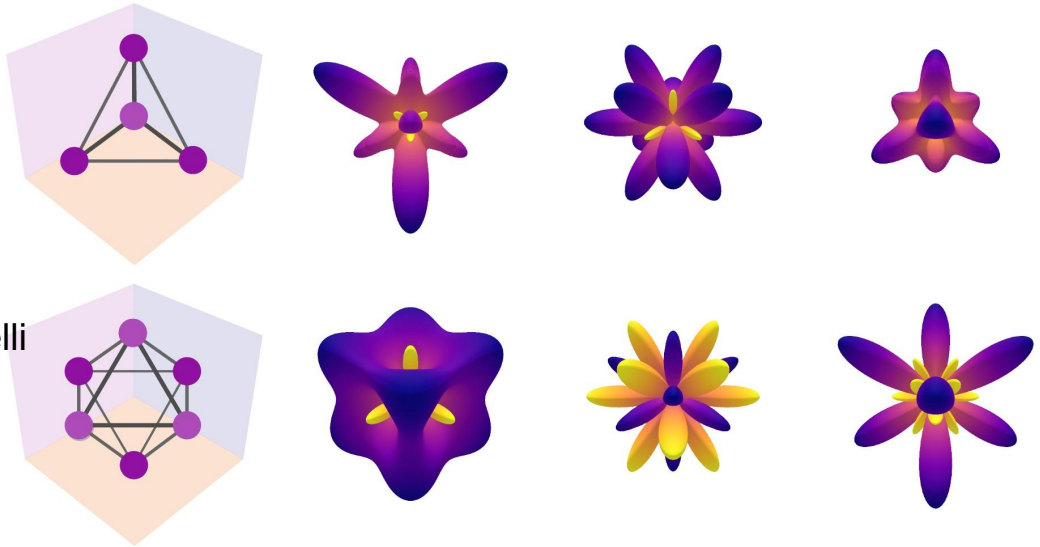
Expected and Unexpected Properties of Neural Networks with Symmetry

with illustrative examples for the 3D Euclidean group

Tess Smidt

PAPERS:

T. E. Smidt
Trends in Chemistry (2021)
T. E. Smidt, M. Geiger, B. K. Miller.
Physical Review Research (2021)
S. Batzner, A. Musaelian… B. Kozinsky
NequIP and Allegro
W. Wang, M. Xu, C. Cai, …, R. Gómez-Bombarelli
ICML 2022
J. A. Rackers, L. Tecot, M. Geiger, T. E. Smidt
arXiv:2201.03726
Y. Liao, T. E. Smidt
arXiv:2206.11990



SOFTWARE:

e3nn.org
https://github.com/e3nn

MIT EECS

BERKELEY LAB

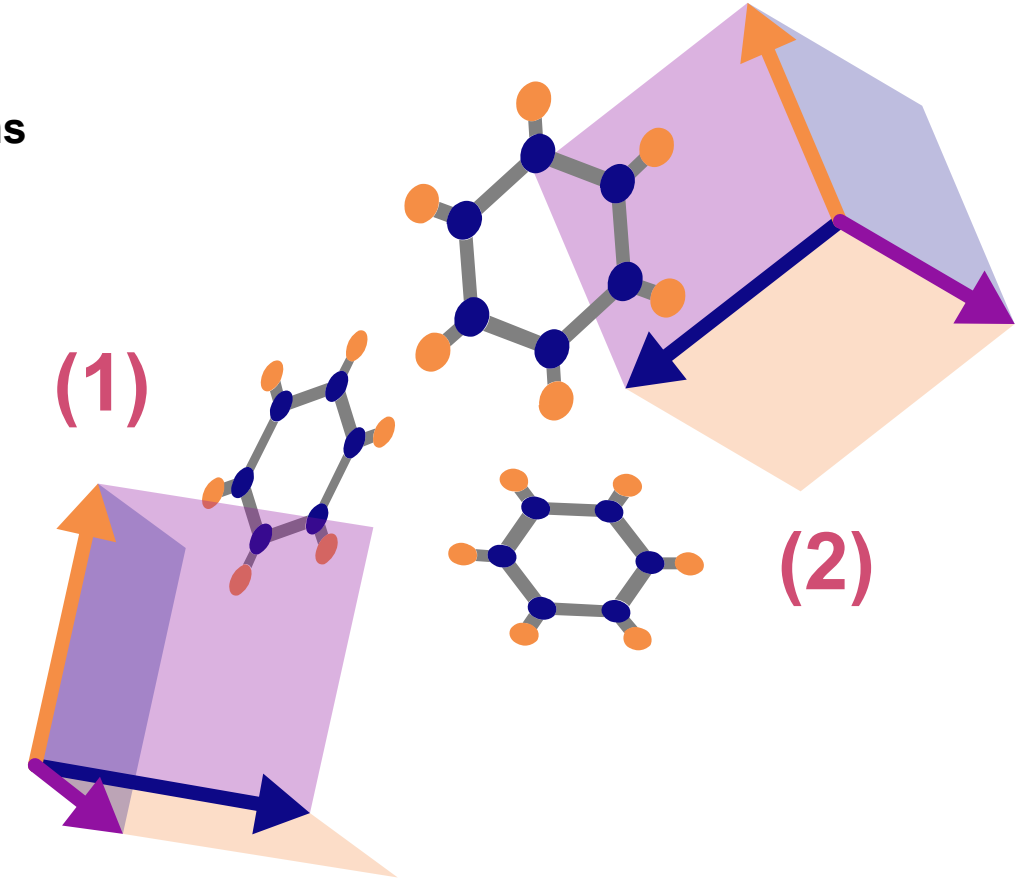
CAMERA Applied Math

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

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 Euclidean space (3D rotations, translations, and inversion)

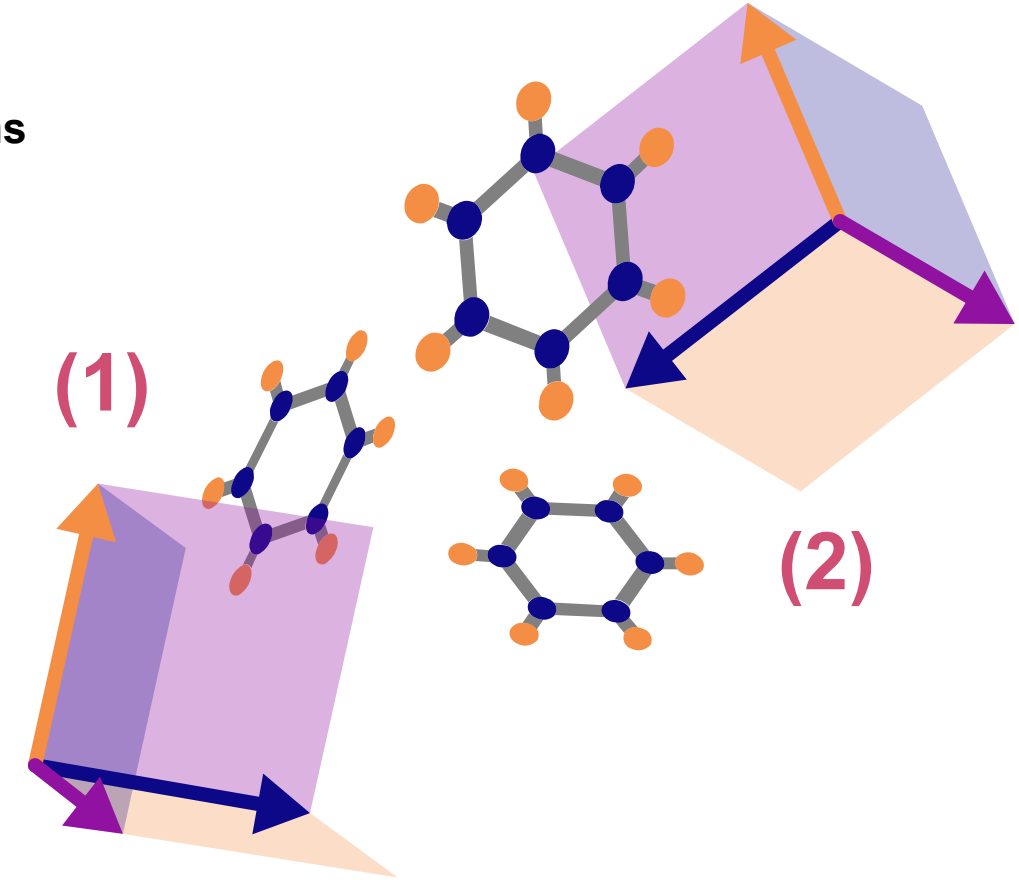


To describe physical systems we use coordinate systems

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

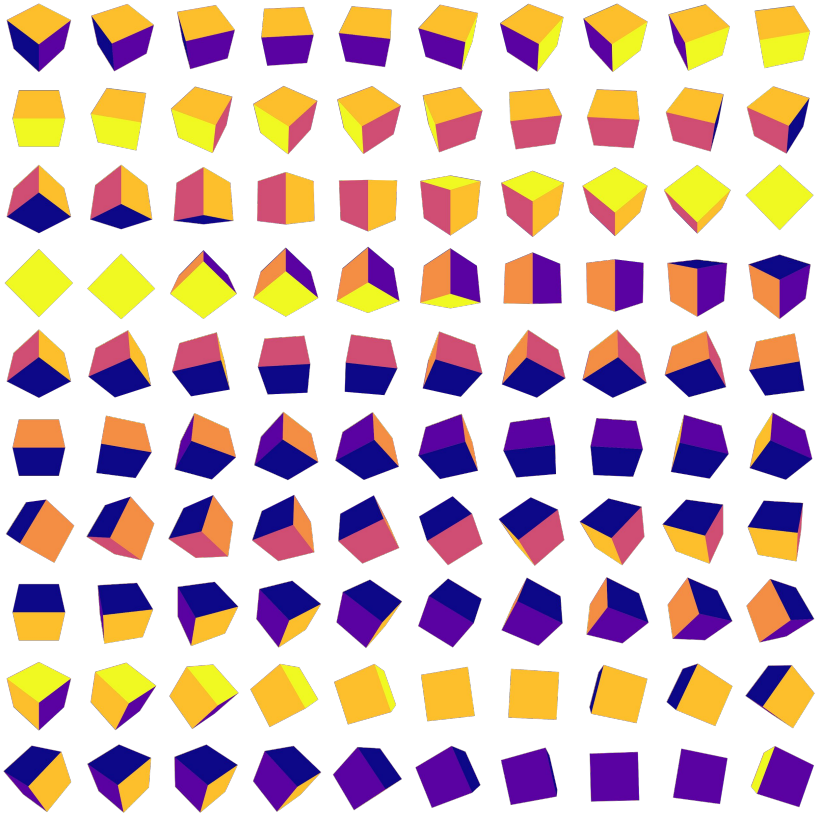
We can transform between coordinate systems using the symmetries of Euclidean space (3D rotations, translations, and inversion)

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



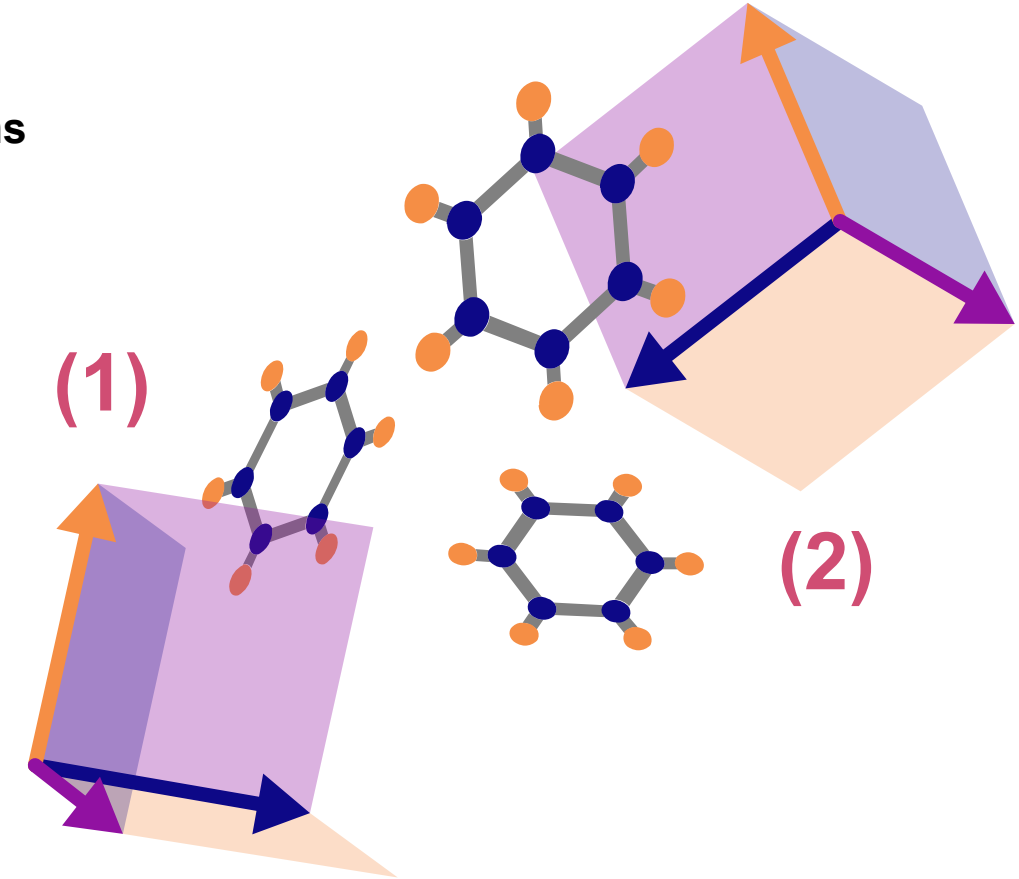
training with symmetry

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 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

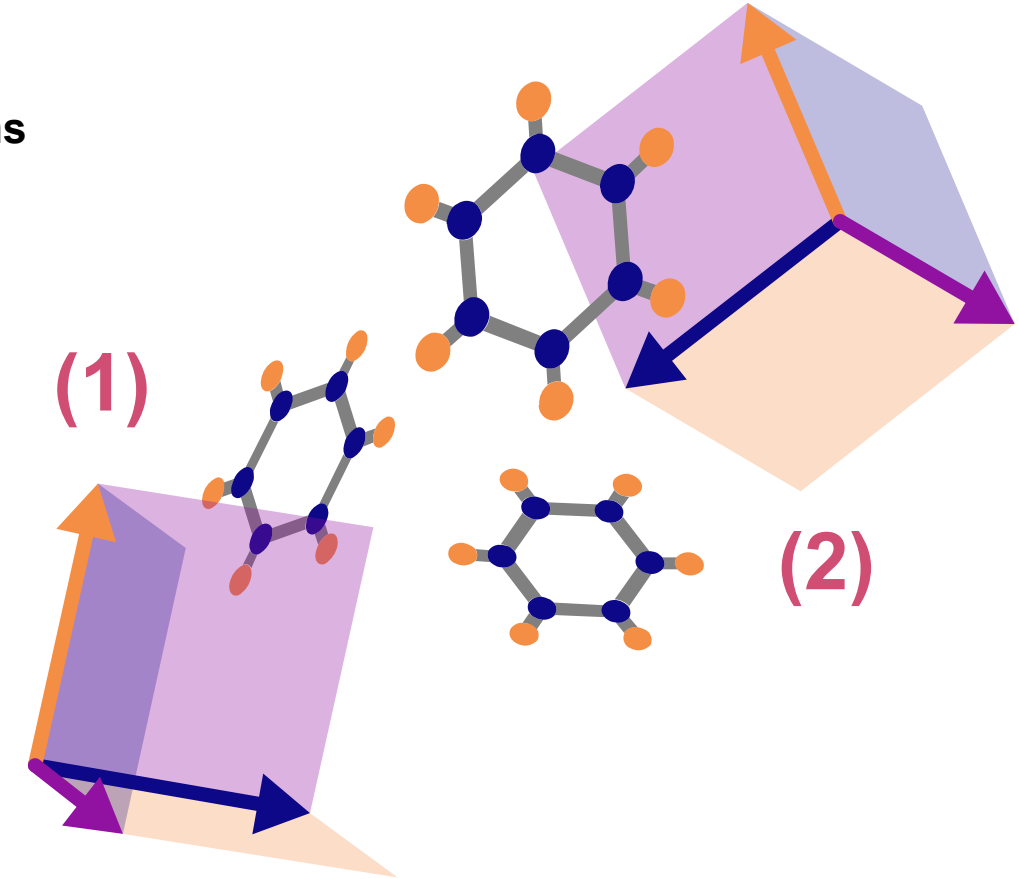
(1) and (2) use different coordinate systems to describe the same physical system.

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

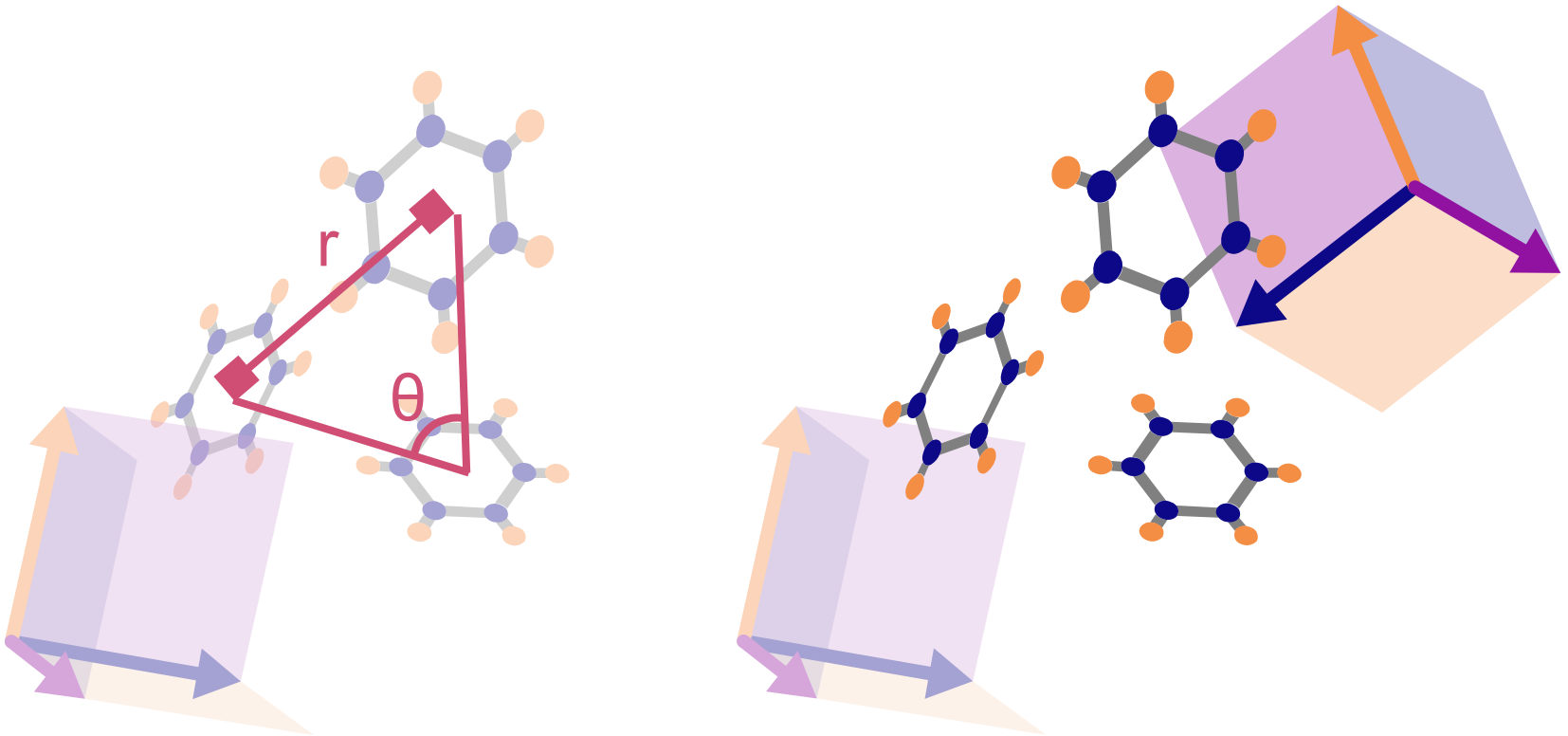
Traditional machine learning see (1) and (2) as completely different!

