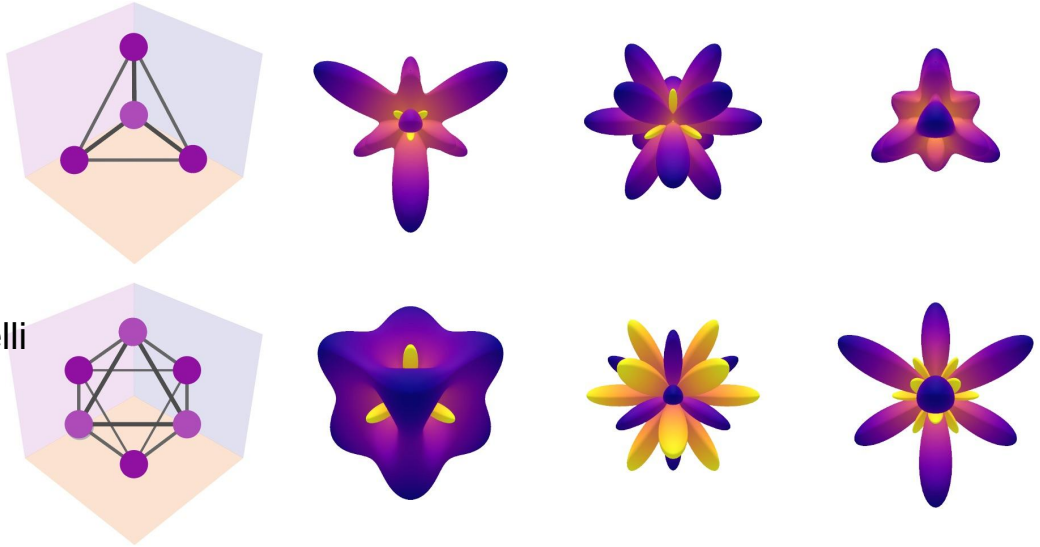
Symmetry's made to be broken:

Learning how to break symmetry with symmetry-preserving neural networks

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

First: Symmetry is useful

- **Motivations for symmetry-equivariant ML**
- **Recent applications and findings**

Second: Symmetry gives natural bases for expressing features

- **Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.**
  - **Irreducible representations and how to find them.**

Third: Emergent behavior of equivariant nns

- **Consequences of being symmetry-preserving**
- **Properties of E(3)NNs that have yet to be fully utilized.**
  - **Symmetry can tell you when you're missing data.**

First: Symmetry is useful

- **Motivations for symmetry-equivariant ML**
- **Recent applications and findings**

Second: Symmetry gives natural bases for expressing features

- **Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.**
  - **Irreducible representations and how to find them.**

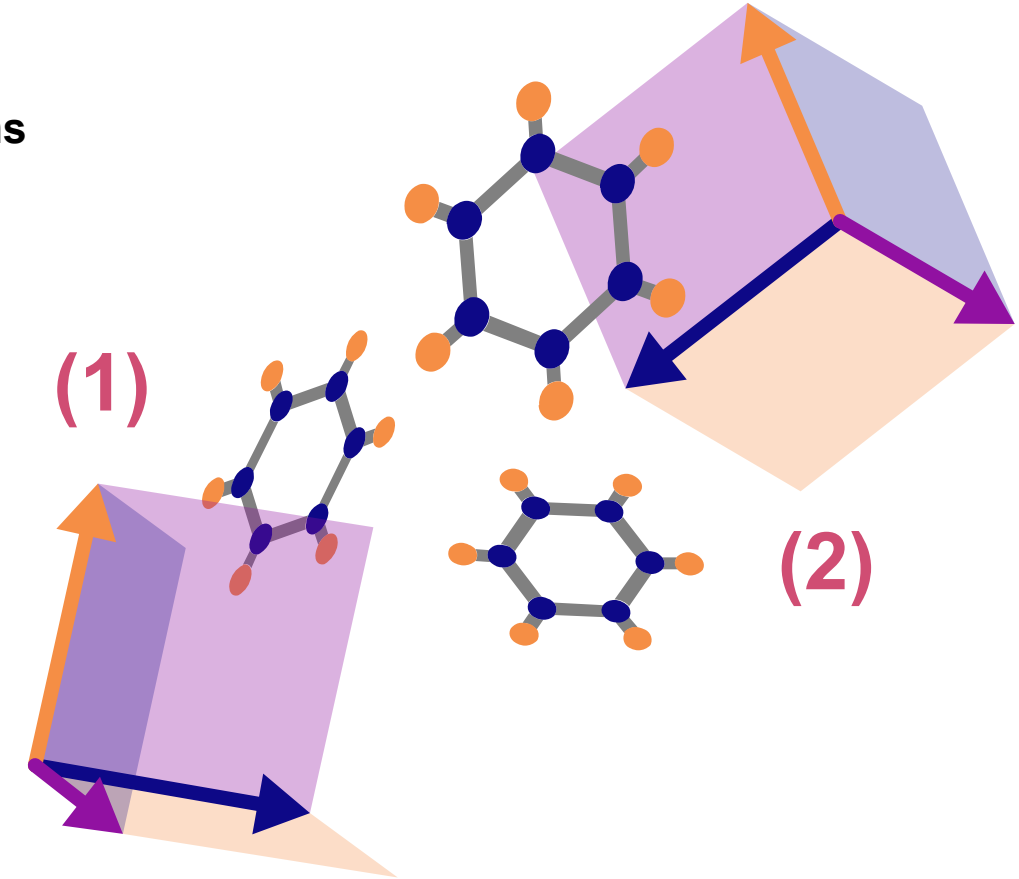
Third: Emergent behavior of equivariant nns

- **Consequences of being symmetry-preserving**
- **Properties of E(3)NNs that have yet to be fully utilized.**
  - **Symmetry can tell you when you're missing data.**

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)

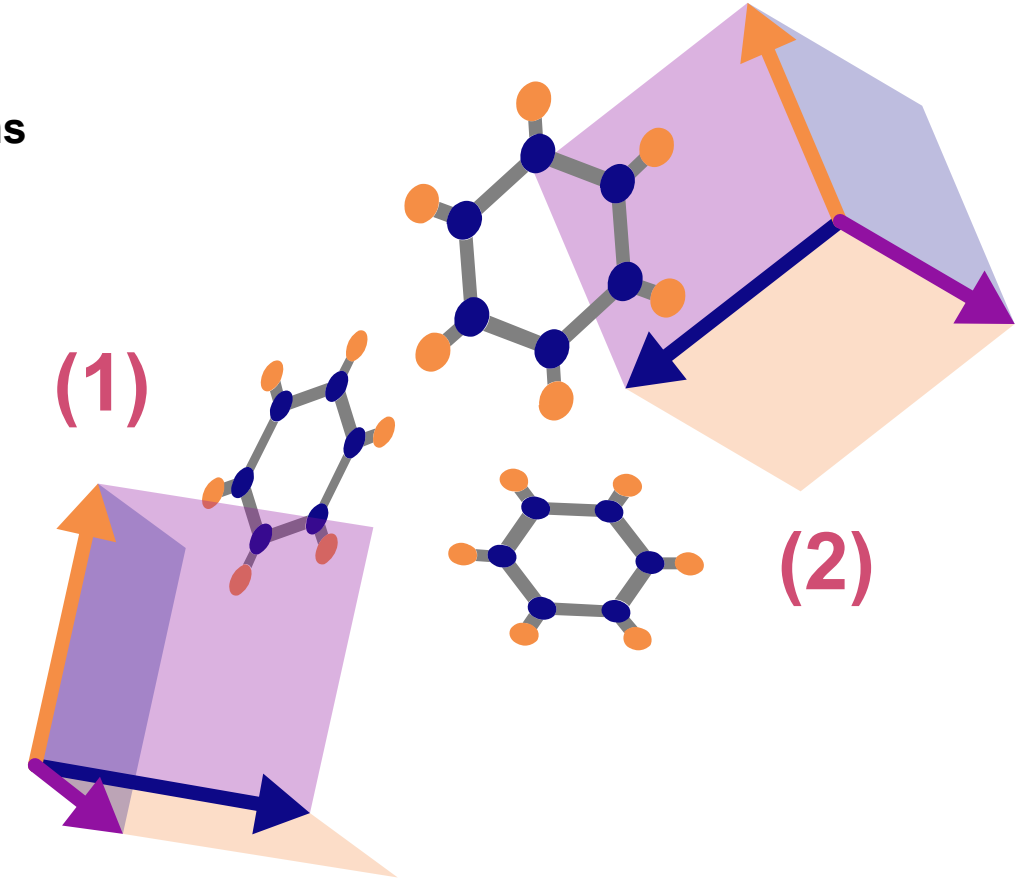


To describe physical systems we use coordinate systems

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

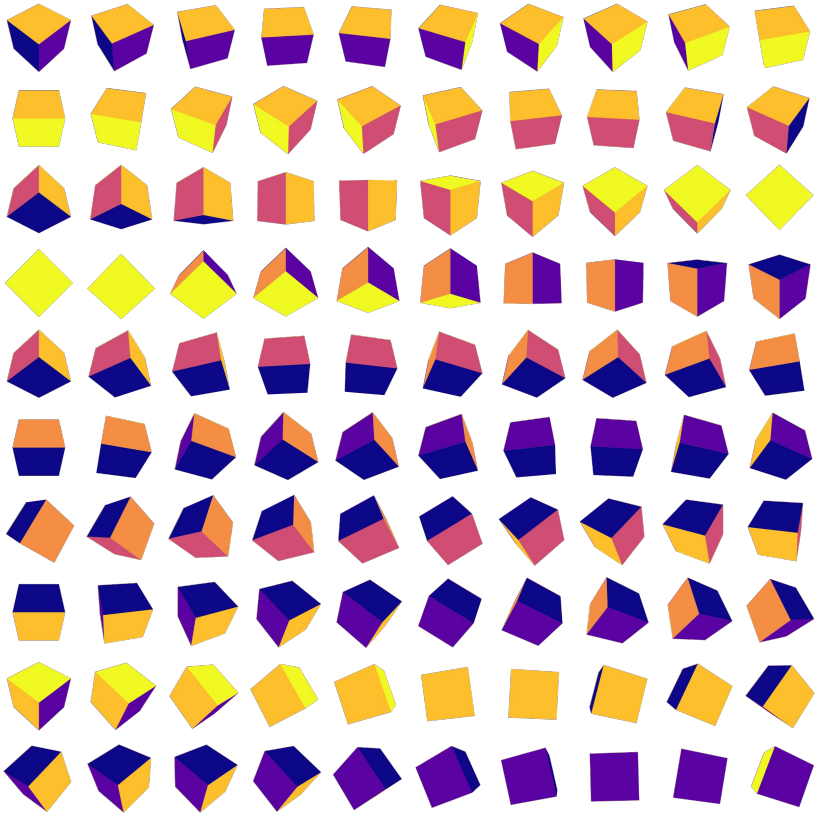
We can transform between coordinate systems using the symmetries of Euclidean space (3D rotations, translations, and inversion)

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



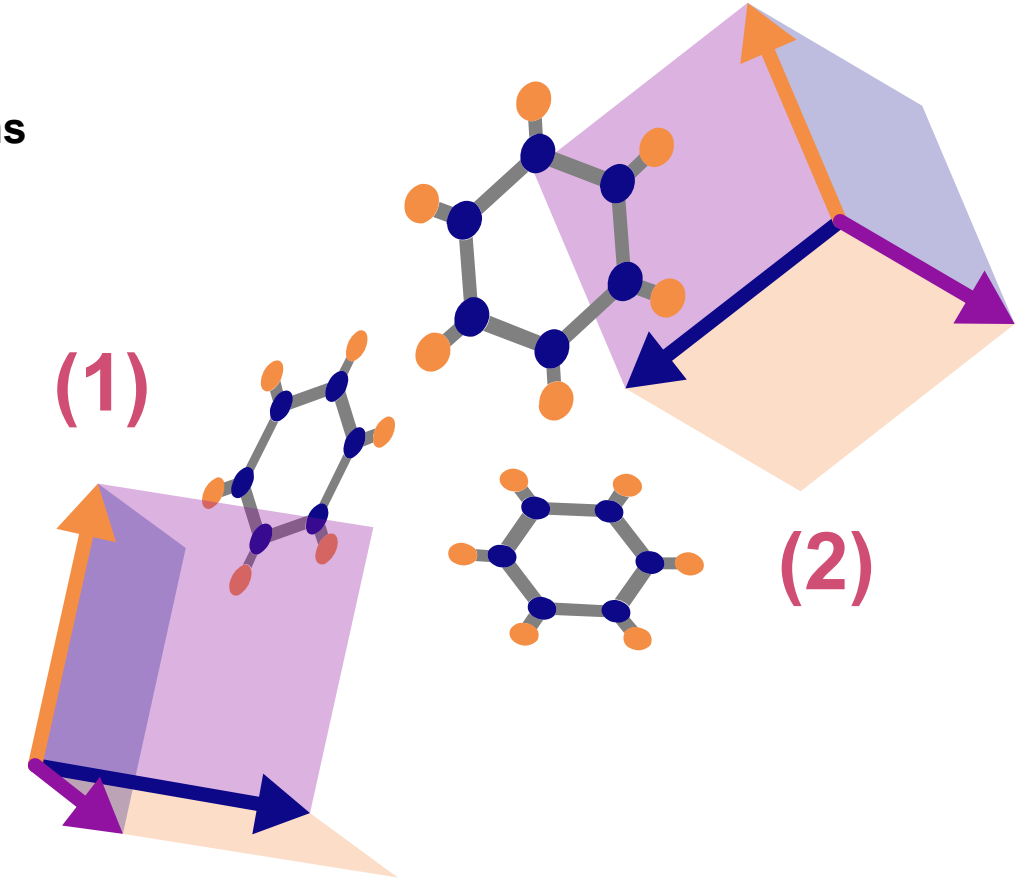
training with symmetry

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

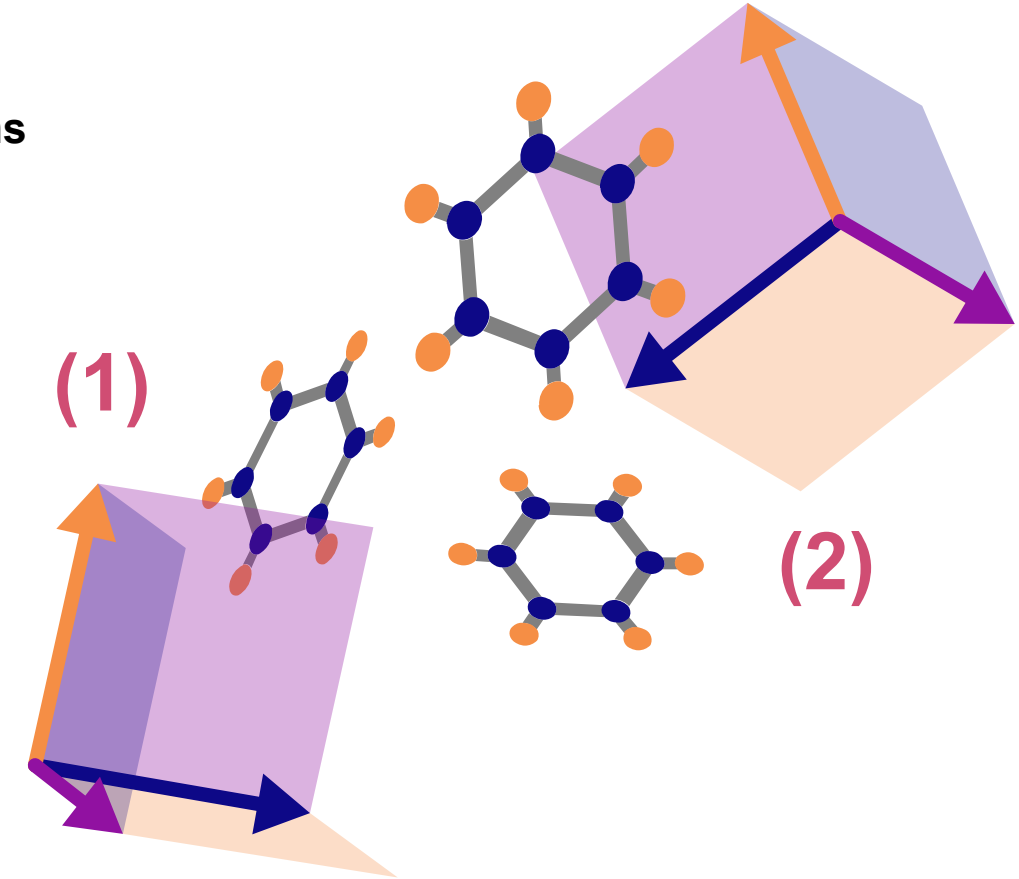
(1) and (2) use different coordinate systems to describe the same physical system.

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

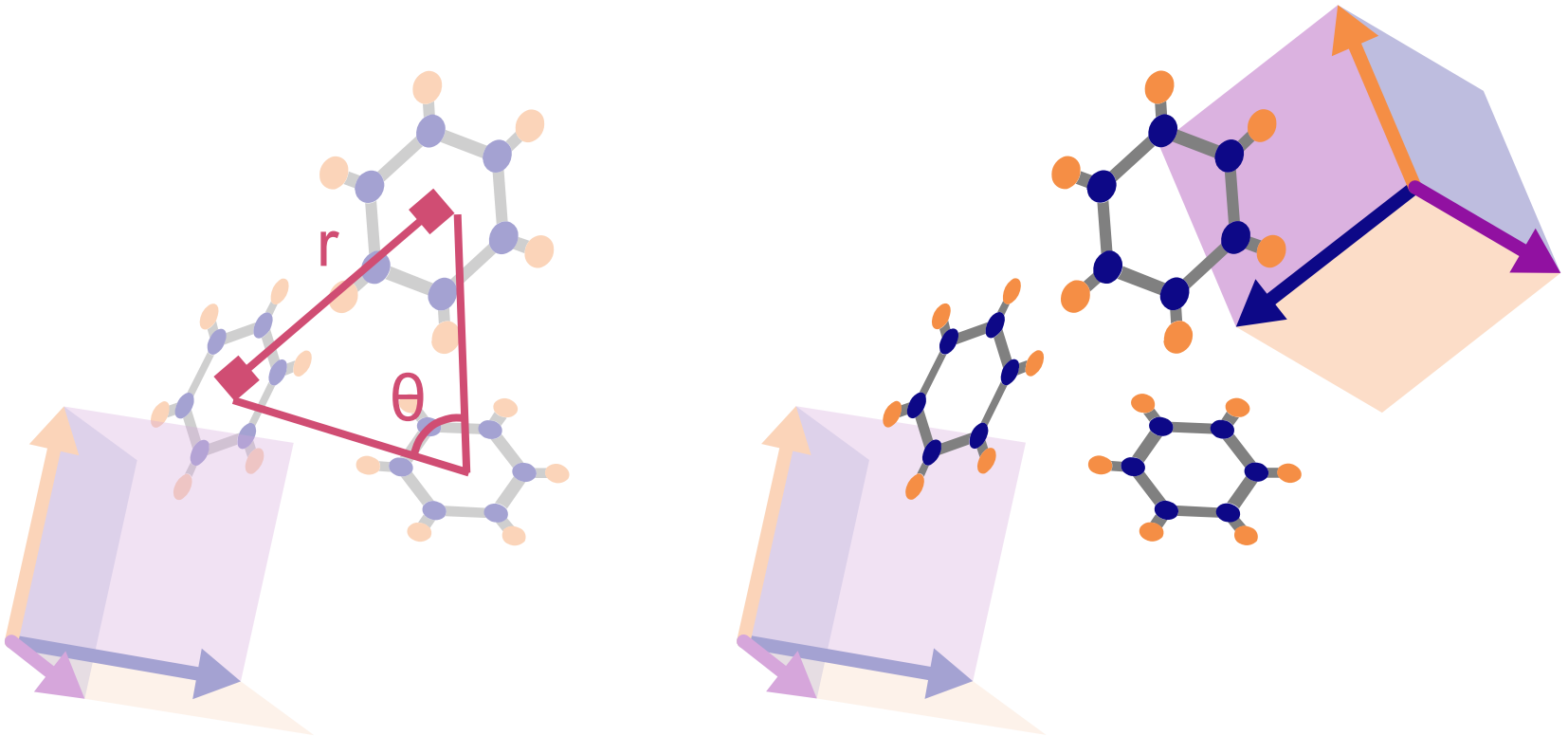
Traditional machine learning see (1) and (2) as completely different!

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

...so want machine learning with symmetry!

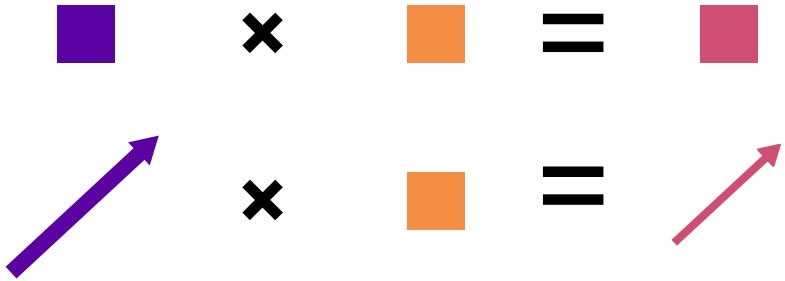


Invariant models pre-compute invariant features and throw away the coordinate system.
Equivariant models keep the coordinate system
AND if the coordinate system changes, the outputs change accordingly.

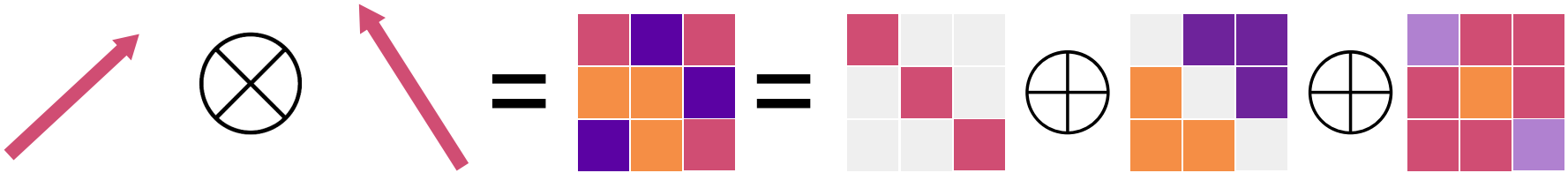


Interactions in equivariant models are more complex than invariant models *but also more expressive…*

How do we interact invariant objects? Scalar multiplication.



How do we interact equivariant objects? Geometric tensor products!



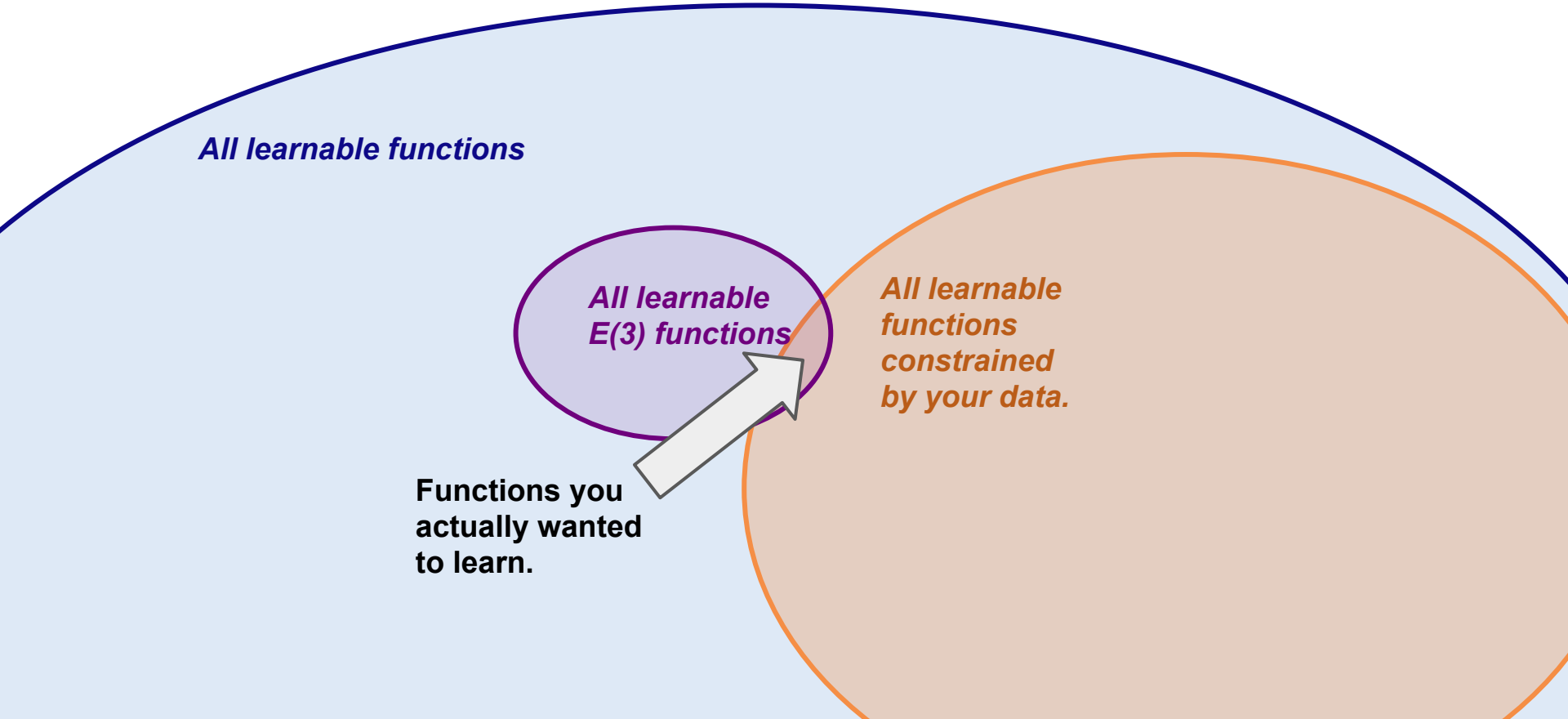
dot product
trace
invariant
L=0
1 degree of freedom

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

symmetric
traceless
equivariant
L=2
5 degrees of freedom

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

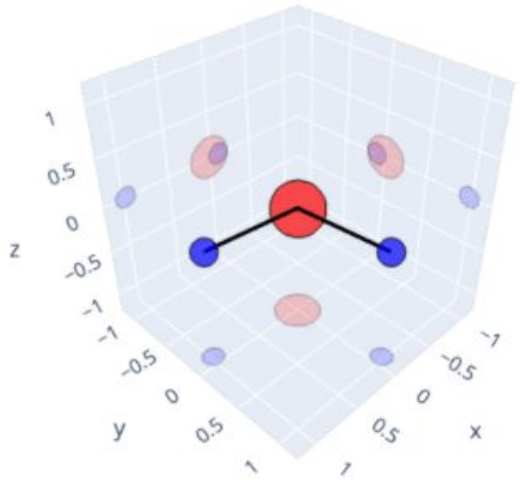
Why limit yourself to functions with (Euclidean) symmetry?
You can *substantially* shrink the space of functions you need to optimize over.
This means you need less data to constrain your function.



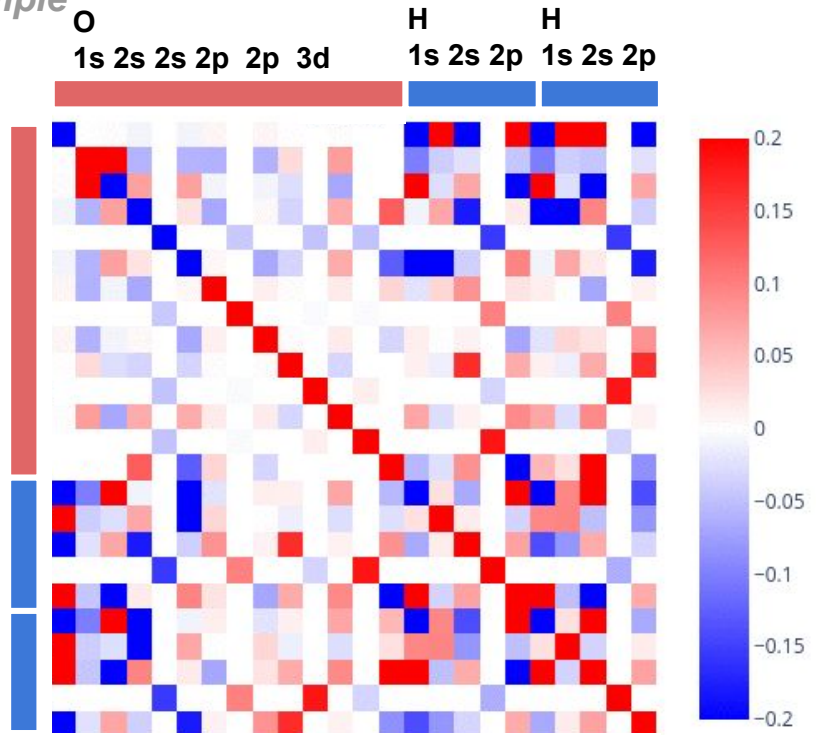
Euclidean symmetry equivariant methods have Euclidean symmetry "built-in".
These methods understand that a physical system described by e.g. two different coordinate systems still "means" the same thing *even without training*.

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

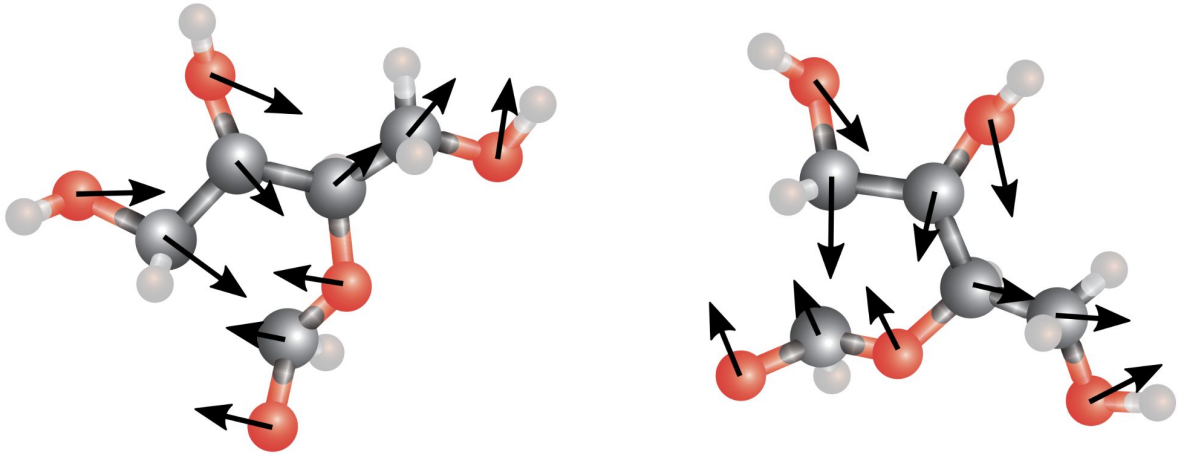
$$R\vec{v}$$



$$RHR^T$$



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.



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

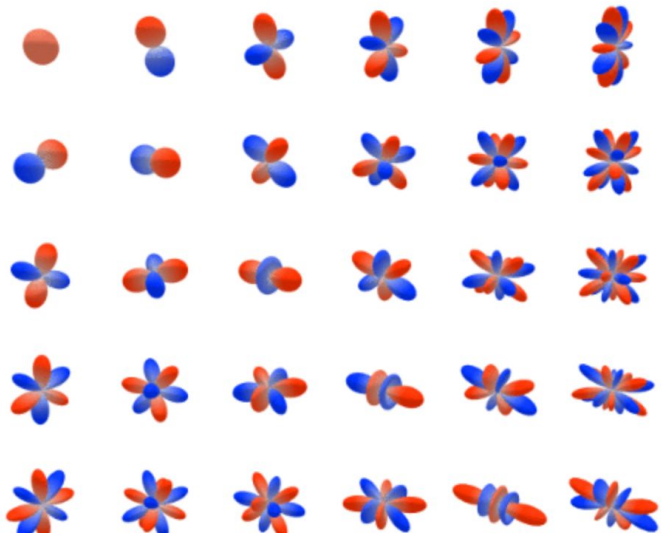
View My GitHub Profile

Hosted on GitHub Pages — Theme by orderedlist

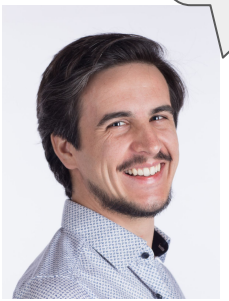
Welcome to e3nn!

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.

