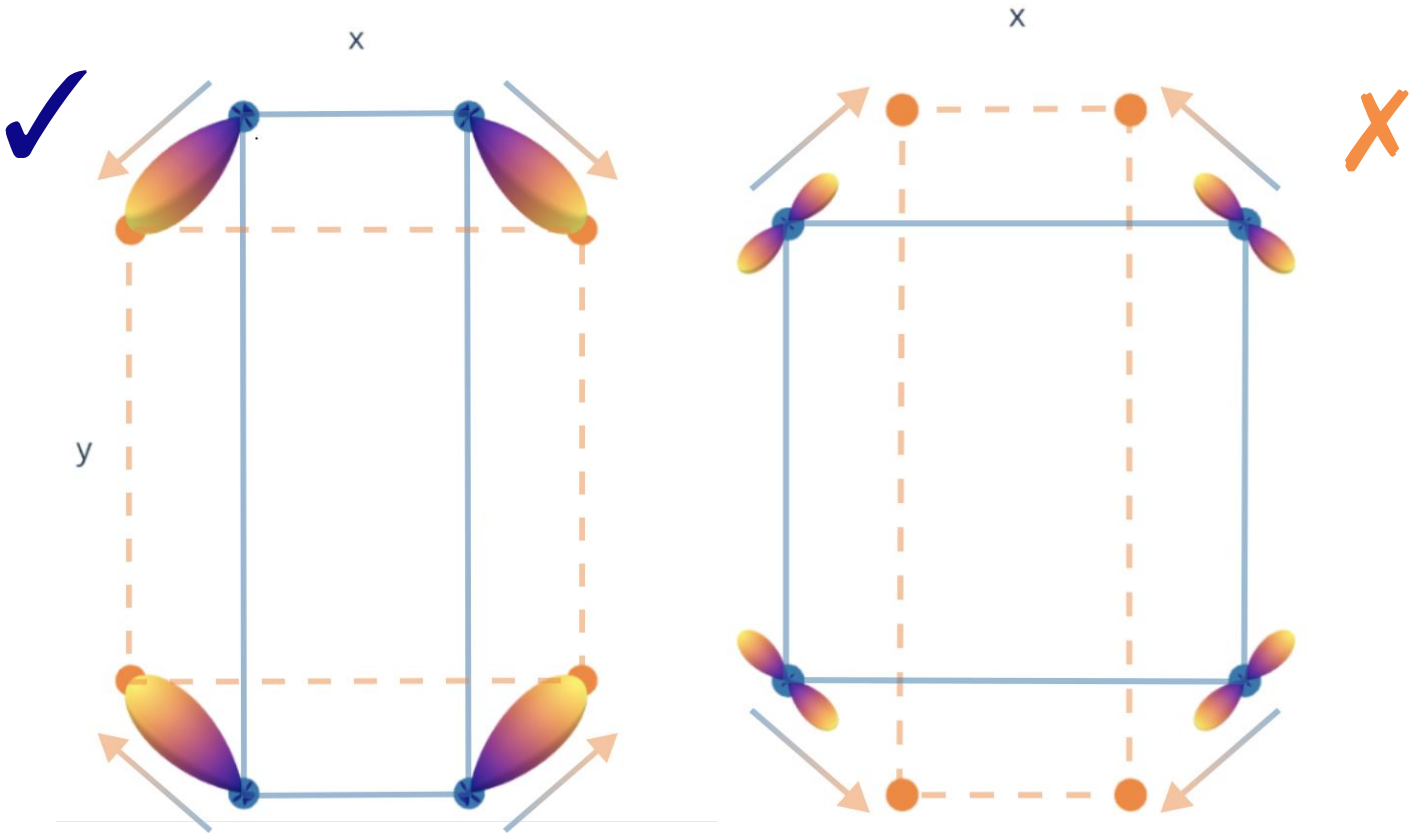

7: Trigonal Planar


Tuong Phung

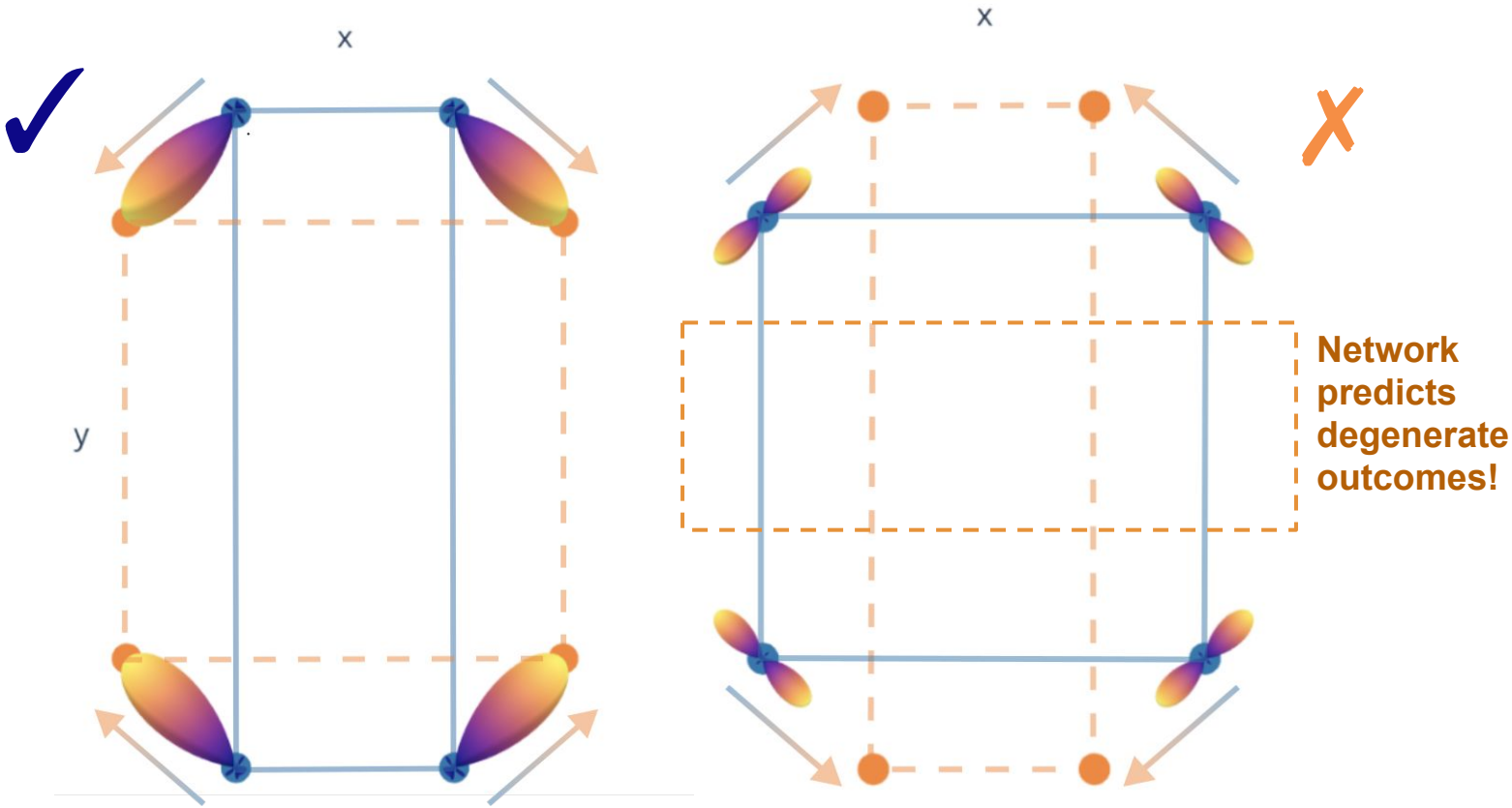
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters"). Gradients have symmetry of "true" outputs.