We want methods that see (1) and (2) as the same system described differently...

...so want machine learning with symmetry!

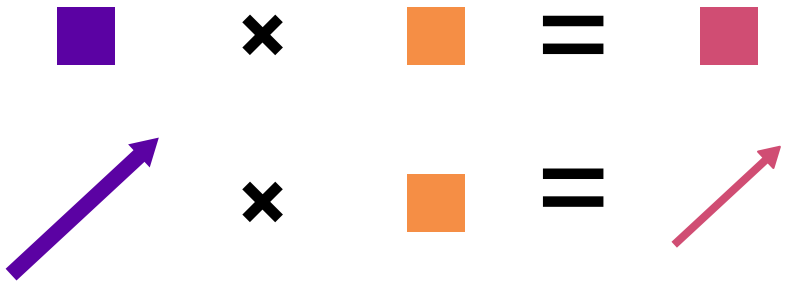


Invariant models pre-compute invariant features and throw away the coordinate system.
Equivariant models keep the coordinate system
AND if the coordinate system changes, the outputs change accordingly.

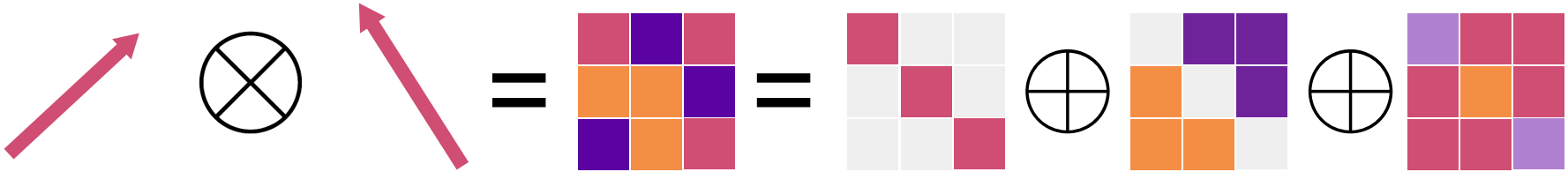


Interactions in equivariant models are more complex than invariant models.

How do we interact invariant objects? Scalar multiplication.



How do we interact equivariant objects? Geometric tensor products!



Generalizes to higher orders.
Same mathematics that describes atomic interactions, e.g. addition of angular momentum.

dot product
trace
invariant
L=0
1 degree of freedom

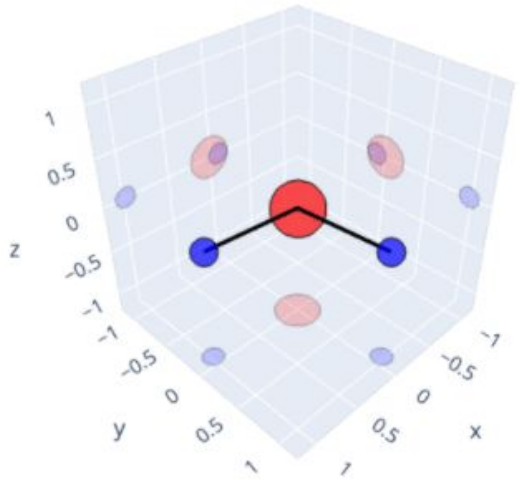
cross-product
antisymmetric
equivariant
L=1
3 degrees of freedom

symmetric
traceless
equivariant
L=2
5 degrees of freedom

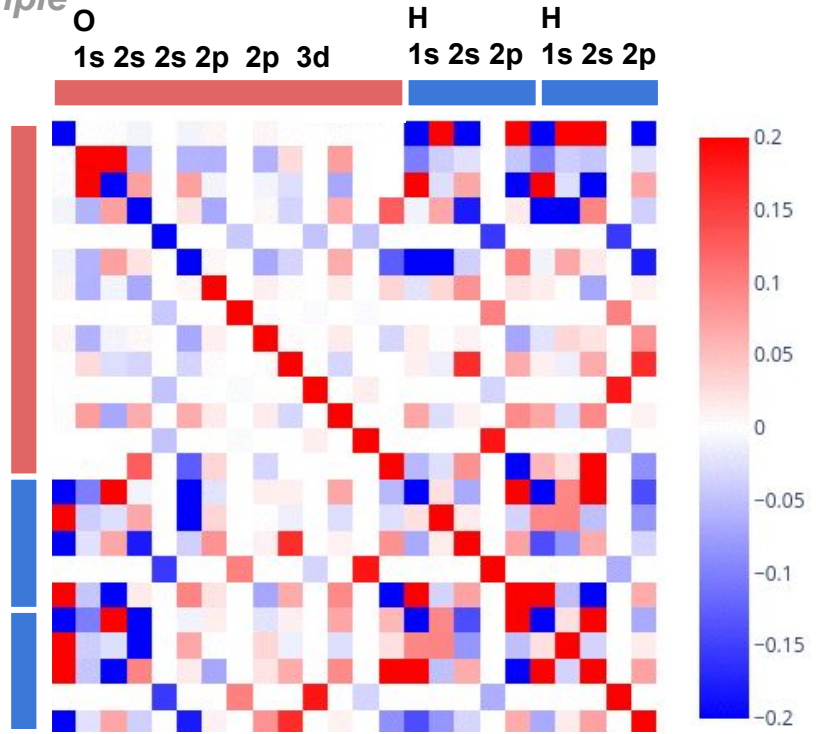
Euclidean symmetry equivariant methods have Euclidean symmetry "built-in".
These methods understand that a physical system described by e.g. two different coordinate systems still "means" the same thing *even without training*.

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

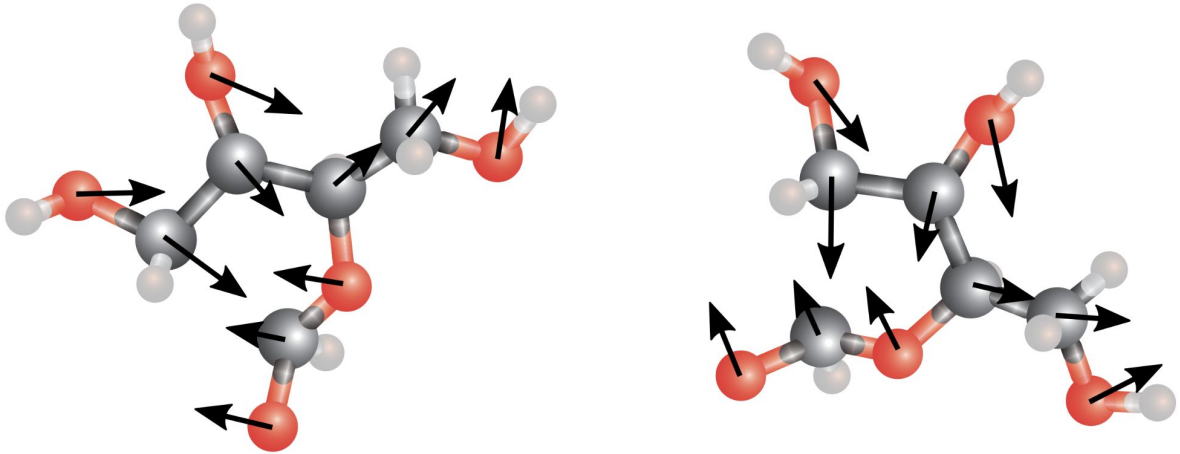
$$R\vec{v}$$



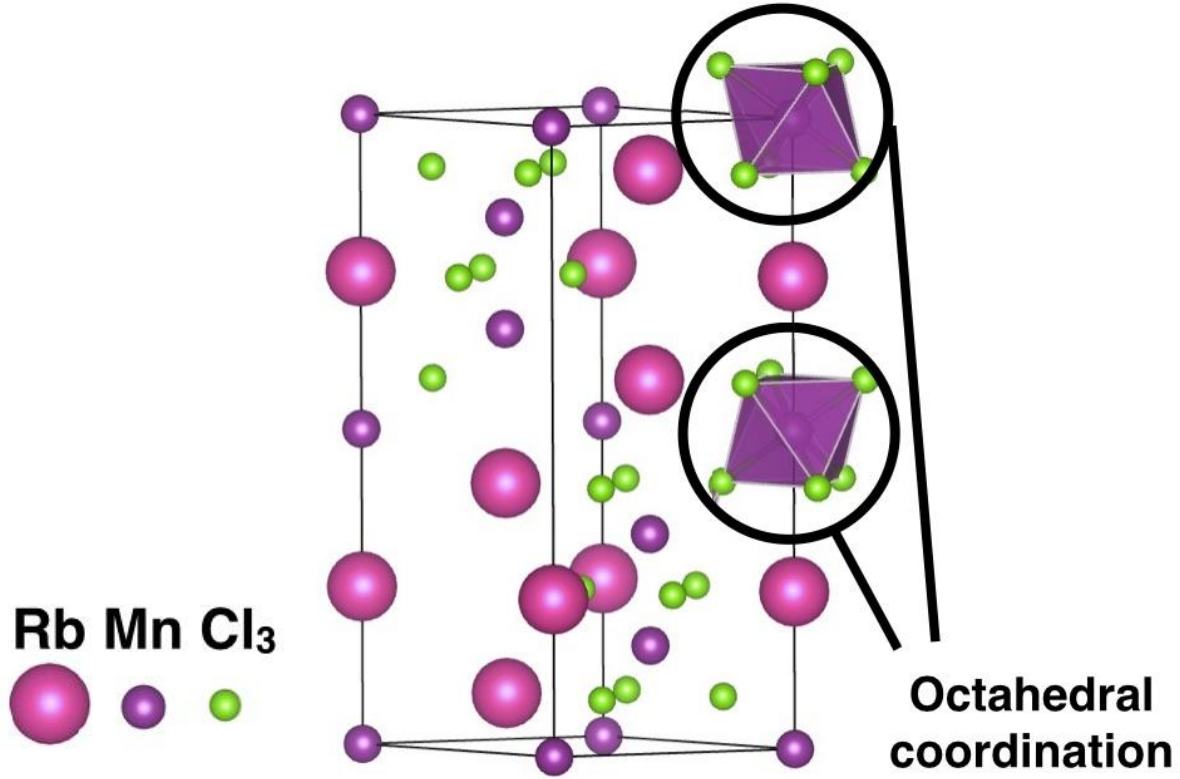
$$RHR^T$$



Given a molecule and a rotated copy,
predicted forces are the same up to rotation.
(Predicted forces are equivariant to rotation.)
Additionally, networks generalize to molecules with similar motifs.



These networks can recognize equivalent recurring geometric patterns that appear in different locations and orientations (from seeing only one example)**.**



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

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

To do this… we first needed to build a general package for prototyping and scaling E(3)NNs.

e3nn

e3nn: a modular PyTorch framework for Euclidean neural networks

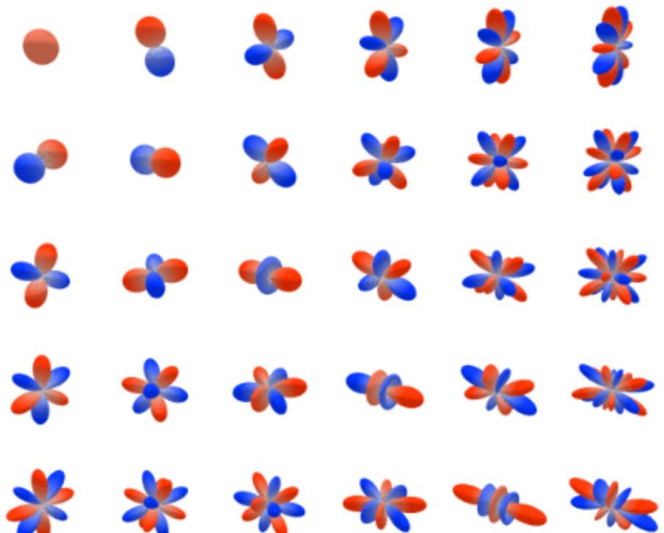
View My GitHub Profile

Hosted on GitHub Pages — Theme by orderedlist

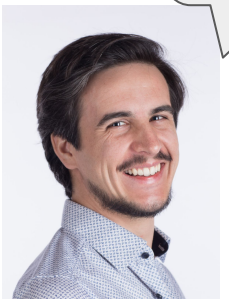
Welcome to e3nn!

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.

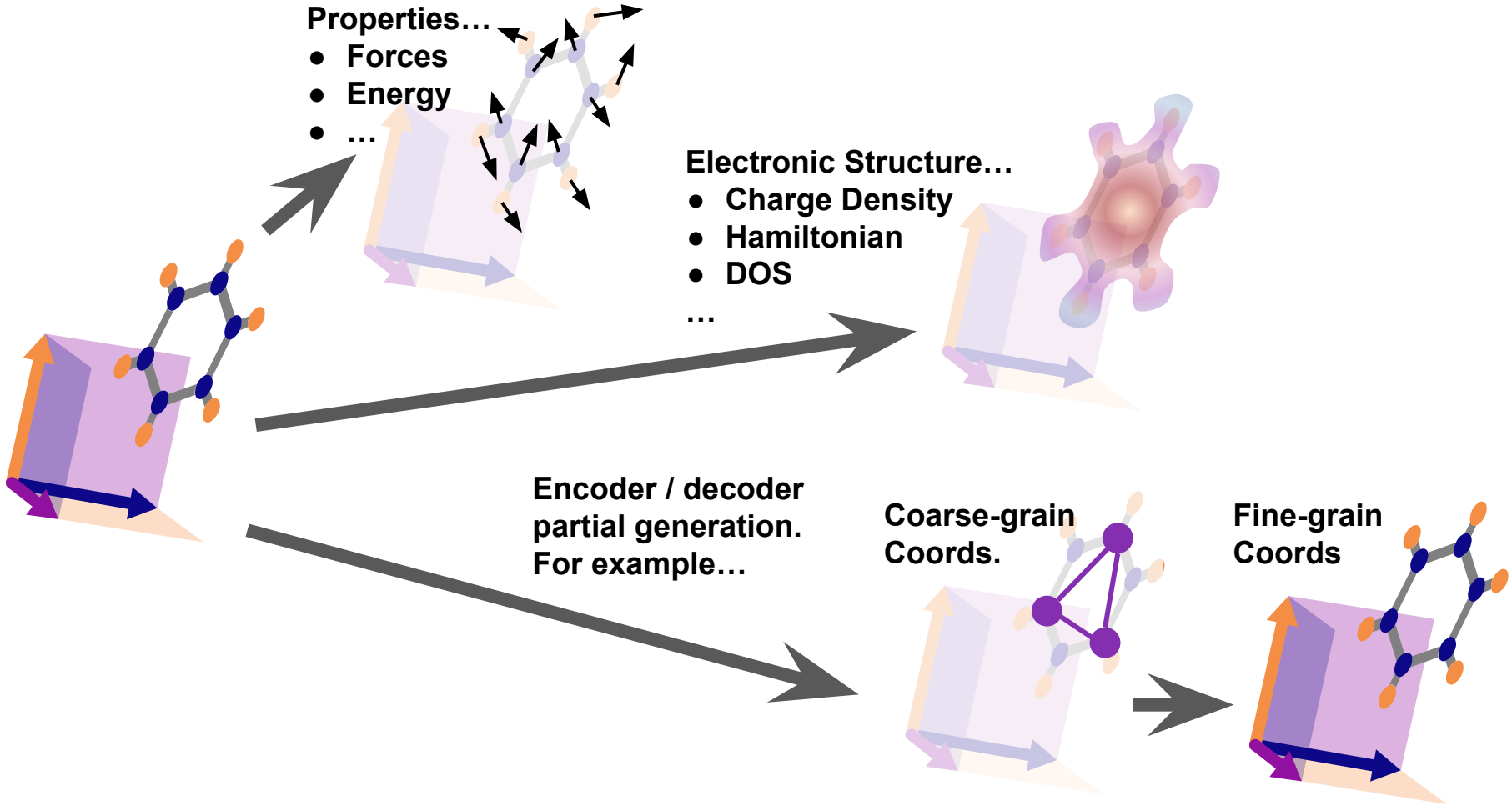


Also e3nn-jax!



Mario Geiger

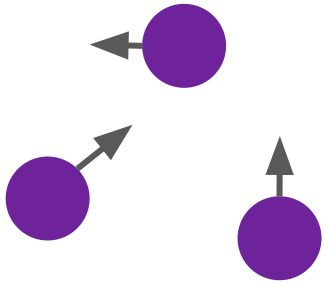
We've used E(3)NNs to build data-efficient and scalable models of physical processes.



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

E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.

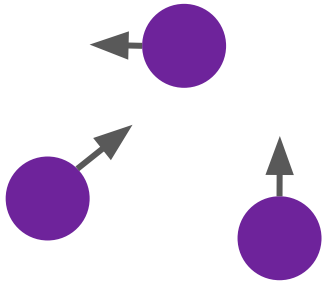
We've used E(3)NNs to build data-efficient and scalable models of physical processes.
E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.