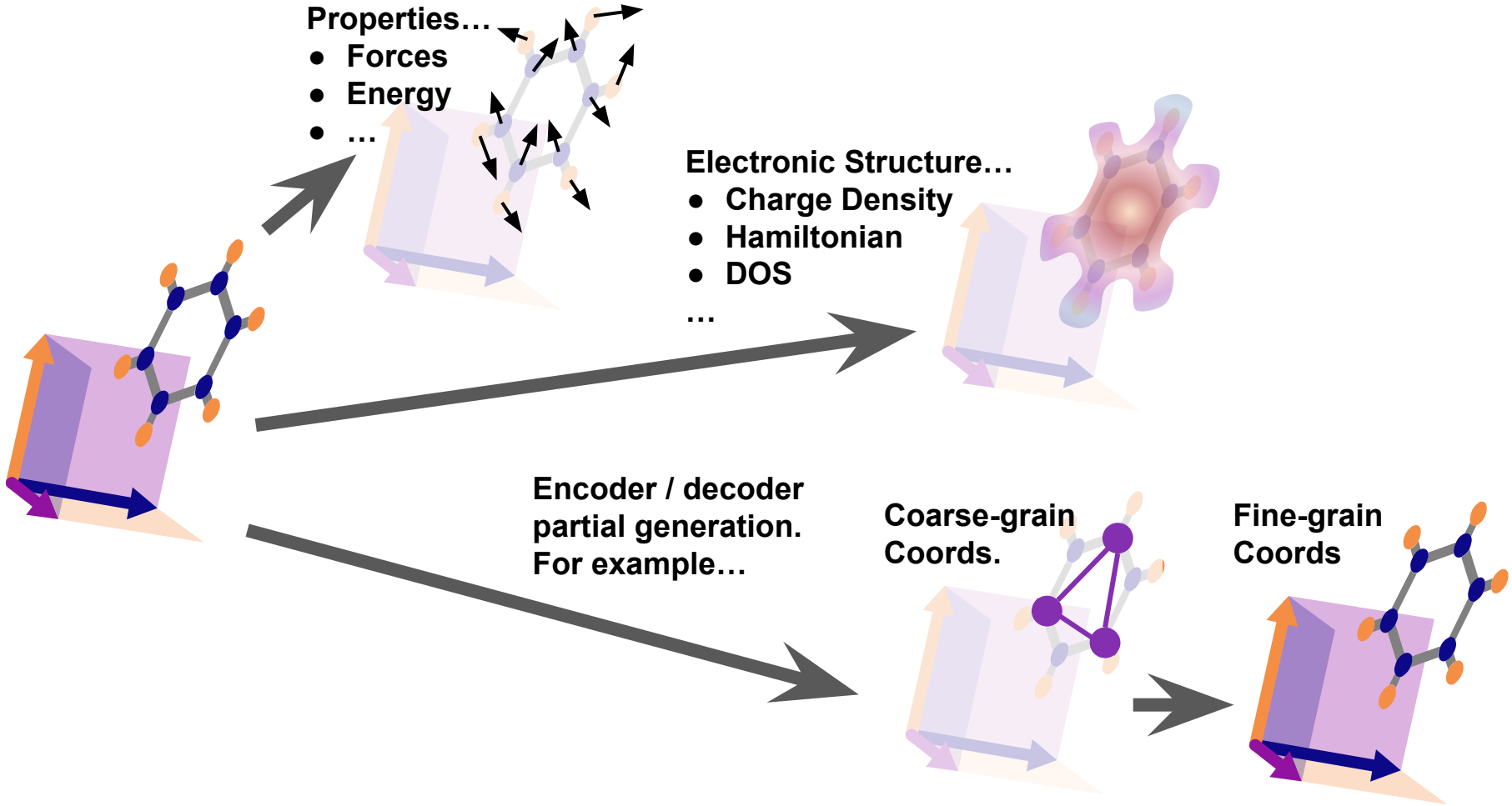


Also e3nn-jax!



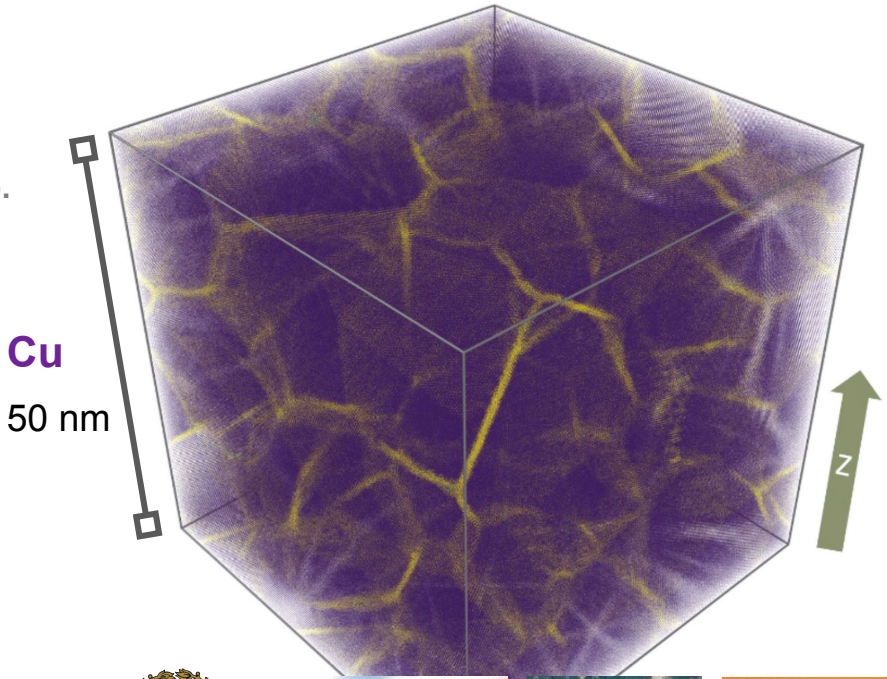
Mario Geiger

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.

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).

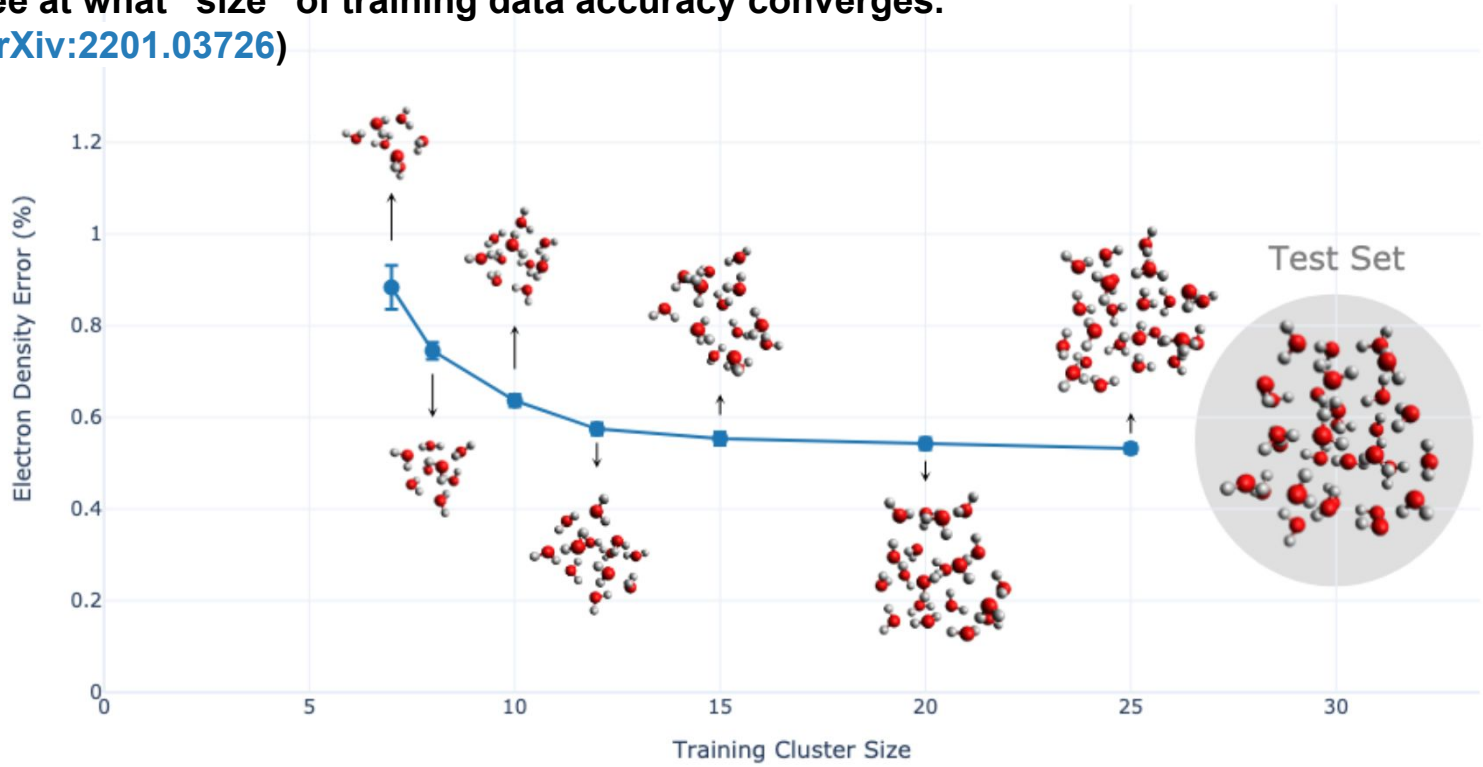
Open source codes
Allegro: https://github.com/mir-group/allegro
NequIP: https://github.com/mir-group/nequip
e3nn: https://github.com/e3nn/e3nn/

Boris Kozinsky
Simon Batzner
Alby Musaelian



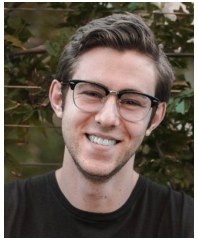
We've used E(3)NNs to build data-efficient and scalable models of physical processes.
…and estimate the "nearsightedness" of water.

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)



Sandia National Laboratories

Josh Rackers

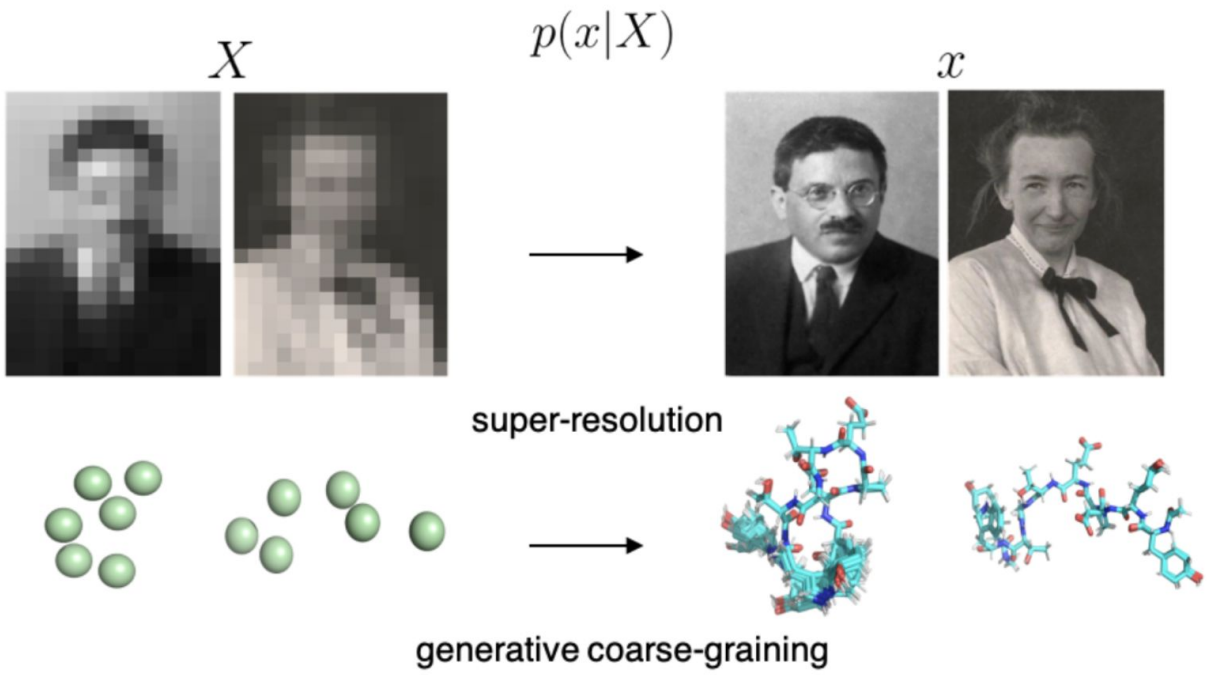


Lucas Tecot

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

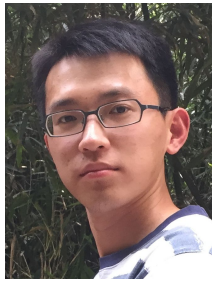
…and generate fine-grain molecular conformations from coarse-grained molecules

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 approaching state-of-art for OC20 and with shorter training times.

Atomic Architects MIT	Equiformer (IS2RE training data only with IS2RS and Noisy Nodes. Direct.)

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

Submitted model is on substantially less data than leading models and is trained for ~10x less epochs.

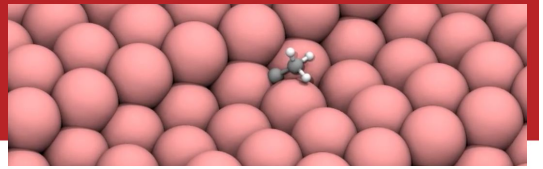
Yi-Lun Liao

FACEBOOK AI Carnegie Mellon University

Home Leaderboard Datasets Challenge Discuss

Public Leaderboard

See the latest results and submit your own on the evaluation server!



Task IS2RE | Task S2EF | Task IS2RS

Test Split

Average | ID | OOD Ads | OOD Cat | OOD Both

			AVERAGE		
	TEAM	METHOD	EWT (%)	ENERGY MAE (EV)	SUBMITTED
1.	FAIR + NERSC	GemNet-XL	11.13	0.3712	2021/09/27
2.	TUM + FAIR	GemNet-T-EFwT-Relaxation-All	9.86	0.3997	2021/08/23
3.	FAIR + CMU	spinconv-force-centric-relaxation-all	7.9	0.4343	2021/06/03
9.	Atomic Architects MIT	Equiformer (IS2RE training data only with IS2RS and Noisy Nodes. Direct.)	5.66	0.466	2022/05/20
10.	Machine Learning	3d-Graphormer (Direct, IS2RE data only)	6.1	0.4722	2021/10/06
11.	Deep Mind	GNS + Noisy Nodes (IS2RE Only)	6.5	0.4728	2022/02/15

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

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

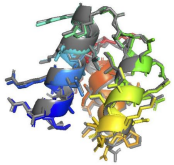
Equiformer + Spherical Channel Network takes 2nd place in OC20 2022 Challenge!

C. L. Zitnick, A Das, … B. Wood (Meta)

SCN: https://arxiv.org/abs/2206.14331

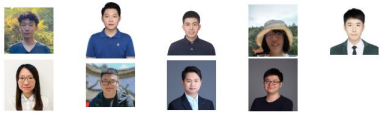
EquiFold by Prescient Design and friends Applies a Equiformer inspired architecture to protein folding.

Comparable accuracy to AlphaFold without "multi-sequence alignment" and much faster
(e.g. some task EF 1 sec vs. AF 1 hour).
bioRxiv:10.1101/2022.10.07.511322
Equifold code



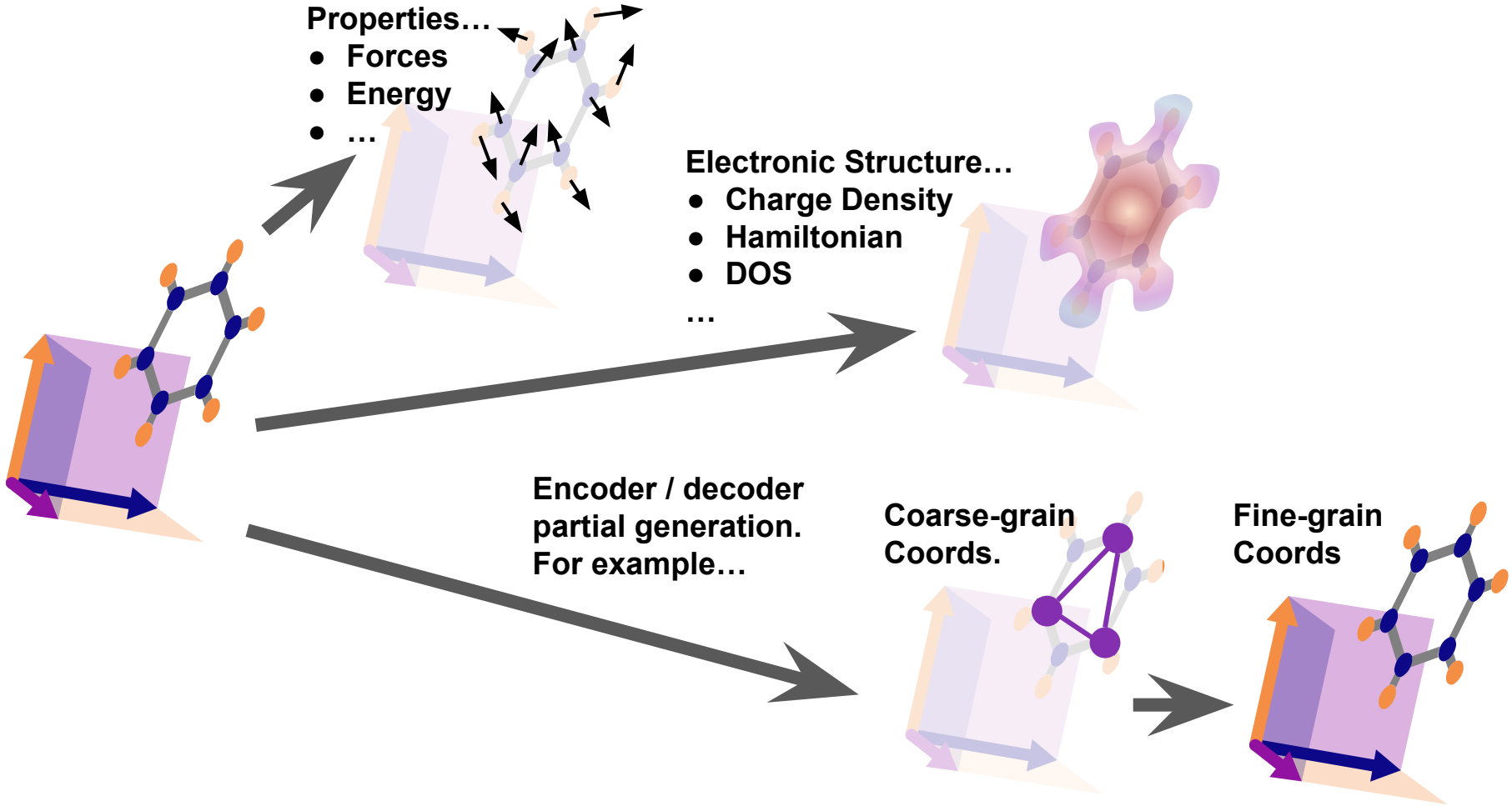
Yi-Lun Liao

Schedule

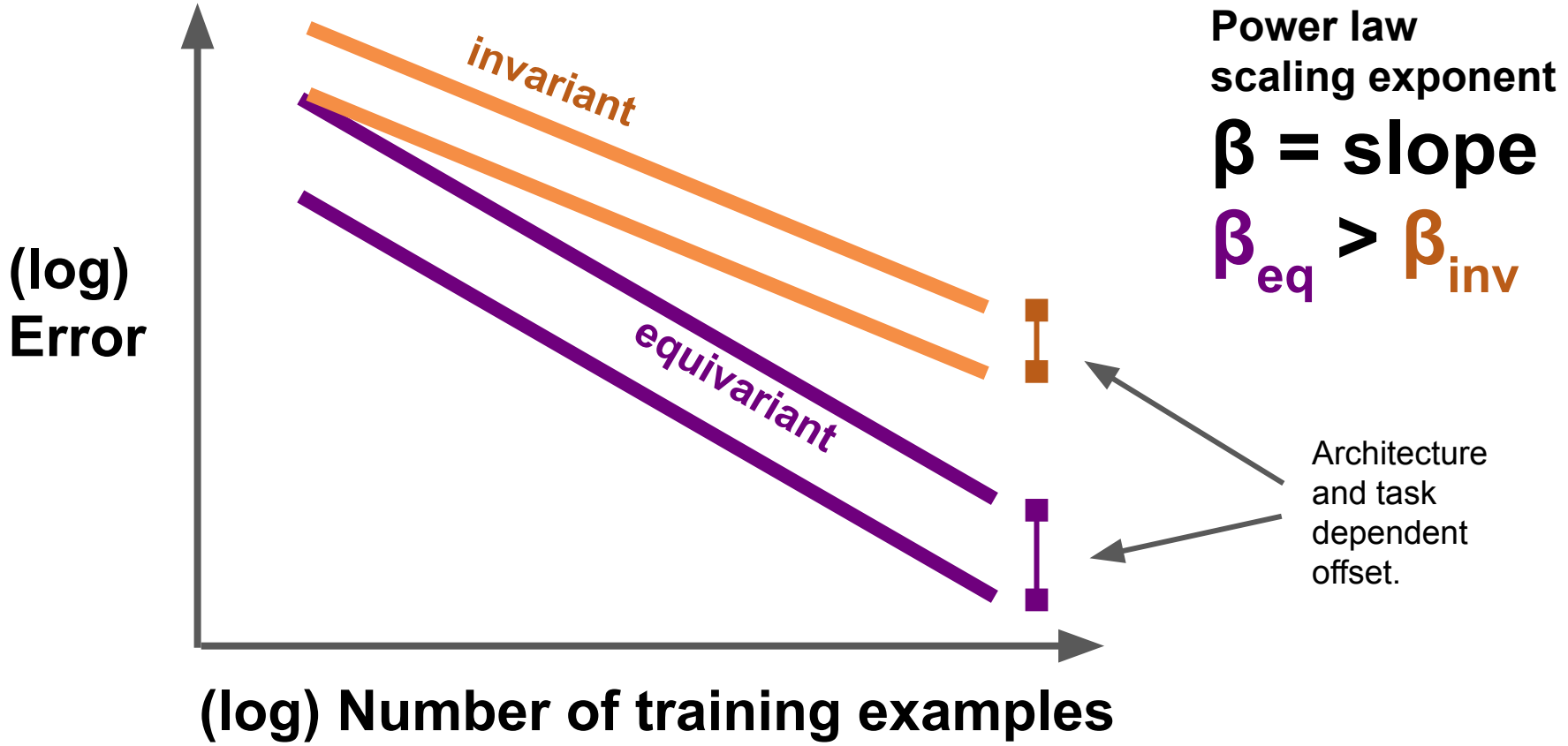
15:00 - 15:05 CST	Buffer for attendees to join Zoom [room link]	-
15:05 - 15:20 CST	Challenge overview, results and analysis	Abhishek Das (Open Catalyst Project team) 
15:20 - 15:50 CST	Invited talk + Q&A	Gabor Csanyi 
15:50 - 16:10 CST	Runner-up talk + Q&A	Yi-Lun Liao, Tess Smidt (Atomic Architects, MIT)  
16:10 - 16:30 CST	Winner talk + Q&A	Jiaqi Han, Tian Bian, Geyan Ye, Kaili Ma, Yuduo Zhi, Kangfei Zhao, Tingyang Xu, Wenbing Huang, Yu Rong (Tencent AI Lab, Tsinghua University, Renmin University of China, The Chinese University of Hong Kong) 
16:30 - 17:00 CST	Invited talk + Q&A	Yoshua Bengio 

Equiformer + SCN

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.

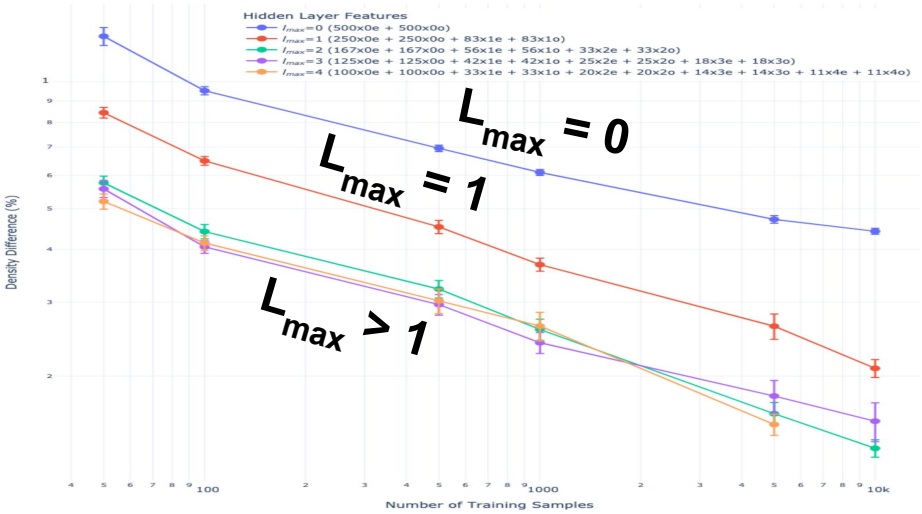


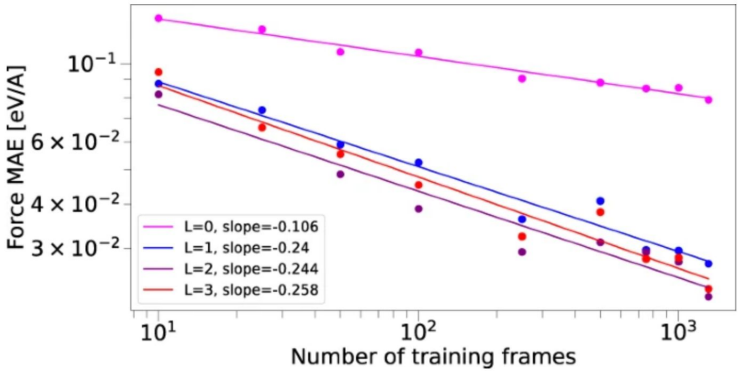
Table A.1 Power laws for neural force field scaling.

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.

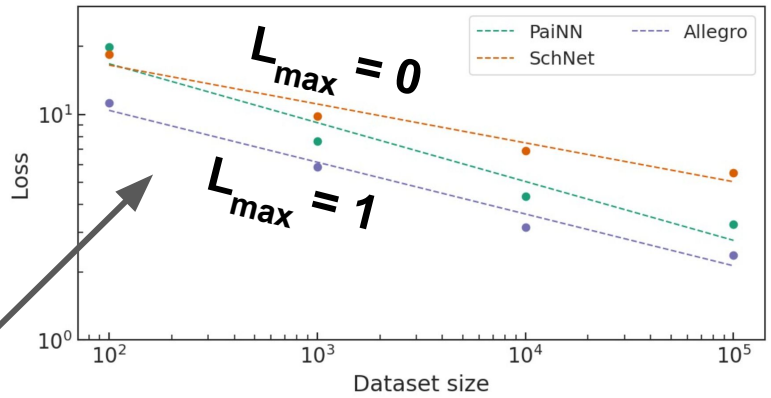


Fig. A.5 Calculating neural scaling power laws for neural force fields. Test loss versus dataset size for PaiNN, Allegro, and SchNet models with fixed capacity, 64.

First: Symmetry is useful

- **Motivations for symmetry-equivariant ML**
- **Recent applications and findings**

Second: Symmetry gives natural bases for expressing features

- **Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.**
  - **Irreducible representations and how to find them.**

Third: Emergent behavior of equivariant nns

- **Consequences of being symmetry-preserving**
- **Properties of E(3)NNs that have yet to be fully utilized.**
  - **Symmetry can tell you when you're missing data.**

First: Symmetry is useful

- Motivations for symmetry-equivariant ML
- Recent applications and findings

Second: Symmetry gives natural bases for expressing features

- **Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.**
  - **Irreducible representations and how to find them.**

Third: Emergent behavior of equivariant nns

- Consequences of being symmetry-preserving
- Properties of E(3)NNs that have yet to be fully utilized.
  - Symmetry can tell you when you're missing data.

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.

g is an element of Euclidean symmetry

$$f(D(g)x, w) = D(g)f(x, w)$$

D(g) is how we "represent" g acting on x (or f(x)).

The form of D(g) depends on what it's acting on! (e.g. x vs. f(x))

Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$

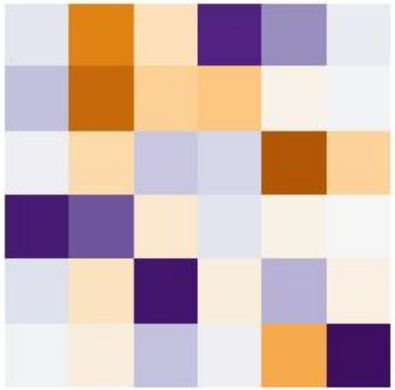
Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$



$$\begin{pmatrix} x^2 \\ xy \\ xz \\ y^2 \\ yz \\ z^2 \end{pmatrix}$$

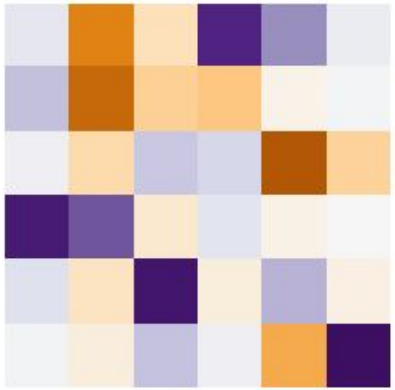
Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$



$$\begin{pmatrix} x^2 \\ xy \\ xz \\ y^2 \\ yz \\ z^2 \end{pmatrix}$$

$$SQ = QR$$

R is scalar or vector representation.

Solve to find whether Q exists to map parts of new vector space to *known* representations (scalar and vector).

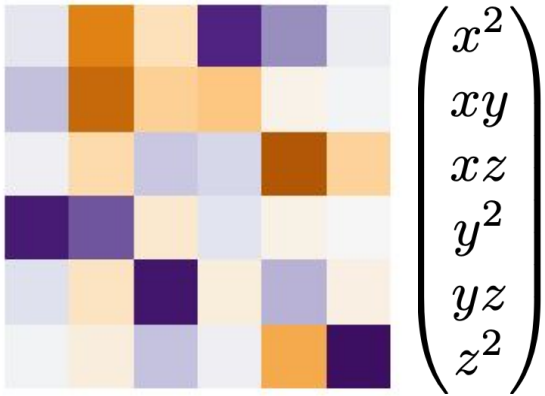
Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

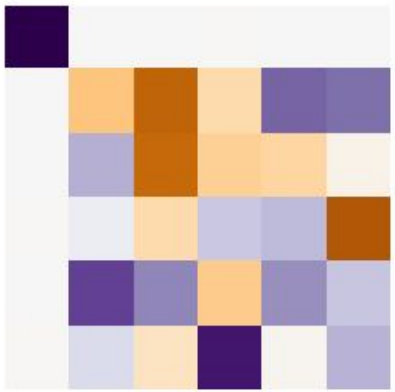
Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$



$$\begin{pmatrix} x^2 \\ xy \\ xz \\ y^2 \\ yz \\ z^2 \end{pmatrix}$$



$$\begin{pmatrix} x^2+y^2+z^2 \\ 2x^2-y^2-z^2 \\ xy \\ xz \\ y^2-z^2 \\ yz \end{pmatrix}$$

Groups can act on many types of mathematical objects.

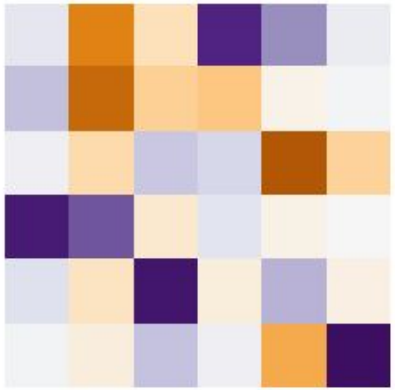
For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

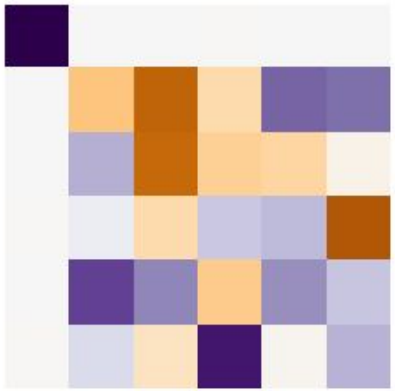
$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$

reducible



$$\begin{pmatrix} x^2 \\ xy \\ xz \\ y^2 \\ yz \\ z^2 \end{pmatrix}$$

irreducible



$$\begin{pmatrix} x^2+y^2+z^2 \\ 2x^2-y^2-z^2 \\ xy \\ xz \\ y^2-z^2 \\ yz \end{pmatrix}$$

Wash. Rinse. Repeat.

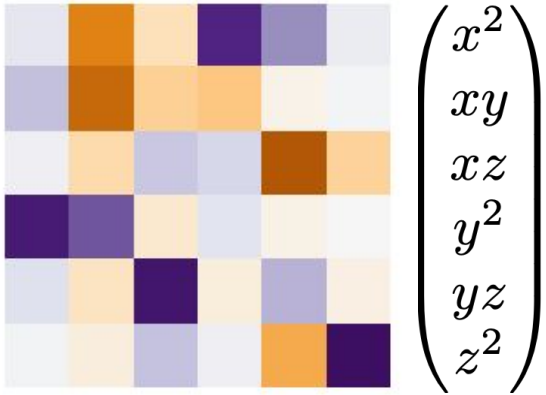
Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

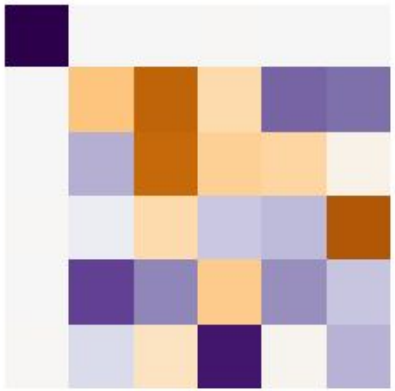
Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$



$$\begin{pmatrix} x^2 \\ xy \\ xz \\ y^2 \\ yz \\ z^2 \end{pmatrix}$$



$$\begin{pmatrix} x^2+y^2+z^2 \\ 2x^2-y^2-z^2 \\ xy \\ xz \\ y^2-z^2 \\ yz \end{pmatrix}$$

L=2 spherical harmonics

Use to derive spherical harmonics!

Groups can act on many types of mathematical objects.

For 3D rotations, the most familiar ones are scalars (no change) and the 3D vector.

Can create more complex objects by tensor producting known vector spaces.

By knowing how rotations act on the smaller spaces, we can figure out how it acts on the larger one.