What information does the network need to "pick" a rectangle?

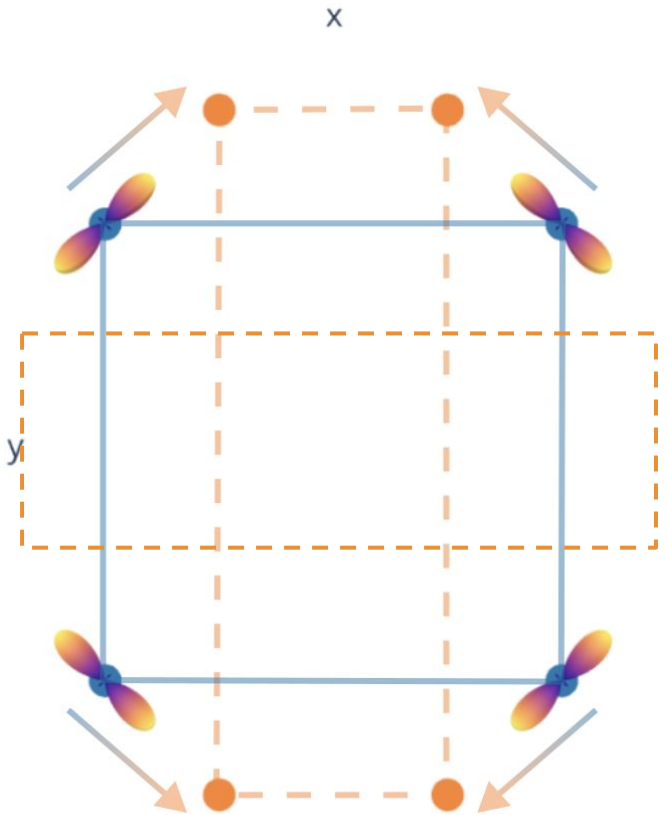
input — learnable parameters — predicted output