Ab initio molecular dynamics

- Predict forces on atoms with quantum mechanical accuracy
- move atoms small distance in direction of force
- wash, rinse, repeat

We've used E(3)NNs to build data-efficient and scalable models of physical processes.
E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.



Ab initio molecular dynamics

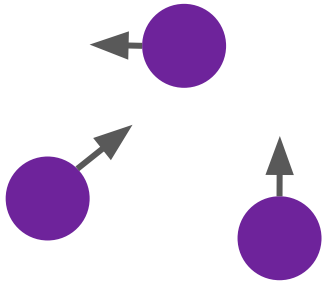
- Predict forces on atoms with quantum mechanical accuracy
- move atoms small distance in direction of force
- wash, rinse, repeat

Problem! Methods scale poorly with number of electrons 😭

- Density functional theory (DFT) ⇨ $O(n^3)$
- Coupled cluster singles doubles triples CCSD(T) ⇨ $O(n^8)$
- Note, this is for EACH step

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

E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.

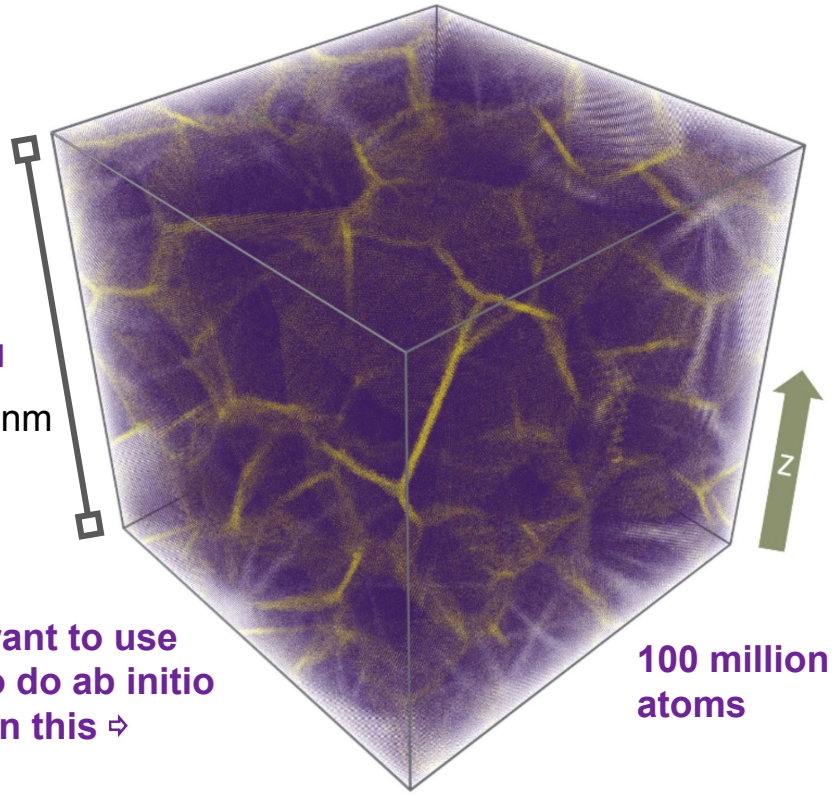


Ab initio molecular dynamics

- Predict forces on atoms with quantum mechanical accuracy
- move atoms small distance in direction of force
- wash, rinse, repeat

Problem! Methods scale poorly with number of electrons 😭

- Density functional theory (DFT) ⇨ $O(n^3)$
- Coupled cluster singles doubles triples CCSD(T) ⇨ $O(n^8)$
- Note, this is for EACH step

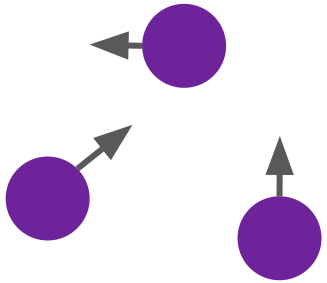


We want to use ML to do ab initio MD on this ⇨

100 million atoms

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

E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.

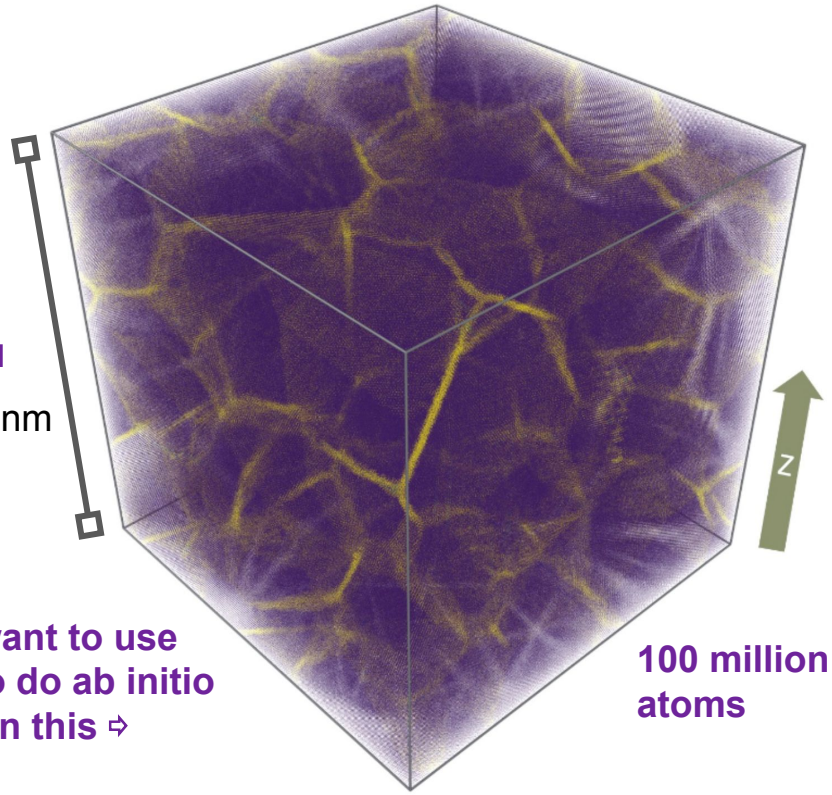


Ab initio molecular dynamics

- Predict forces on atoms with quantum mechanical accuracy
- move atoms small distance in direction of force
- wash, rinse, repeat

Problem! Methods scale poorly with number of electrons 😭

- Density functional theory (DFT) ⇨ $O(n^3)$
- Coupled cluster singles doubles triples CCSD(T) ⇨ $O(n^8)$
- Note, this is for EACH step



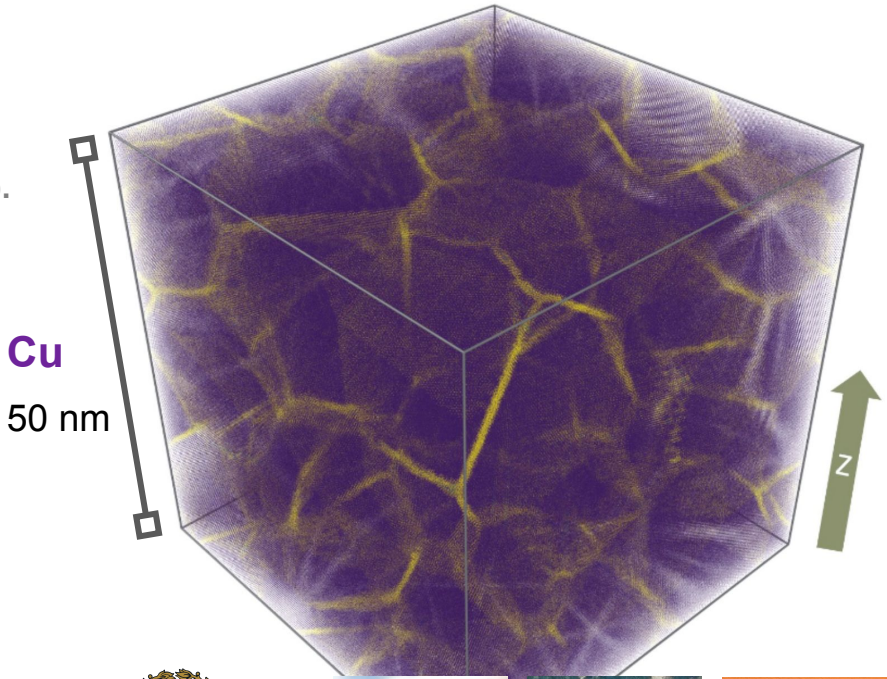
We want to use ML to do ab initio MD on this ⇨

100 million atoms

Because with DFT it takes longer than the age of the universe to compute 😭😭😭

We've used E(3)NNs to build data-efficient and scalable models of physical processes.
E(3)NNs are state-of-the-art in accuracy for *ab initio* machine learned molecular dynamics.

Dec. 2020 – DeePMD
Gordon Bell Prize (the Nobel Prize of Supercomputing)
goes to DeePMD for machine learned MD on
100 million atoms with ab initio accuracy (27,000 GPUs).



With collaborators, Kozinsky Group @ Harvard

Jan. 2021 – NequIP *(Batzner et al.)*
E(3)NN methods *1000x more data efficient*
(more accurate with less data).

Apr. 2022 – Allegro *(Musaelian et al.)*
E(3)NN methods are *more accurate than and as scalable as* DeePMD on 100 million atom systems.
(~100 GPUs).

Open source codes
Allegro: https://github.com/mir-group/allegro
NequIP: https://github.com/mir-group/nequip
e3nn: https://github.com/e3nn/e3nn/

Boris Kozinsky
Simon Batzner
Alby Musaelian

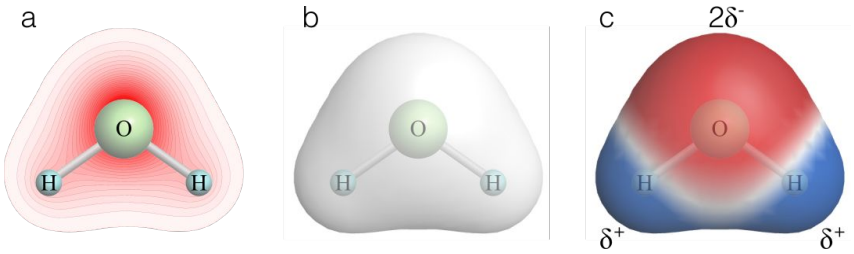


We've used E(3)NNs to build data-efficient and scalable models of physical processes.

…and estimate the "nearsightedness" of water.

We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and estimate the "nearsightedness" of water.

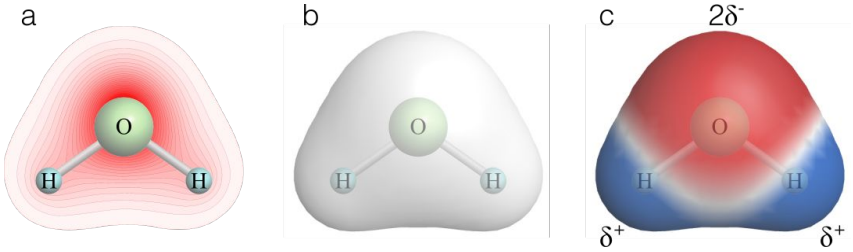
(a) **Ground state charge density of water molecule**
(b) **"Surface" of water molecule**
(c) **Electrostatic potential red - / blue +**



link

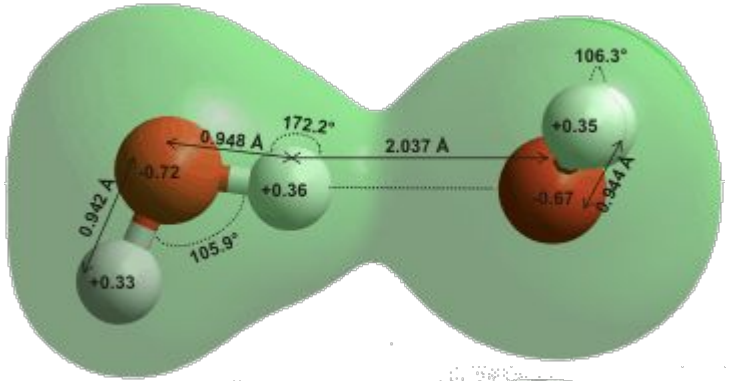
We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and estimate the "nearsightedness" of water.

(a) Ground state charge density of water molecule
(b) "Surface" of water molecule
(c) Electrostatic potential red - / blue +

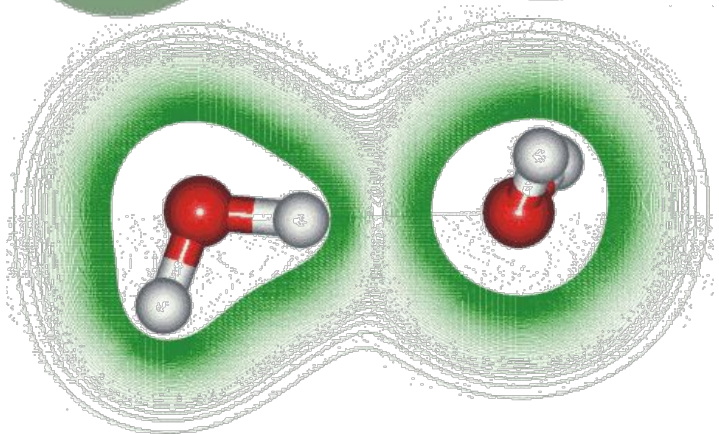


link

"Surface" and charge density of water dimer

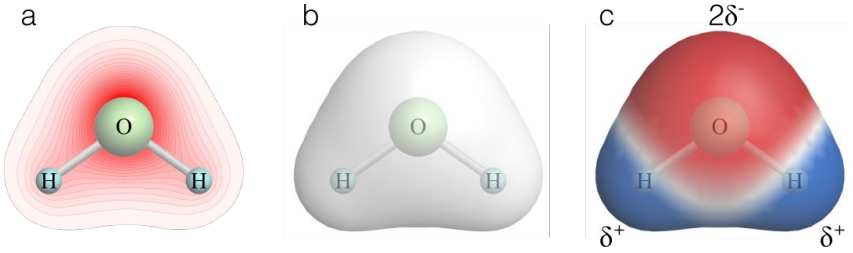


link



We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and estimate the "nearsightedness" of water.

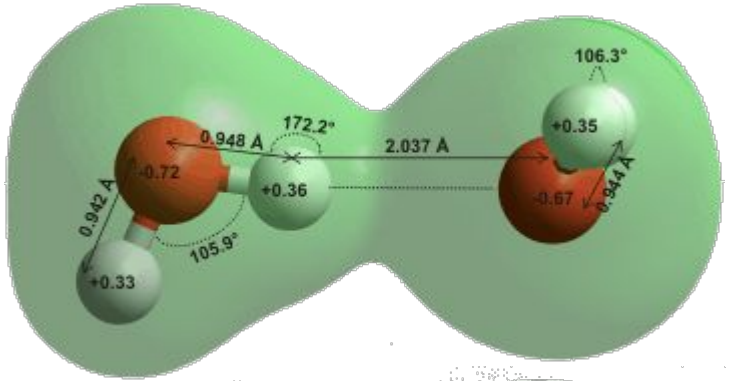
(a) **Ground state charge density of water molecule**
(b) **"Surface" of water molecule**
(c) **Electrostatic potential red - / blue +**



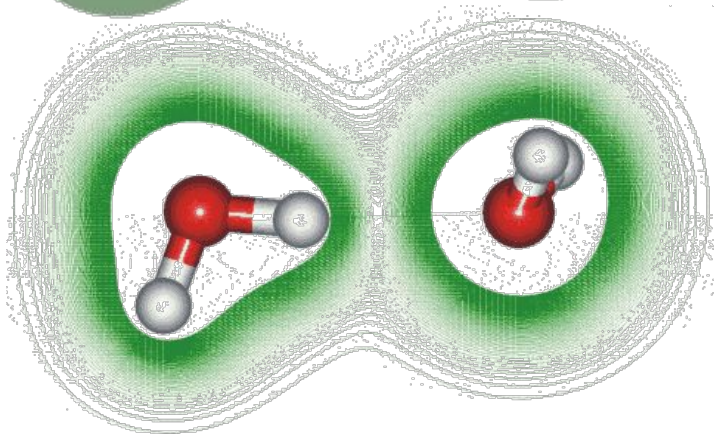
link

- Distributions readily expressed by spherical harmonics and radial functions centered on atoms
- Natural to express with E(3)NNs

"Surface" and charge density of water dimer



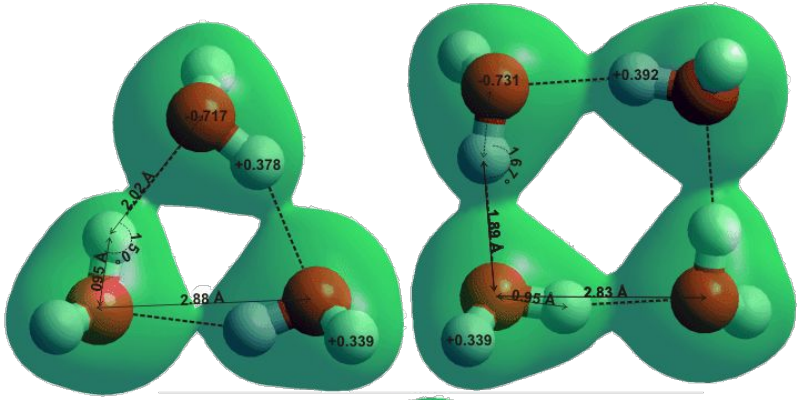
link

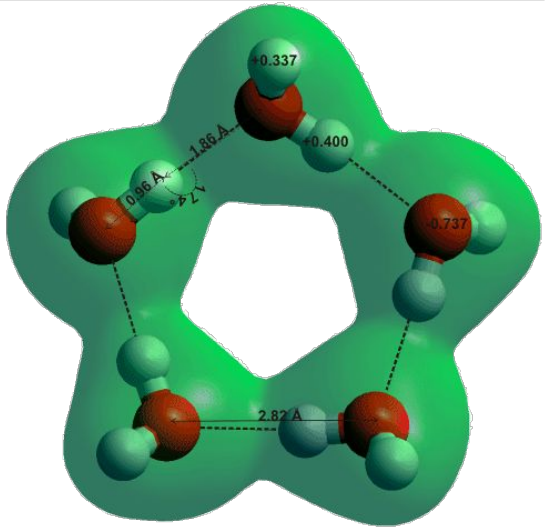


We've used E(3)NNs to build data-efficient and scalable models of physical processes.

…and estimate the "nearsightedness" of water.