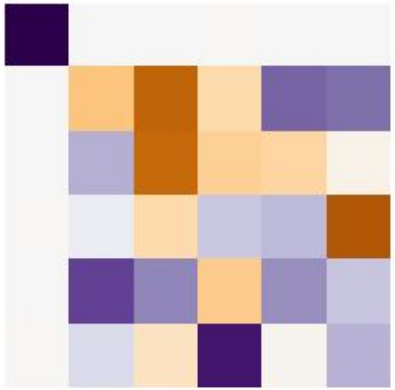
$$(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)$$

Change of basis

2 vector spaces of irreps

⇨ 1 vector space of irreps

"Clebsch-Gordan Coefficients"



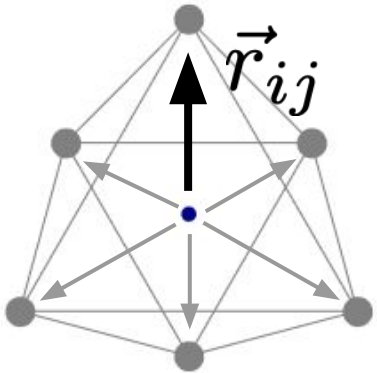
$$\begin{pmatrix} x^2+y^2+z^2 \\ 2x^2-y^2-z^2 \\ xy \\ xz \\ y^2-z^2 \\ yz \end{pmatrix}$$

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

Spherical harmonic projections

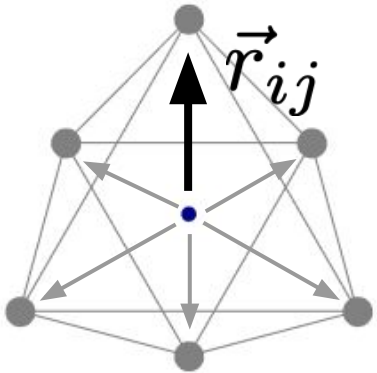
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections



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

Spherical harmonic projections

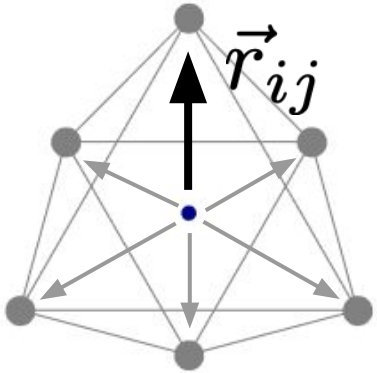


"Coefficients"

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

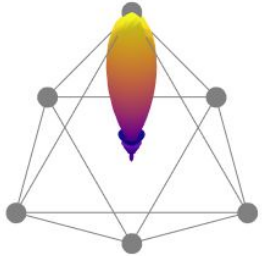
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections



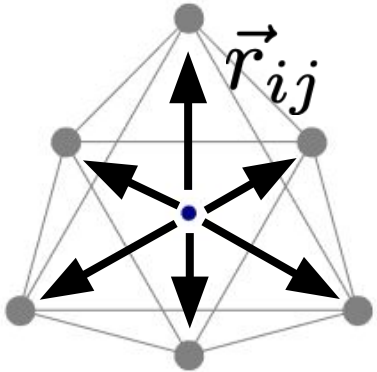
"Coefficients"

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



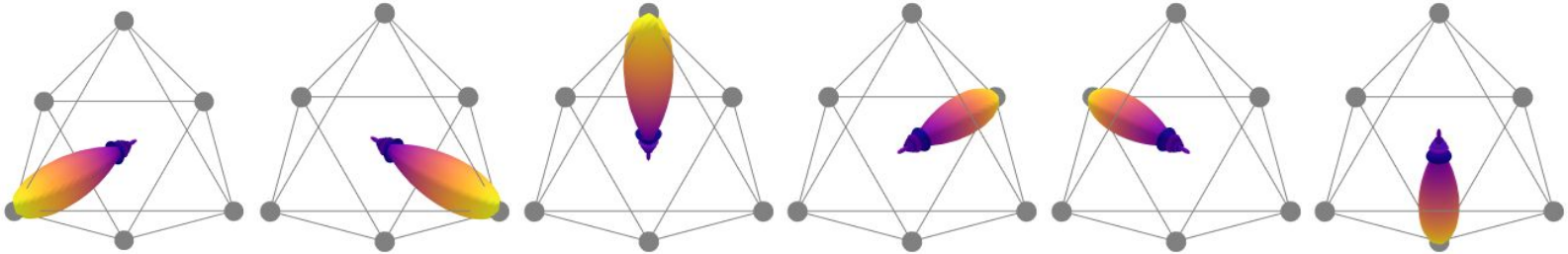
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections



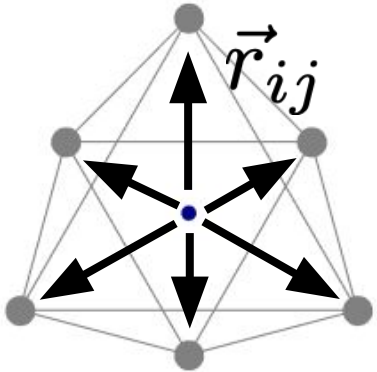
"Coefficients"

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



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

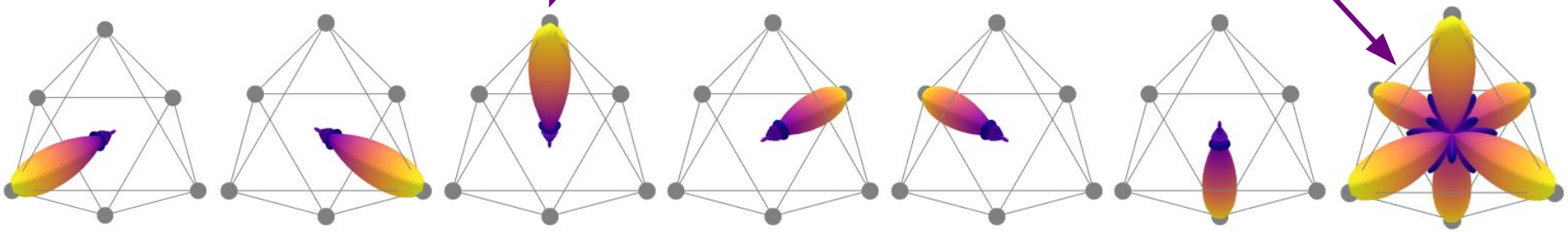
Spherical harmonic projections



"Coefficients"

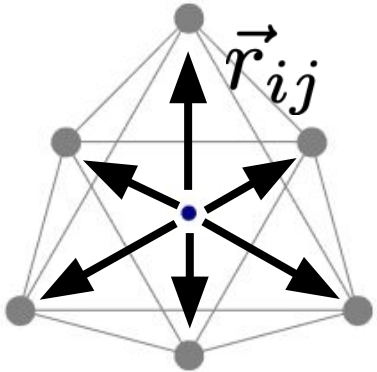
$$\sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$

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



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

Spherical harmonic projections

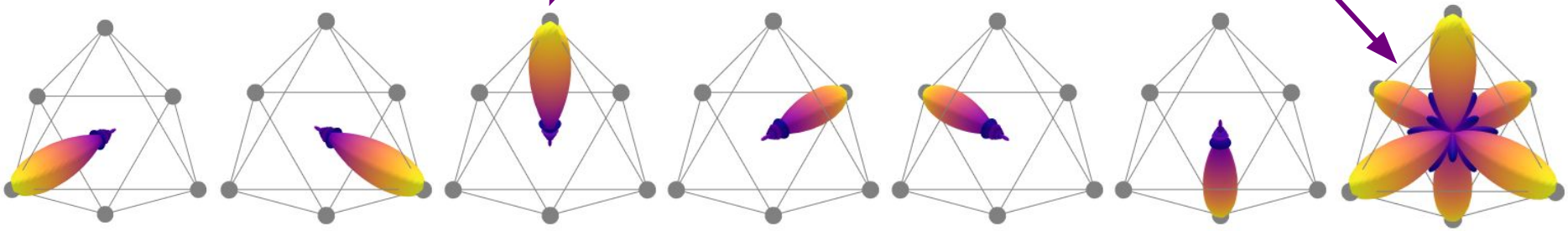


"Coefficients"

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

Why useful?
Variable peaks but a fixed length signal!
Can describe a distribution of vectors.

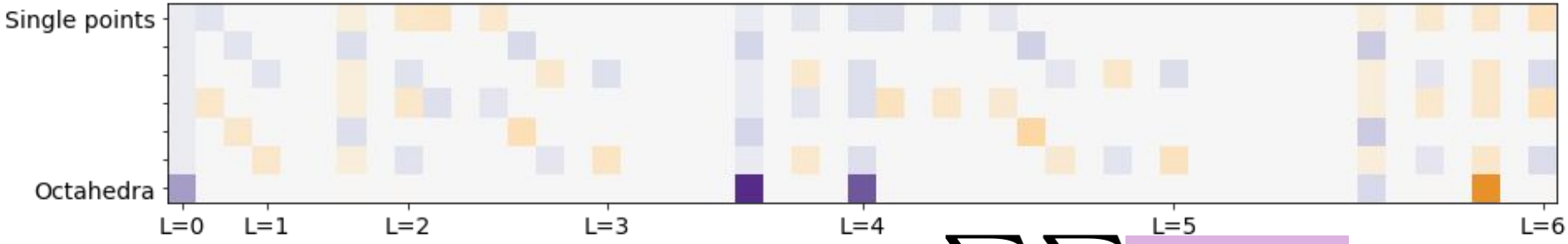
$$\sum_{j} \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$



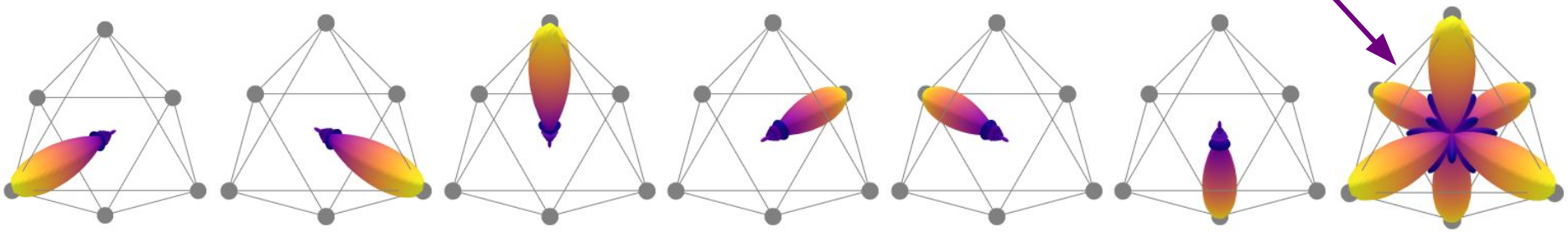
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections

Coefficients attached to each spherical harmonic



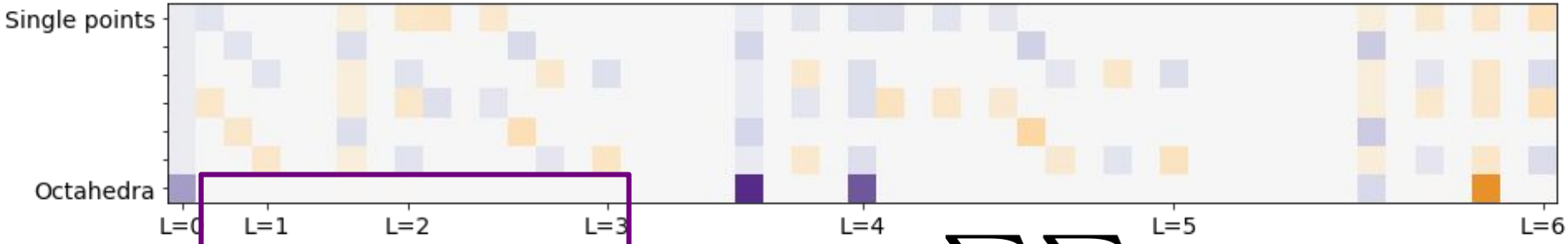
$$\sum_j \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})$$



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

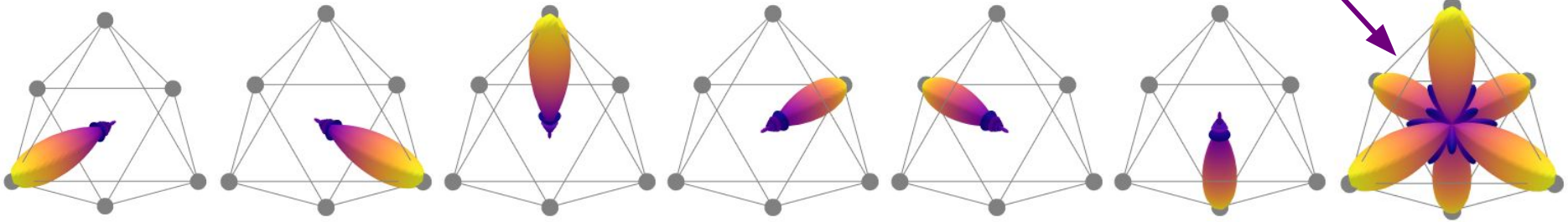
Spherical harmonic projections

Coefficients attached to each spherical harmonic



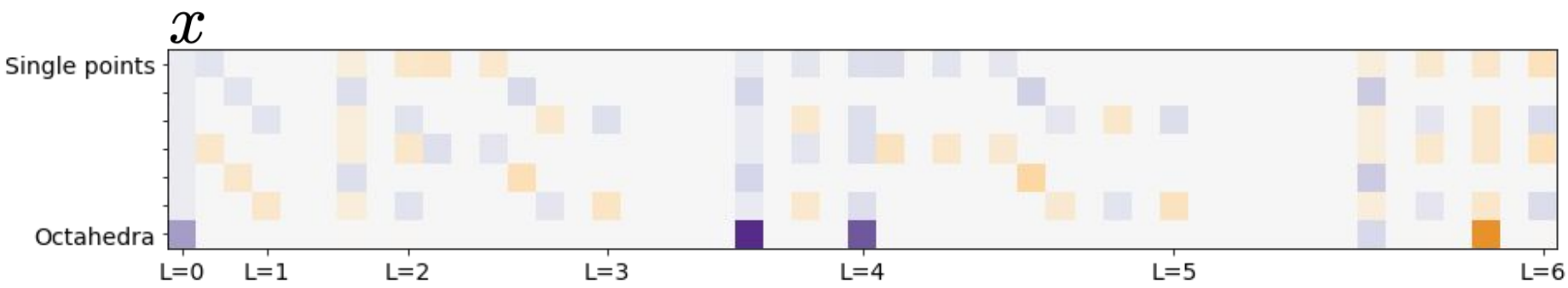
Typical signature of high symmetry objects: cancelation of terms.

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



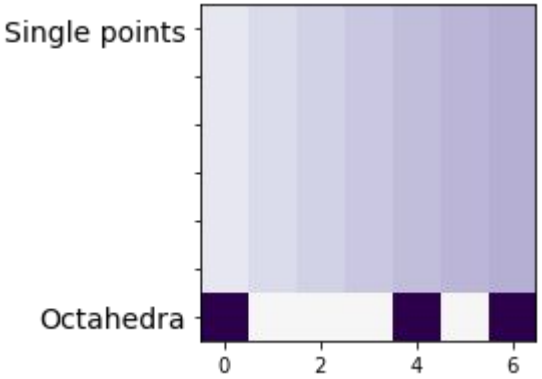
From spherical harmonic projections we can create invariants using tensor products.

Power spectra, bispectra, …



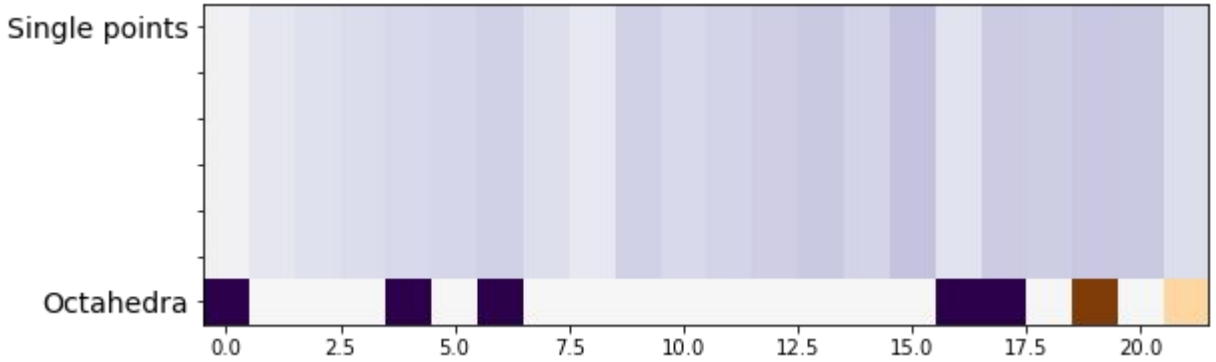
$x \otimes x \rightarrow$ **Scalars**

Power spectra



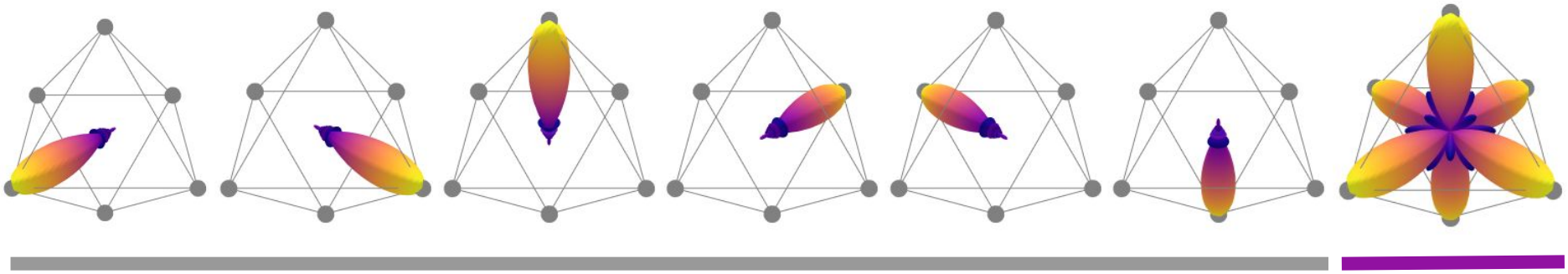
$x \otimes x \otimes x \rightarrow$ **Scalars**

Bispectra



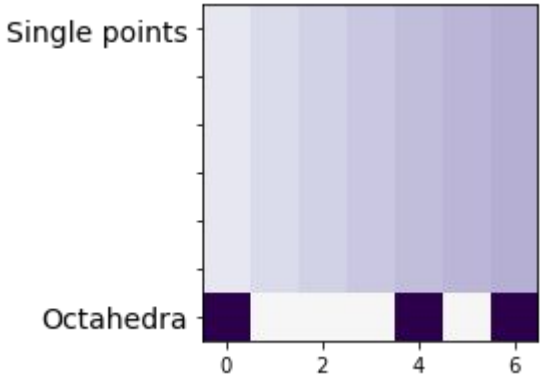
From spherical harmonic projections we can create invariants using tensor products.

Power spectra, bispectra, … but in equivariant nns we can use the equivariant data.



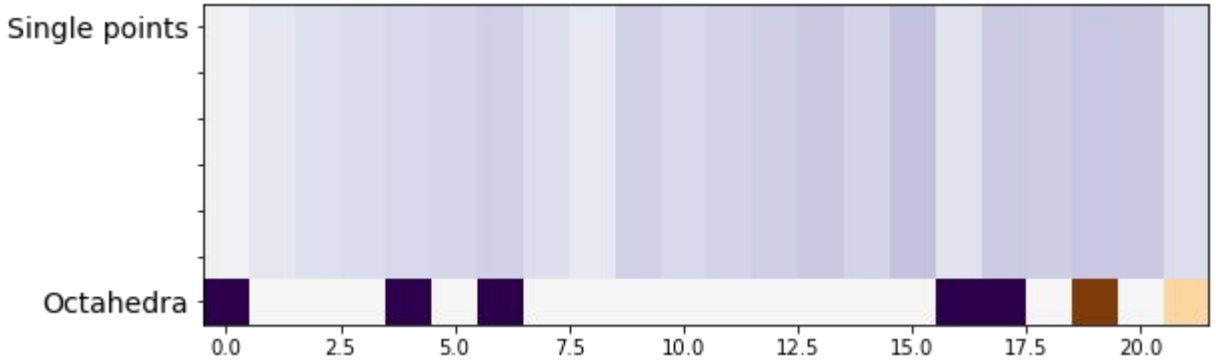
$x \otimes x \rightarrow$ **Scalars**

Power spectra

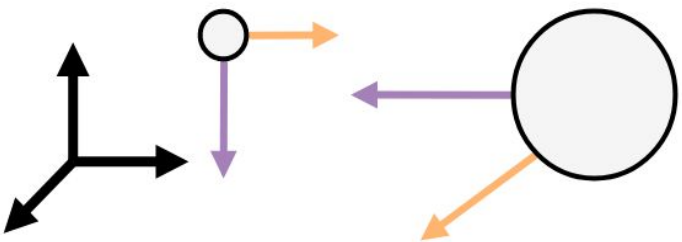


$x \otimes x \otimes x \rightarrow$ **Scalars**

Bispectra

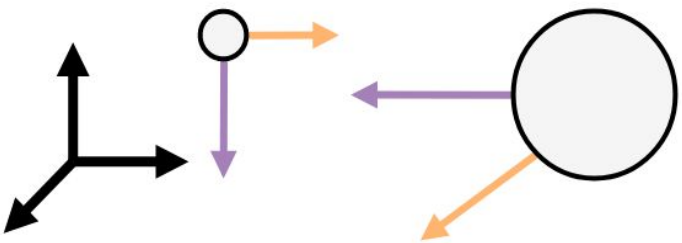


The input to our network is geometry and (geometric tensor) features on that geometry.



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

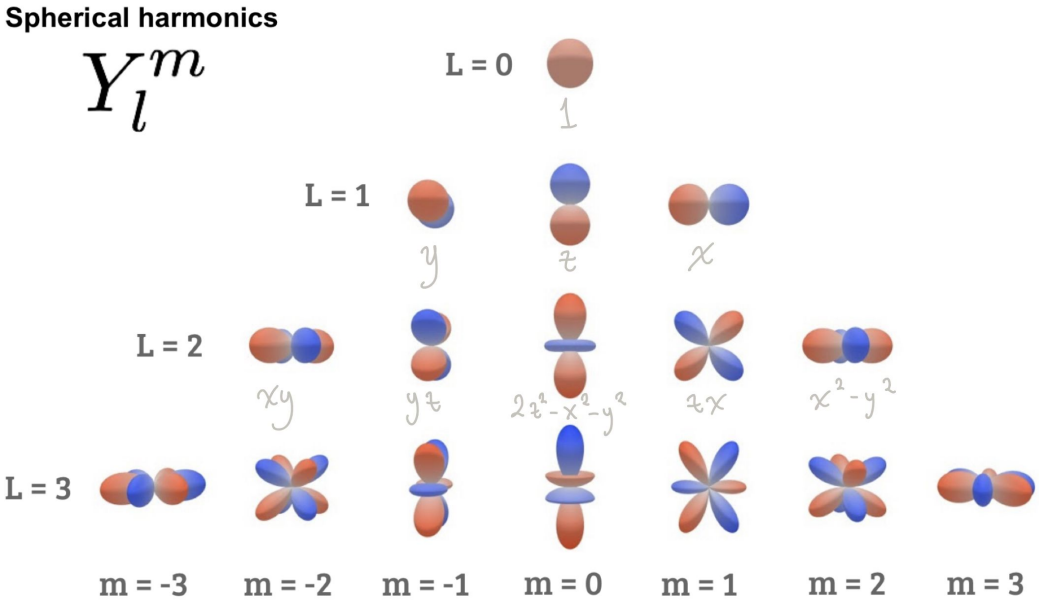
The input to our network is geometry and (geometric tensor) features on that geometry.
We categorize our features by how they transform under rotation and parity
as *irreducible representations of O(3).*



```
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
```

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

First: Symmetry is useful

- Motivations for symmetry-equivariant ML
- Recent applications and findings

Second: Symmetry gives natural bases for expressing features

- **Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.**
  - **Irreducible representations and how to find them.**

Third: Emergent behavior of equivariant nns

- Consequences of being symmetry-preserving
- Properties of E(3)NNs that have yet to be fully utilized.
  - Symmetry can tell you when you're missing data.

First: Symmetry is useful

- Motivations for symmetry-equivariant ML
- Recent applications and findings

Second: Symmetry gives natural bases for expressing features

- Test out some content I'm developing for the course I'm teaching this Spring on symmetry + ML.
  - Irreducible representations and how to find them.

Third: Emergent behavior of equivariant nns

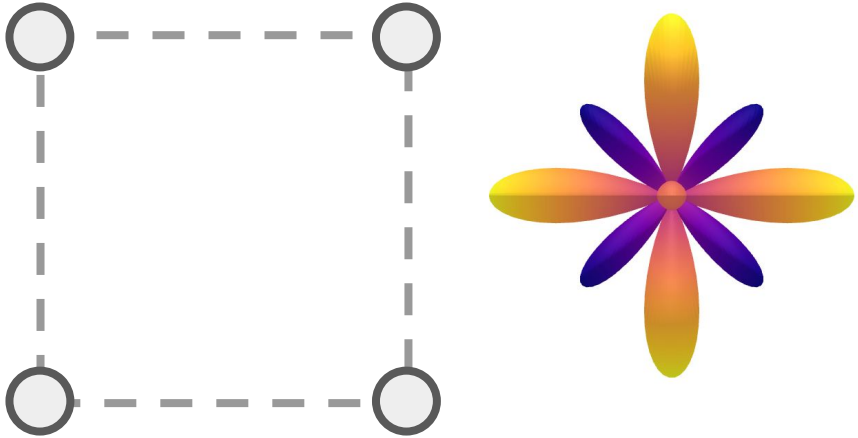
- **Consequences of being symmetry-preserving**
- **Properties of E(3)NNs that have yet to be fully utilized.**
  - **Symmetry can tell you when you're missing data.**

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

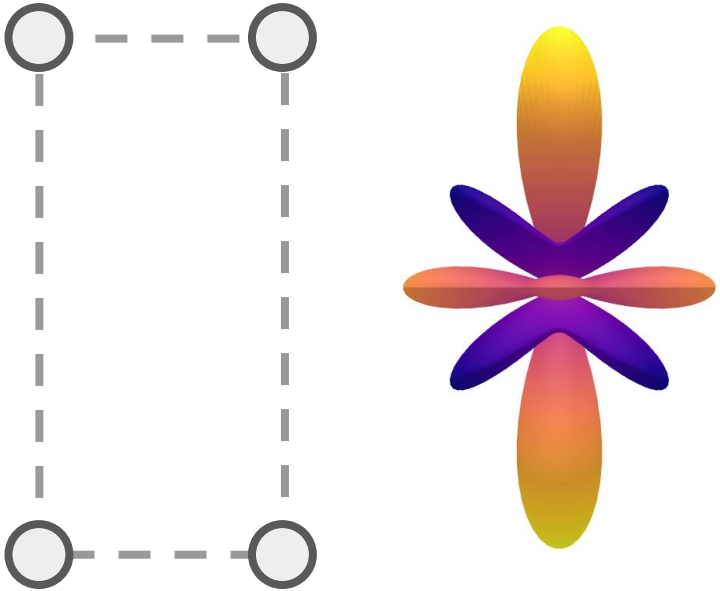
However, the symmetry of the inputs also matter! ENNs *preserve* symmetry.

Many different types of geometric objects have the same symmetry.

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



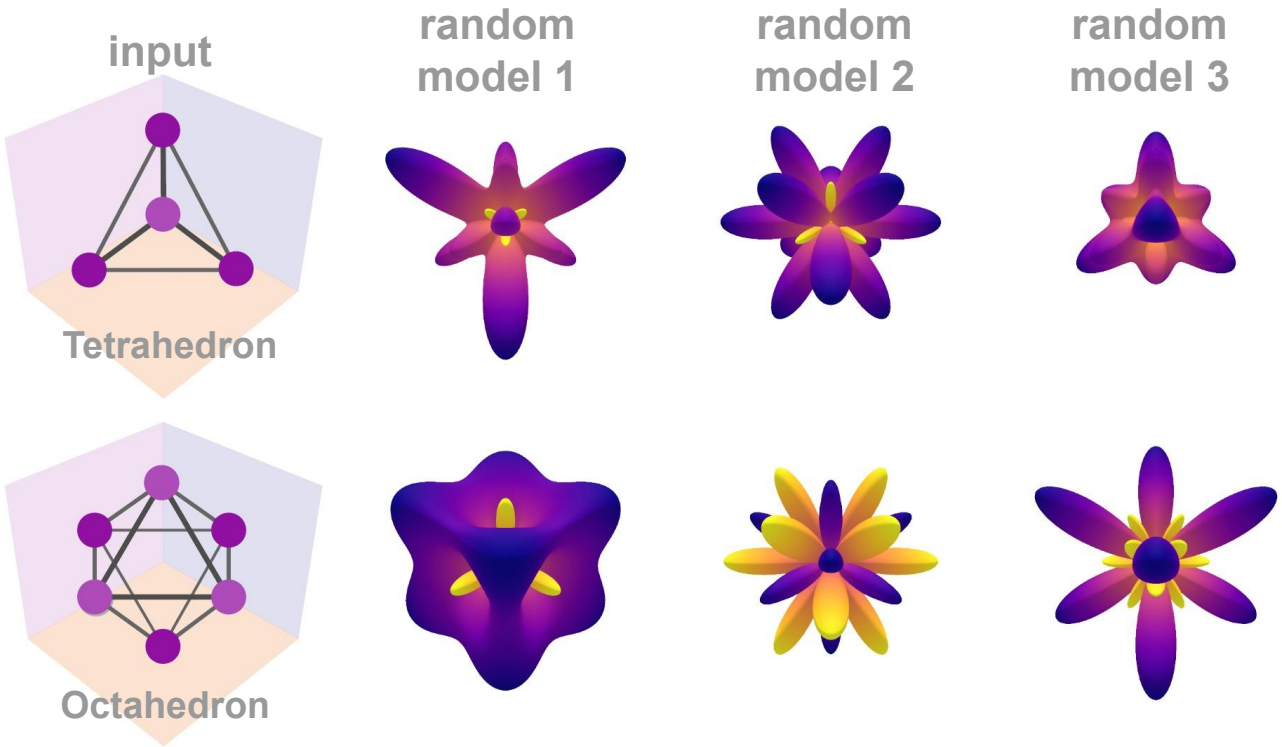
$$\mathrm{Sym}(x) \cong D_{4h}$$



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

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."



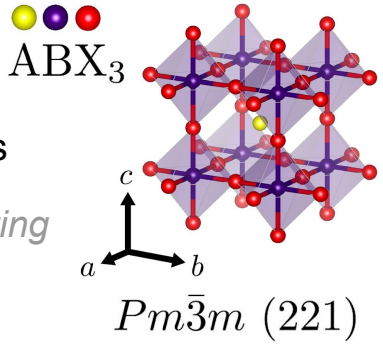
Cases where we need to predict lower symmetry outputs?

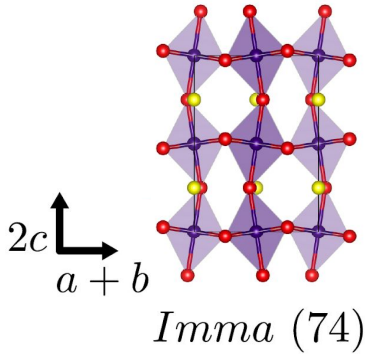
Cases where we need to predict lower symmetry outputs?

Structural phase transitions of crystals (and other symm breaking properties, e.g. magnetic order)

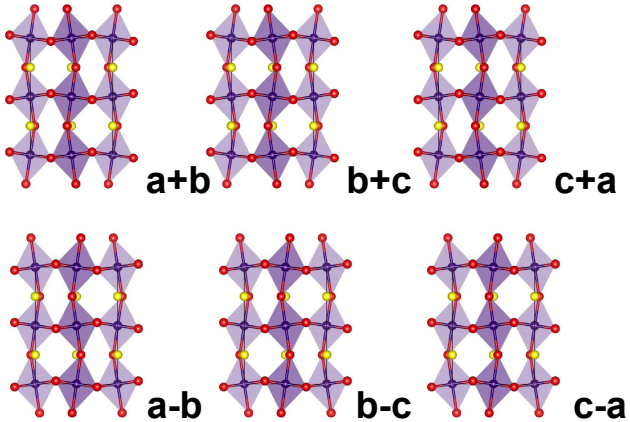
Predict distortion displacements

Octahedral tilting in perovskites





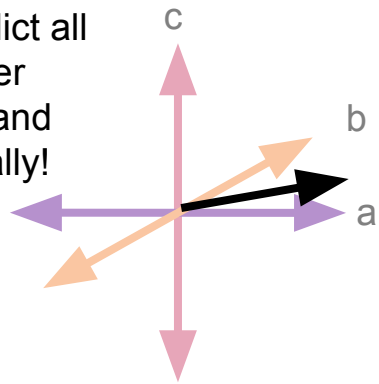
Target distorted structure



6 different vectors in reciprocal space

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

Want to predict all possible order parameters and sample equally!



Cases where we need to predict lower symmetry outputs?

Dataset may be missing the needed information for inputs and outputs to be compatible by symmetry.