$$f(x, w) = y$$

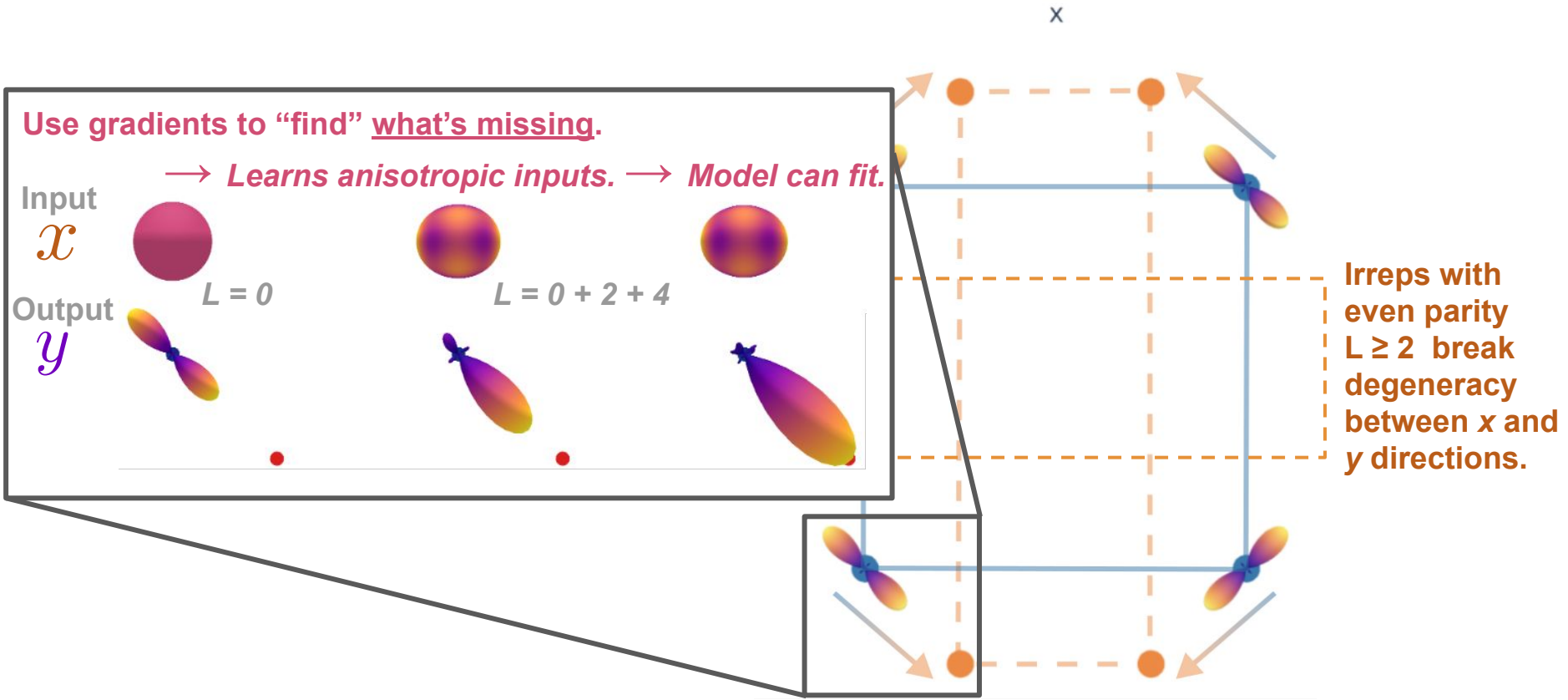
inputs

Update weights using...

$$w_i^{x} = w_i^{x} - \eta \frac{\partial \text{loss}}{\partial w_i^{x}}$$



Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters"). Gradients have symmetry of "true" outputs.

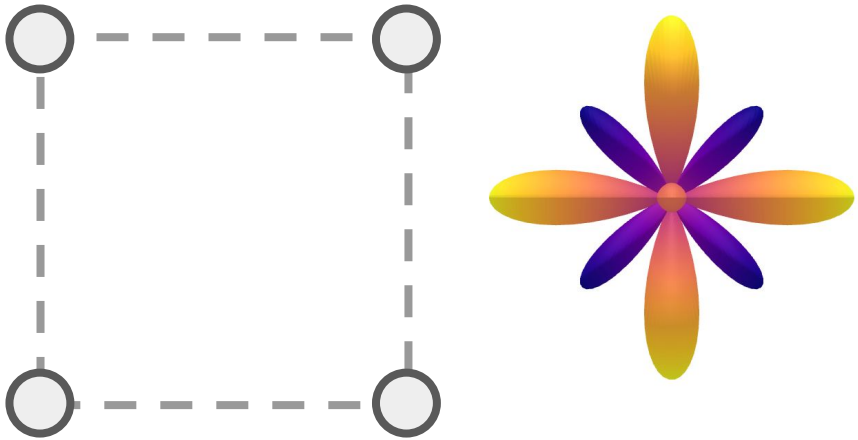


ENNs have the symmetry of the *representation* built-in.

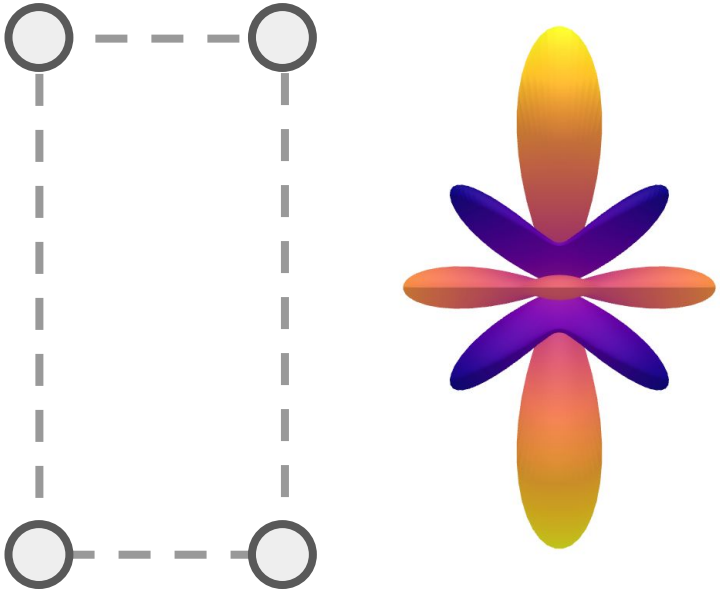
ENNs also *preserve* the subgroup symmetry of the inputs.