"Surfaces" of water
trimer, tetramer, pentamer…



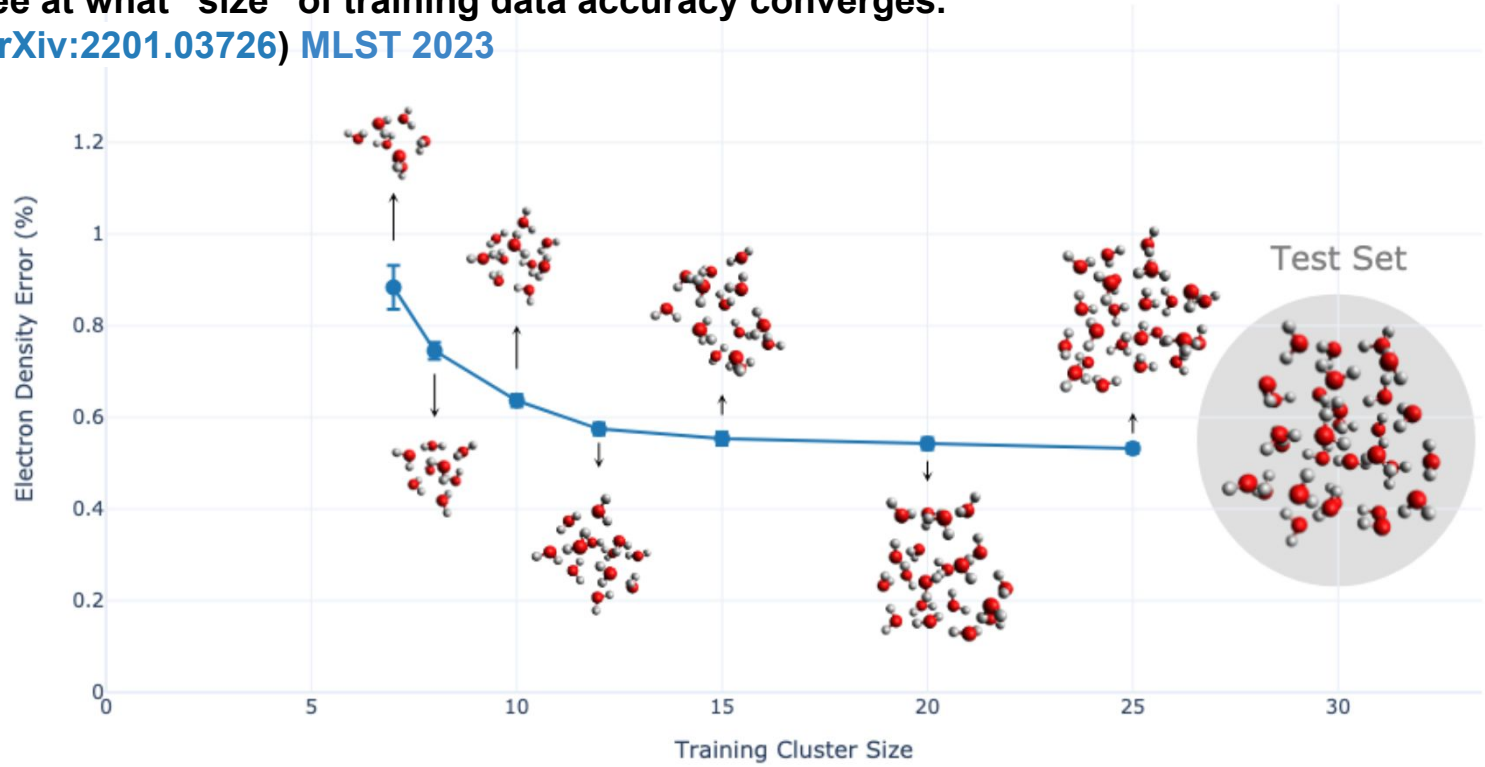


What size of clusters does our model need to see before it "gets the idea"?

link

We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and estimate the "nearsightedness" of water.

Predict electron density (DFT and CCSD) **of larger water cluster when trained on smaller water clusters.**
See at what "size" of training data accuracy converges.
(arXiv:2201.03726) MLST 2023



Sandia National Laboratories

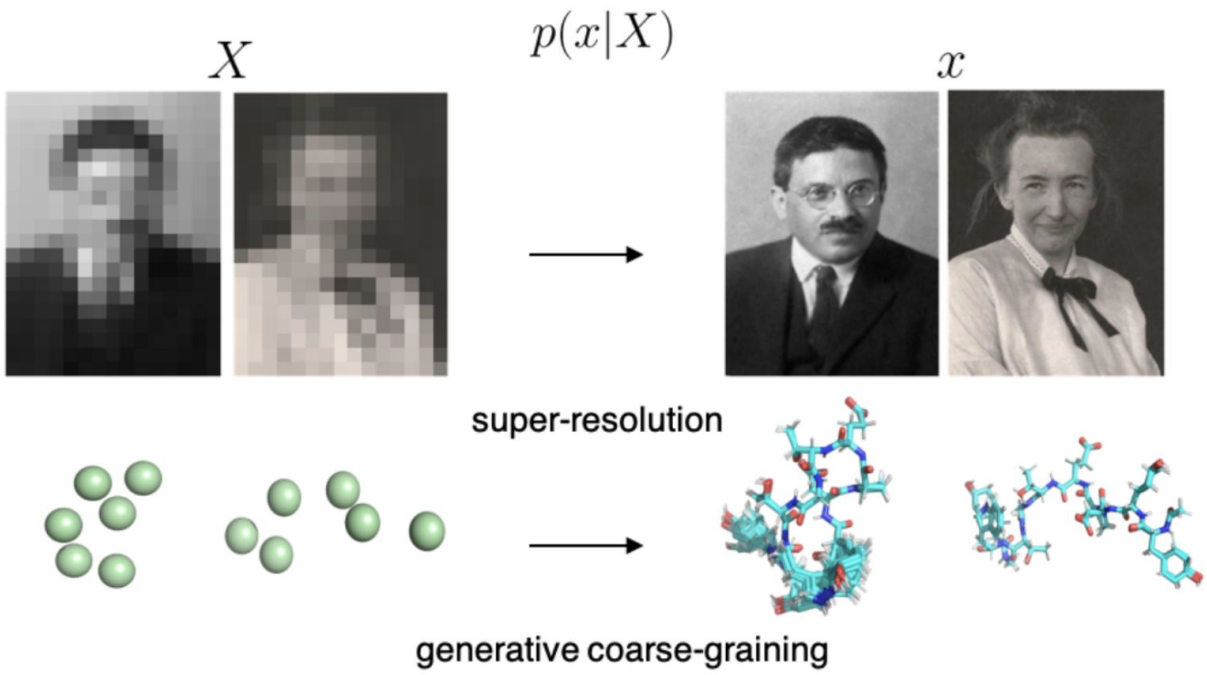
Josh Rackers

Lucas Tecot

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

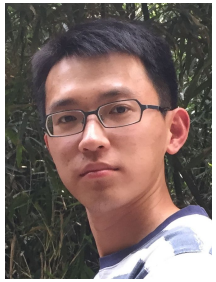
…and generate fine-grain molecular conformations from coarse-grained molecules

Learn to coarsen and "re-fine" molecules
(arXiv:2201.12176)



Rafael Gomez-Bombarelli
Wujie Wang
Minkai Xu
Chen Cai
…

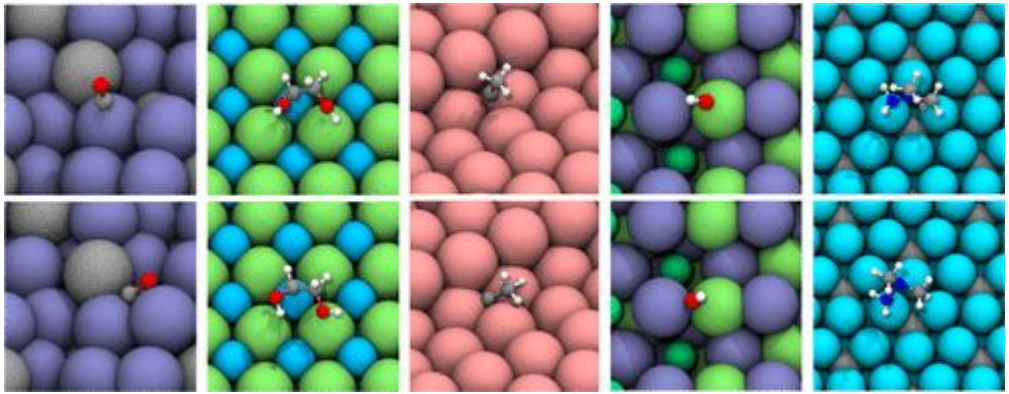


We've used E(3)NNs to build data-efficient and scalable models of physical processes.

…and E(3)NNs are state-of-art for OC20 and with shorter training times.

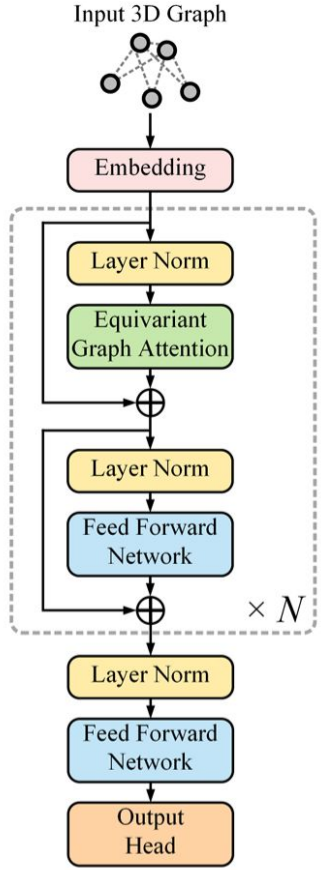
Open Catalysis 2020 Dataset (examples)



Predict energy, forces of given configurations and relaxed structures.

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

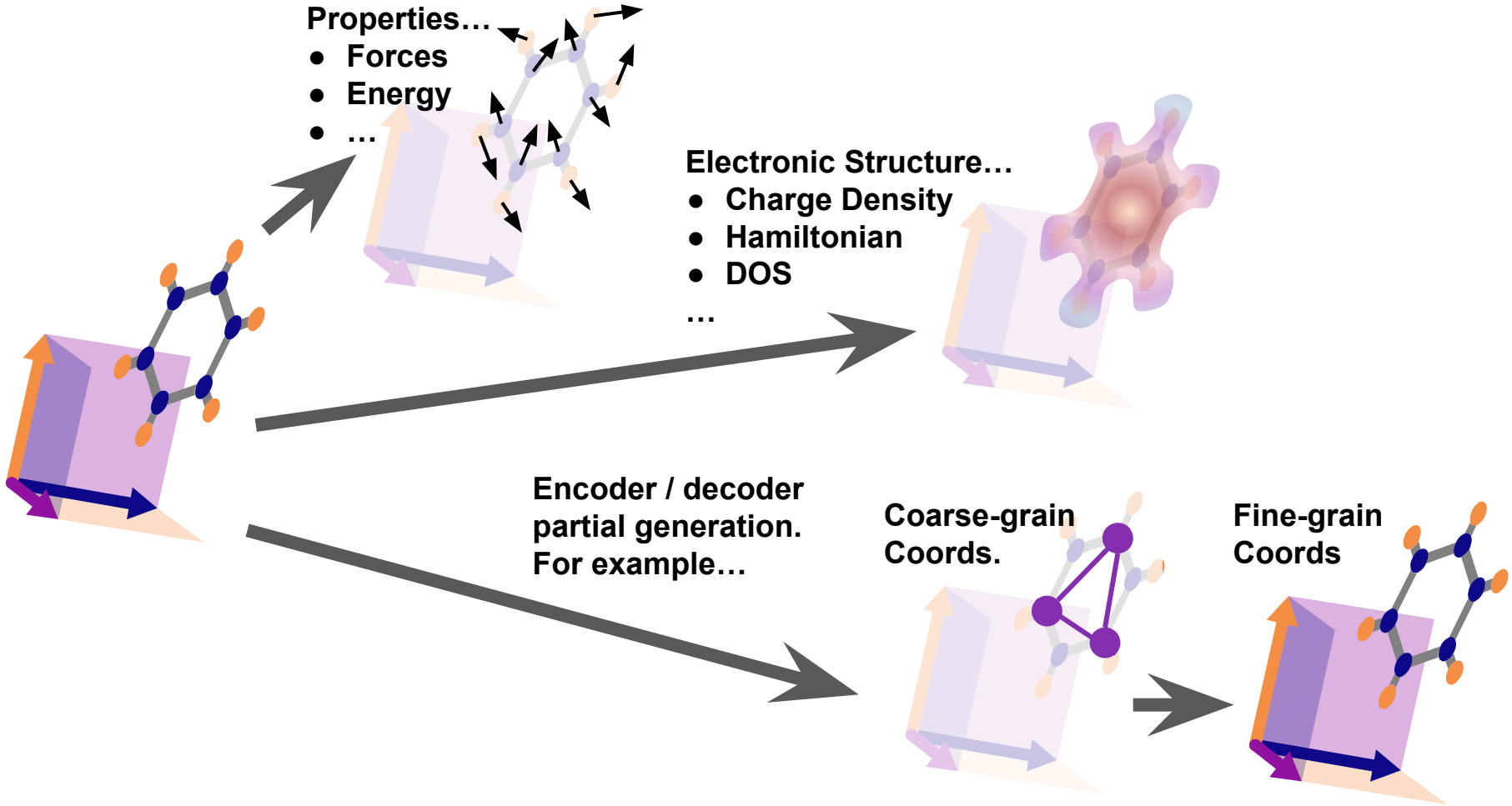
First equivariant transformer to be state-of-art on multiple atomistic benchmarks (QM9, MD17, OC20).



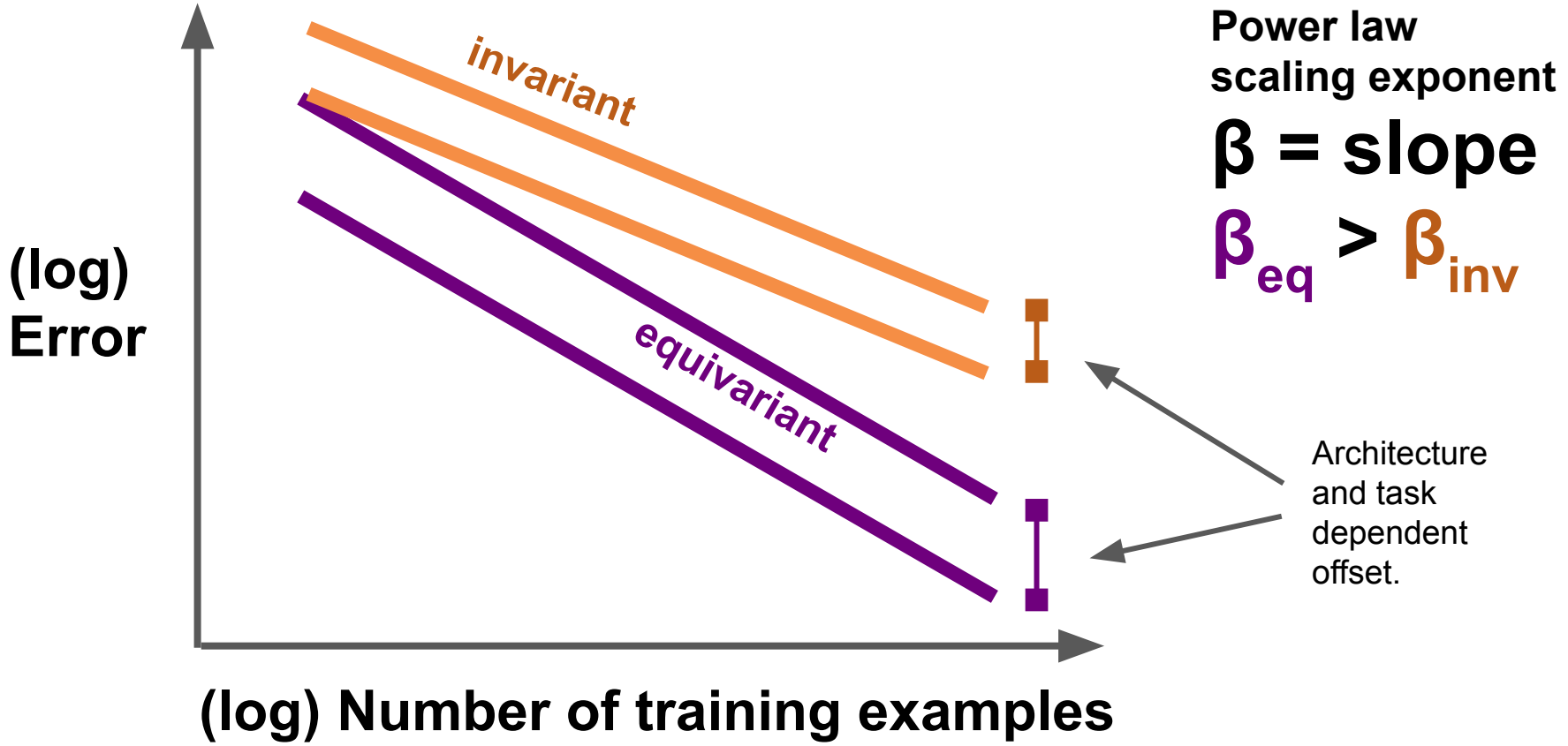
Often, you can often take any technique in ML and adapt it to be equivariant – but it can be subtle!

Yi-Lun Liao

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



Equivariant models are more data efficient than invariant models (even when predicting invariants).
Error reduces more quickly with equivariant than invariant models.



This phenomena is observed across different architectures and training tasks.