Approximately Equivariant Networks for Imperfectly Symmetric Dynamics
R. Wang, R. Walters, R. Yu, ICML 2022

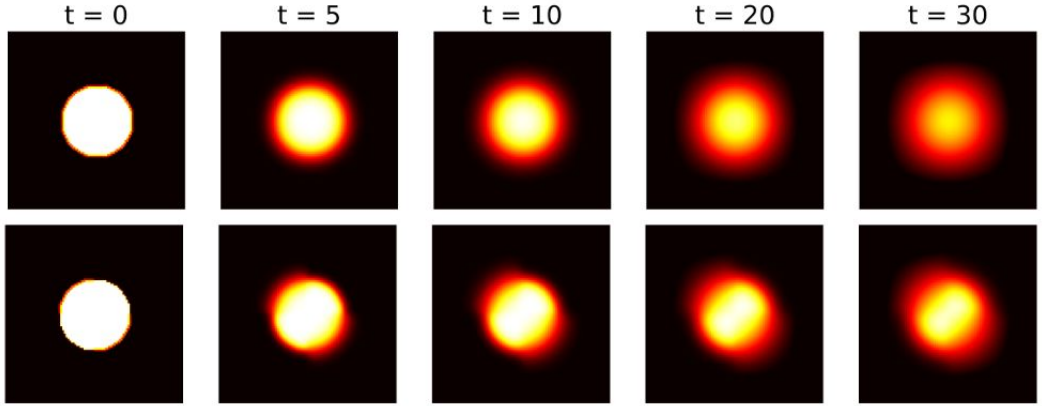
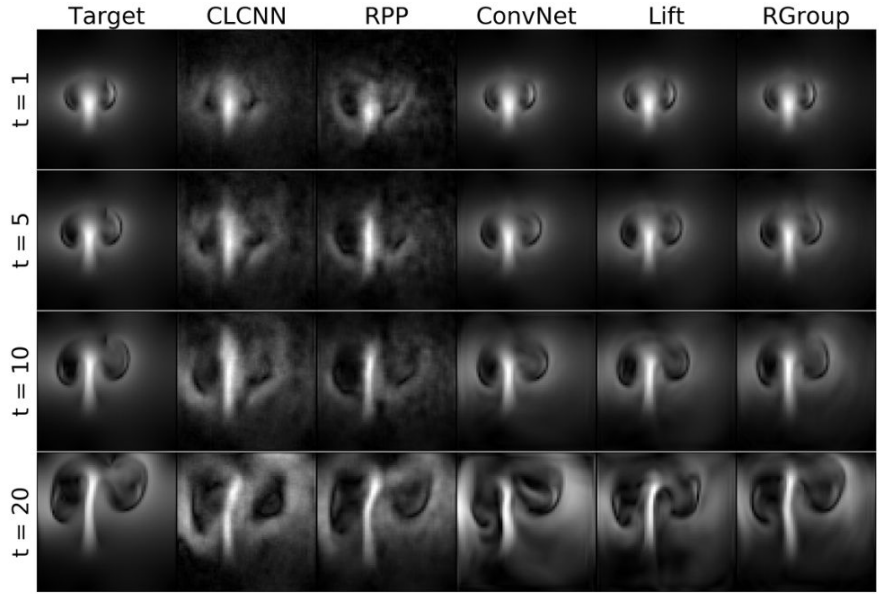


Figure 2: Simulated diffusion of heat in a metal plate with (top) uniform diffusion coefficient resulting in perfect symmetry and (bottom) slightly varying diffusion coefficient resulting in approximate symmetry.

Cases where we need to predict lower symmetry outputs?

Dataset may be missing the needed information for inputs and outputs to be compatible by symmetry.

Approximately Equivariant Networks for Imperfectly Symmetric Dynamics
R. Wang, R. Walters, R. Yu, ICML 2022



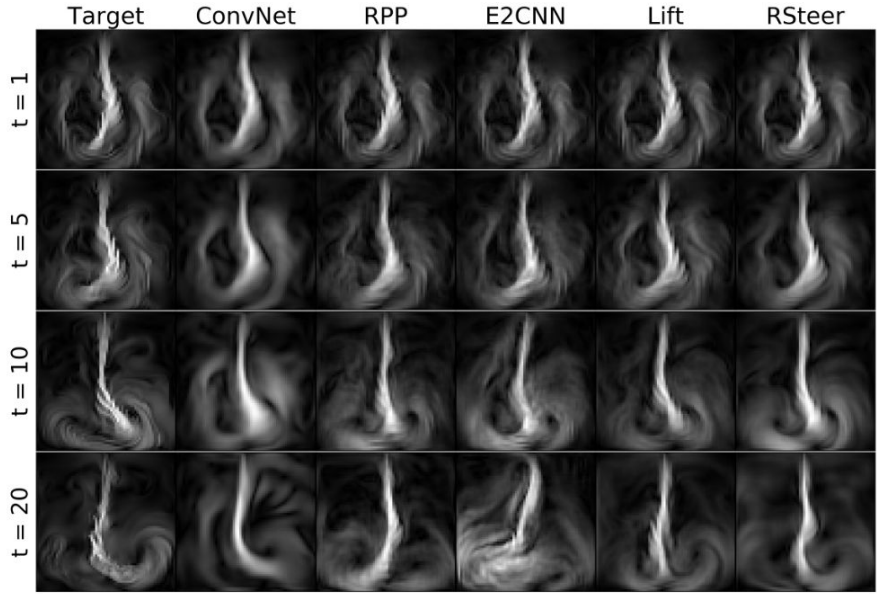
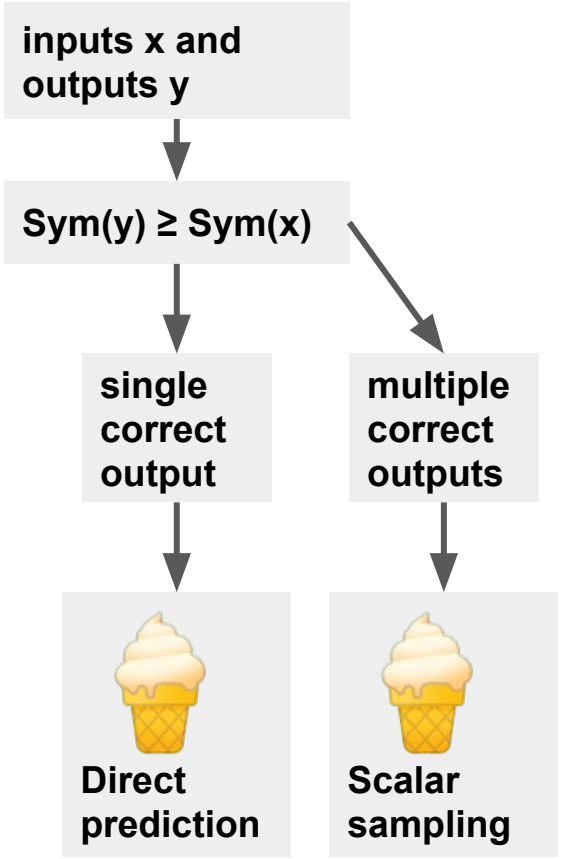


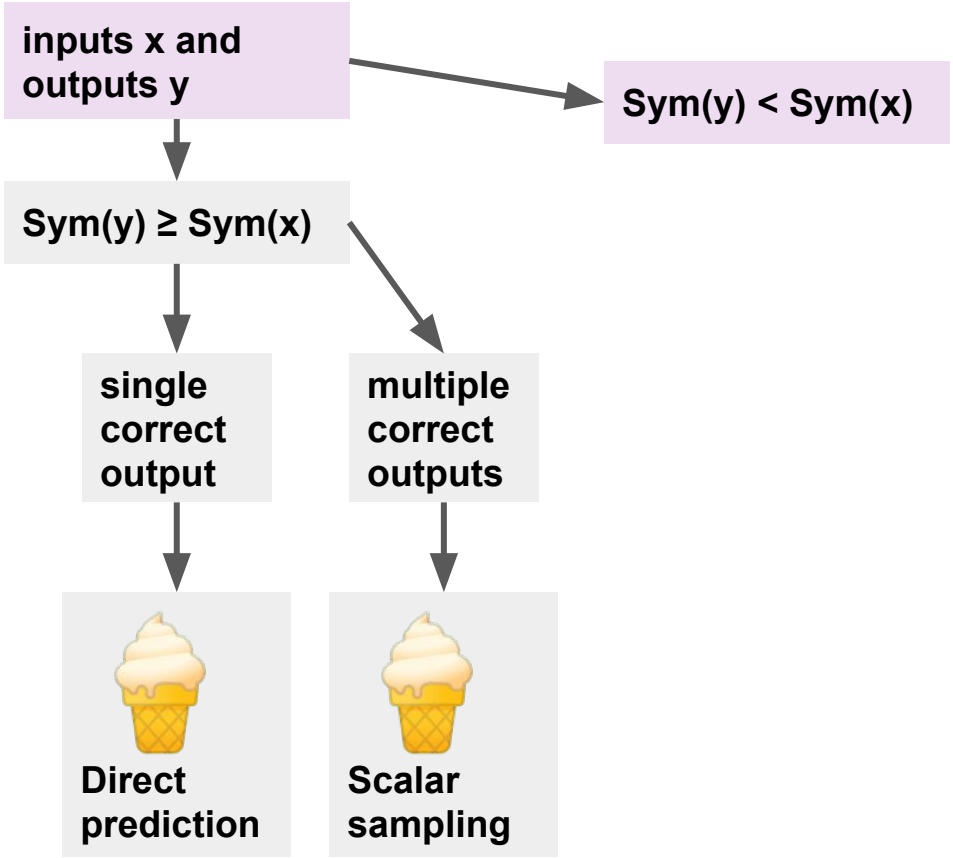
Figure 3: Target (ground truth) and model predictions comparison at time step 1, 5, 10, 20 for smoke simulation with approximate translation (left) and rotation (right) symmetries.

When can symmetry tell you something you don't know...

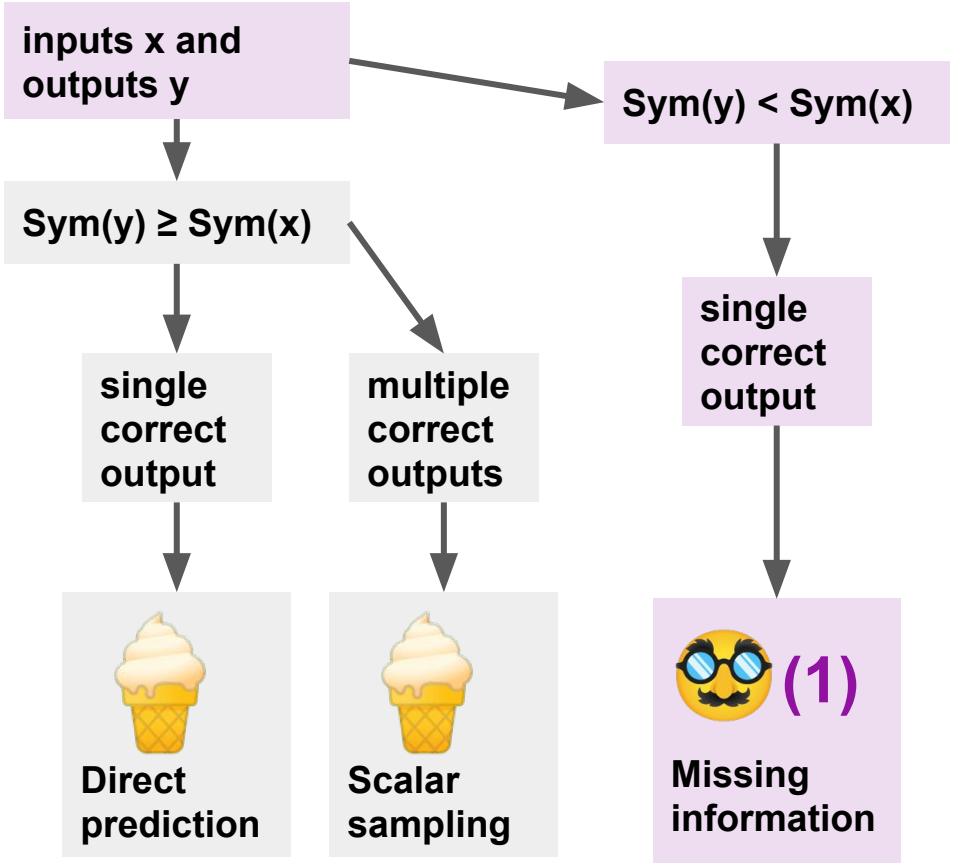
When can symmetry tell you something you don't know...



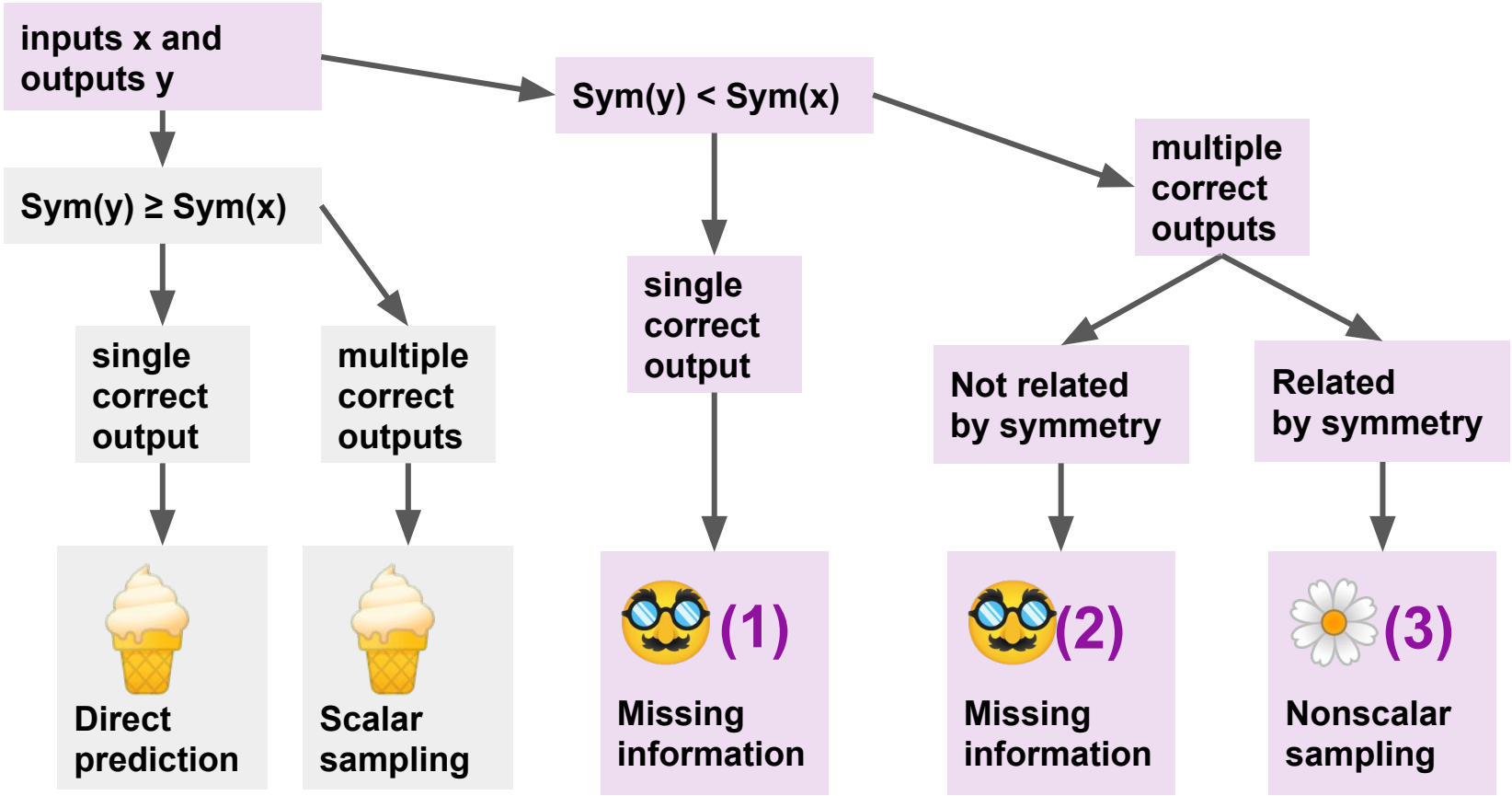
When can symmetry tell you something you don't know...



When can symmetry tell you something you don't know...

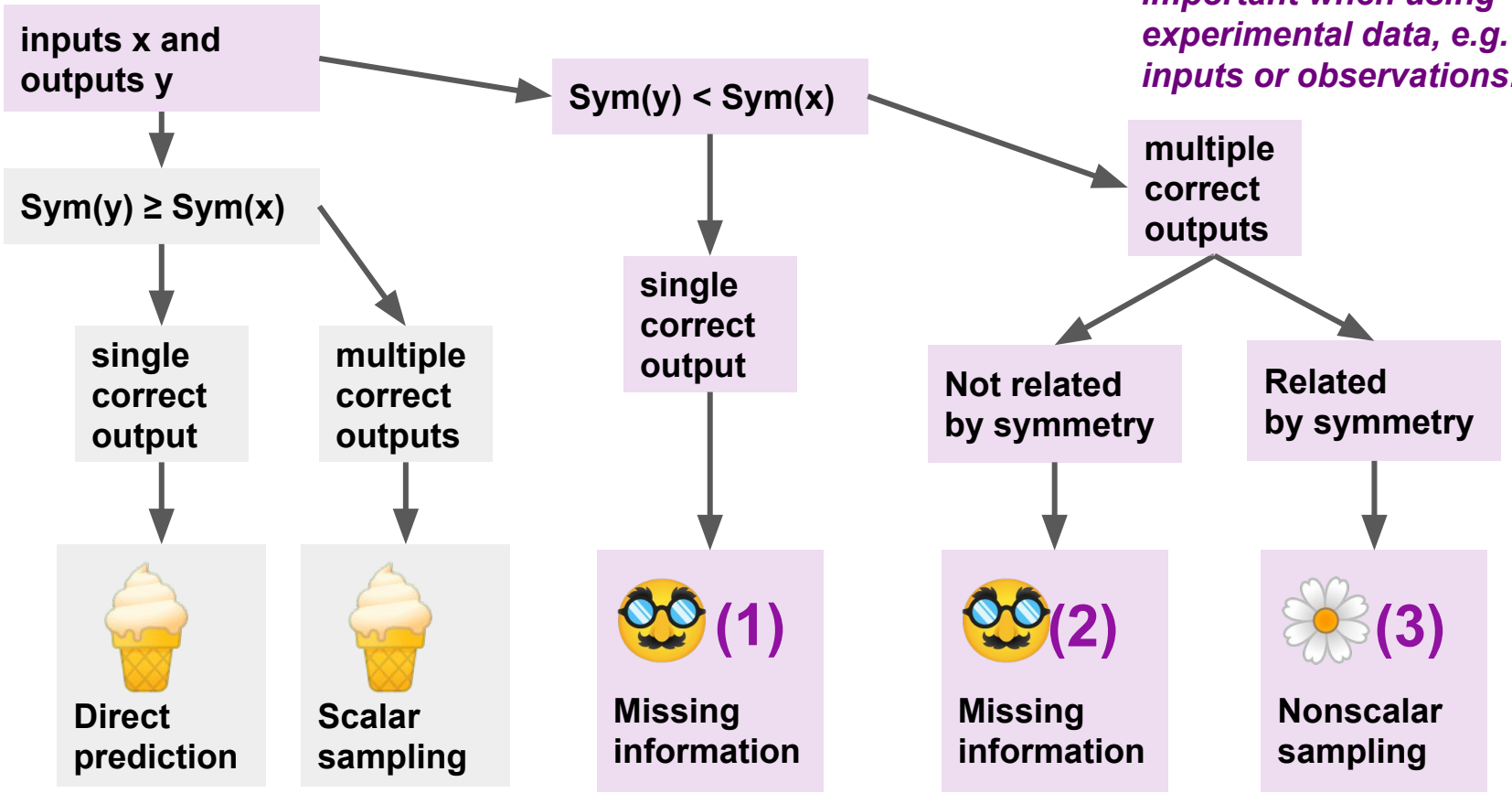


When can symmetry tell you something you don't know...

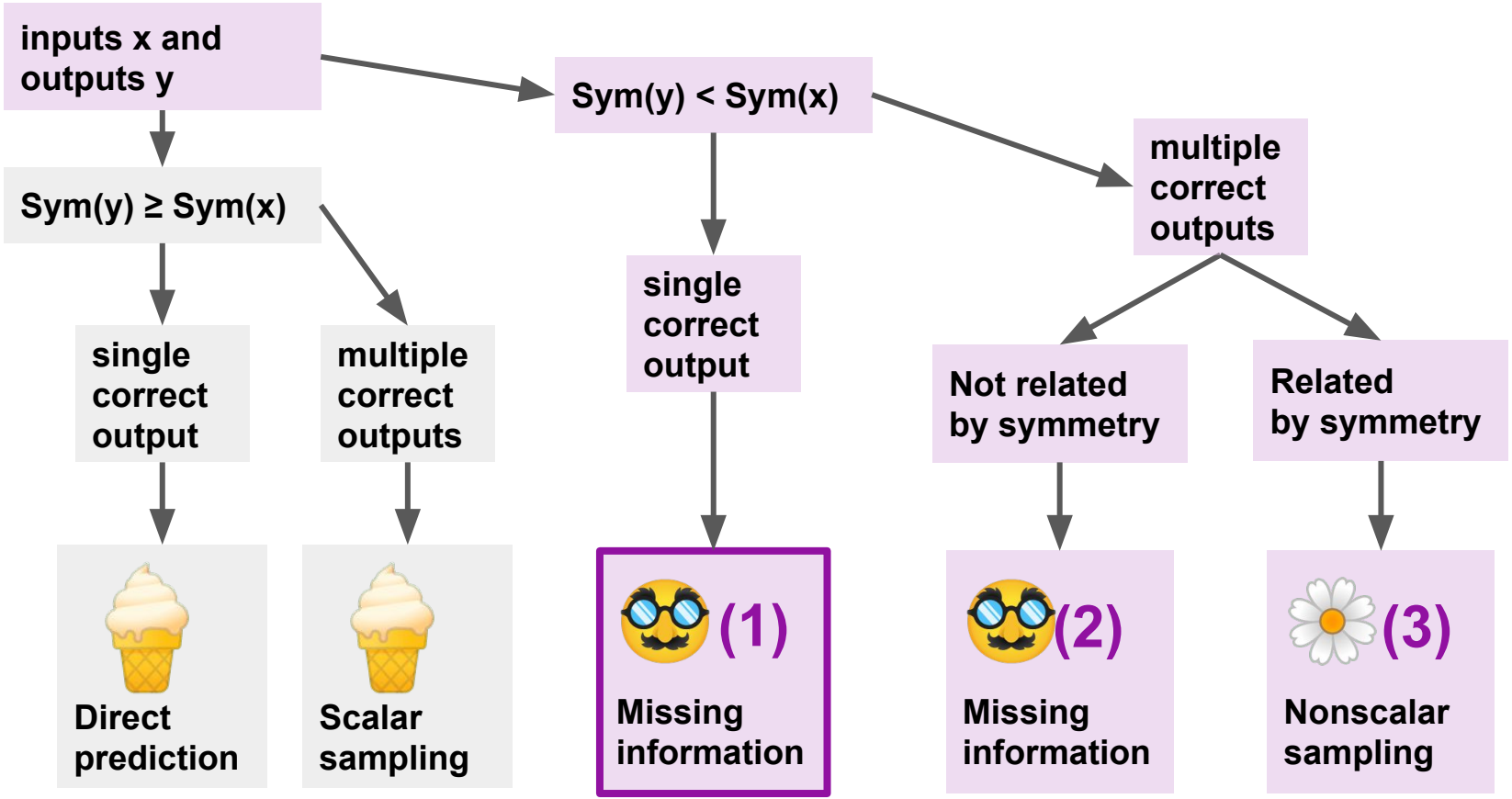


When can symmetry tell you something you don't know...

This becomes particularly important when using experimental data, e.g. partial inputs or observations!



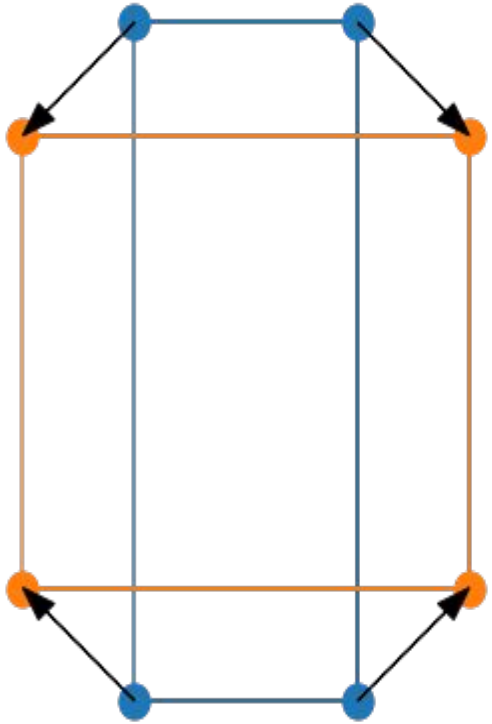
When can symmetry tell you something you don't know...



(1) Sym(y) < Sym(x) ⇨ Single output ⇨ Missing Information

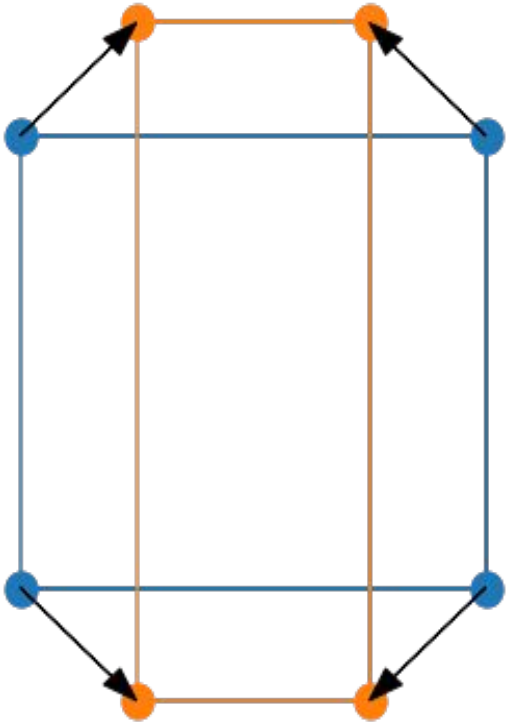
Sym(y) ≥ Sym(x)

Task 1: Rectangle to Square

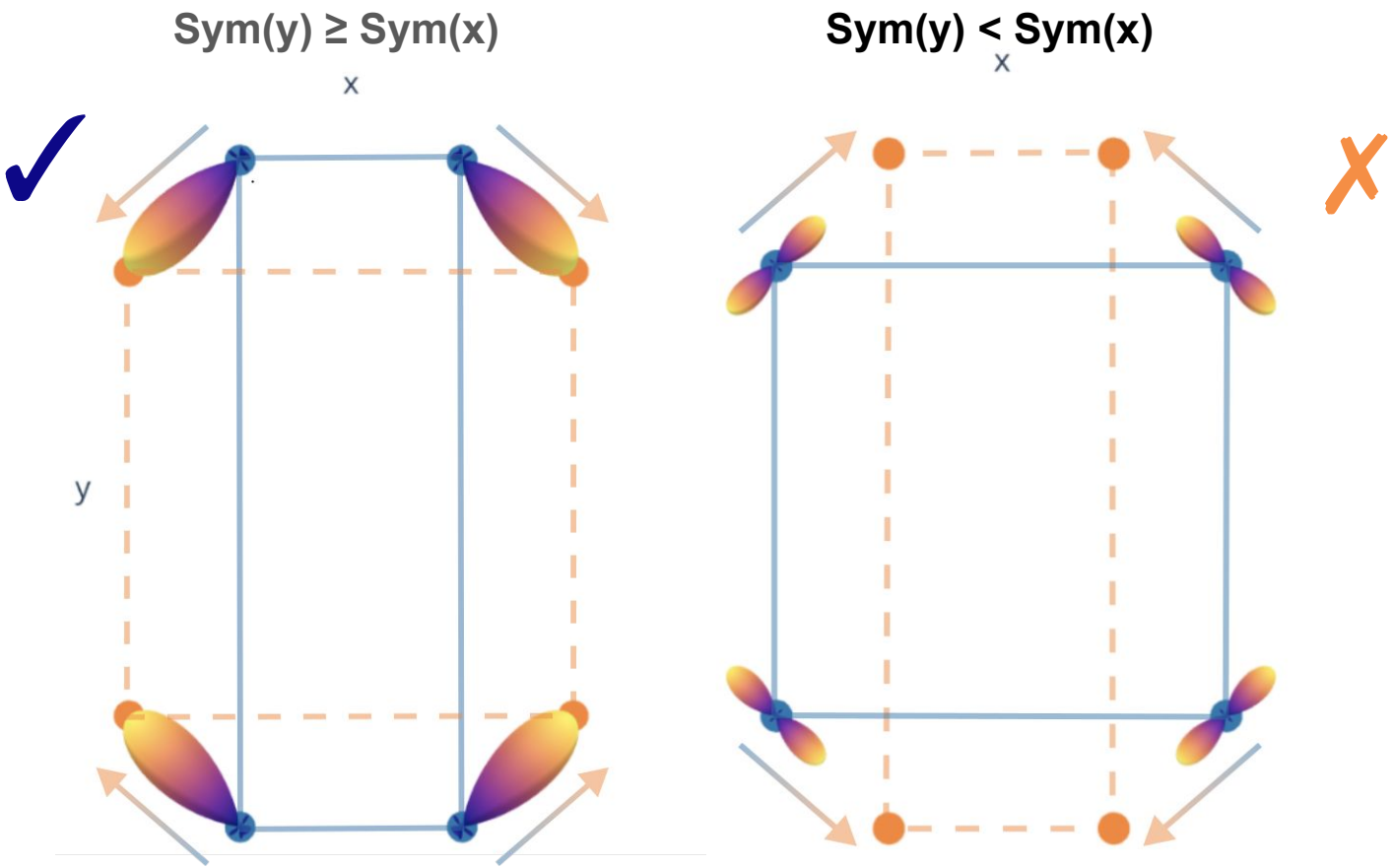


Sym(y) < Sym(x)

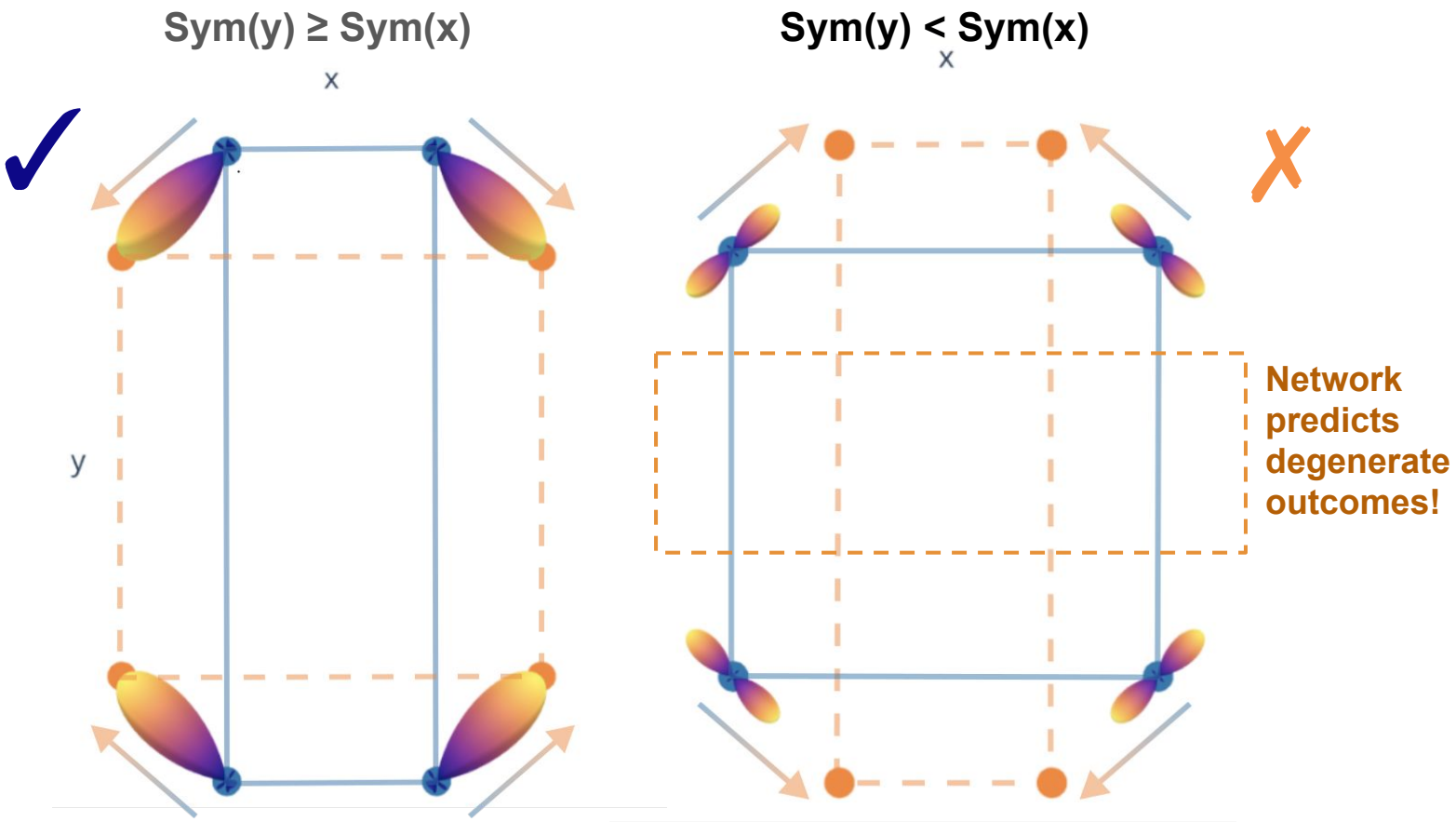
Task 2: Square to Rectangle



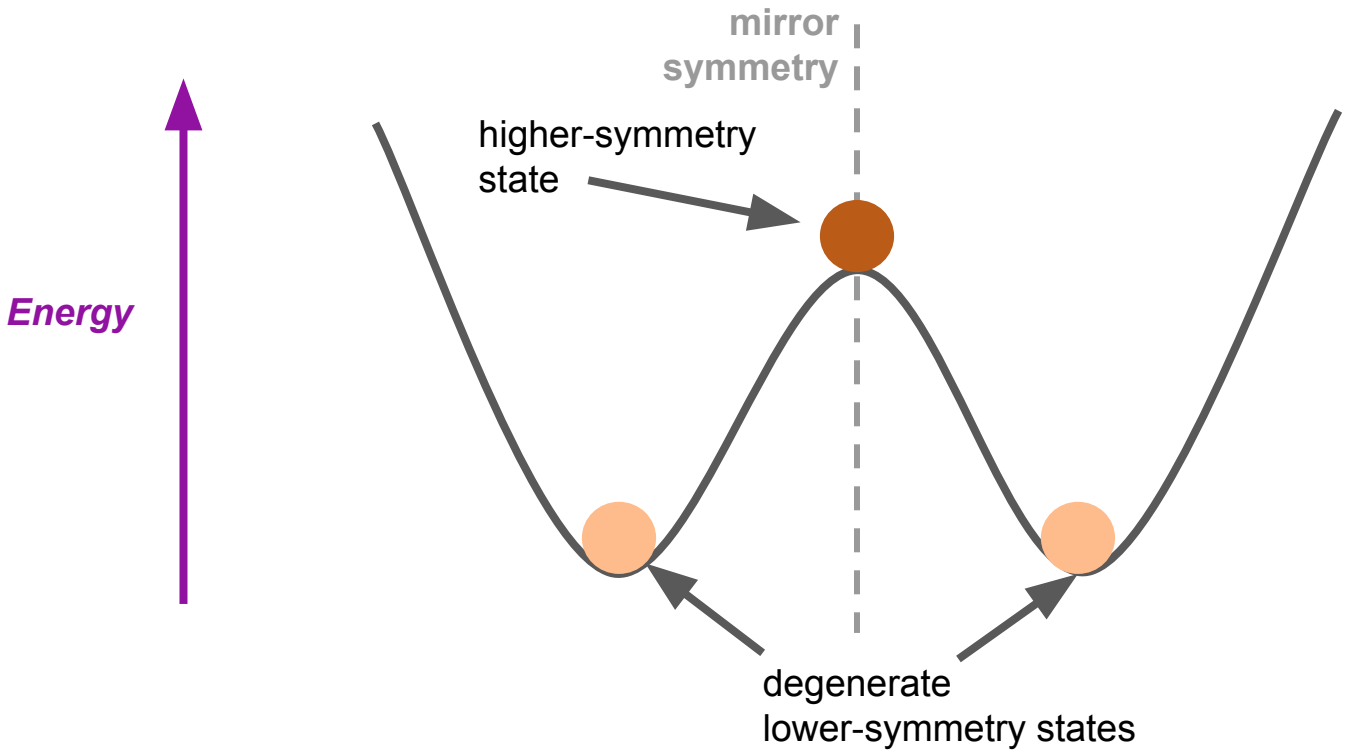
(1) Sym(y) < Sym(x) ⇨ Single output ⇨ Missing Information



