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
  *NeguLP 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](https://github.com/e3nn)
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.
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.

\[ \square \times \square = \square \]

How do we interact **equivariant** objects? Geometric tensor products!

\[ \times \times = \]

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.

All learnable functions

All learnable $E(3)$ functions

Functions you actually wanted to learn.

All learnable functions constrained by your data.
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.
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

**Welcome!**

**Getting Started**

- How to use the Resources
- Installation

**Help**

**Contributing**

**Resources**

- Math that’s good to know
- e3nn_tutorial
- e3nn_book
- Papers
- Previous Talks
- Poster
- Slack
- Recurring Meetings / Events

**Calendar**

**e3nn Team**

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 neural networks.

Also e3nn-jax!

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

- Properties...
  - Forces
  - Energy
  - ...

- Electronic Structure...
  - Charge Density
  - Hamiltonian
  - DOS
  - ...

- Encoder / decoder partial generation. For example...

- Coarse-grain Coords.

- Fine-grain Coords
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).

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/
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)
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

(\textit{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.

Equiformer = Equivariant graph attention transformer
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
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
We’ve used E(3)NNs to build data-efficient and scalable models of physical processes.

Properties...
- Forces
- Energy
- ...

Electronic Structure...
- Charge Density
- Hamiltonian
- DOS
- ...

Encoder / decoder partial generation. For example...

Coarse-grain Coords.

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

Power law scaling exponent

\[ \beta = \text{slope} \]

\[ \beta_{\text{eq}} > \beta_{\text{inv}} \]

Architecture and task dependent offset.
This phenomena is observed across different architectures and training tasks.

<table>
<thead>
<tr>
<th>Model</th>
<th>$R^2$</th>
<th>Scaling exponent $\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchNet</td>
<td>0.95</td>
<td>0.17 ± 0.03</td>
</tr>
<tr>
<td>PaiNN</td>
<td>0.94</td>
<td>0.26 ± 0.05</td>
</tr>
<tr>
<td>Allegro</td>
<td>0.97</td>
<td>0.23 ± 0.03</td>
</tr>
</tbody>
</table>

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

\[ f(x, w) = y \]

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...

\[
\omega_i = \omega_i - \eta \frac{\partial \text{loss}}{\partial \omega_i}
\]
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.

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

$D(g)$ 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. $f(x)$)

[tsmidt@mit.edu](mailto:tsmidt@mit.edu) | T. E. Smidt. [Trends in Chemistry (2021)](https://e3nn.org) | e3nn.org
Groups can act on many types of mathematical objects. For 3D rotations, the most familiar ones are scalars 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 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.

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

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

R is scalar or vector representation.
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.

\[(x, y, z) \otimes (x, y, z) = (x^2, xy, xz, y^2, yz, z^2)\]
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.

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

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

\[L=2 \text{ spherical harmonics}\]

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”
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

\[ \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x}) \]
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

Spherical harmonic projections

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

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

Spherical harmonic projections

\[ \sum_{lm} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x}) \]
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

*Spherical harmonic projections*

\[
\sum_{lm} \sum_{ij} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x})
\]
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

**Spherical harmonic projections**

\[ \sum_{l\,m} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x}) \]

“Coefficients”

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

\[ \sum_{j} \sum_{l\,m} Y_{lm}(\hat{r}_{ij}) Y_{lm}(\hat{x}) \]
The (angular) basis of many descriptors and convolutions in E(3) equivariant networks

*Spherical harmonic projections*

**Coefficients attached to each spherical harmonic**

\[
\sum_j \sum_{l m} Y_{l m}(\hat{r}_{ij}) Y_{l m}(\hat{x})
\]
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, ...*
From spherical harmonic projections we can create invariants using tensor products. *Power spectra, bispectra, ... but in equivariant nns we can use the equivariant data.*
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]
]...

tsmidt@mit.edu | T. E. Smidt. Trends in Chemistry (2021) | e3nn.org
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 $O(3)$.

gamey = [[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")
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.

\[
\text{Sym}(x) \cong D_{4h}
\]

\[
\text{Sym}(x) \cong D_{2h}
\]
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

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!

tsmidt@mit.edu | T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021) | e3nn.org
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...

inputs x and outputs y

Sym(y) ≥ Sym(x)

single correct output

multiple correct outputs

Direct prediction

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

inputs x and outputs y

\[ \text{Sym}(y) \geq \text{Sym}(x) \]

\[ \text{Sym}(y) < \text{Sym}(x) \]

single correct output

multiple correct outputs

Direct prediction

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

inputs x and outputs y

Sym(y) ≥ Sym(x)

single correct output

Direct prediction

Scalar sampling

Sym(y) < Sym(x)

single correct output

(1)

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

inputs x and outputs y

Sym(y) ≥ Sym(x)

single correct output

Sym(y) < Sym(x)

single correct output

multiple correct outputs

multiple correct outputs

Not related by symmetry

(1) Missing information

(2) Missing information

(3) Nonscalar sampling

Related by symmetry

Direct prediction

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

inputs x and outputs y

Sym(y) ≥ Sym(x)