Predicting electron densities

J. Rackers et al.

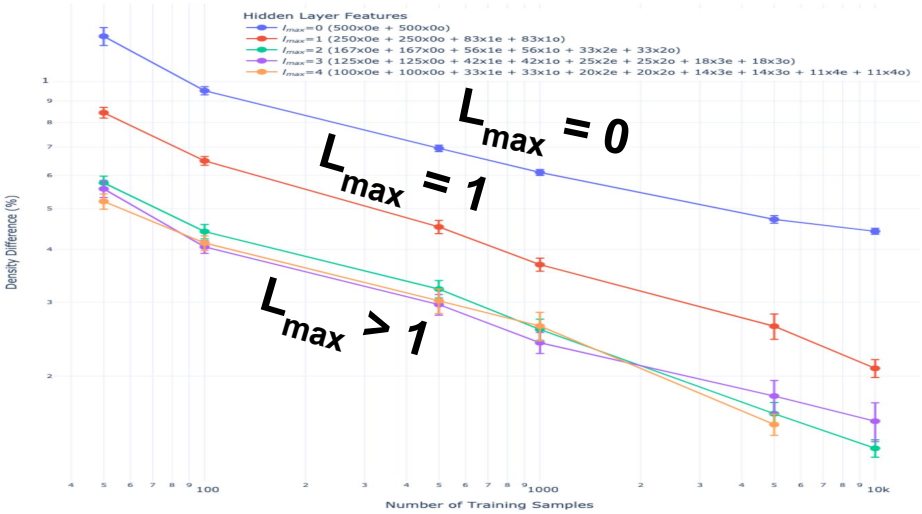


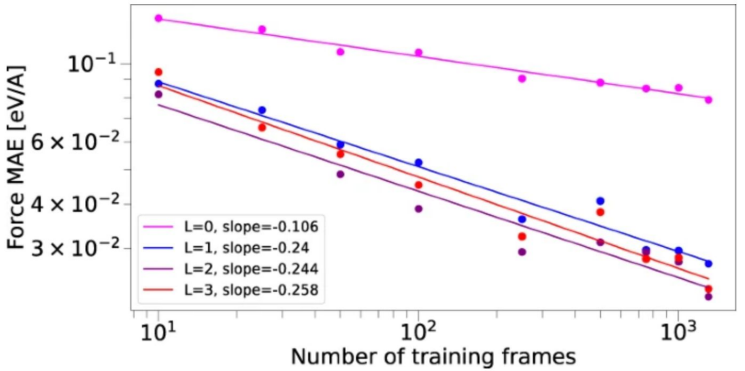
Table A.1 Power laws for neural force field scaling.

Model	R^2	Scaling exponent β
SchNet	0.95	0.17 ± 0.03
PaiNN	0.94	0.26 ± 0.05
Allegro	0.97	0.23 ± 0.03

Force fields

S. Batzner et al.

Fig. 5: Learning curves.



Log-log plot of the predictive error on the water data set from[50] using NequIP with $l \in \{0, 1, 2, 3\}$ as a function of training set size, measured via the force MAE.

N. Frey et al.

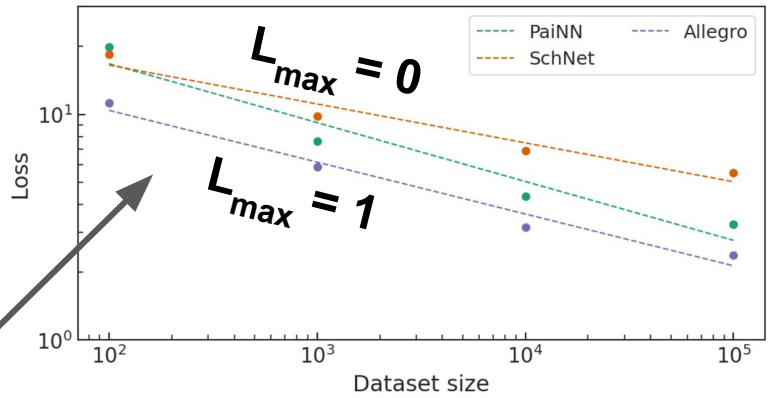


Fig. A.5 Calculating neural scaling power laws for neural force fields. Test loss versus dataset size for PaiNN, Allegro, and SchNet models with fixed capacity, 64.

Euclidean neural networks: neural networks + representation theory

Euclidean neural networks: neural networks + representation theory

neural networks = deep learning ⊂ machine learning ⊂ artificial intelligence

Any machine learned model

input — learnable parameters — predicted output

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

Evaluate performance using a loss / error function

$$\text{loss} = \text{mean}\left((y - y_{true})^2\right)$$

Neural networks must be differentiable so we can update the weights with...

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

neural networks with emojis?

Euclidean neural networks: neural networks + representation theory

(group) representation theory: how do things transform under group action

point groups, space groups, selection rules, symmetry allowed / forbidden properties

Euclidean neural networks: neural networks + representation theory

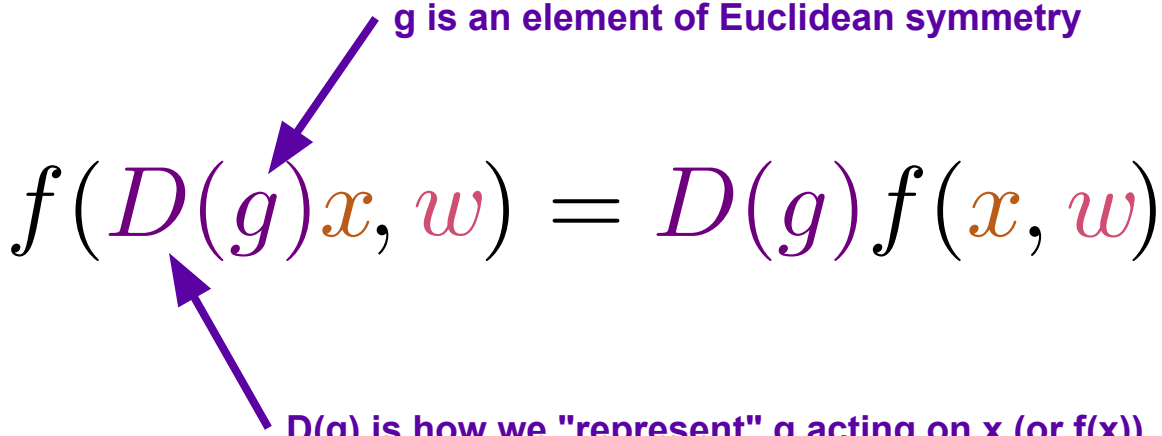
(group) representation theory: how do things transform under group action

point groups, space groups, selection rules, symmetry allowed / forbidden properties

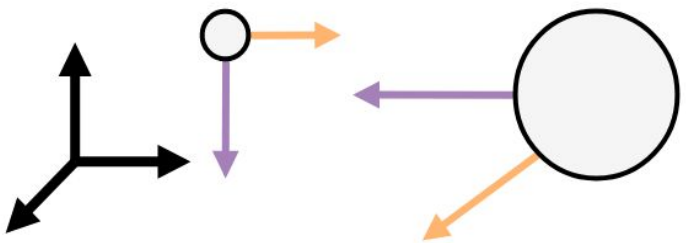
All neural network operations are constructed to commute with group action D(g).

Rotations, translations, inversion

i.e. we can "rotate" the inputs or the outputs and we get the same thing.

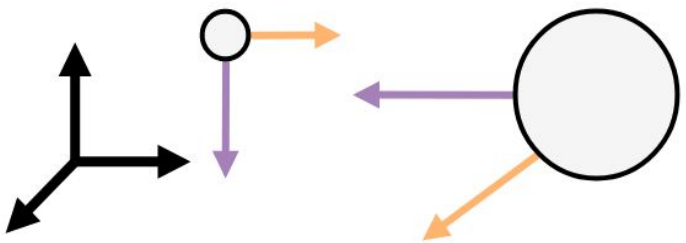


The input to our network is geometry and (geometric tensor) features on that geometry.



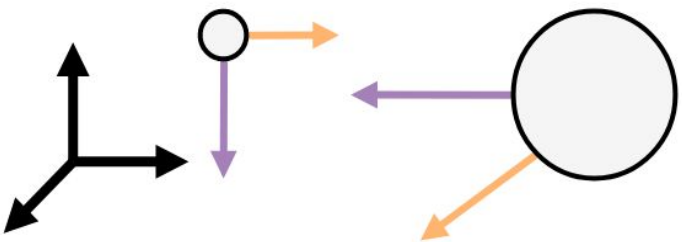
```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
...
```

The input to our network is geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity
as *irreducible representations of O(3).*



```
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 our network is geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity
as *irreducible representations of O(3).*

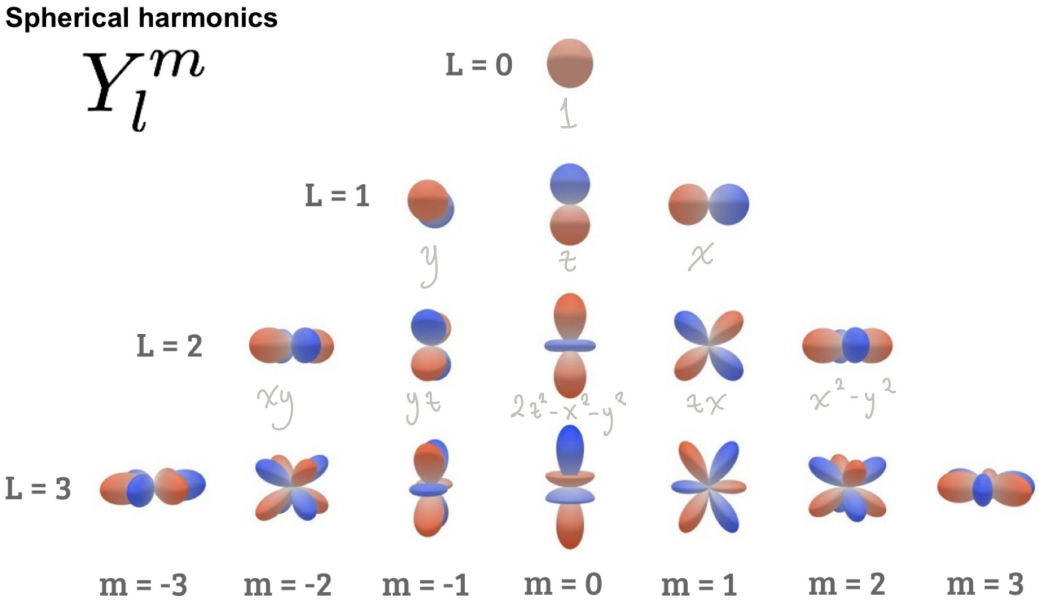


```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even
vector = e3nn.o3.Irrep("1o")  # L=1, odd
irreps = 1 * scalar + 1 * vector + 1 * vector
```

In order for the network to preserve symmetry, we need to tell it what symmetry there is to begin with (e.g. scalars, vectors, ...)

All data (input, intermediates, output) **in E(3)NNs are geometric tensors.**
Geometric tensors are the "data types" of 3D space and have many forms.
But they all transform predictably under rotation, translation, and inversion.

(angular portion of hydrogen atomic orbitals)




```
from e3nn import o3

Rs_s_orbital = o3.Irrep("0e")

Rs_p_orbital = o3.Irrep("1o")

Rs_d_orbital = o3.Irrep("2e")

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

tsmidt@mit.edu | T. E. Smidt. Trends in Chemistry (2021) | e3nn.org

We learn complex descriptions by interacting given features and functions of geometry.

>> e.g. convolutions with Euclidean symmetry

In standard image convolutions, filter depends on coordinate system.

convolutional neural networks:

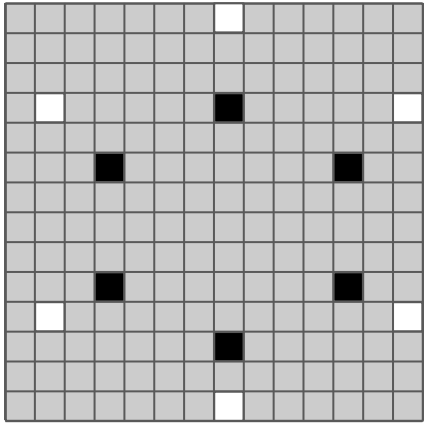
Used for images. In each layer, scan over image with learned filters.

http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

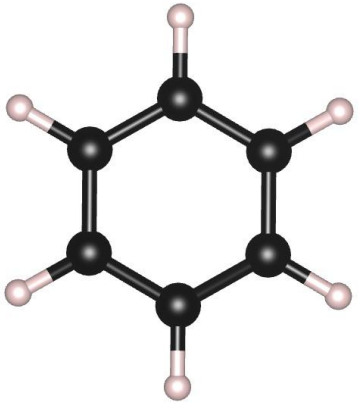
For atoms and other point set data, rather than image convolutions, we perform continuous convolutions…

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

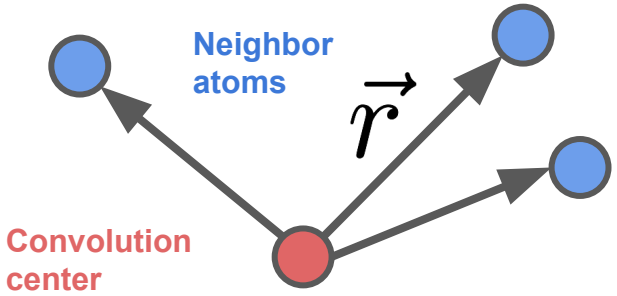
We use points. Images of atomic systems are sparse and imprecise.



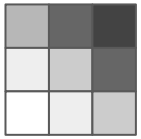
vs.



We use continuous convolutions with atoms as convolution centers.



filter



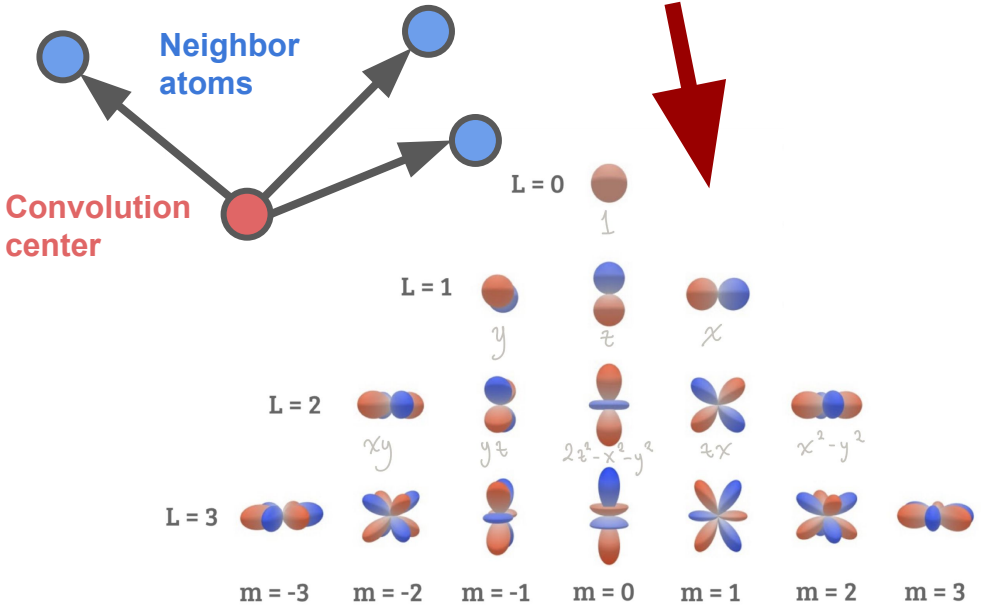
filter function

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

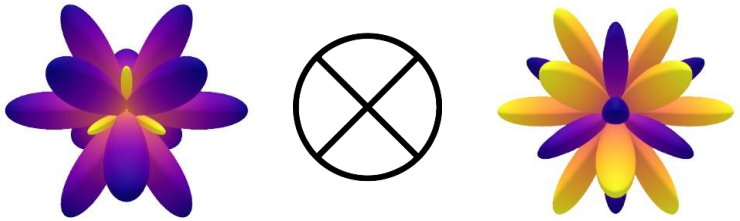
… and we require the convolutional filter to be symmetry-preserving.

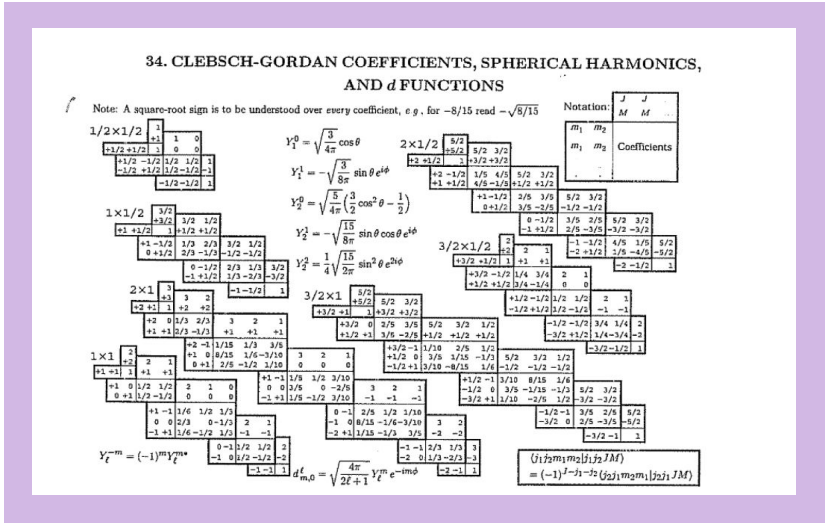
E(3) symmetry preserving convolutional filters are based on learned radial functions and spherical harmonics...

$$W(\vec{r}) = R(r) Y_l^m(\hat{r})$$



...and in order to interact our filters with our inputs we need geometric tensor algebra.




34. CLEBSCH-GORDAN COEFFICIENTS, SPHERICAL HARMONICS, AND d FUNCTIONS

Note: A square-root sign is to be understood over every coefficient, $e.g.$, for $-8/15$ read $-\sqrt{8/15}$.