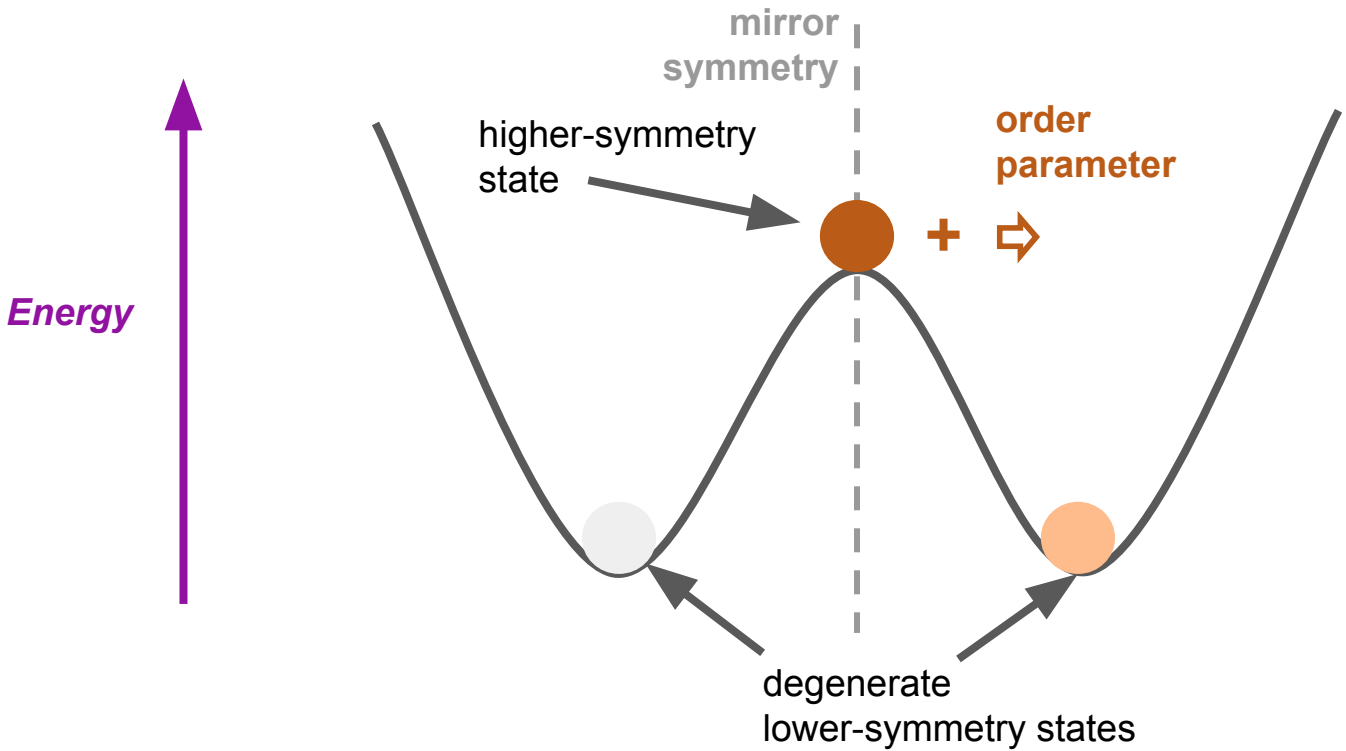
(1) Sym(y) < Sym(x) ⇨ Single output ⇨ Missing Information



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?

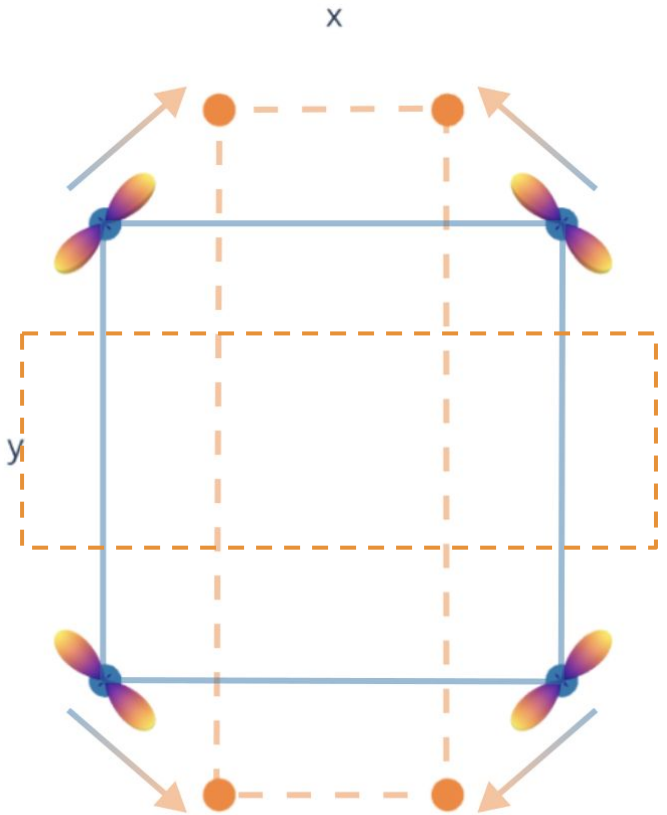
input — learnable parameters — predicted output

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

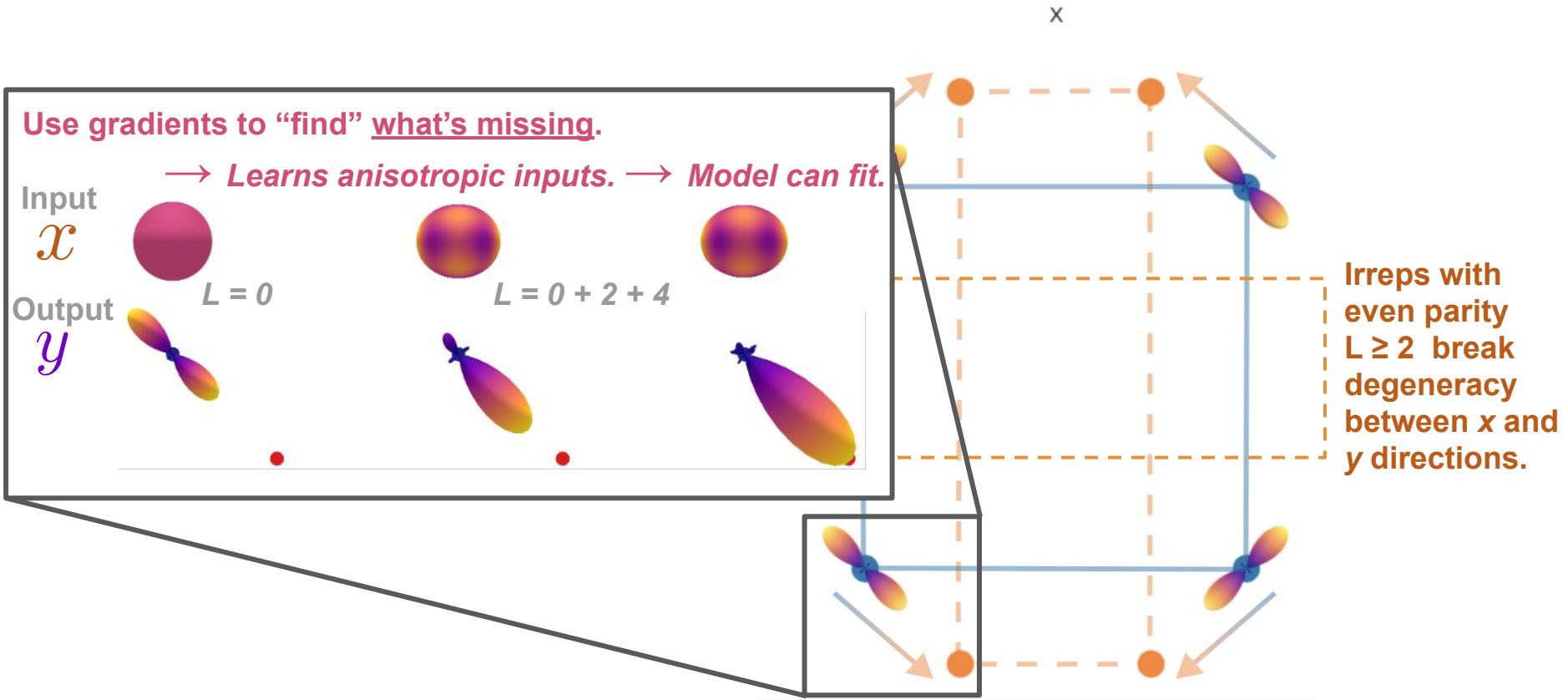
inputs

Update weights using...

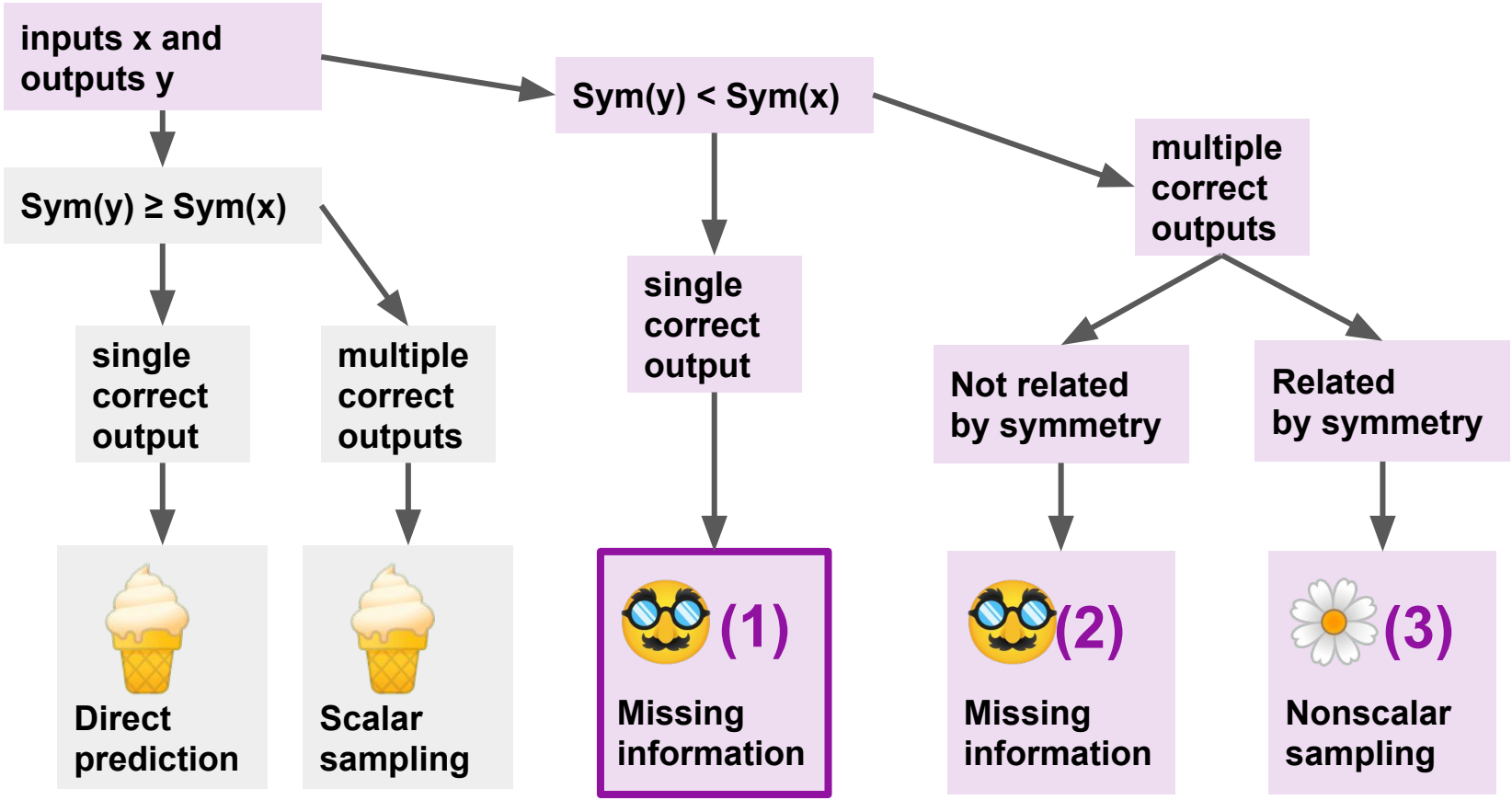
$$w_i^{x} = w_i^{x} - \eta \frac{\partial \text{loss}}{\partial w_i} x$$



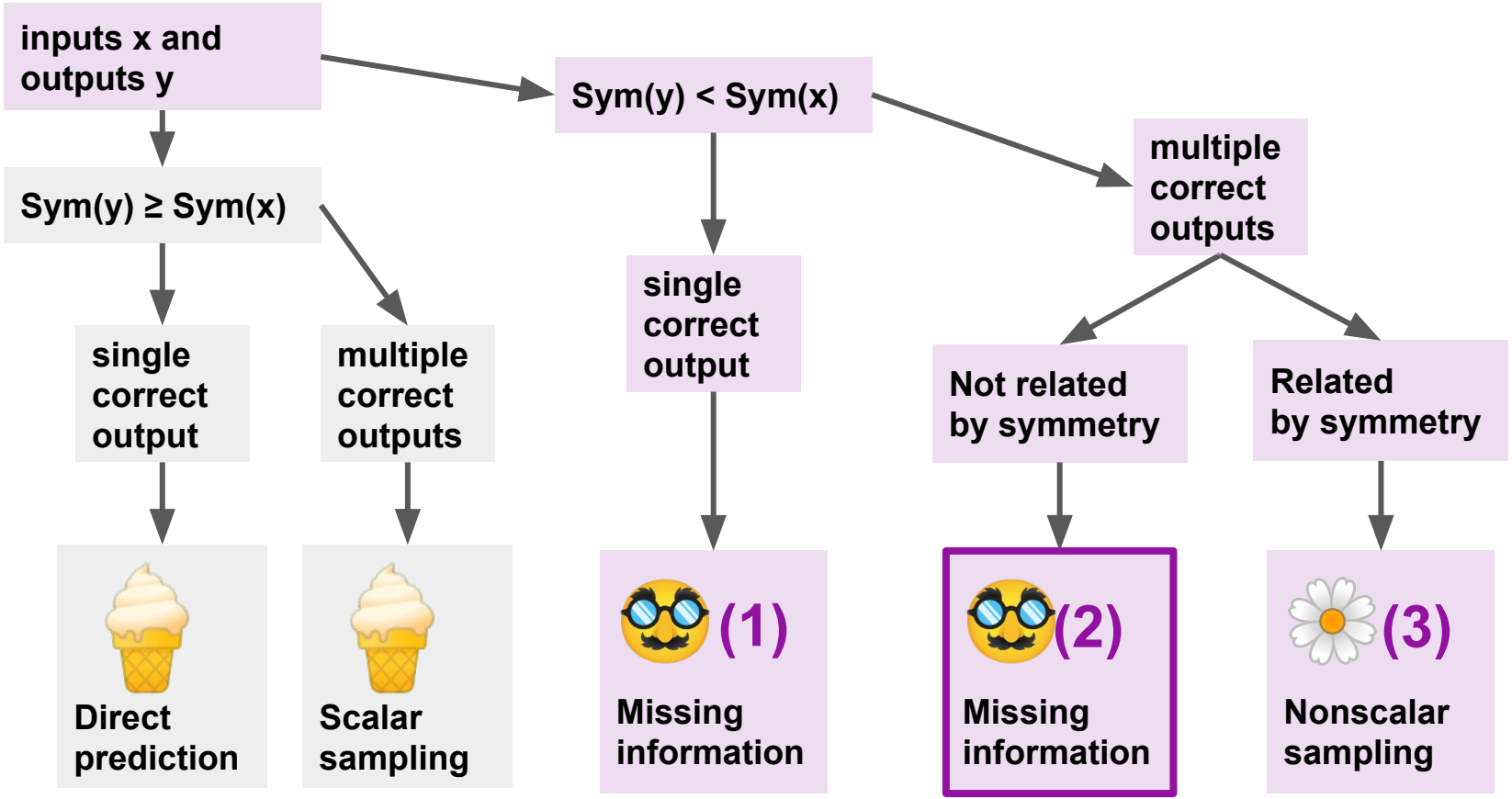
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").



When can symmetry tell you something you don't know...

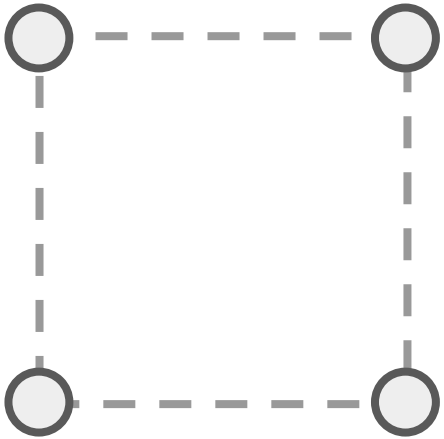


When can symmetry tell you something you don't know...

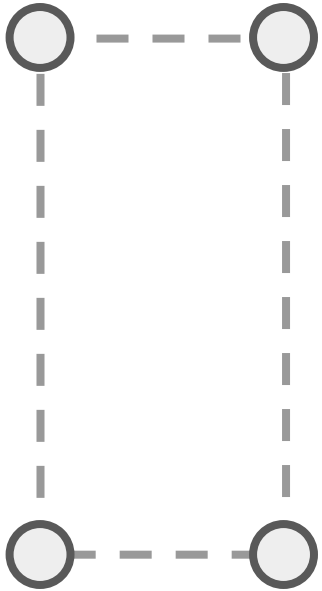


(2) Multiple outputs ⇨ Not related by symmetry ⇨ Missing Information

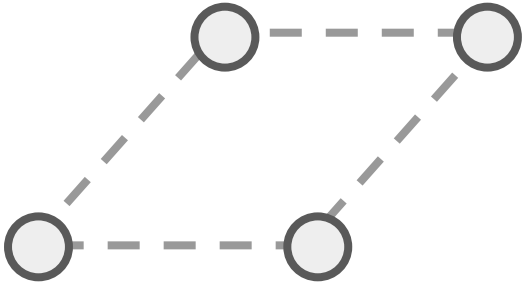
input



output 1



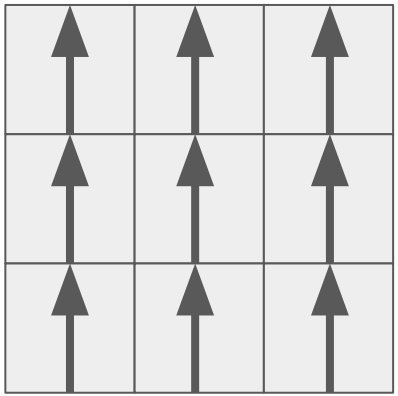
output 2



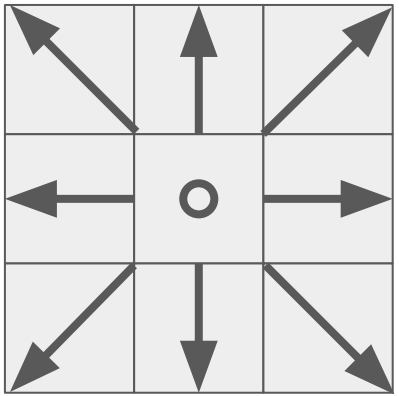
By symmetry these outputs must be generated by different inputs.
e.g. unobserved system features.

(2) Multiple outputs ⇨ Not related by symmetry ⇨ Missing Information

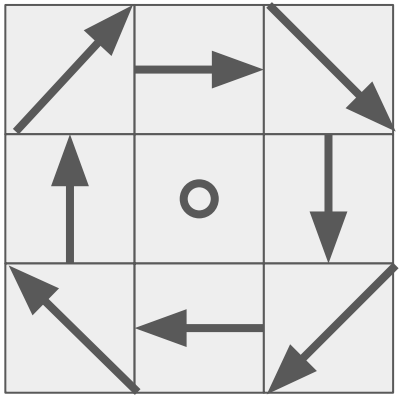
input



output 1

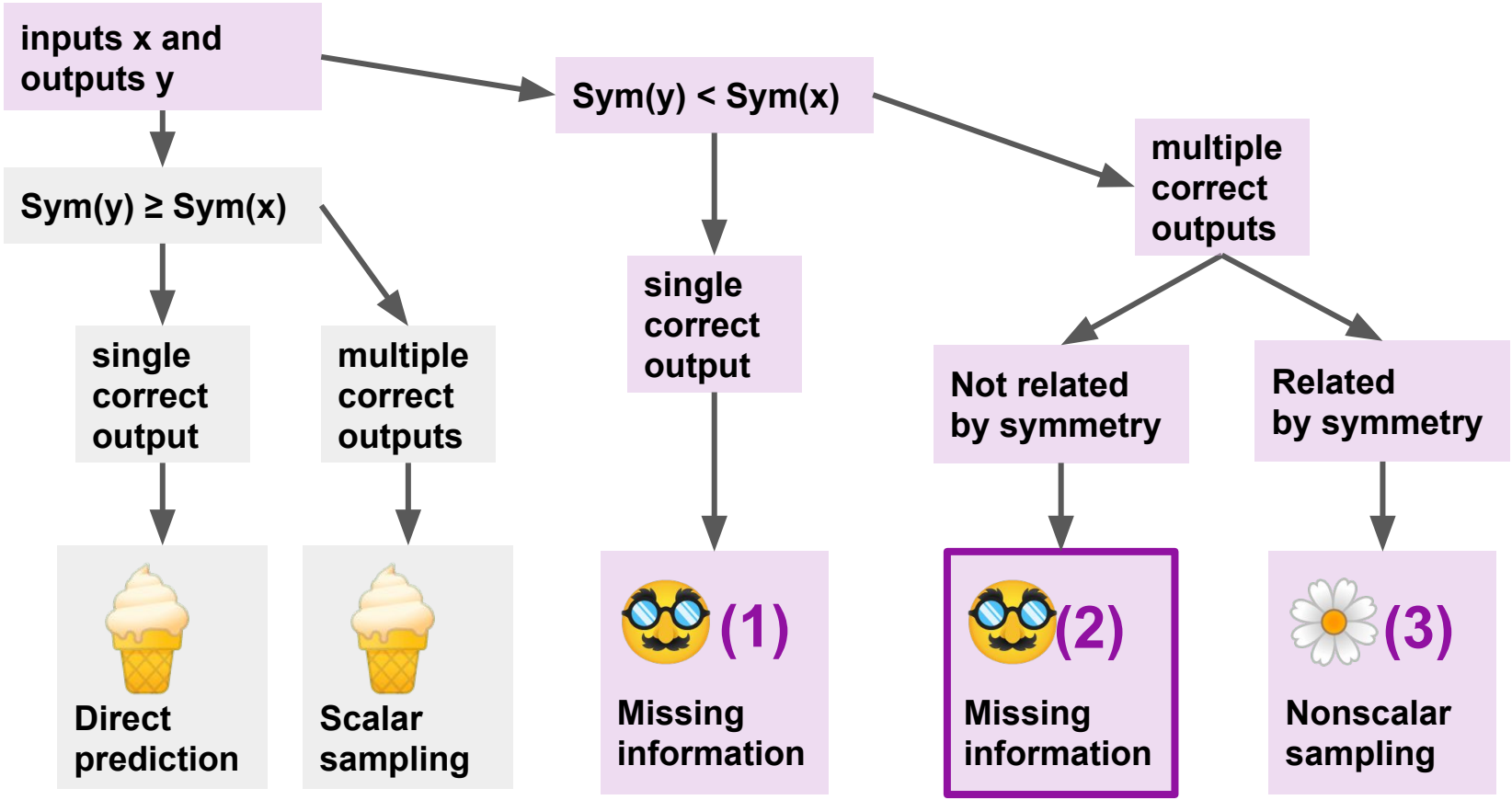


output 2

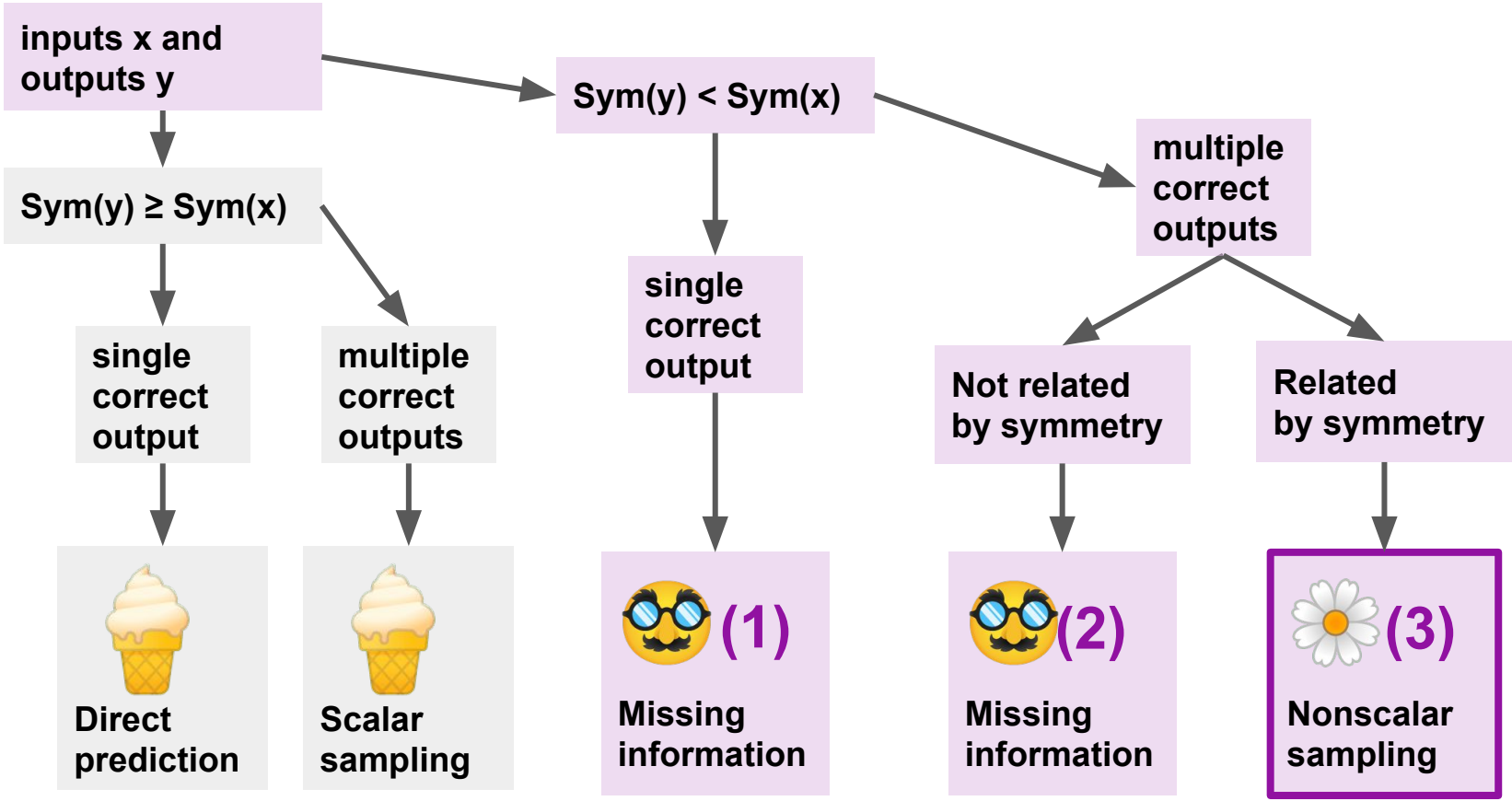


By symmetry these outputs must be generated by different inputs.
e.g. unobserved system features.

When can symmetry tell you something you don't know...



When can symmetry tell you something you don't know...



(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

<u>Goal:</u> Partition this graph into two subgraphs

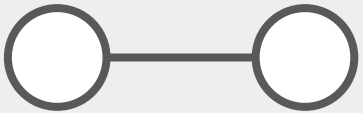
subgraph 1 [1, 0]

subgraph 2 [0, 1]

(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

Goal: **Partition this graph into two subgraphs**



subgraph 1 [1, 0]

subgraph 2 [0, 1]

y

Trained

[1, 0, 0, 1]

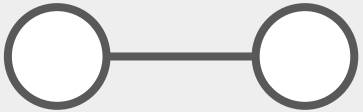
Learned

[½, ½, ½, ½]

(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

Goal: **Partition this graph into two subgraphs**



subgraph 1 [1, 0]

subgraph 2 [0, 1]

y

Trained

[1, 0, 0, 1]

Learned

[½, ½, ½, ½]

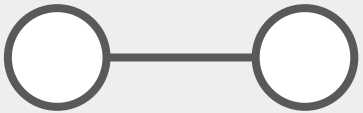
Implied

[0, 1, 1, 0]

(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

Goal: **Partition this graph into two subgraphs**



subgraph 1 [1, 0]

subgraph 2 [0, 1]

y

Trained

[1, 0, 0, 1]

Learned

[½, ½, ½, ½] ← *Not particular helpful. Can't recover distinct partitions.*

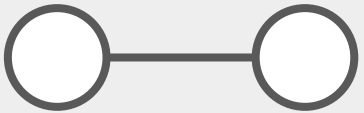
Implied

[0, 1, 1, 0]

(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

Goal: **Partition this graph into two subgraphs**



subgraph 1 [1, 0]

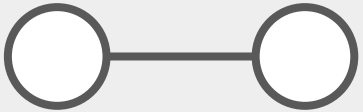
subgraph 2 [0, 1]

		y	$y \otimes y$
Trained		[1, 0, 0, 1]	$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
Learned		[½, ½, ½, ½]	$\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
Implied		[0, 1, 1, 0]	$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling

An analogous problem to the square ⇨ rectangle for the permutation group.

Goal: **Partition this graph into two subgraphs**



subgraph 1 [1, 0]

subgraph 2 [0, 1]

		y	$y \otimes y$
Trained	[1, 0, 0, 1]		$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
Learned	[½, ½, ½, ½]		$\frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$
Implied	[0, 1, 1, 0]		$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$

Does not have unique decomposition!
... but this (often) does!