Many different types of geometric objects have the same symmetry.

ENN transforms geometric objects into different objects with same or higher symmetry.

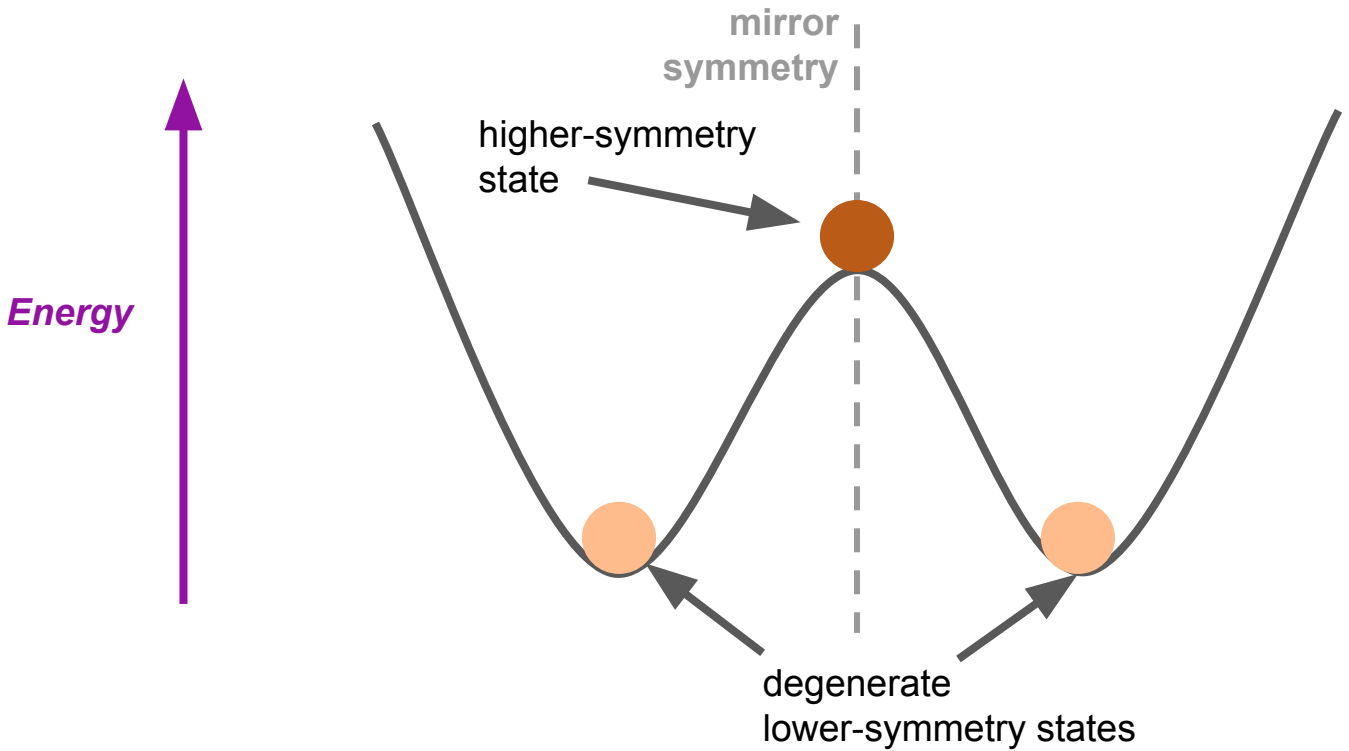


$$\mathrm{Sym}(x) \cong D_{4h}$$

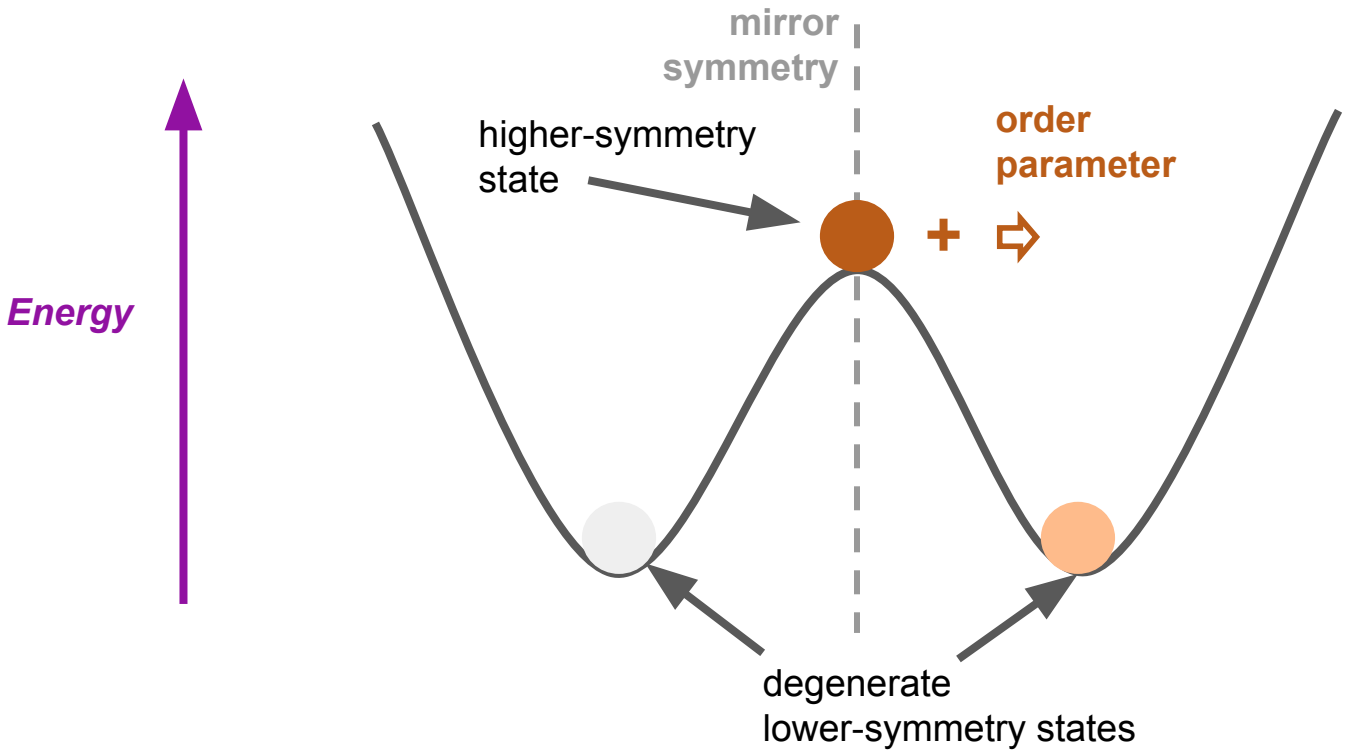


$$\mathrm{Sym}(x) \cong D_{2h}$$

Order parameters describe symmetry breaking and distinguish between degenerate states.



Order parameters describe symmetry breaking and distinguish between degenerate states.



The Atomic Architects Slay Into Battle!