Notation: $\begin{array}{cc} J & J \\ M & M \\ m_1 & m_2 \\ m_1 & m_2 \end{array}$ Coefficients

$Y_1^0 = \sqrt{\frac{3}{4\pi}}\cos\theta$

$Y_1^1 = -\sqrt{\frac{3}{8\pi}}\sin\theta\, e^{i\phi}$

$Y_2^0 = \sqrt{\frac{5}{4\pi}}\left(\frac{3}{2}\cos^2\theta - \frac{1}{2}\right)$

$Y_2^1 = -\sqrt{\frac{15}{8\pi}}\sin\theta\cos\theta\, e^{i\phi}$

$Y_2^2 = \frac{1}{4}\sqrt{\frac{15}{2\pi}}\sin^2\theta\, e^{2i\phi}$

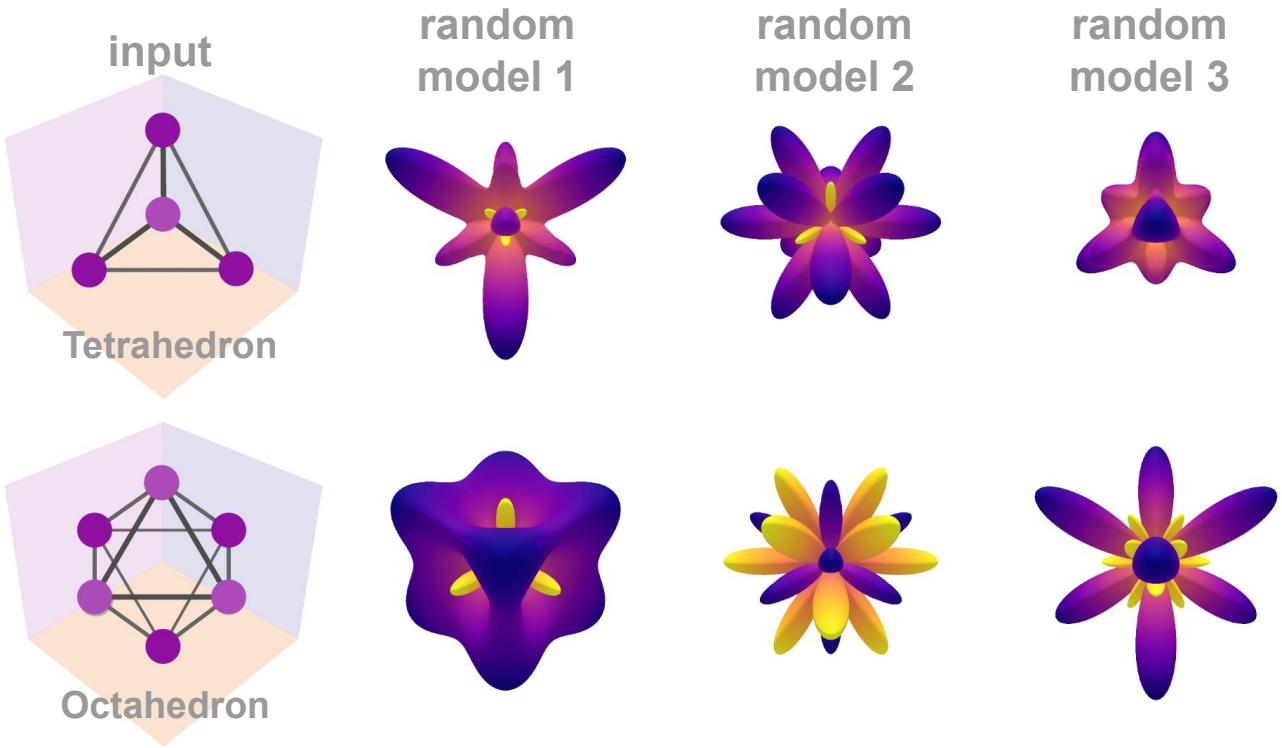
$Y_\ell^{-m} = (-1)^m Y_\ell^{m*}$

$d^\ell_{m,0} = \sqrt{\frac{4\pi}{2\ell+1}}\, Y_\ell^m\, e^{-im\phi}$

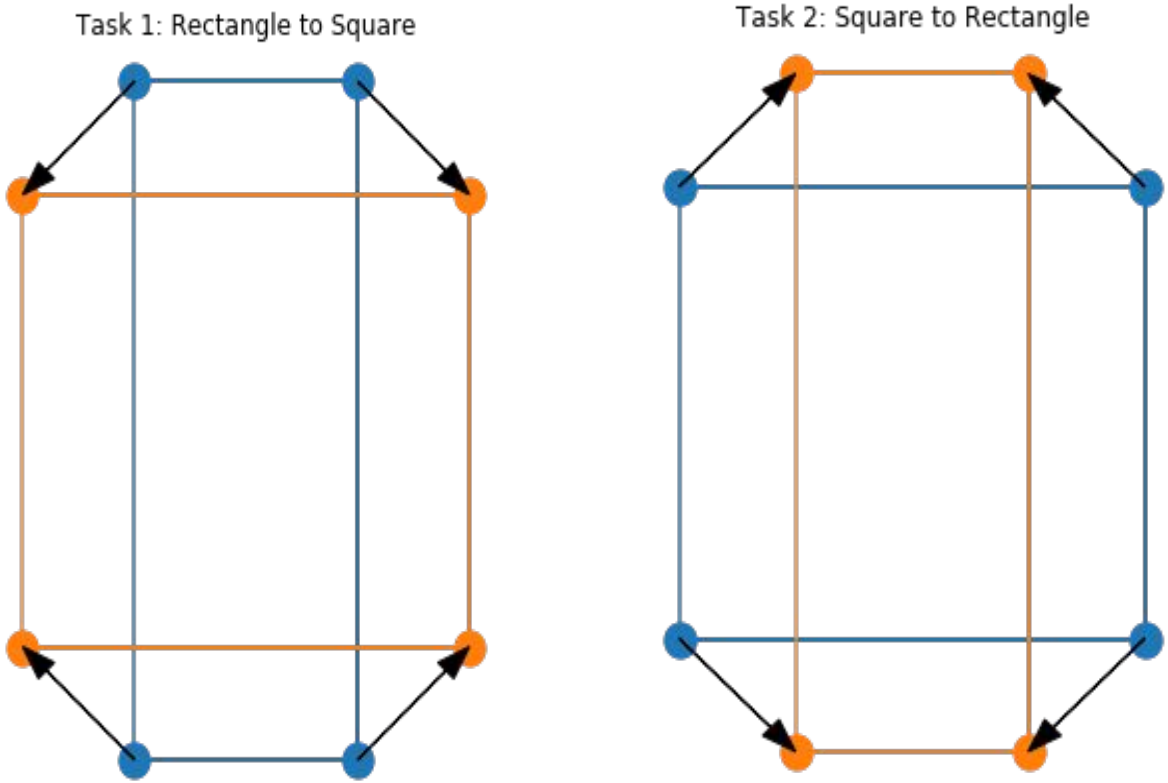
$\langle j_1 j_2 m_1 m_2 | j_1 j_2 JM \rangle = (-1)^{j-j_1-j_2} \langle j_2 j_1 m_2 m_1 | j_2 j_1 JM \rangle$

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

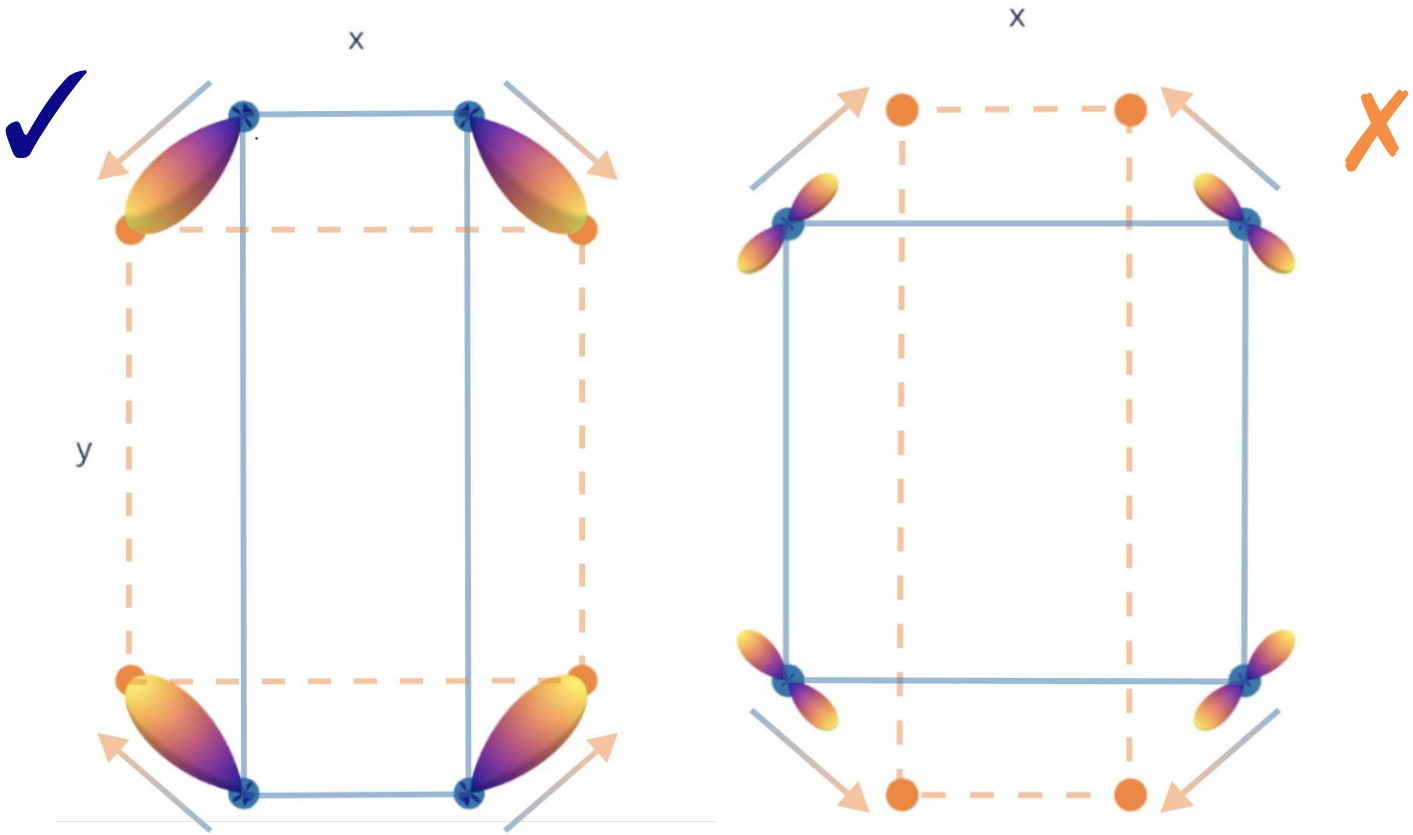
Curie's principle (1894): "When effects show certain asymmetry, this asymmetry must be found in the causes that gave rise to them."



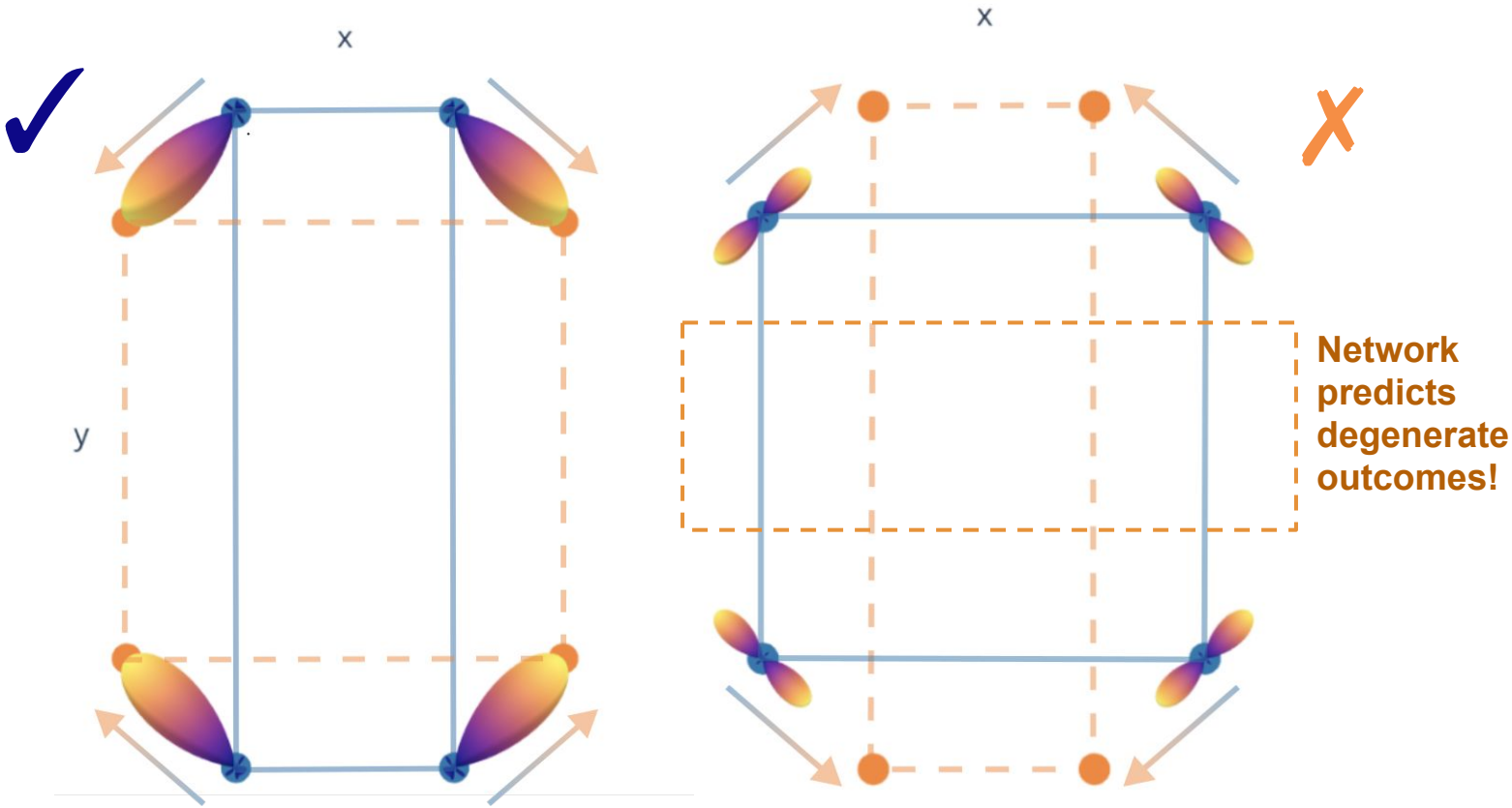
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



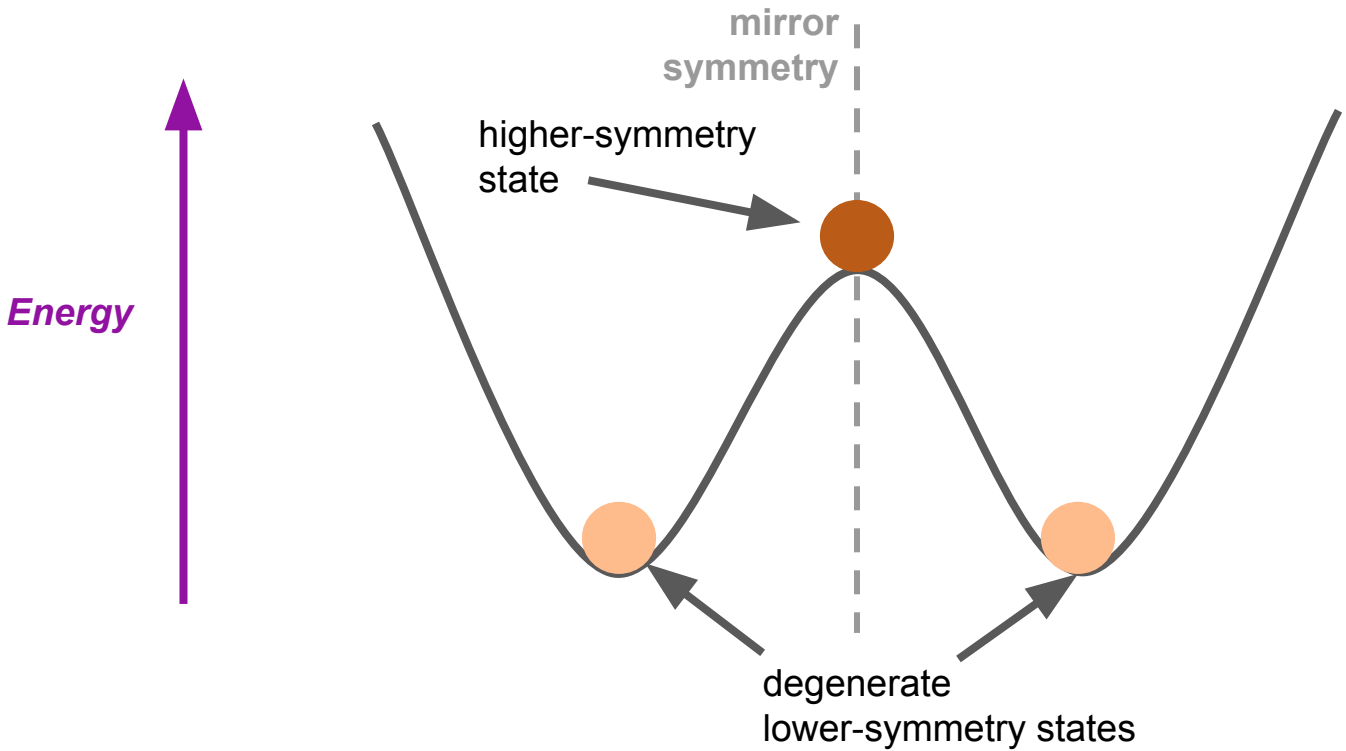
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



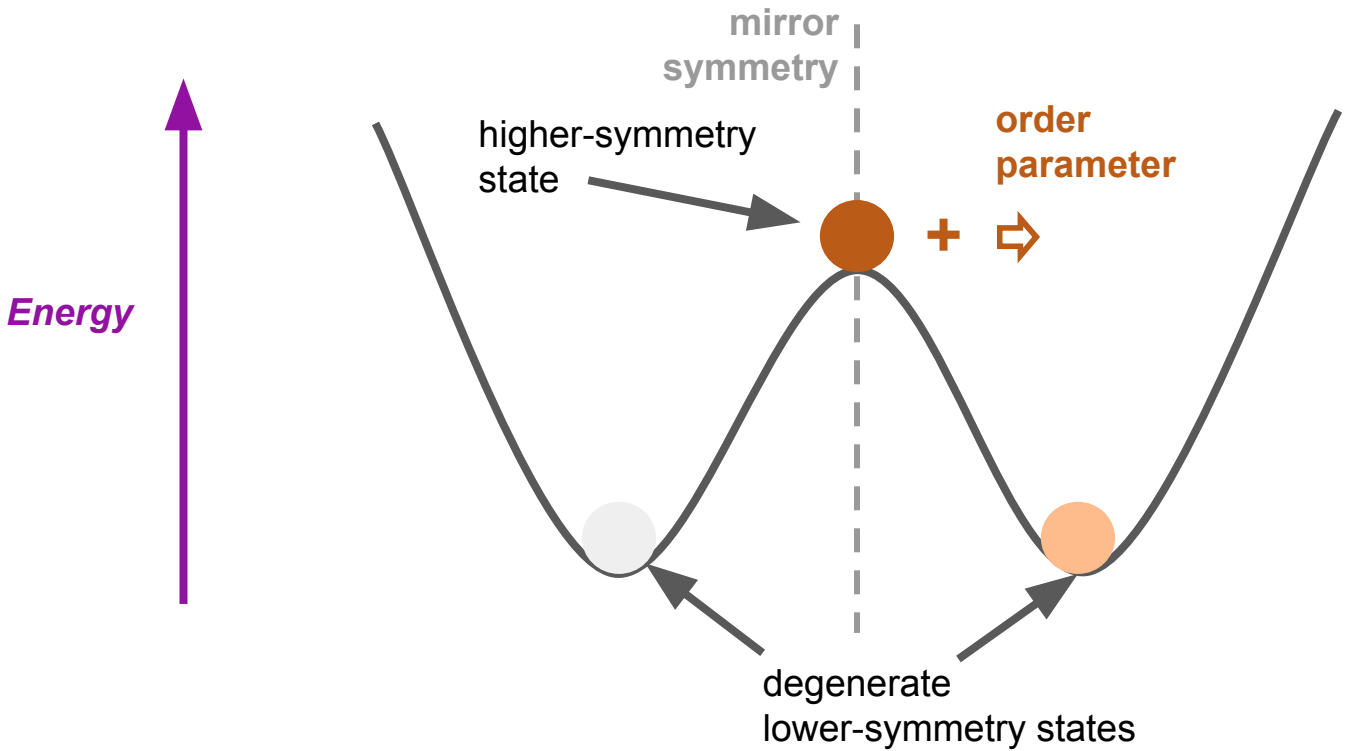
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.