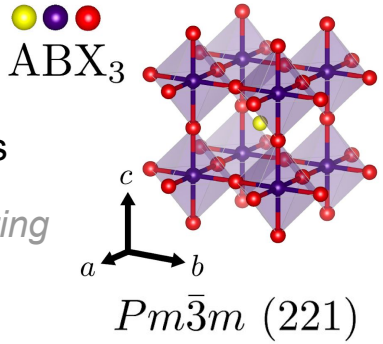
$y \otimes y \otimes y$

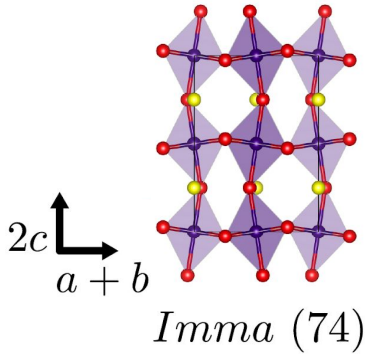
(3) Multiple outputs ➪ Related by symmetry ➪ Nonscalar sampling

A physical example: Structural phase transitions of crystals

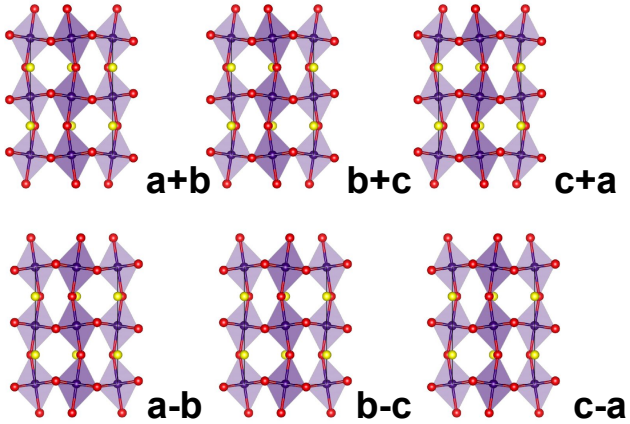
Predict distortion displacements

Octahedral tilting in perovskites





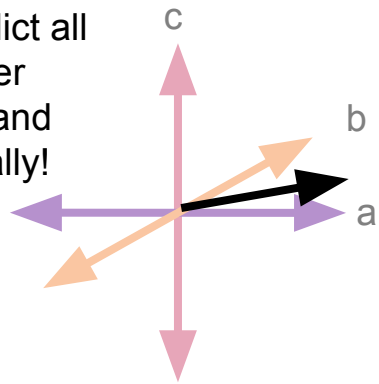
Target distorted structure



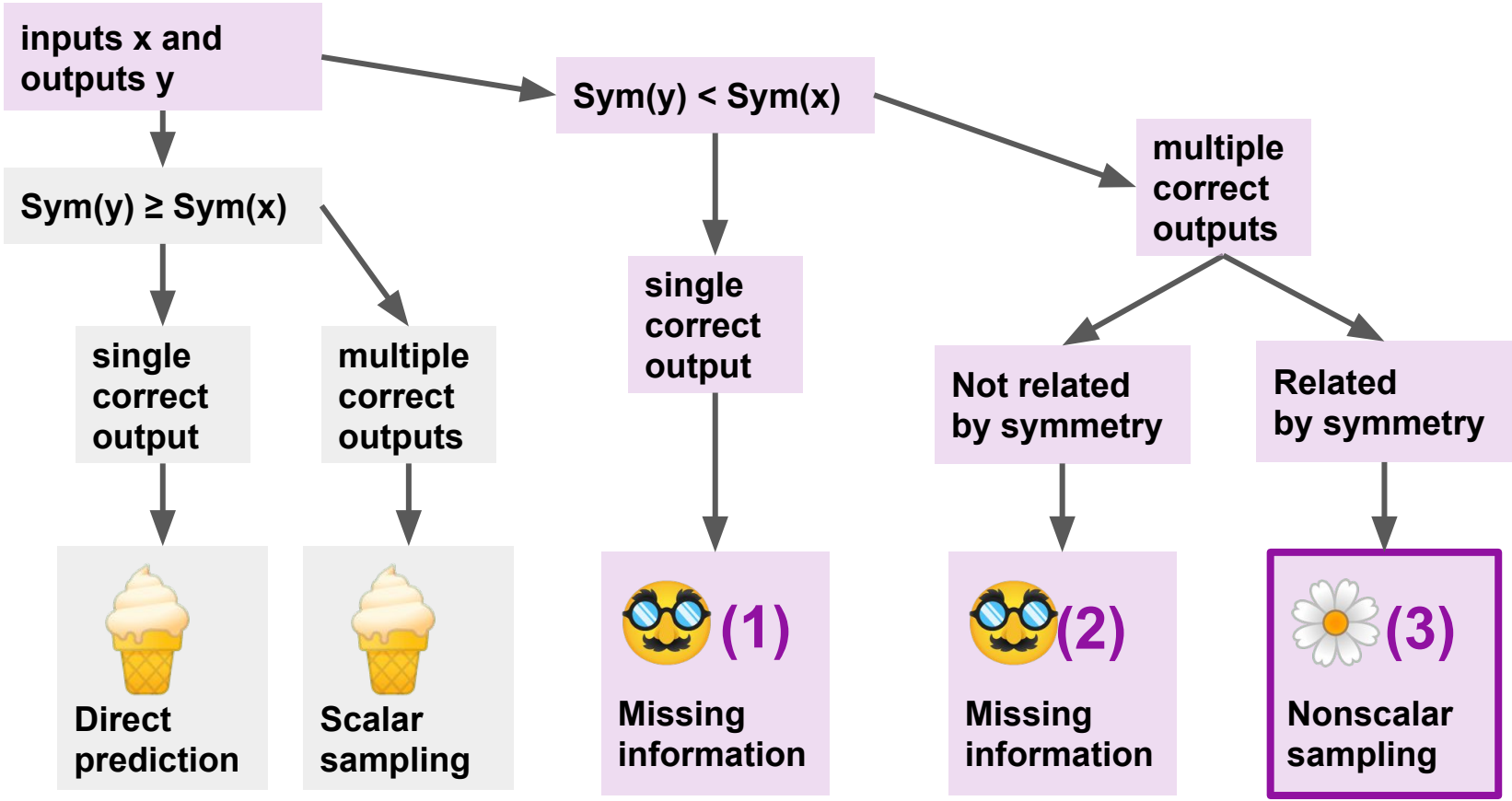
6 different vectors in reciprocal space

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

Want to predict all possible order parameters and sample equally!

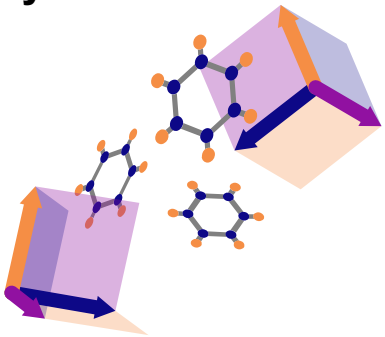


When can symmetry tell you something you don't know...

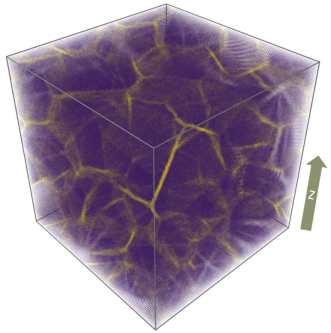


TAKEAWAYS

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

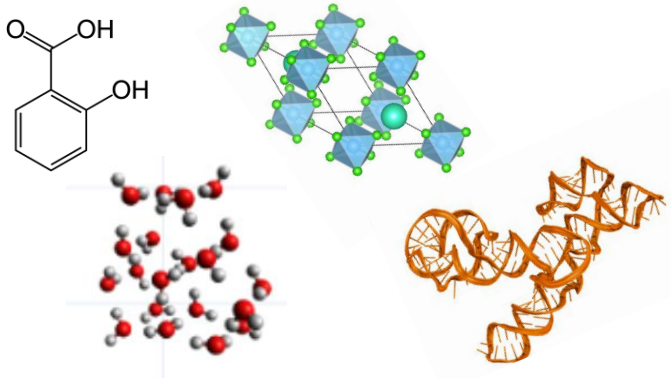


This makes these models data-efficient, robust, scalable, and generalizable.

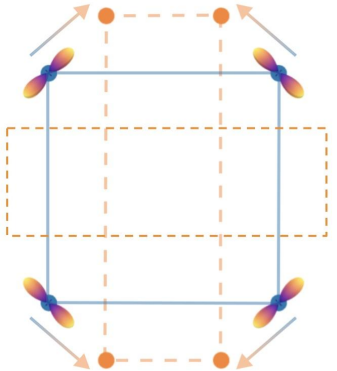


QM accurate MD on 100s of millions of atoms.

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



ENN properties can uncover missing information and efficiently learn statistics of physical systems.



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

Calling in backup (slides)!

Methods	Task Units	α bohr^3	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_ν cal/mol K	G meV	H meV	R^2 bohr^3	U meV	U_0 meV	ZPVE meV
NMP [29]		.092	69	43	38	.030	.040	19	17	.180	20	20	1.50
SchNet [64]†		.235	63	41	34	.033	.033	14	14	.073	19	14	1.70
Cormorant [1]		.085	61	34	38	.038	.026	20	21	.961	21	22	2.03
LieConv [21]		.084	49	30	25	.032	.038	22	24	.800	19	19	2.28
DimeNet++ [25]†		.044	33	25	20	.030	.023	8	7	.331	6	6	1.21
TFN [72]‡		.223	58	40	38	.064	.101	-	-	-	-	-	-
SE(3)-Transformer [23]		.142	53	35	33	.051	.054	-	-	-	-	-	-
EGNN [63]		.071	48	29	25	.029	.031	12	12	.106	12	11	1.55
SphereNet [48]†		.046	32	23	18	.026	.021	8	6	.292	7	6	1.12
SEGNN [5]		.060	42	24	21	.023	.031	15	16	.660	13	15	1.62
EQGAT [46]		.063	44	26	22	.014	.027	12	13	.257	13	13	1.50
Equiformer		.056	33	17	16	.014	.025	10	10	.227	11	10	1.32

Table 1: **MAE results on QM9 testing set.** † denotes using different training, validation, testing data partitions as mentioned in SEGNN [5]. ‡ denotes results from SE(3)-Transformer [23].

Methods	Energy MAE (eV) ↓					EwT (%) ↑				
	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
SchNet [64]†	0.6465	0.7074	0.6475	0.6626	0.6660	2.96	2.22	3.03	2.38	2.65
DimeNet++ [25]†	0.5636	0.7127	0.5612	0.6492	0.6217	4.25	2.48	4.40	2.56	3.42
GemNet-T [42]†	0.5561	0.7342	0.5659	0.6964	0.6382	4.51	2.24	4.37	2.38	3.38
SphereNet [48]	0.5632	0.6682	0.5590	0.6190	0.6024	4.56	2.70	4.59	2.70	3.64
(S)EGNN [5]	0.5497	0.6851	0.5519	0.6102	0.5992	4.99	2.50	4.71	2.88	3.77
SEGNN [5]	0.5310	0.6432	0.5341	0.5777	0.5715	5.32	2.80	4.89	3.09	4.03
Equiformer	0.5088	0.6271	0.5051	0.5545	0.5489	4.88	2.93	4.92	2.98	3.93

Table 2: **Results on OC20 IS2RE validation set.** † denotes results reported by SphereNet [48].

Methods	Energy MAE (eV) ↓					EwT (%) ↑				
	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
CGCNN [83]	0.6149	0.9155	0.6219	0.8511	0.7509	3.40	1.93	3.10	2.00	2.61
SchNet [64]	0.6387	0.7342	0.6616	0.7037	0.6846	2.96	2.33	2.94	2.21	2.61
DimeNet++ [25]	0.5621	0.7252	0.5756	0.6613	0.6311	4.25	2.07	4.10	2.41	3.21
SpinConv [68]	0.5583	0.7230	0.5687	0.6738	0.6310	4.08	2.26	3.82	2.33	3.12
SphereNet [48]	0.5625	0.7033	0.5708	0.6378	0.6186	4.47	2.29	4.09	2.41	3.32
SEGNN [5]	0.5327	0.6921	0.5369	0.6790	0.6101	5.37	2.46	4.91	2.63	3.84
Equiformer	0.5037	0.6881	0.5213	0.6301	0.5858	5.14	2.41	4.67	2.69	3.73

Table 3: **Results on OC20 IS2RE testing set.**

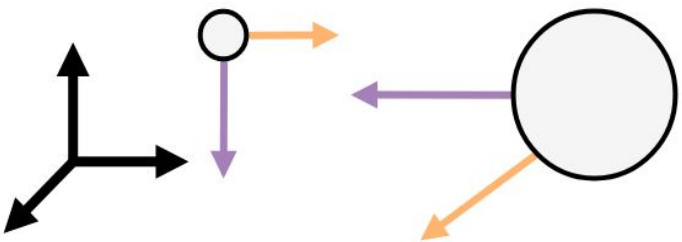
Methods	Energy MAE (eV) ↓					EwT (%) ↑				
	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
GNS [30]	0.54	0.65	0.55	0.59	0.5825	-	-	-	-	-
Noisy Nodes [30]	0.47	0.51	0.48	0.46	0.4800	-	-	-	-	-
Graphormer [66]	0.4329	0.5850	0.4441	0.5299	0.4980	-	-	-	-	-
Equiformer	0.4222	0.5420	0.4231	0.4754	0.4657	7.23	3.77	7.13	4.10	5.56
+ Noisy Nodes	0.4156	0.4976	0.4165	0.4344	0.4410	7.47	4.64	7.19	4.84	6.04

Table 4: **Results on OC20 IS2RE validation set when IS2RS node-level auxiliary task is adopted during training.** "GNS" denotes the 50-layer GNS trained without Noisy Nodes data augmentation, and "Noisy Nodes" denotes the 100-layer GNS trained with Noisy Nodes. "Equiformer + Noisy Nodes" uses data augmentation of interpolating between initial structure and relaxed structure and adding Gaussian noise as described by Noisy Nodes [30].

Methods	Energy MAE (eV) ↓					EwT (%) ↑				
	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
GNS + Noisy Nodes [30]	0.4219	0.5678	0.4366	0.4651	0.4728	9.12	4.25	8.01	4.64	6.5
Graphormer [66]†	0.3976	0.5719	0.4166	0.5029	0.4722	8.97	3.45	8.18	3.79	6.1
Equiformer + Noisy Nodes	0.4171	0.5479	0.4248	0.4741	0.4660	7.71	3.70	7.15	4.07	5.66

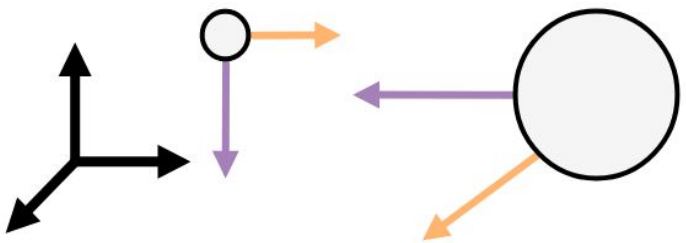
Table 5: **Results on OC20 IS2RE testing set when IS2RS node-level auxiliary task is adopted during training.** † denotes using ensemble of models trained with both IS2RE training and validation sets. In contrast, we use the same single Equiformer model in Table 4, which is trained with only the training set, for evaluation on the testing set.

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



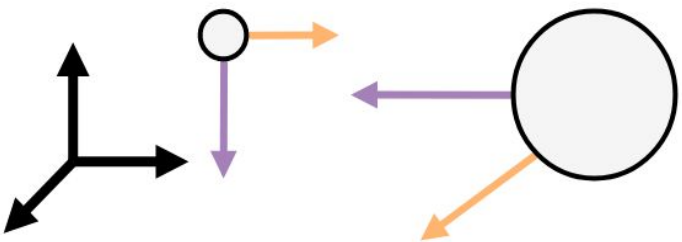
```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
...
```

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).*

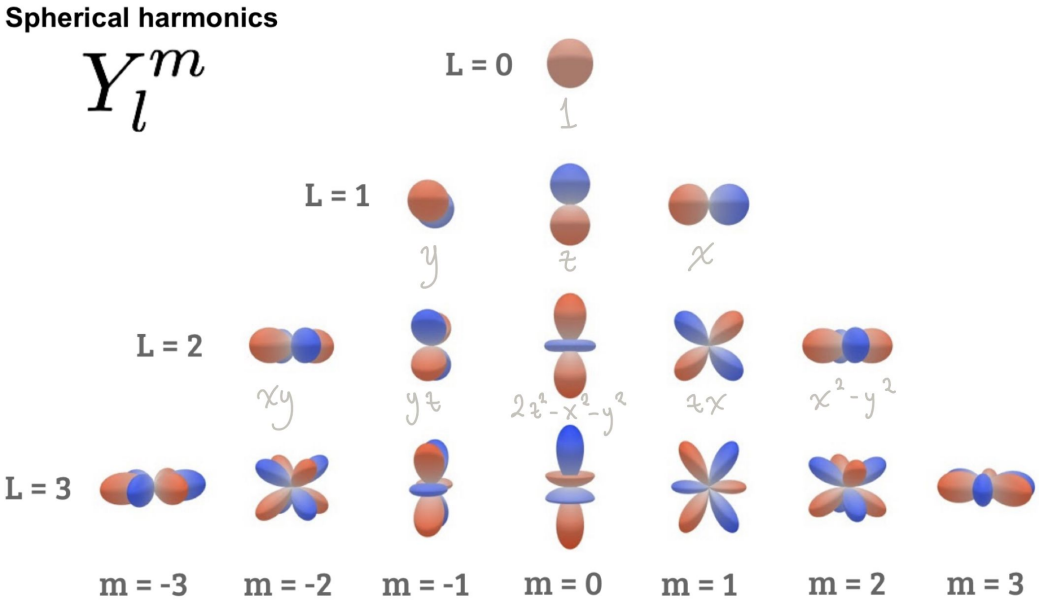


```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even
vector = e3nn.o3.Irrep("1o")  # L=1, odd
irreps = 1 * scalar + 1 * vector + 1 * vector
```

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:

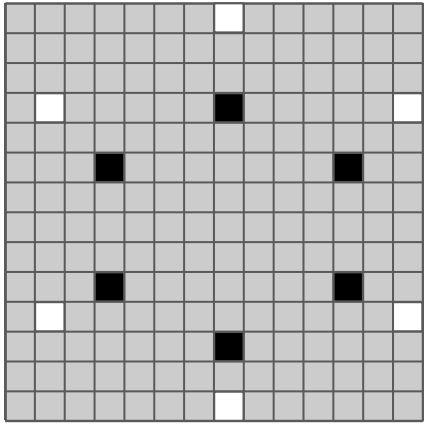
Used for images. In each layer, scan over image with learned filters.

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

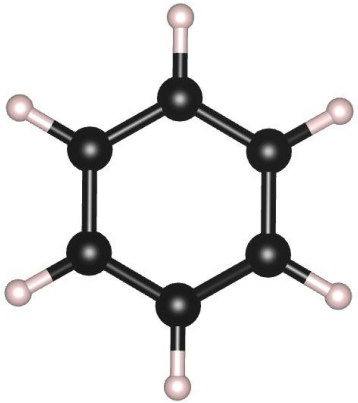
For atoms and other point set data, rather than image convolutions, we perform continuous convolutions…

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

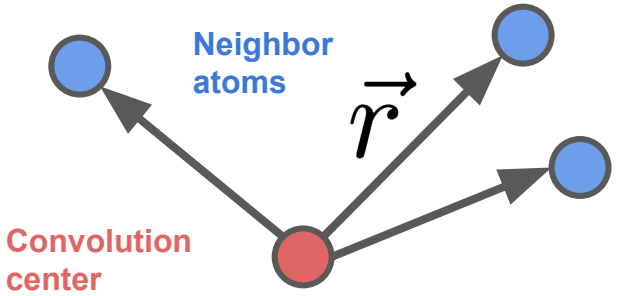
We use points. Images of atomic systems are sparse and imprecise.



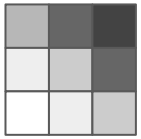
vs.



We use continuous convolutions with atoms as convolution centers.



filter



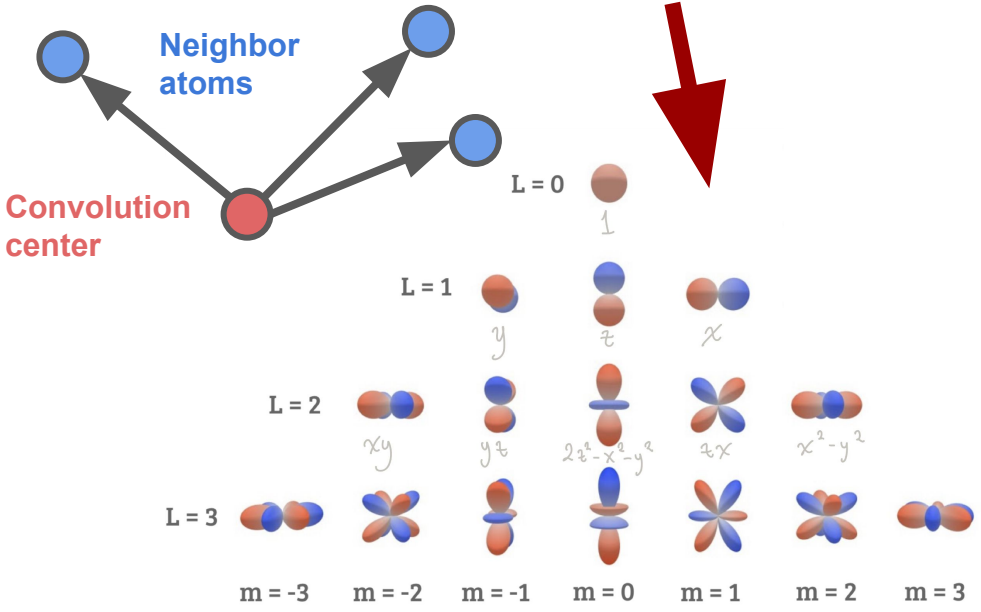
filter function

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

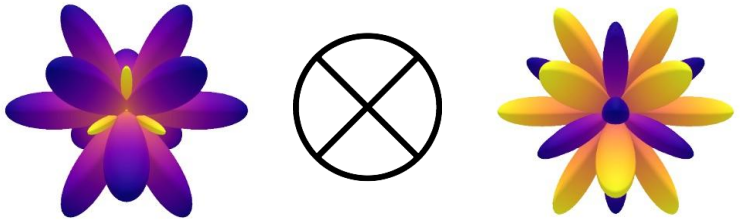
… and we require the convolutional filter to be symmetry-preserving.

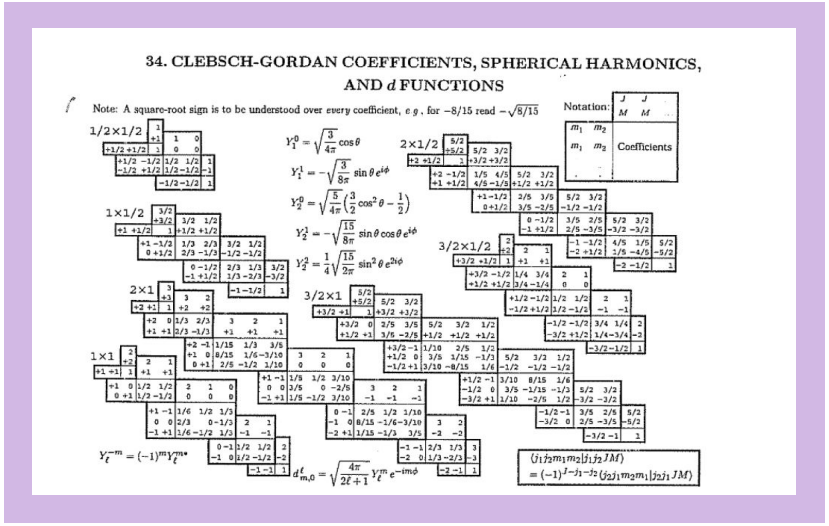
E(3) symmetry preserving convolutional filters are based on learned radial functions and spherical harmonics...

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



...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

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

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

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

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

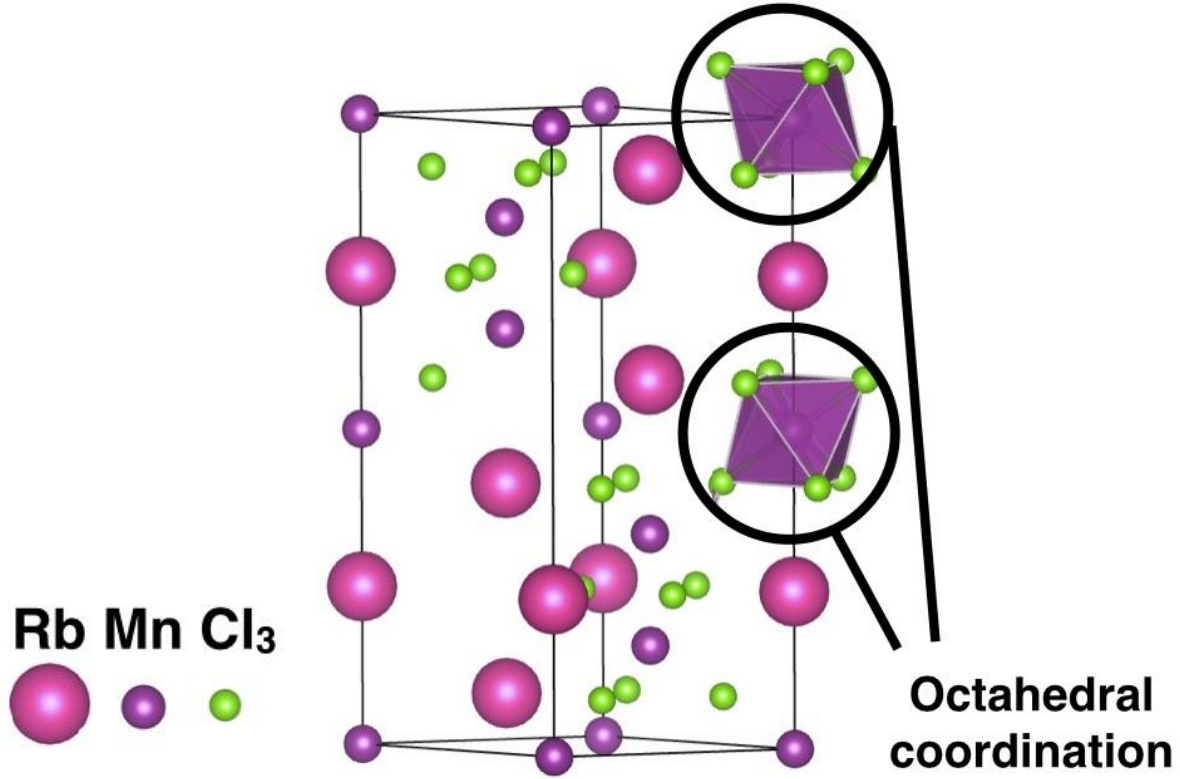
$Y_2^2 = \frac{1}{4}\sqrt{\frac{15}{2\pi}}\sin^2\theta\, e^{2i\phi}$

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

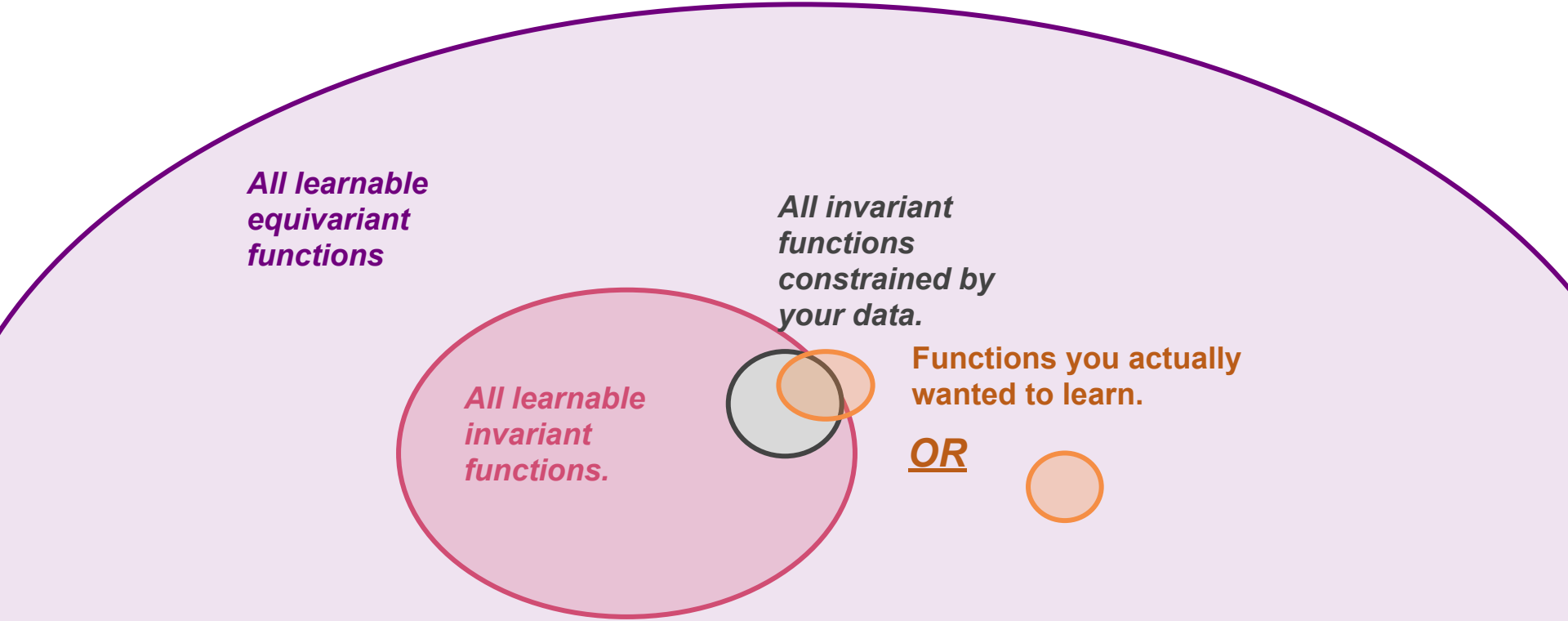
$d^\ell_{m,0} = \sqrt{\frac{4\pi}{2\ell+1}}\, Y_\ell^m\, e^{-im\phi}$

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

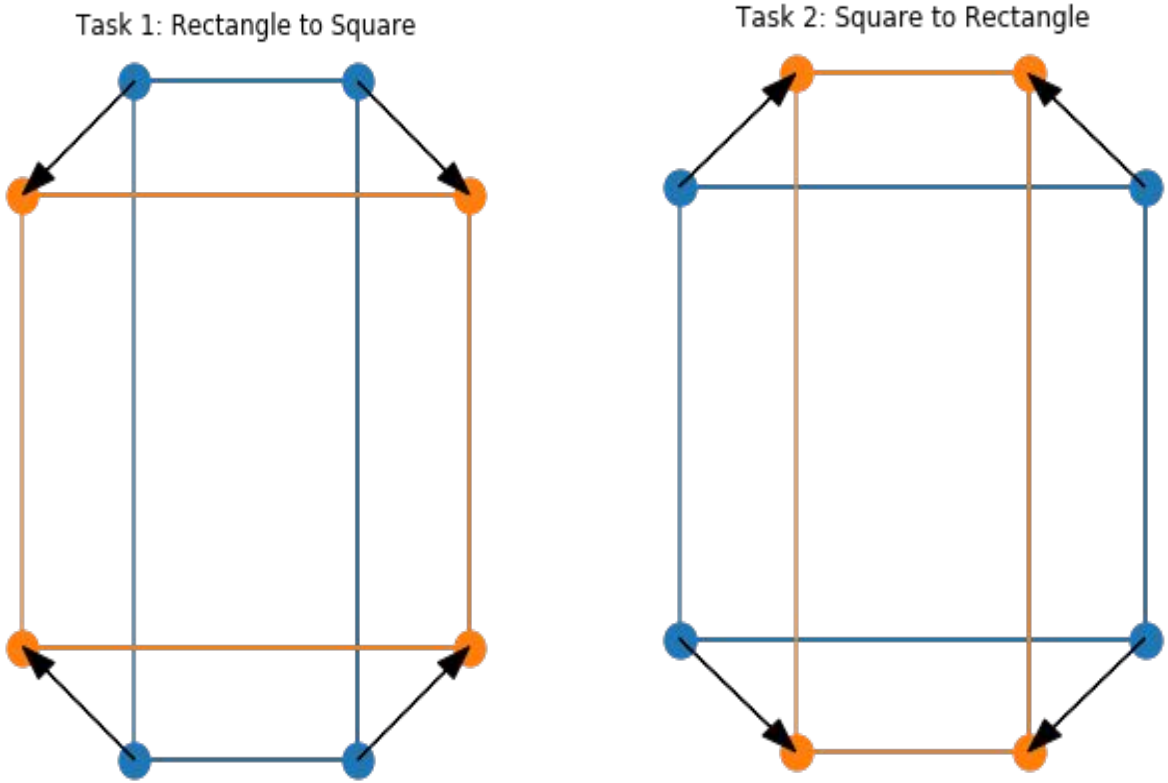
These networks can recognize equivalent recurring geometric patterns that appear in different locations and orientations (from seeing only one example)**.**



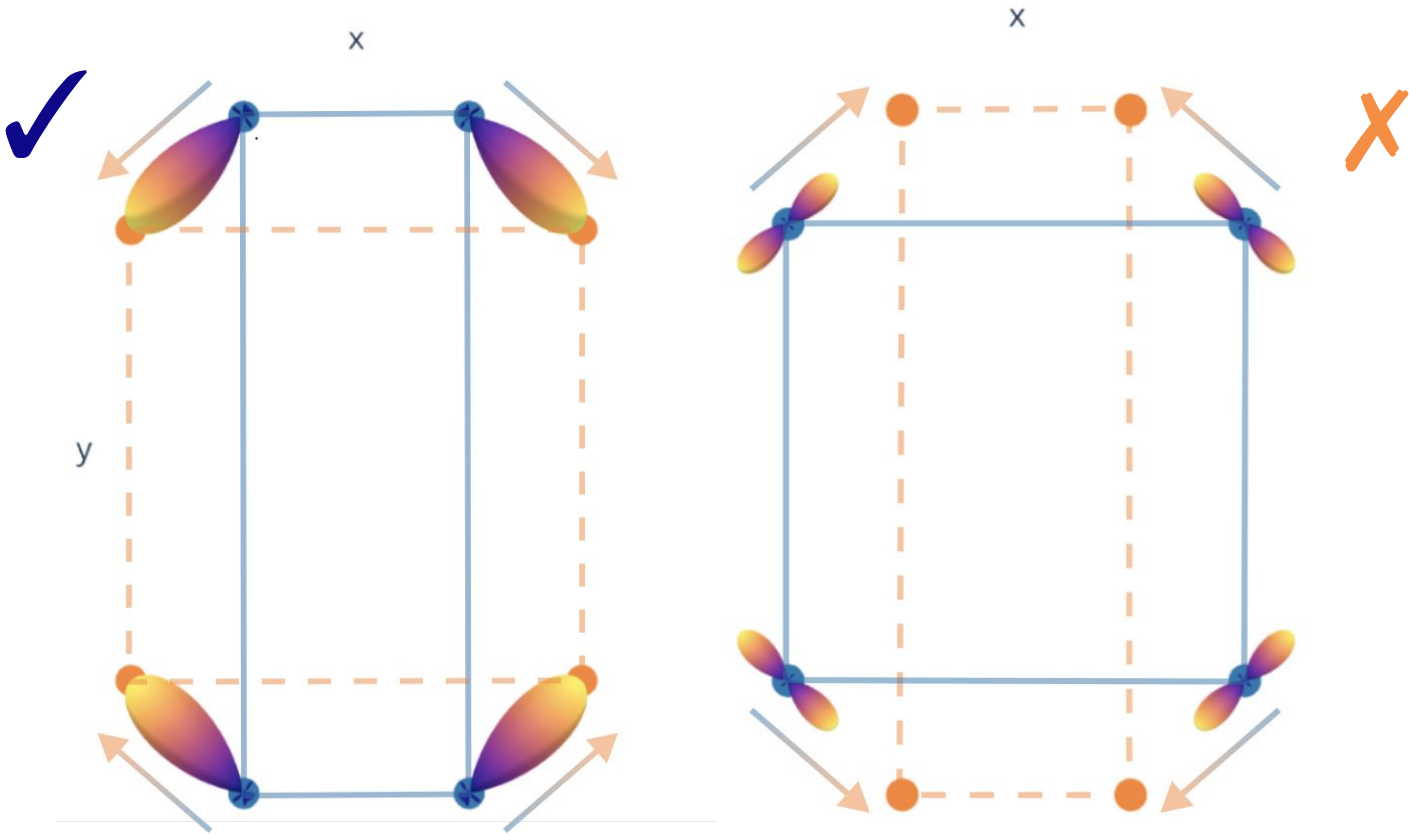
Why not limit yourself to invariant functions?
You have to guarantee that your input features already contain any necessary equivariant interactions *(e.g. cross-products)*.