Order parameters describe symmetry breaking and distinguish between degenerate states.



Order parameters describe symmetry breaking and distinguish between degenerate states.



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

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

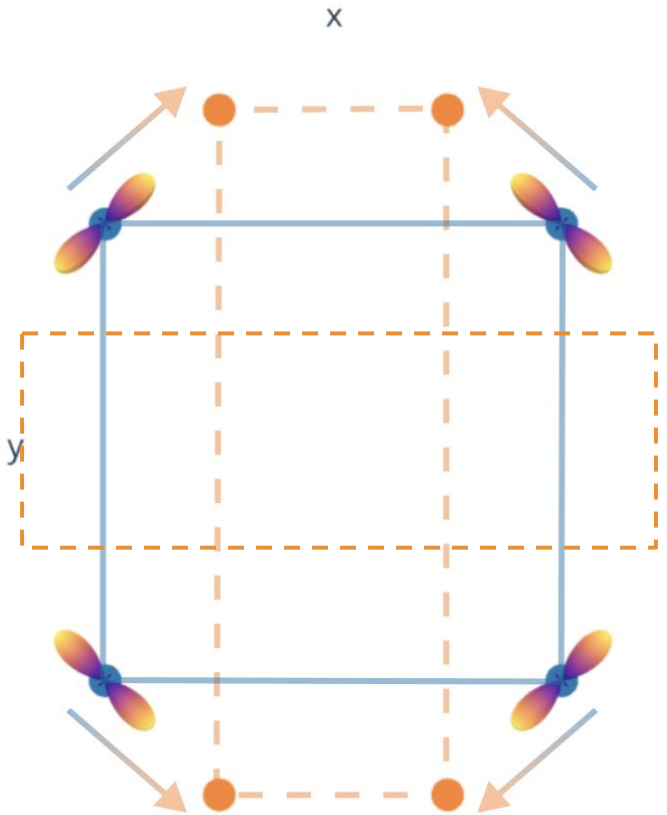
input — learnable parameters — predicted output

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

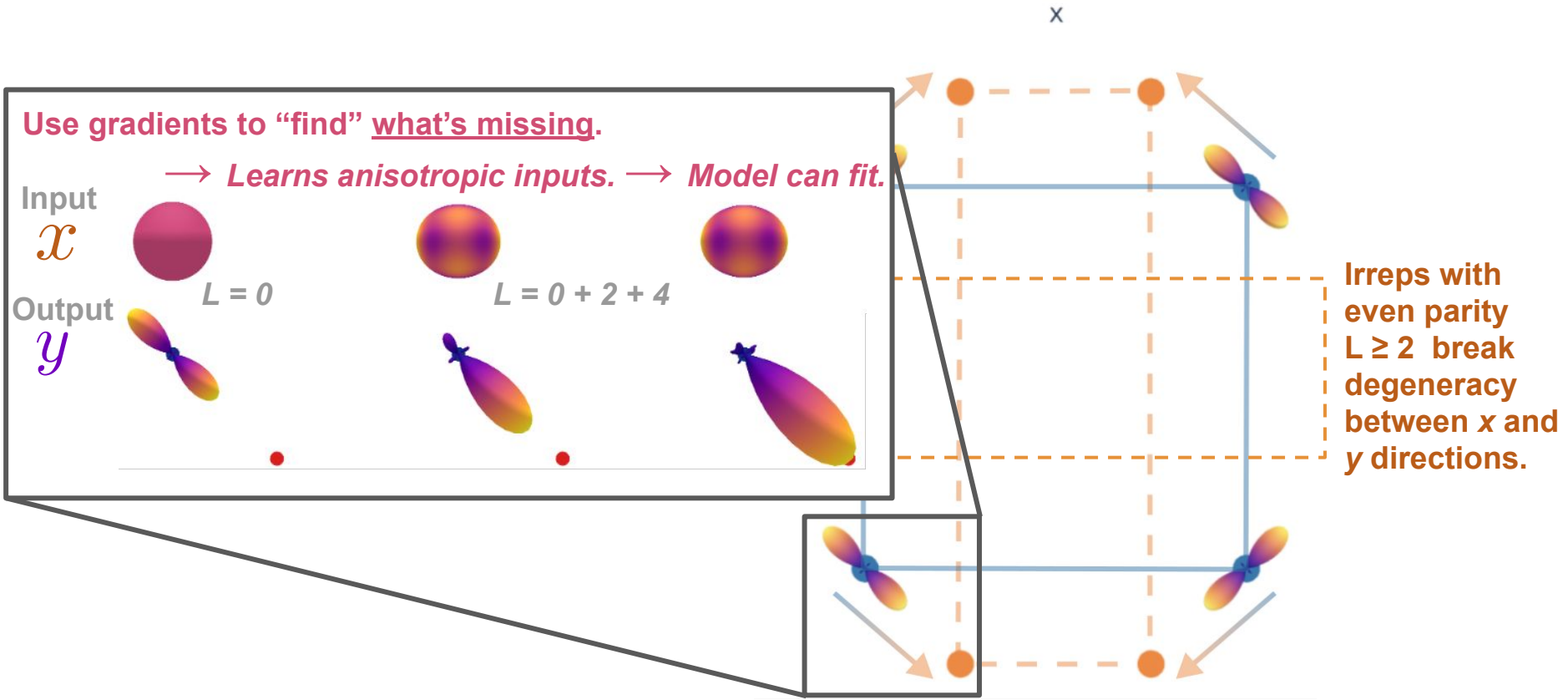
inputs

Update weights using...

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

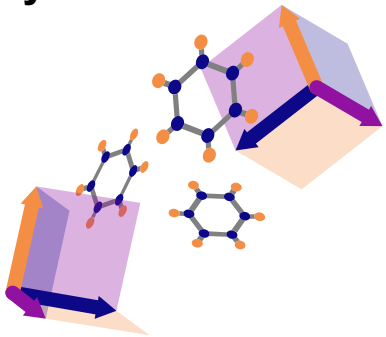


Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").

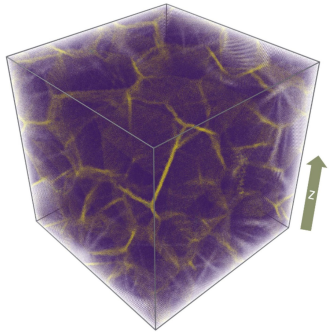


TAKEAWAYS

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

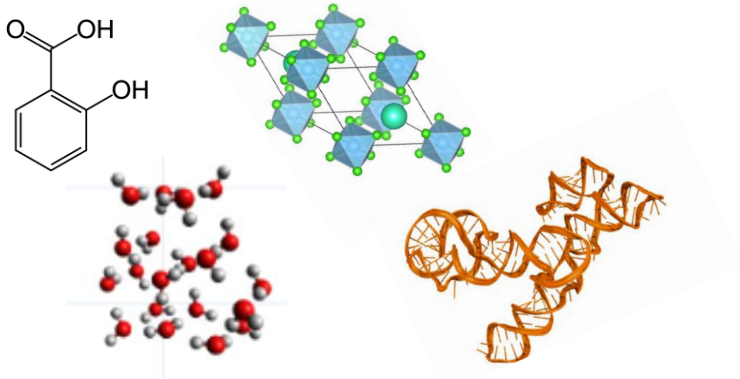


This makes these models data-efficient, robust, scalable, and generalizable.

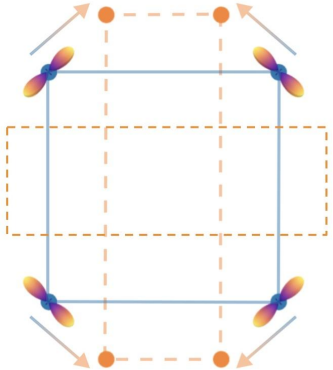


QM accurate MD on 100s of millions of atoms.

E(3)NNs have demonstrated accuracy on a wide range of atomistic systems.



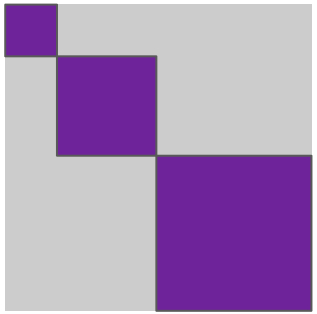
Building assumptions into our models can lead to unexpected consequences.



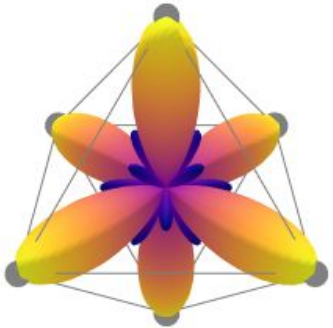
Tess Smidt | tsmidt@mit.edu | e3nn.org

Calling in backup (slides)!

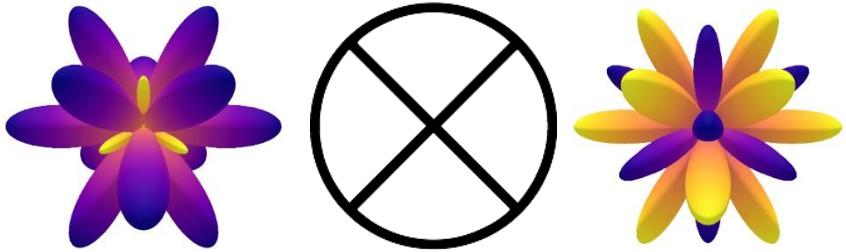
Three essential ingredients for equivariant neural networks



Group representations on tensor vector spaces



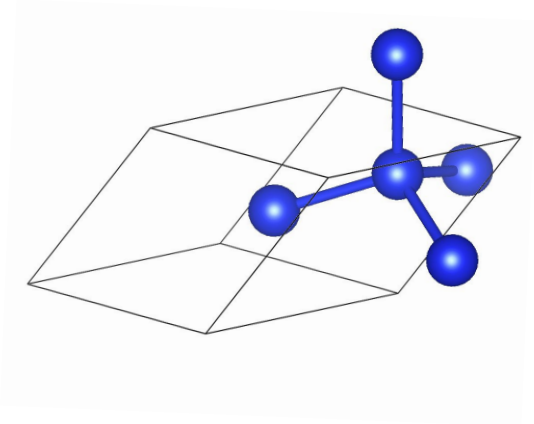
Basis functions (simplest functions that transform as fundamental vector spaces)

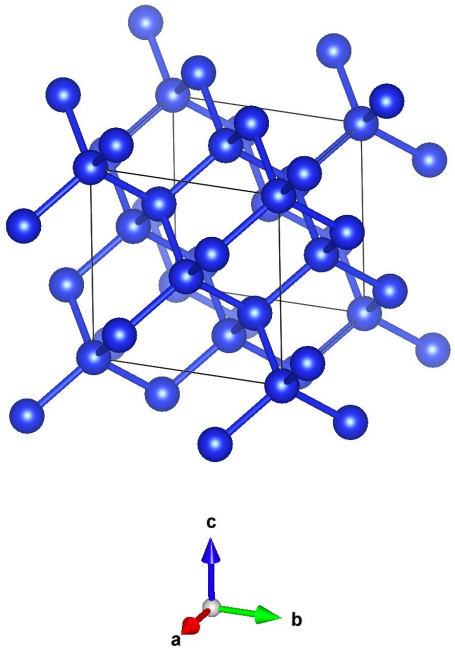


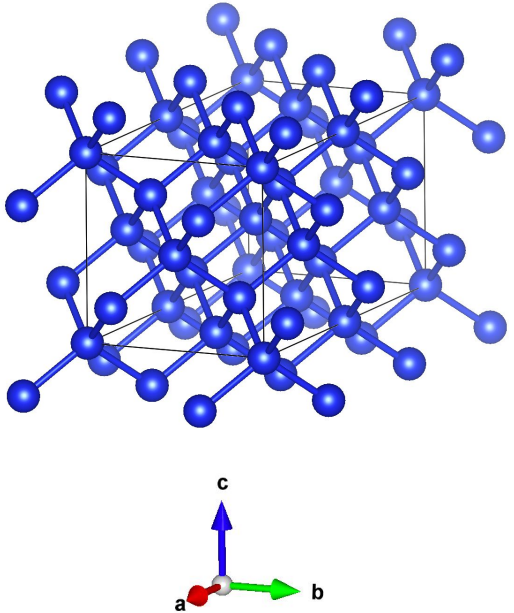
Tensor products (and decomposition back into favored tensor basis)

When given primitive unit cells, conventional unit cells, and supercells of the same crystal the network makes the predictions that mean the same thing.

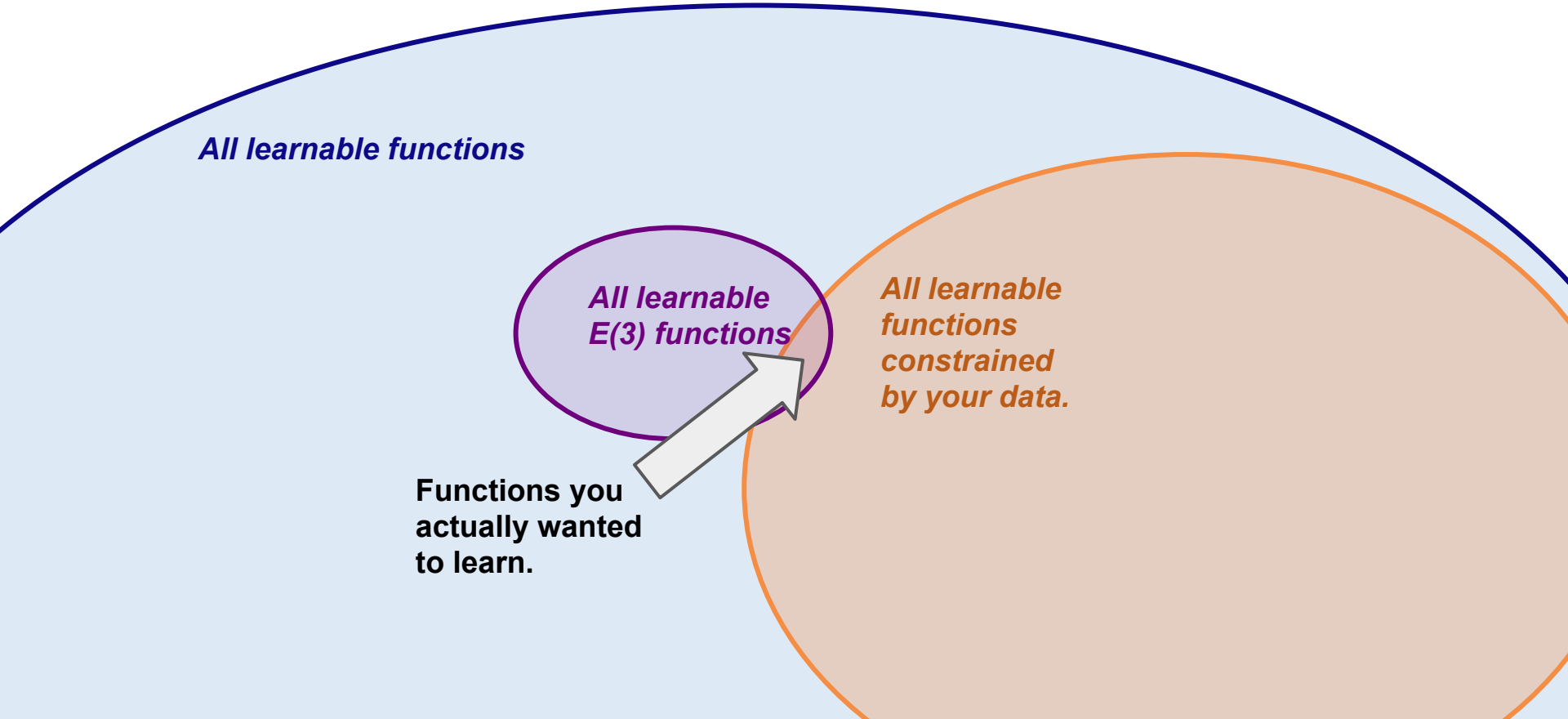
(assuming periodic boundary conditions)



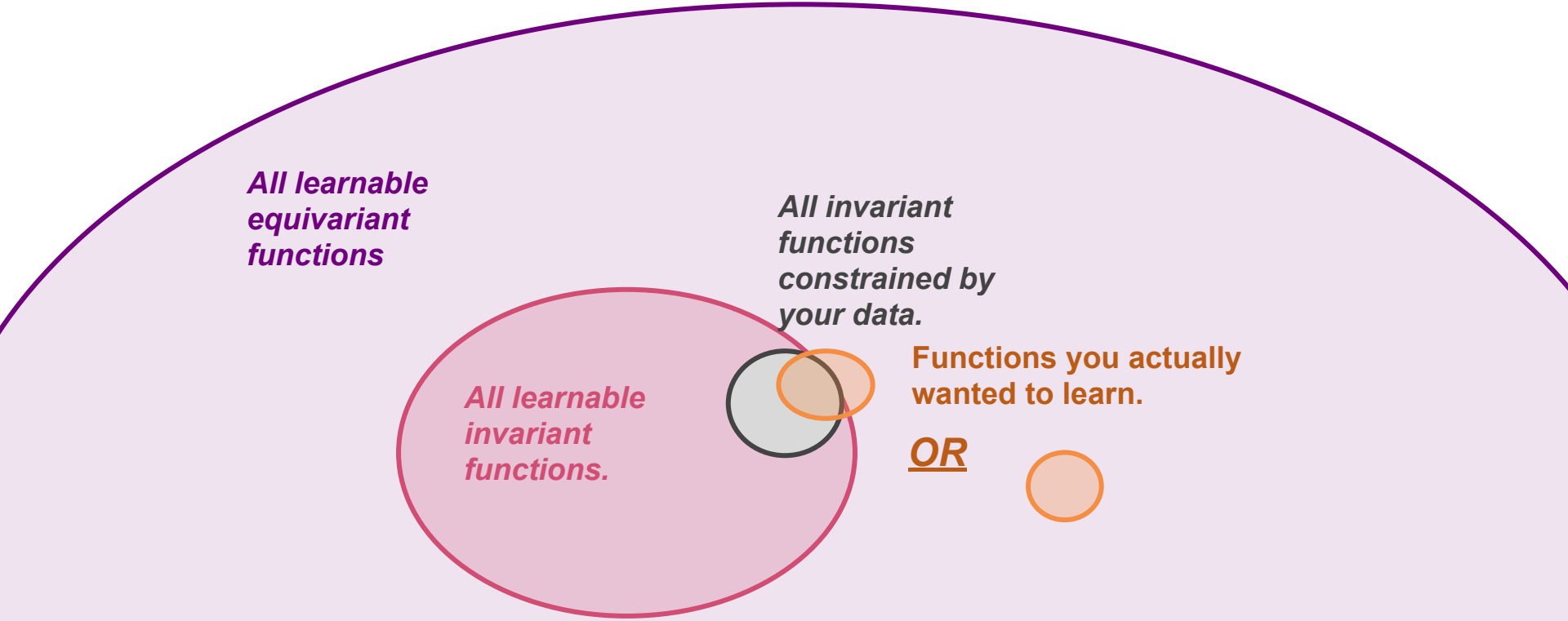




Why limit yourself to functions with (Euclidean) symmetry?
You can *substantially* shrink the space of functions you need to optimize over.
This means you need less data to constrain your function.