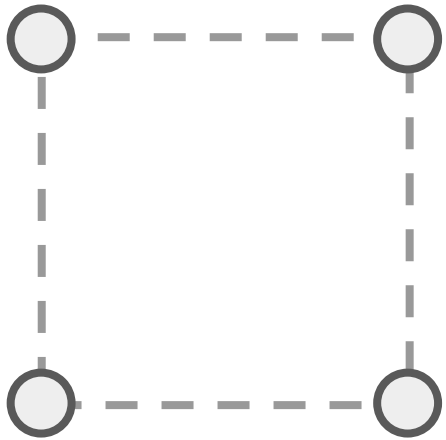
Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.

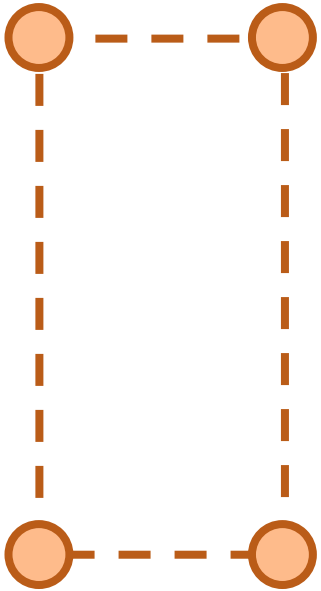


Just like the properties of physical systems,
the outputs of E(3)NNs have equal or higher symmetry than the inputs.

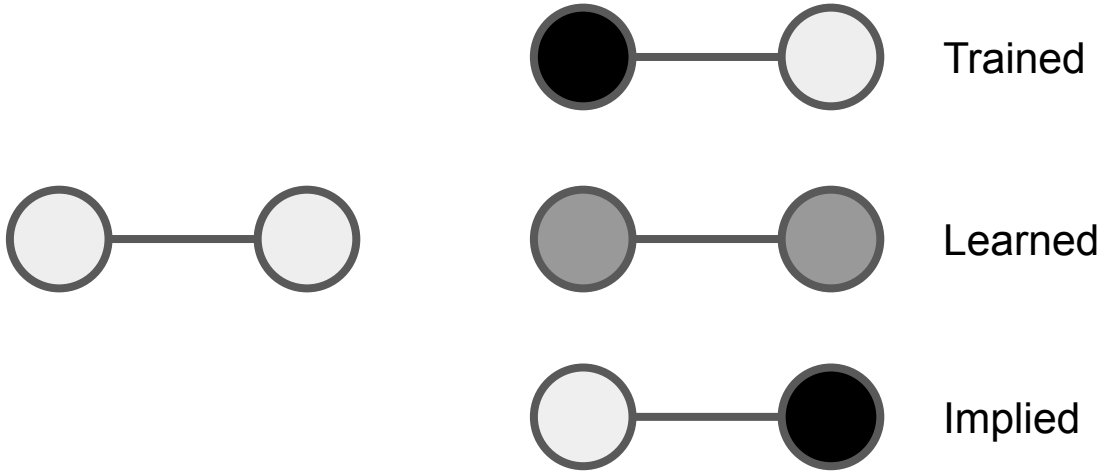


(1) Single output ⇨ Missing Information





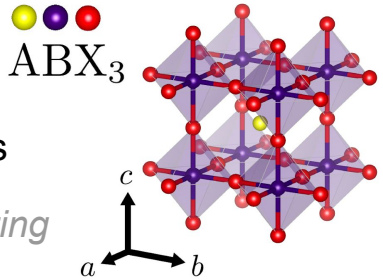
(3) Multiple outputs ⇨ Related by symmetry ⇨ Nonscalar sampling



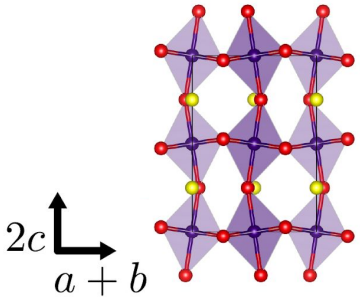
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").

Predict distortion displacements

Octahedral tilting in perovskites



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



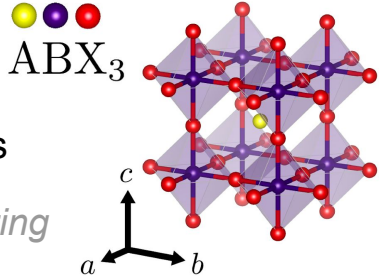
$Imma$ (74)

Target distorted structure

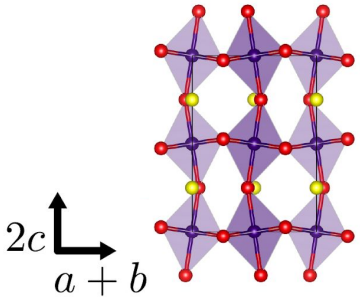
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking "order parameters").

Predict distortion displacements

Octahedral tilting in perovskites

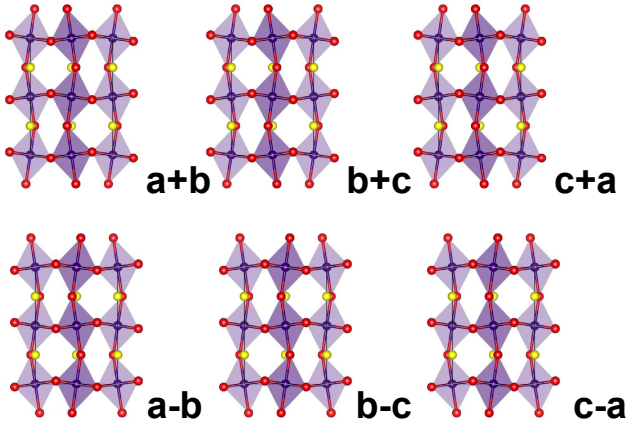


$Pm\bar{3}m$ (221)



$Imma$ (74)

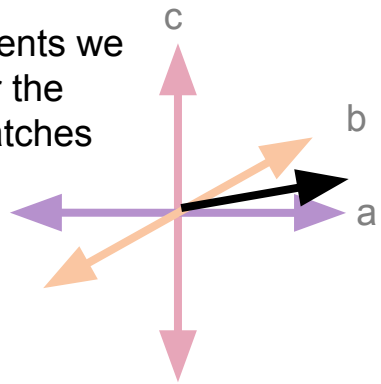
Target distorted structure



6 different vectors in reciprocal space

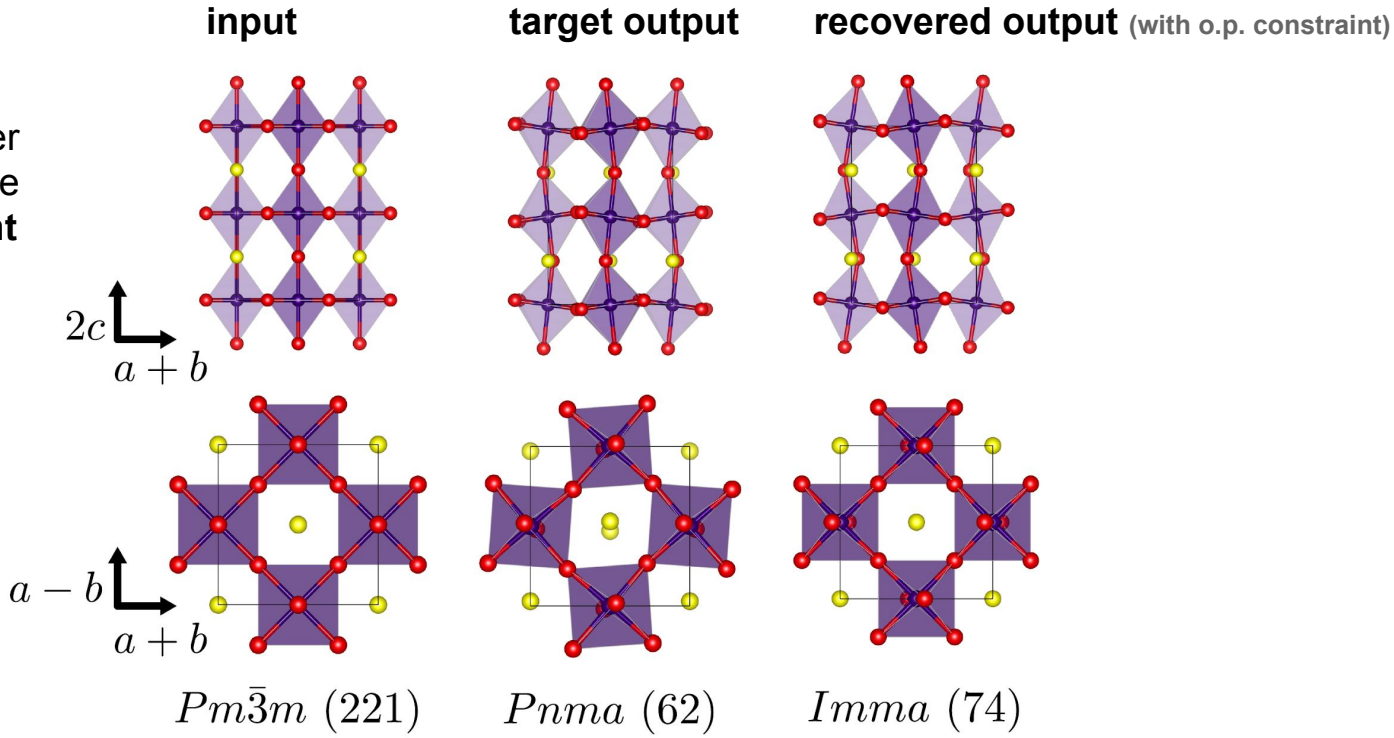
(+½, +½, 0)
(+½, -½, 0)
(0, +½, +½)
(0, +½, -½)
(+½, 0, +½)
(+½, 0, -½)

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



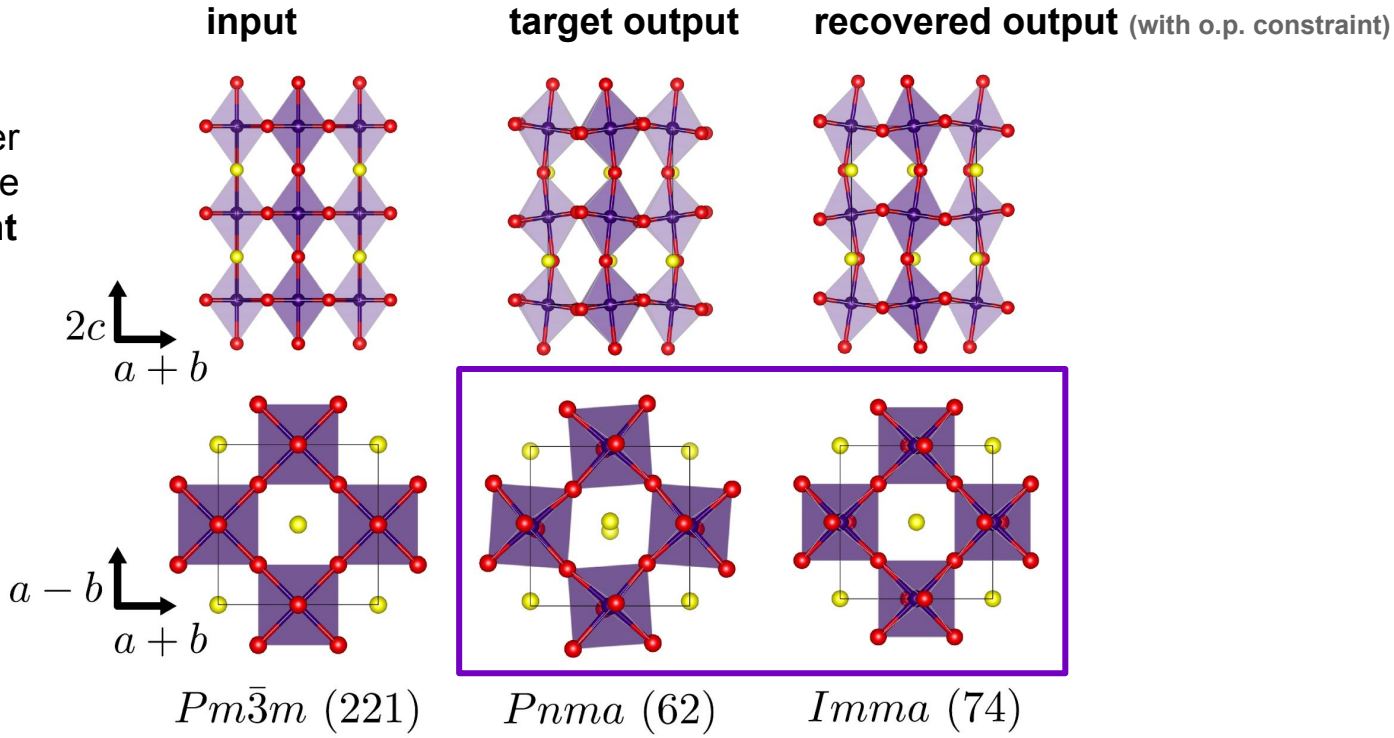
We can additionally put constraints on the "learned" order parameters to recover structures of intermediate symmetry.

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**

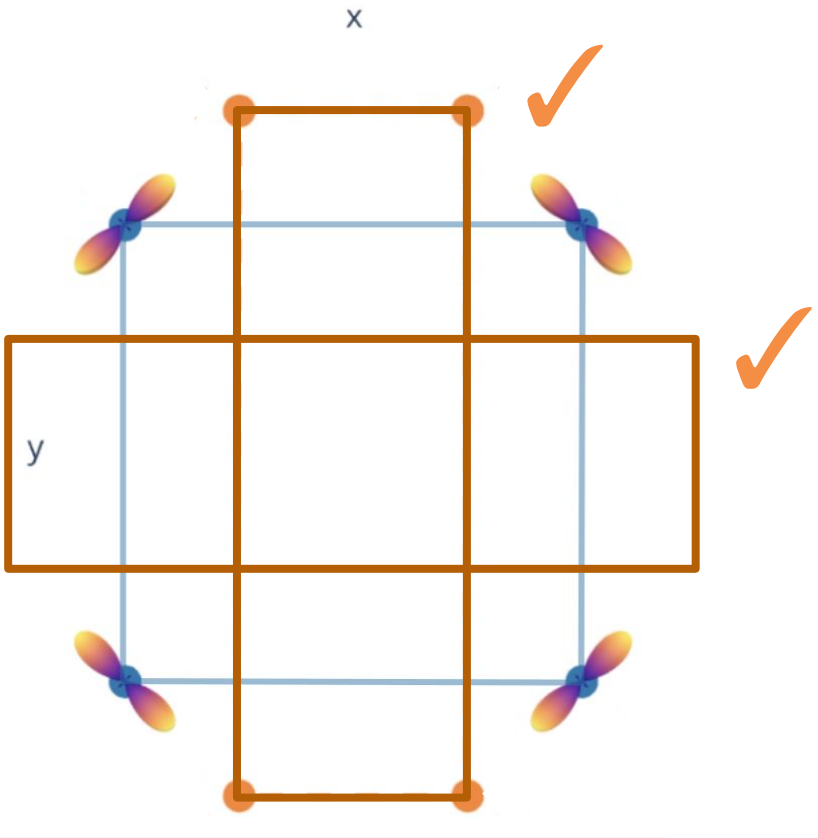


One open question:
Dealing with correlated outputs

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.**

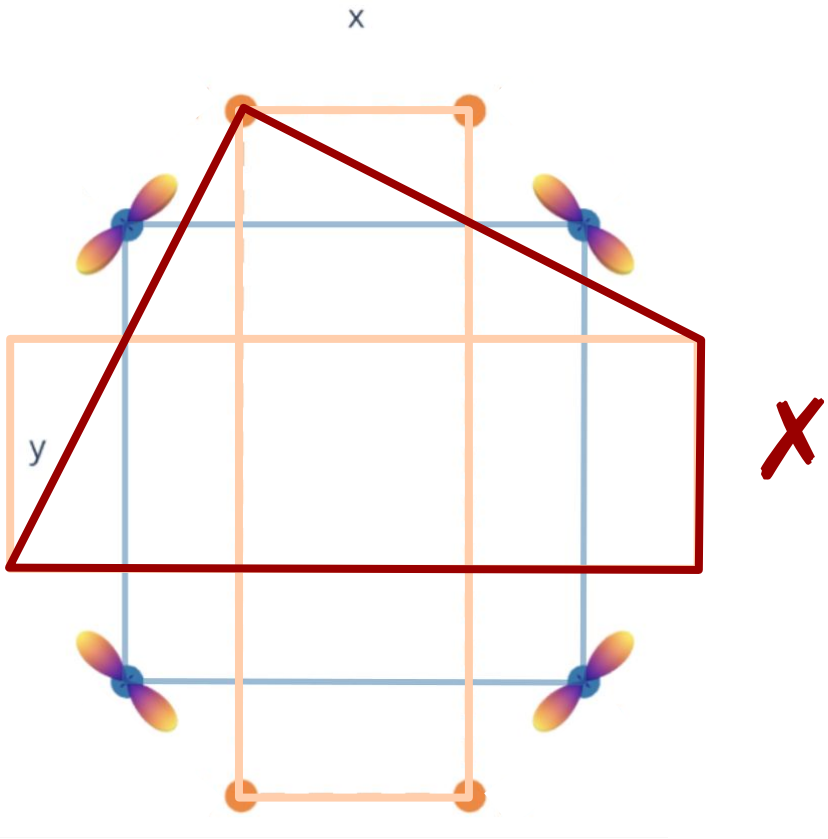


One open question:
Dealing with correlated outputs

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.**

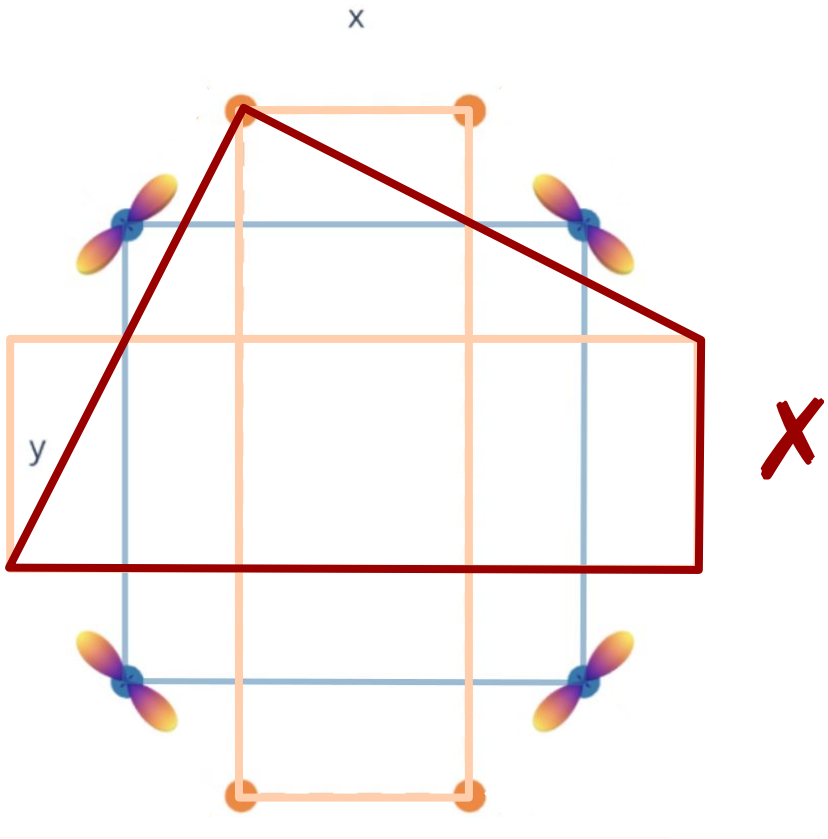


One open question:
Dealing with correlated outputs

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.**

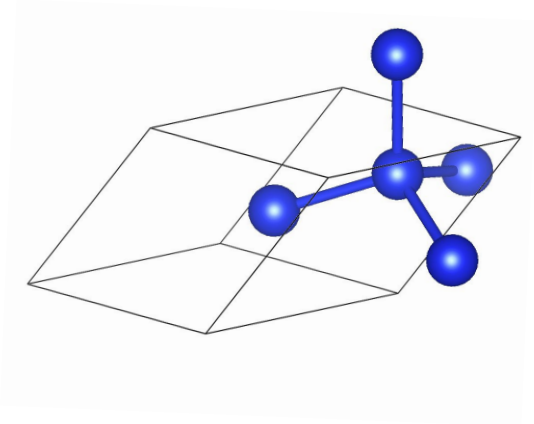


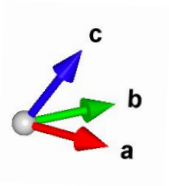
Application: Generative models / design tools for physical systems

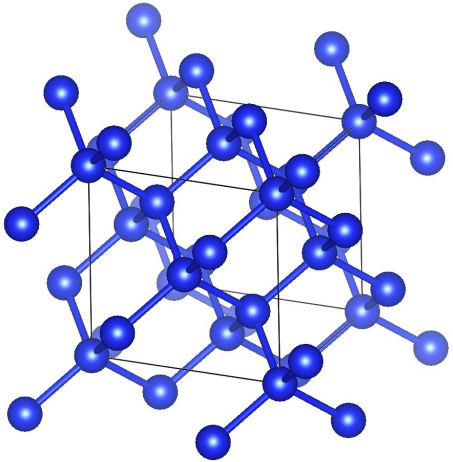
- **Lay down patterns, not just single points at a time**
- **Learn hierarchical representations of physical systems**

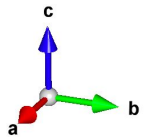
When given primitive unit cells, conventional unit cells, and supercells of the same crystal the network makes the predictions that mean the same thing.

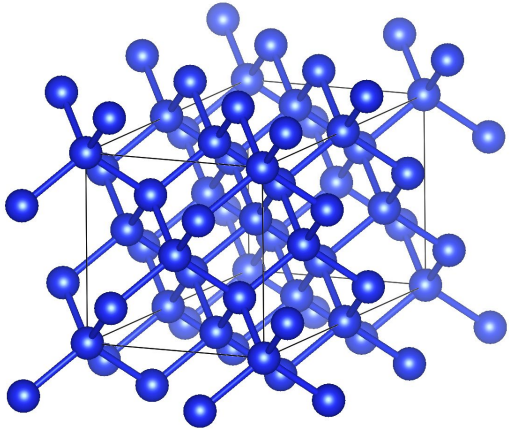
(assuming periodic boundary conditions)

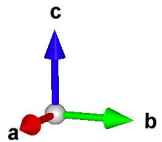




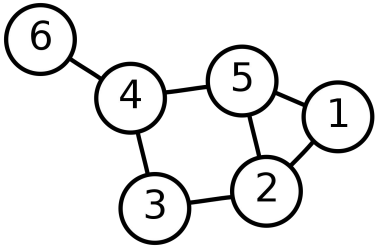
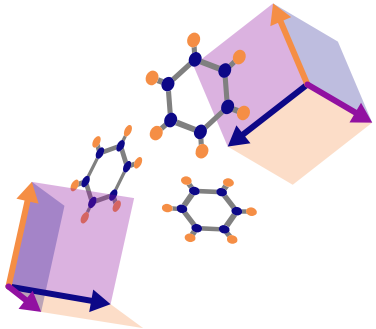






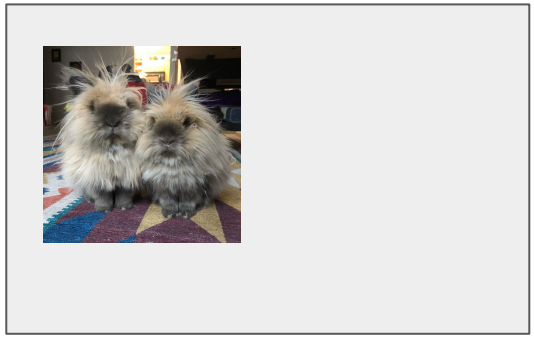


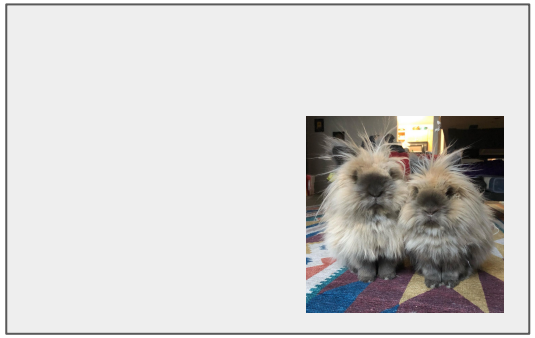
Neural networks are specially designed for different **data types** in order to make use of special features (symmetries) of the data.

Data type	Images	Text	Graph	Science data in 3D
		Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ultricies, justo ac viverra euismod, justo odio eleifend dolor, a imperdiet quam nibh finibus mauris. Morbi lobortis a lorem id dapibus. Interdum et malesuada fames...		
Type of neural network	**Convolutional** Pixels closer together are more important to each other.	**Recurrent** The meaning of a current word depends on what came before.	**Graph** Data on nodes interacts via edges	**Euclidean** Physical data "means" the same thing even when we use different coordinate systems
	Spatial translation symmetry	***Time translation symmetry***	***Permutation symmetry***	***Euclidean symmetry***

If the network can identify an instance of a *local* or *global* pattern, it is <u>guaranteed</u> to identify the pattern in any location / orientation / mirror (change of coordinate system).

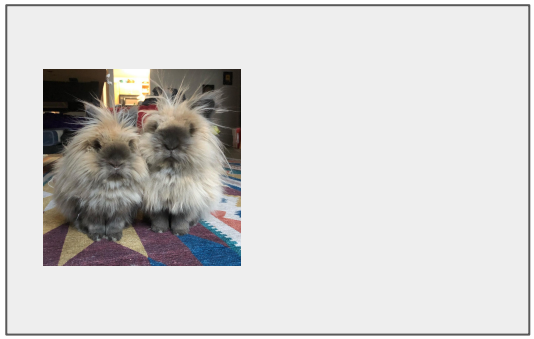
if ✓
then...

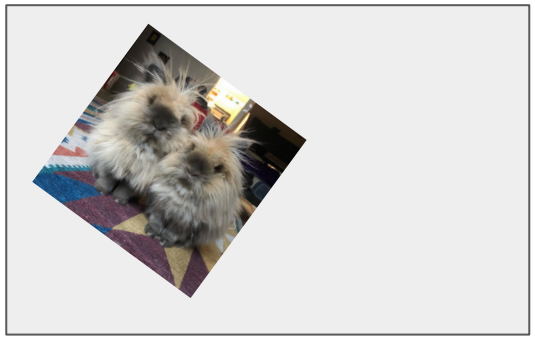




✓translations

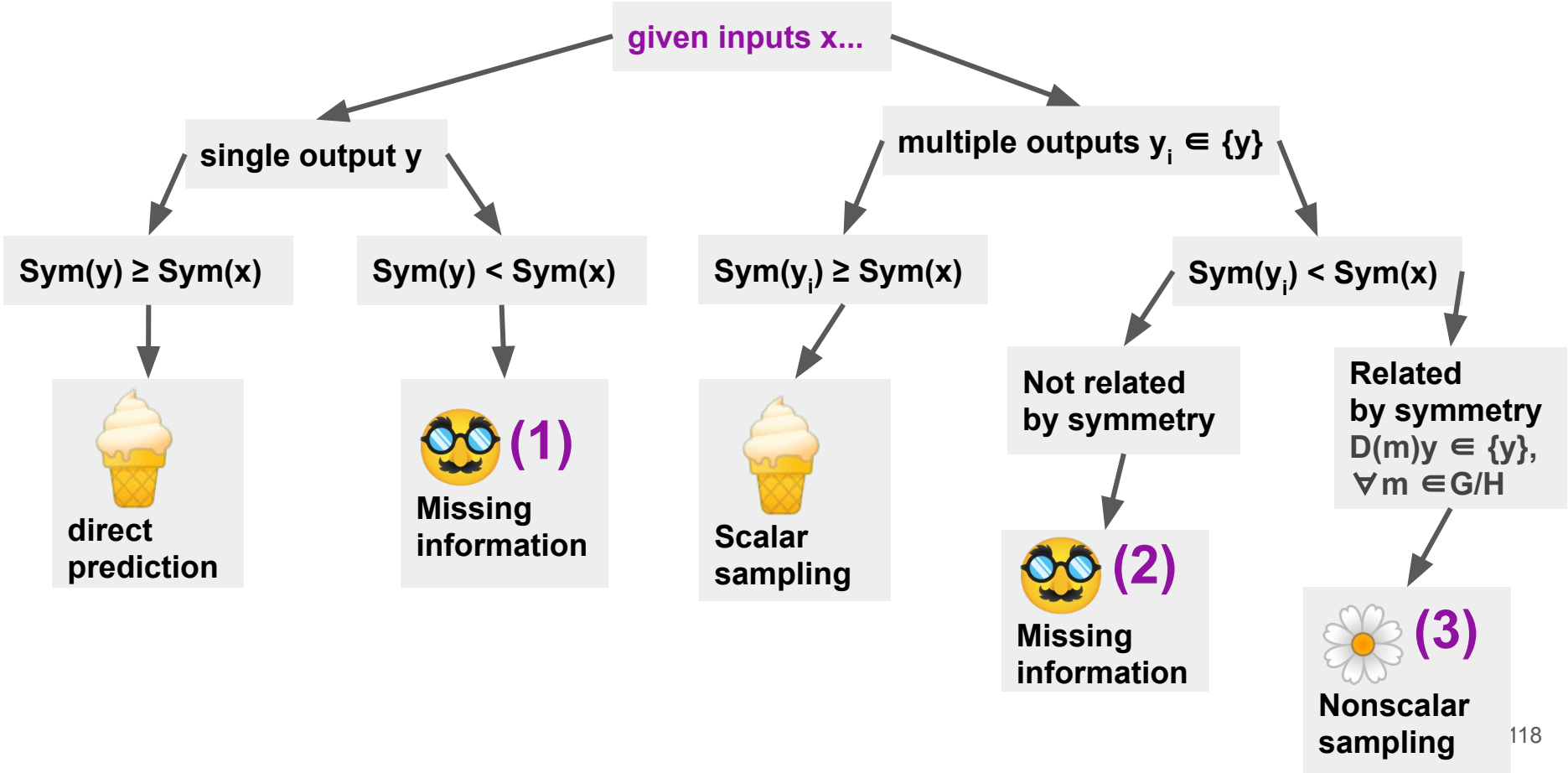
✓mirrors
(rotation + inversion)





✓rotations

When can symmetry tell you something you don't know...



nn with emojis

An introduction to machine learning with emojis

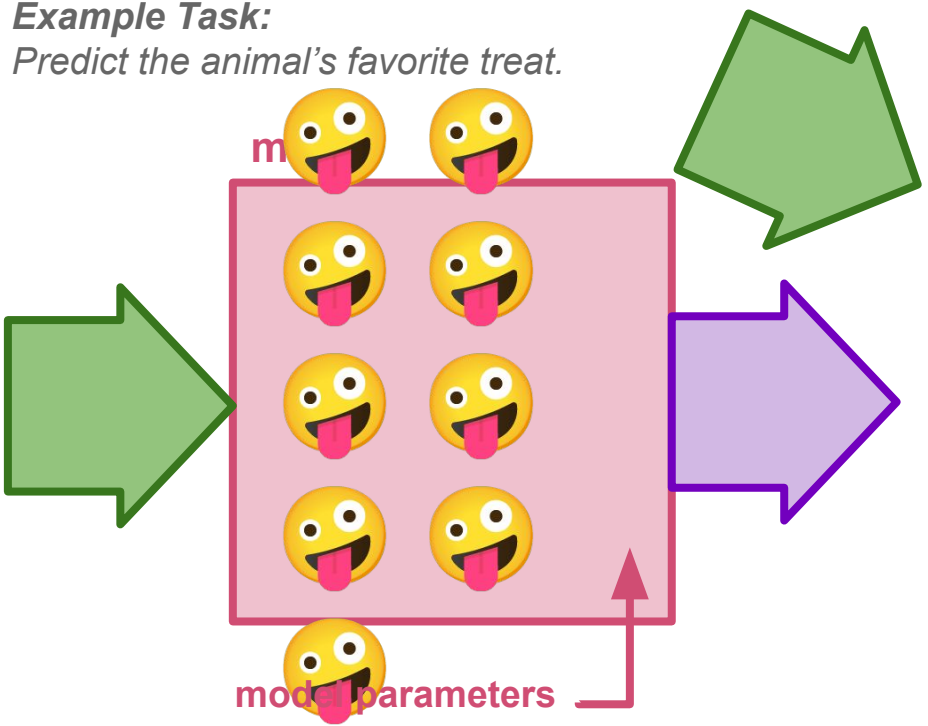
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns**
to give **predictions**
based on **examples**...

Example Task:
Predict the animal's favorite treat.



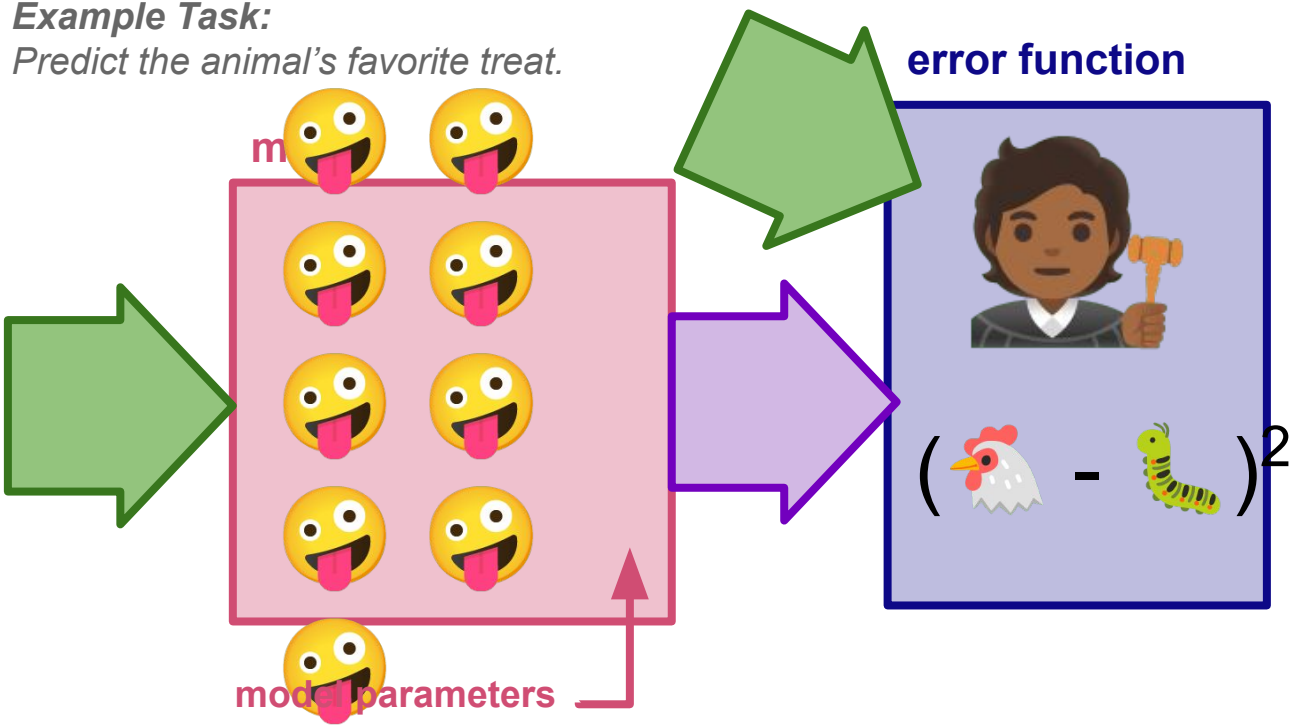
An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns**
to give **predictions**
based on **examples** and
feedback it gets during
training.

Error function evaluates
how well model is doing.

Example Task:
Predict the animal's favorite treat.



An introduction to machine learning with emojis

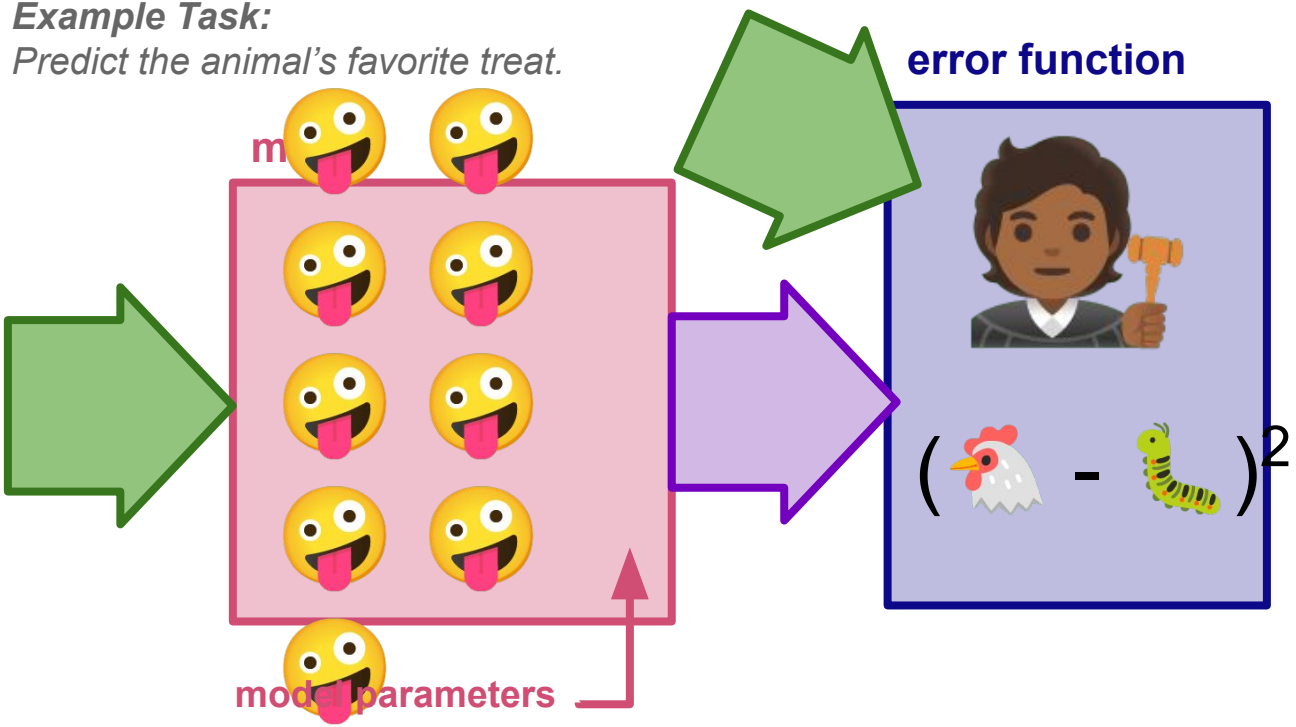
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.

Example Task:
Predict the animal's favorite treat.



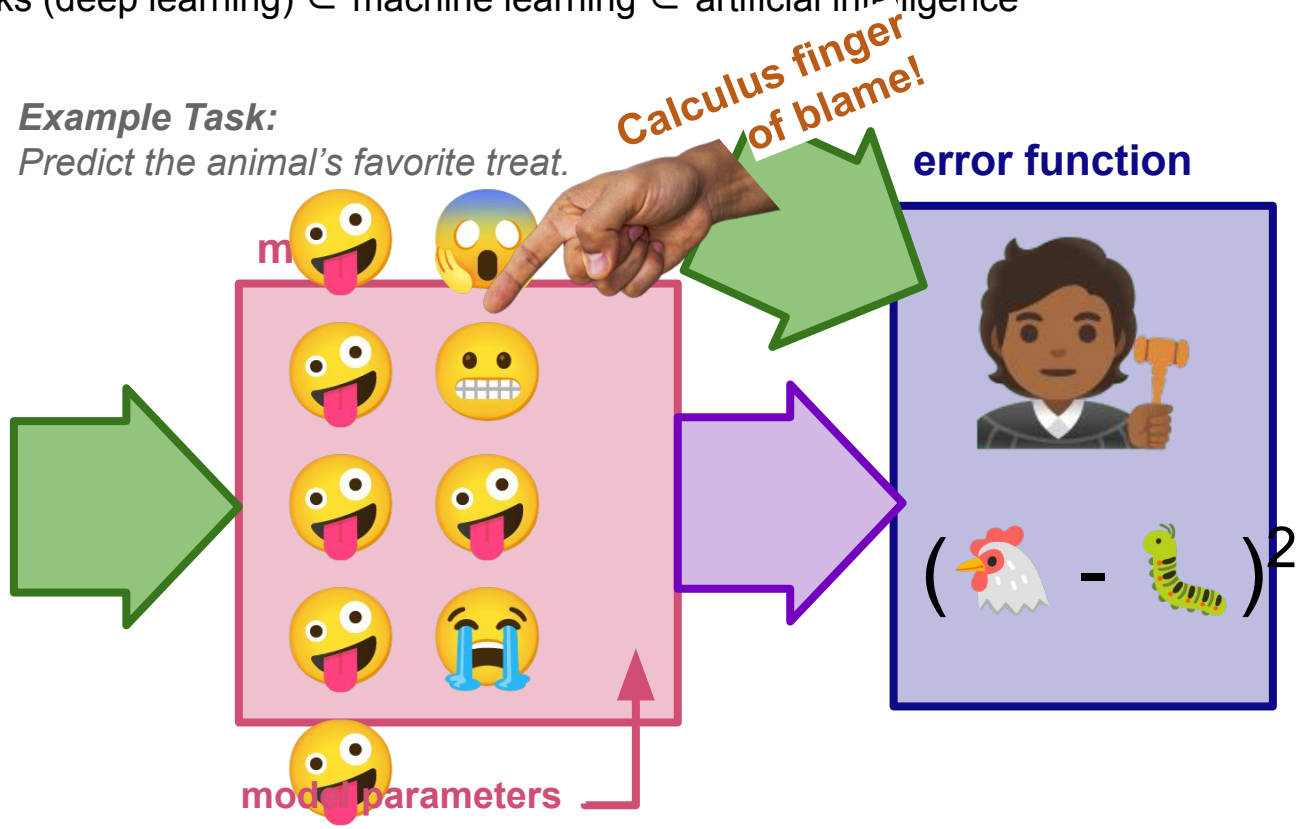
An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.



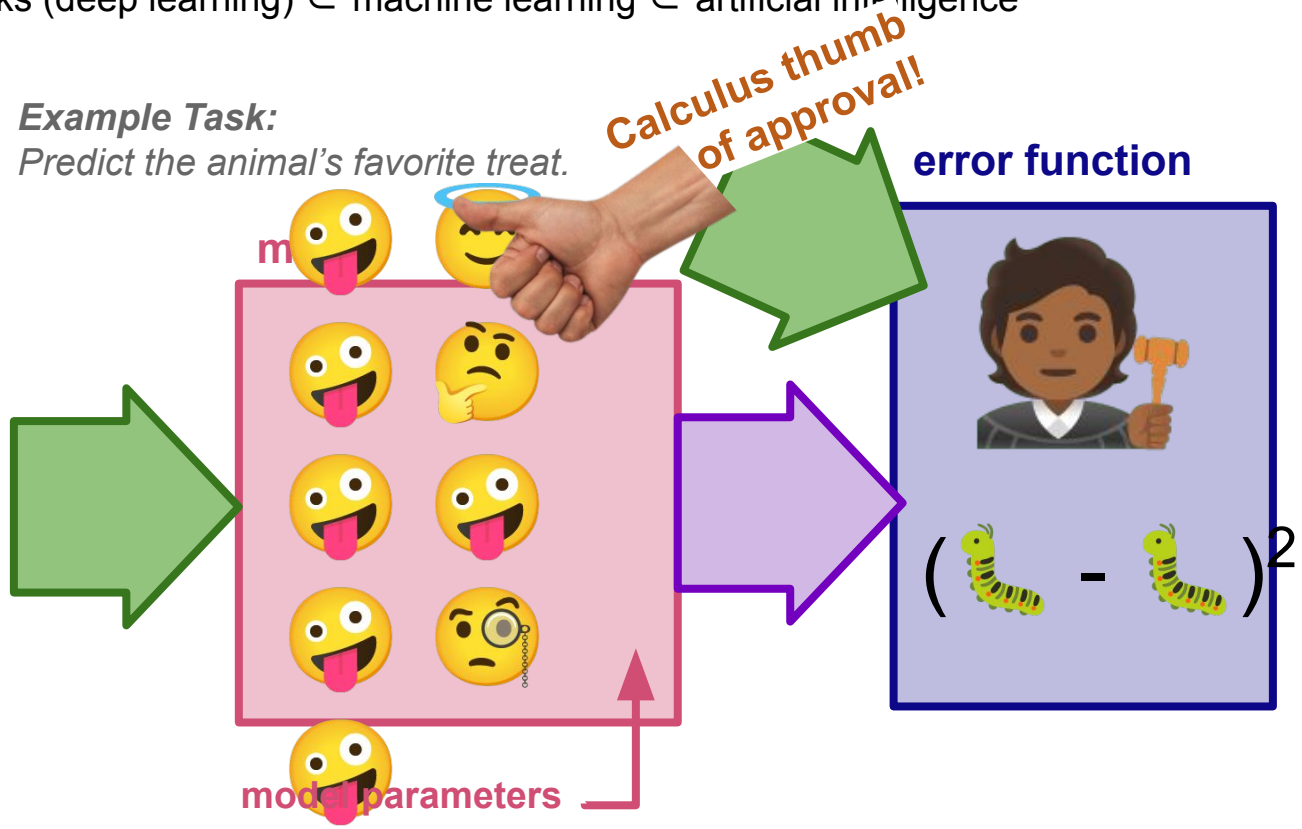
An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.



An introduction to machine learning with emojis

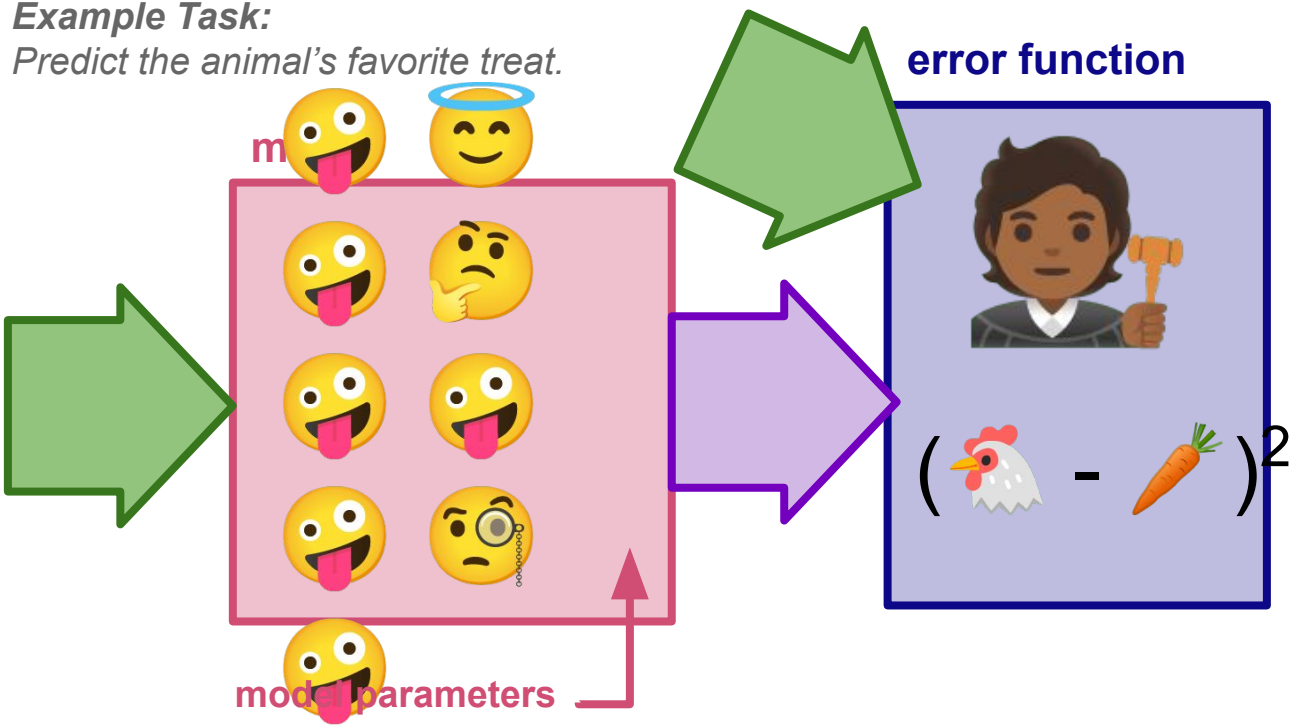
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns**
to give **predictions**
based on **examples** and
feedback it gets during
training.

Error function evaluates
how well model is doing.

Neural networks use
derivatives (calculus) to
update **model**
parameters.

Example Task:
Predict the animal's favorite treat.



An introduction to machine learning with emojis

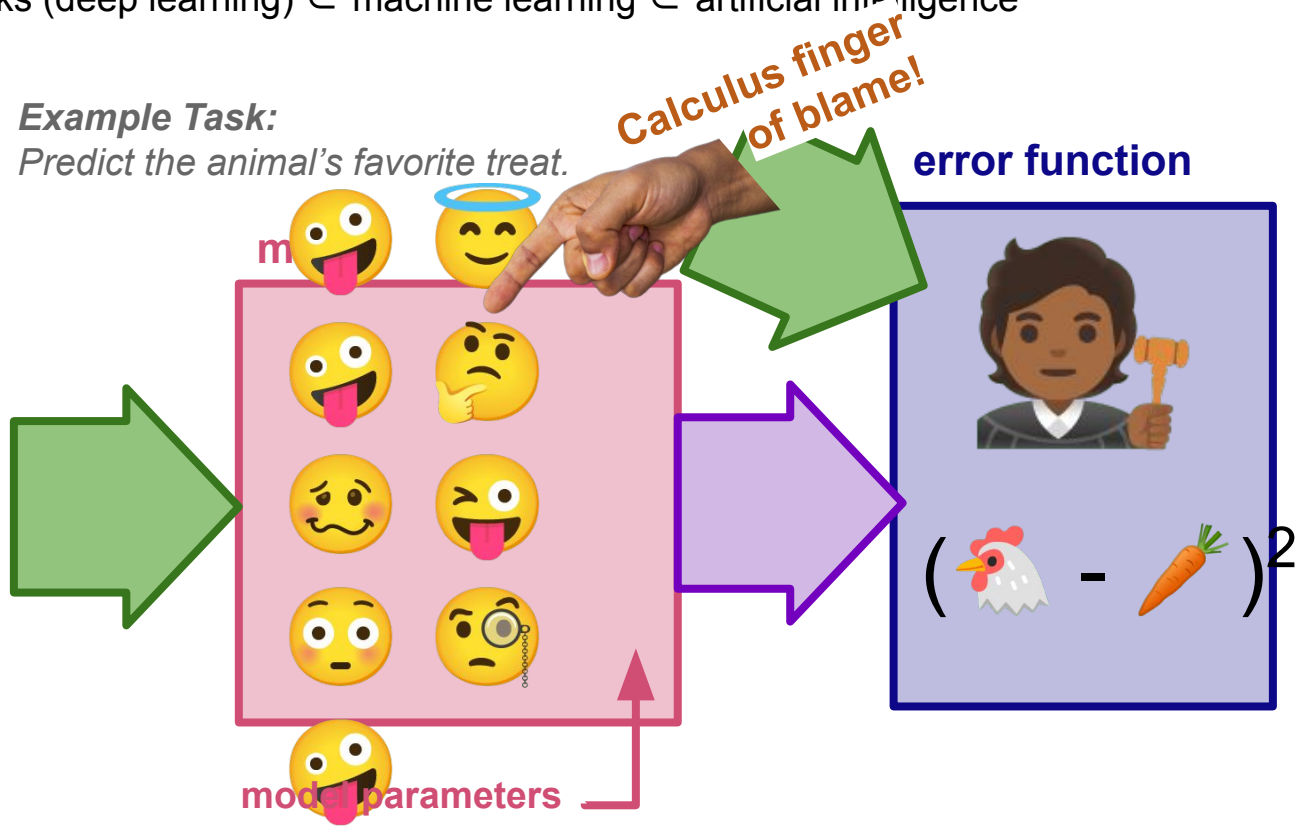
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.

Example Task:
Predict the animal's favorite treat.



An introduction to machine learning with emojis

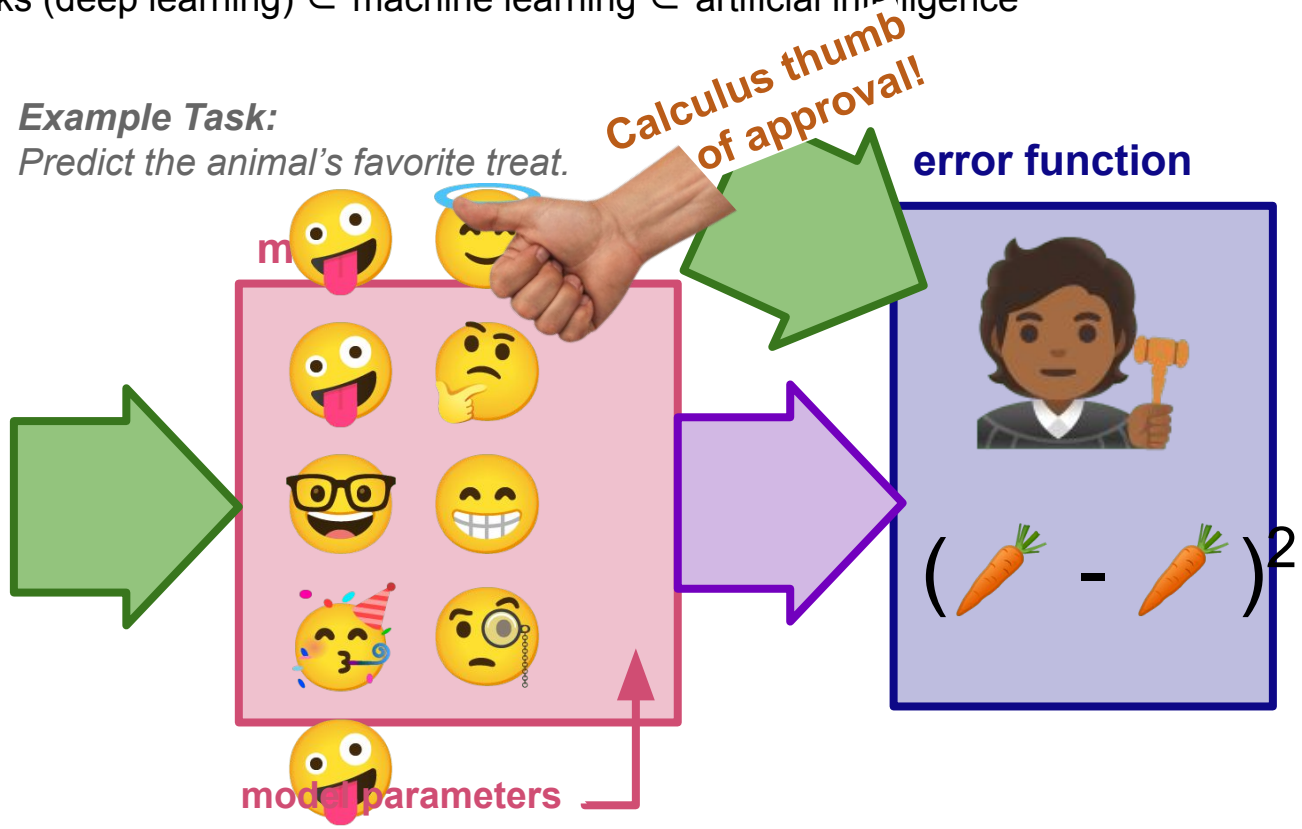
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.

Example Task:
Predict the animal's favorite treat.



An introduction to machine learning with emojis

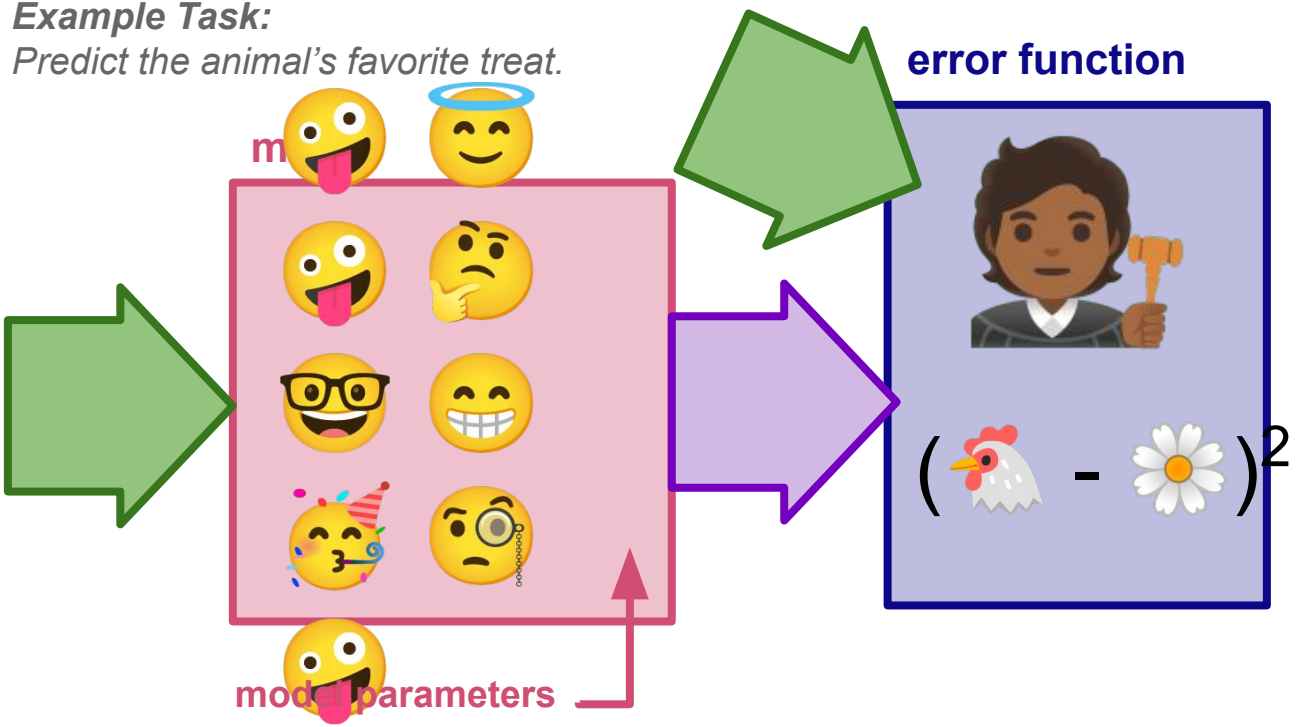
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.

Example Task:
Predict the animal's favorite treat.



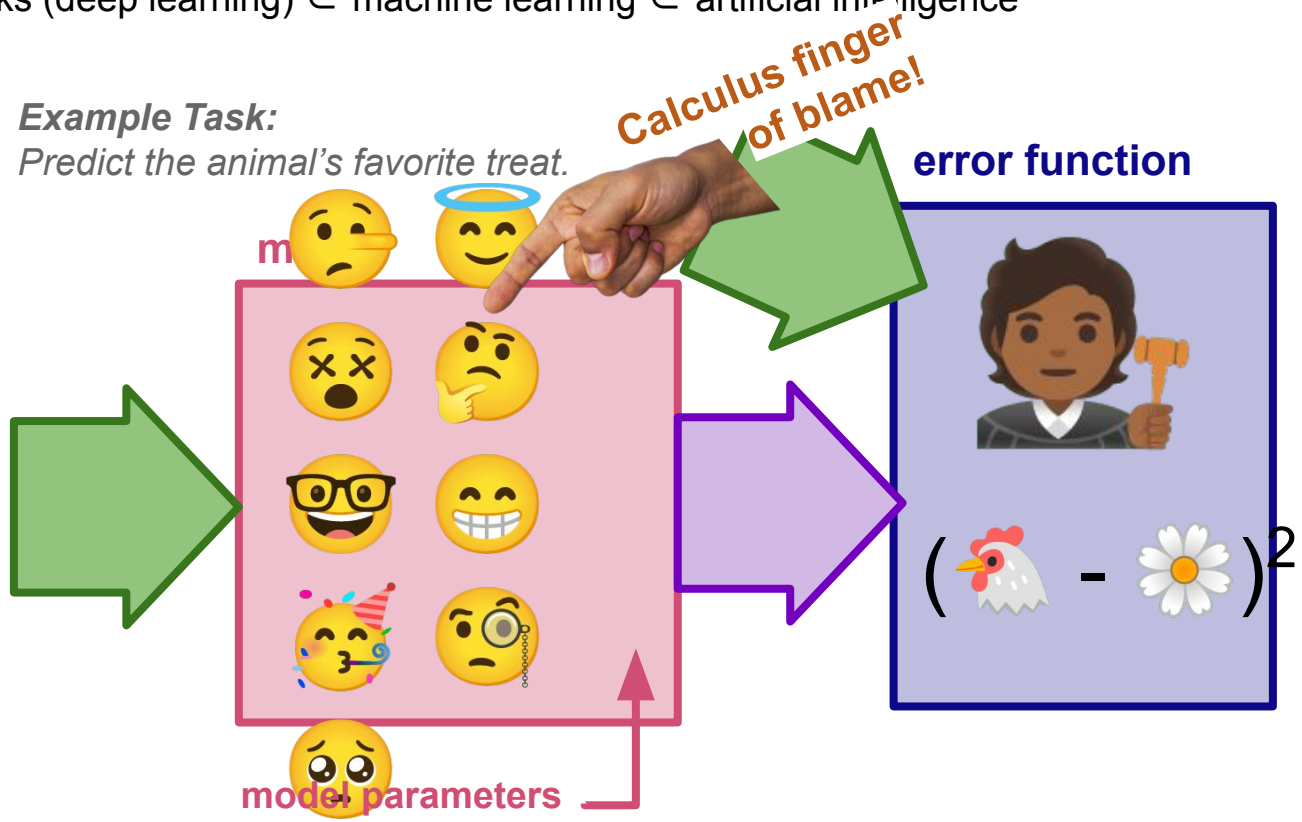
An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.



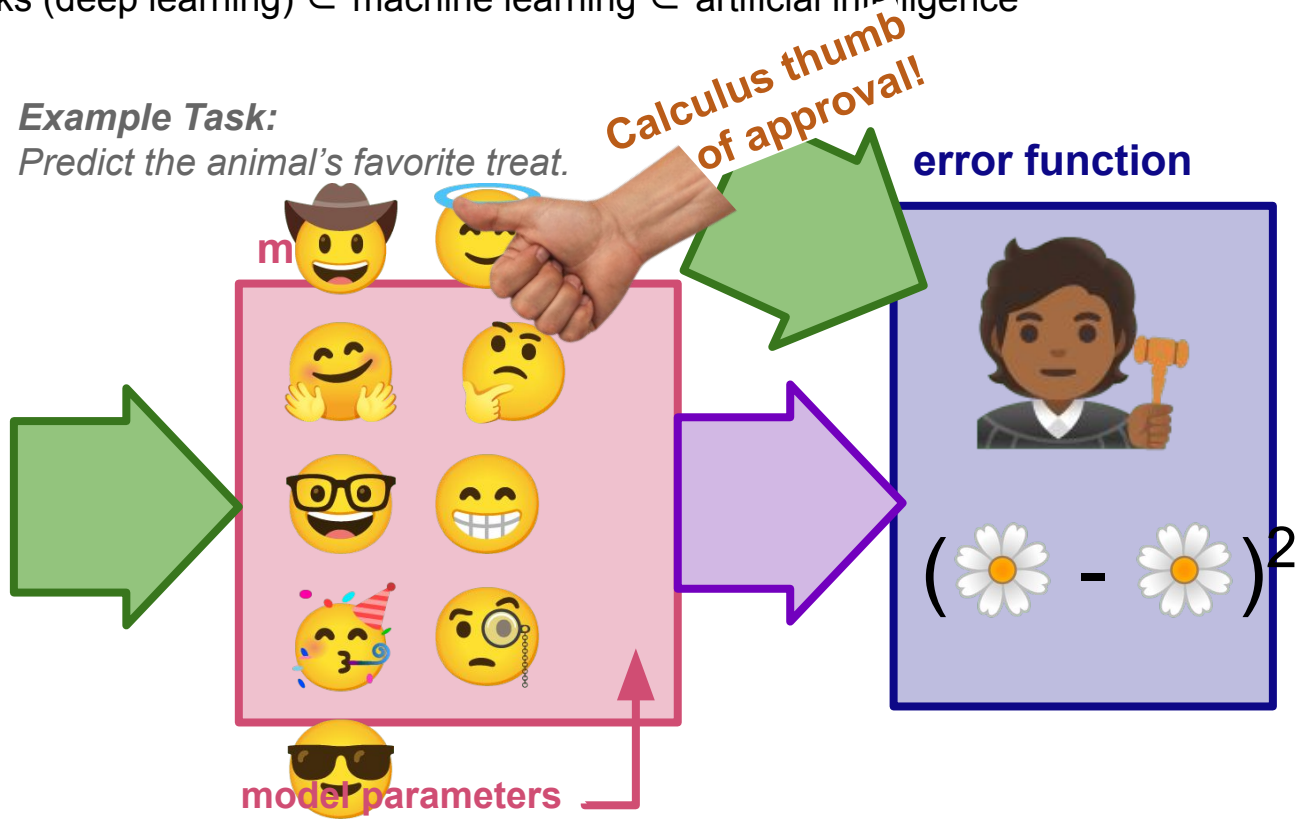
An introduction to machine learning with emojis

neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

Neural networks use **derivatives (calculus)** to update **model parameters**.



Back to main