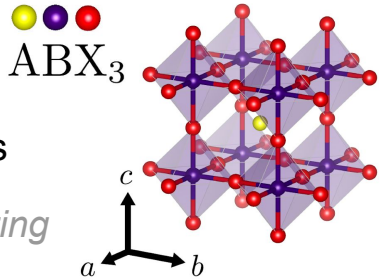
Why not limit yourself to invariant functions?
You have to guarantee that your input features already contain any necessary equivariant interactions *(e.g. cross-products)*.



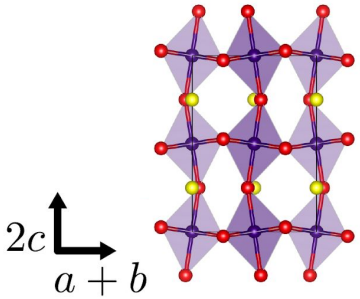
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").

Predict distortion displacements

Octahedral tilting in perovskites



$Pm\bar{3}m$ (221)



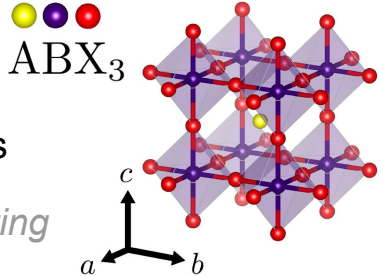
$Imma$ (74)

Target distorted structure

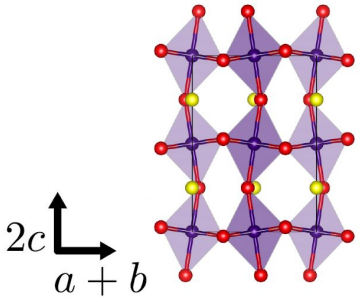
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").

Predict distortion displacements

Octahedral tilting in perovskites

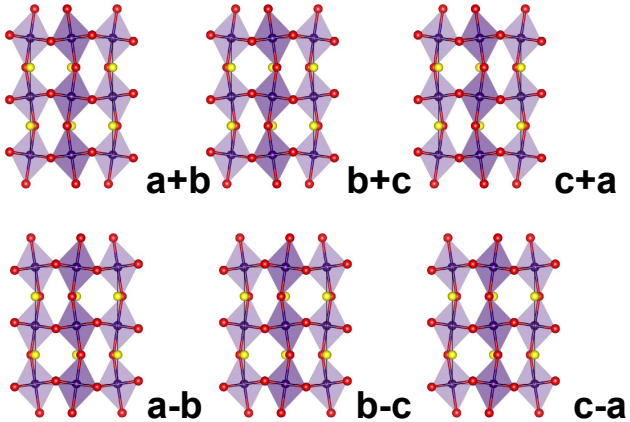


$Pm\bar{3}m$ (221)



$Imma$ (74)

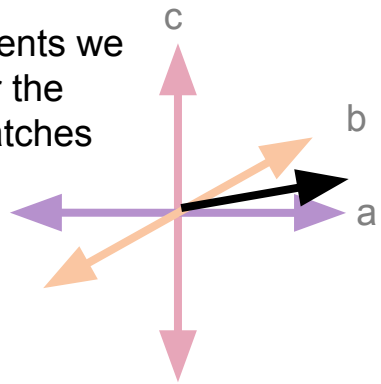
Target distorted structure



6 different vectors in reciprocal space

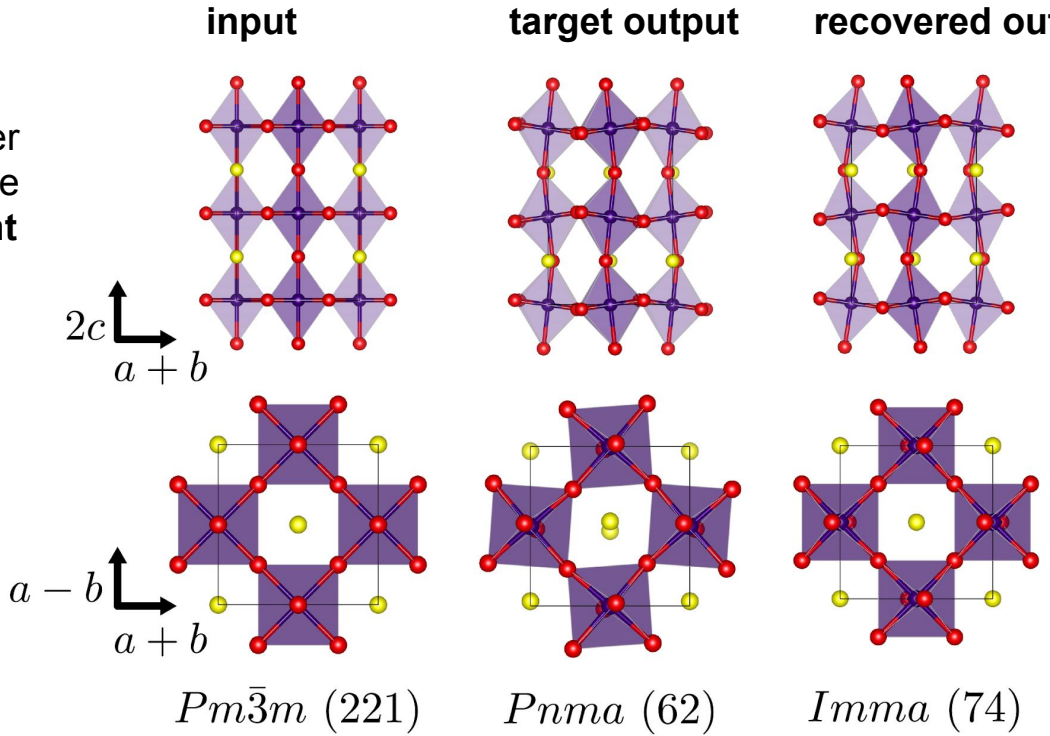
(+½, +½, 0)
(+½, -½, 0)
(0, +½, +½)
(0, +½, -½)
(+½, 0, +½)
(+½, 0, -½)

Using gradients we can recover the o.p. that matches the data.



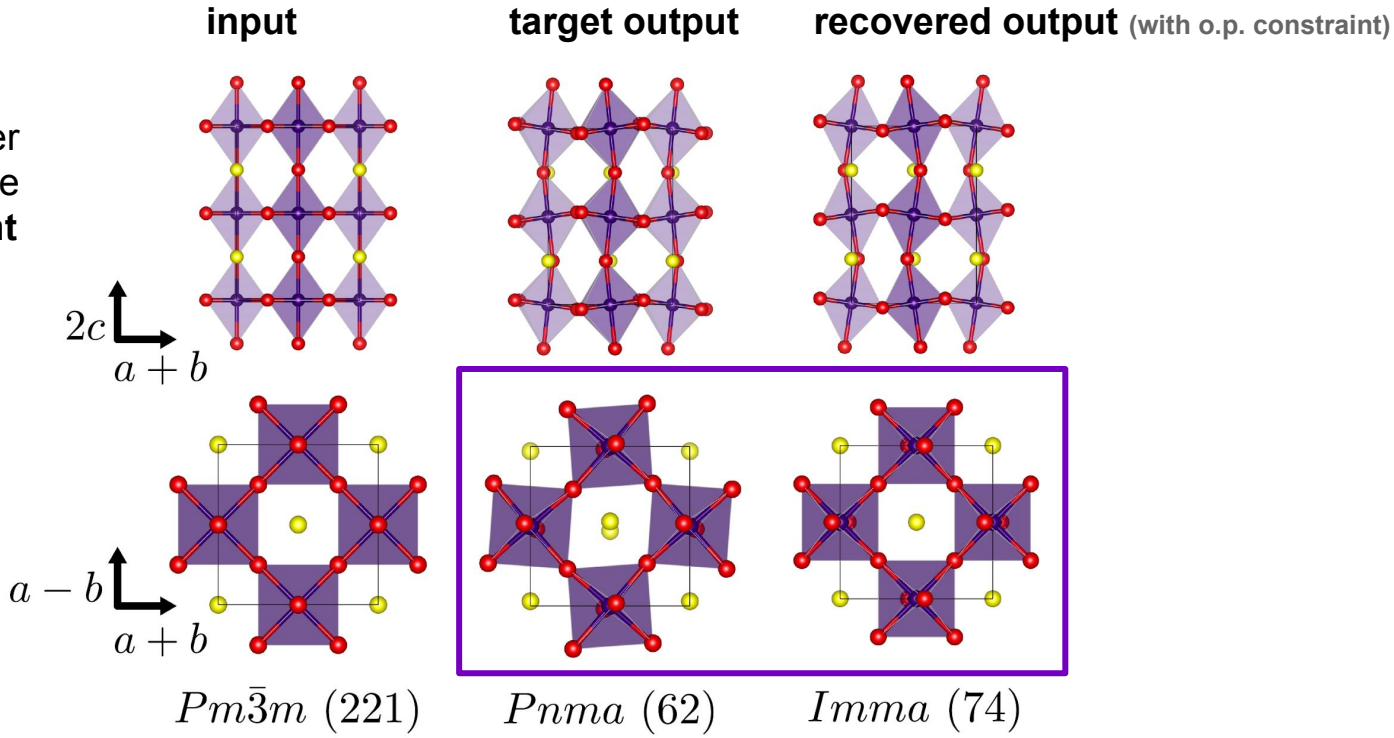
We can additionally put constraints on the "learned" order parameters to recover structures of intermediate symmetry.

e.g. constrain pseudovector order parameters to have **zero z-component**



We can additionally put constraints on the "learned" order parameters to recover structures of intermediate symmetry.

e.g. constrain pseudovector order parameters to have **zero z-component**

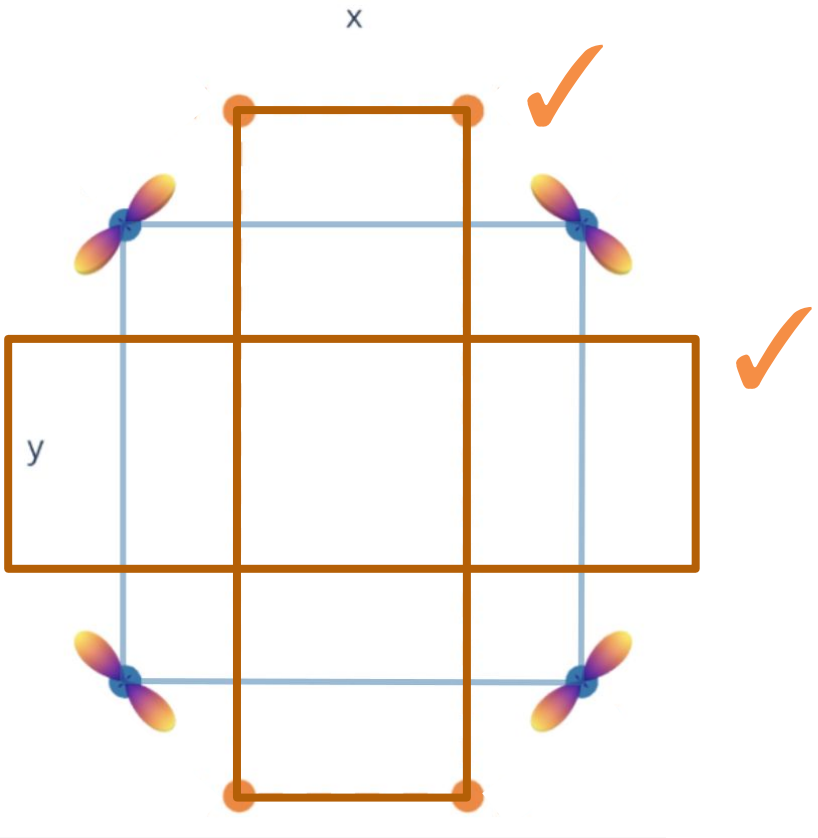


One open question:
Dealing with correlated outputs

Instead of order parameters, what if we just make our outputs more useful, e.g. sampleable?

This requires *higher order correlations*.

**For per atom predictions,
we trace over these correlations.**

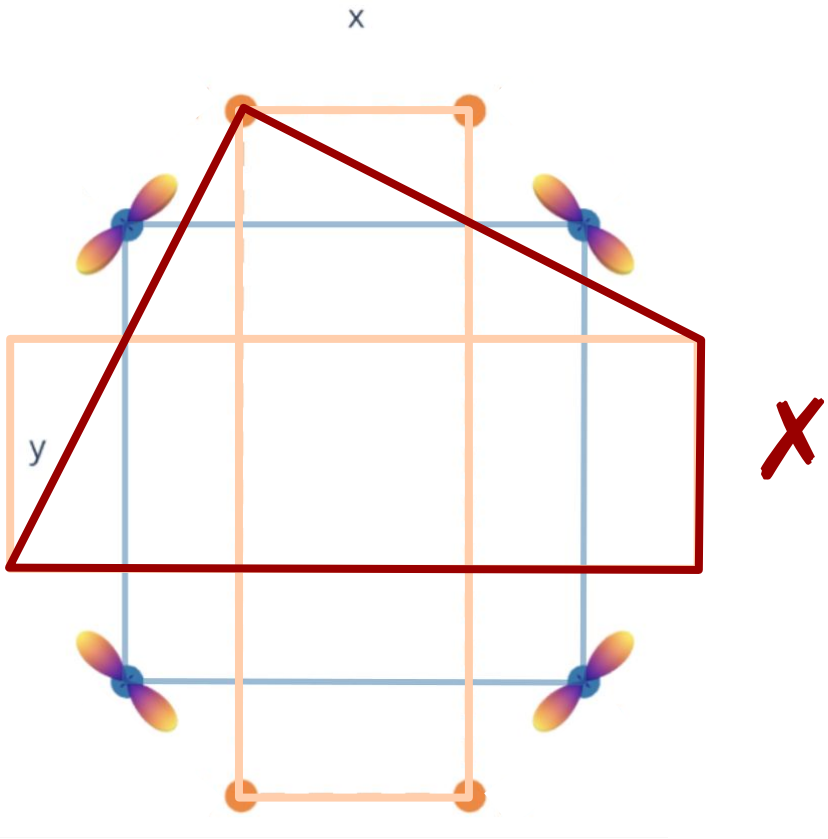


One open question:
Dealing with correlated outputs

Instead of order parameters, what if we just make our outputs more useful, e.g. sampleable?

This requires *higher order correlations*.

**For per atom predictions,
we trace over these correlations.**

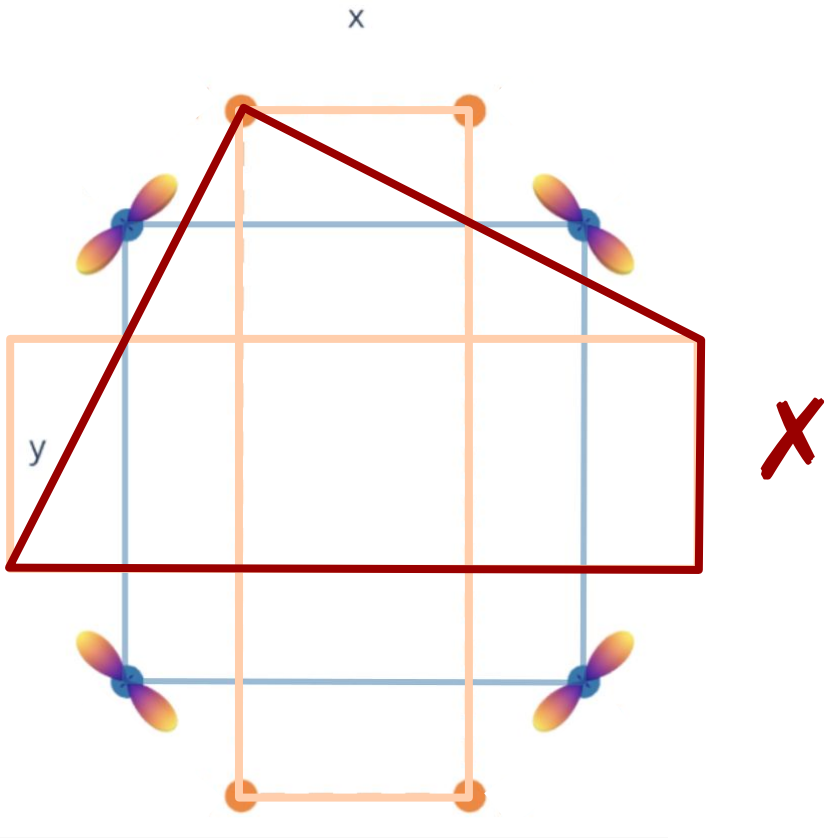


One open question:
Dealing with correlated outputs

Instead of order parameters, what if we just make our outputs more useful, e.g. sampleable?

This requires *higher order correlations*.

**For per atom predictions,
we trace over these correlations.**



Application: Generative models / design tools for physical systems

- **Lay down patterns, not just single points at a time**
- **Learn hierarchical representations of physical systems**