- single correct output
  - Direct prediction
  - Scalar sampling

Sym(y) < Sym(x)

- single correct output
  - (1) Missing information

- multiple correct outputs
  - (2) Missing information

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

- multiple correct outputs
  - Related by symmetry
    - (3) Nonscalar sampling

- Not related by symmetry
  - Missing information
When can symmetry tell you something you don't know...

inputs x and outputs y

Sym(y) ≥ Sym(x)

- single correct output
  - Direct prediction
    - (1) Missing information
  - Scalar sampling
    - (2) Missing information
- multiple correct outputs
  - Related by symmetry
  - Not related by symmetry
    - (3) Nonscalar sampling

Sym(y) < Sym(x)

multiple correct outputs
(1) $\text{Sym}(y) < \text{Sym}(x) \Rightarrow \text{Single output} \Rightarrow \text{Missing Information}$

\[
\begin{align*}
\text{Sym}(y) \geq \text{Sym}(x) & \quad \text{Task 1: Rectangle to Square} \\
\text{Sym}(y) < \text{Sym}(x) & \quad \text{Task 2: Square to Rectangle}
\end{align*}
\]
(1) $\text{Sym}(y) < \text{Sym}(x) \Rightarrow \text{Single output} \Rightarrow \text{Missing Information}$

$\text{Sym}(y) \geq \text{Sym}(x) \checkmark$

$\text{Sym}(y) < \text{Sym}(x) \times$
(1) Sym(y) < Sym(x) ⇒ Single output ⇒ Missing Information

\[ \text{Sym}(y) \geq \text{Sym}(x) \quad \checkmark \]

\[ \text{Sym}(y) < \text{Sym}(x) \quad \boldsymbol{\times} \]

Network predicts degenerate outcomes!
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?

\[ f(x, w) = y \]

Update weights using...

\[ w_i = w_i - \eta \frac{\partial \text{loss}}{\partial w_i} x \]
Using the training procedure itself, we can find data that is implied by symmetry (symmetry-breaking “order parameters”).

Use gradients to “find” what’s missing.

\[ \mathcal{X} \]

\[ \mathcal{Y} \]

L = 0

L = 0 + 2 + 4

---

\[ \text{Learns anisotropic inputs.} \quad \text{Model can fit.} \]

Irreps with even parity \( L \geq 2 \) break degeneracy between \( x \) and \( y \) directions.
When can symmetry tell you something you don't know...

inputs $x$ and outputs $y$

$\text{Sym}(y) \geq \text{Sym}(x)$

- **single correct output**
  - Direct prediction
  - Scalar sampling

- **multiple correct outputs**
  - Not related by symmetry
  - Related by symmetry

$\text{Sym}(y) < \text{Sym}(x)$

- **single correct output**
  - Missing information

- **multiple correct outputs**
  - Missing information
  - Missing information
  - Nonscalar sampling
When can symmetry tell you something you don't know...

inputs x and outputs y

Sym(y) ≥ Sym(x)

Sym(y) < Sym(x)

single correct output

multiple correct outputs

Sym(y) < Sym(x)

single correct output

multiple correct outputs

Not related by symmetry

Related by symmetry

Direct prediction

Scalar sampling

(1) Missing information

(2) Missing information

(3) Nonscalar sampling
(2) Multiple outputs $\Rightarrow$ Not related by symmetry $\Rightarrow$ Missing Information

By symmetry these outputs must be generated by different inputs. e.g. unobserved system features.
(2) Multiple outputs ⇒ Not related by symmetry ⇒ Missing Information

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...

inputs x and outputs y

Sym(y) ≥ Sym(x)

- single correct output
  - Direct prediction
  - Scalar sampling

Sym(y) < Sym(x)

- single correct output
  - (1) Missing information
- multiple correct outputs
  - Not related by symmetry
    - (2) Missing information
  - Related by symmetry
    - (3) Nonscalar sampling

multiple correct outputs
When can symmetry tell you something you don't know...

inputs x and outputs y

Sym(y) ≥ Sym(x)

single correct output

Sym(y) < Sym(x)

single correct output

multiple correct outputs

Not related by symmetry

Missing information

(1)

Related by symmetry

Missing information

(2)

multiple correct outputs

Related by symmetry

Nonscalar sampling

(3)

Direct prediction

Scalar sampling
(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

![Graph diagram](attachment:graph.png)
(3) Multiple outputs $\Rightarrow$ Related by symmetry $\Rightarrow$ Nonscalar sampling

An analogous problem to the square $\Rightarrow$ rectangle for the permutation group.

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

- **subgraph 1**
  - $[1, 0]$
- **subgraph 2**
  - $[0, 1]$

$\begin{bmatrix} 1 \end{bmatrix}$

$\begin{bmatrix} 0 \end{bmatrix}$

An analogous problem to the square $\Rightarrow$ rectangle for the permutation group.

**Trained**

- $[1, 0, 0, 1]$

**Learned**

- $[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$
(3) Multiple outputs $\Rightarrow$ Related by symmetry $\Rightarrow$ Nonscalar sampling

An analogous problem to the square $\Rightarrow$ rectangle for the permutation group.

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

![Graph with subgraphs and labels](image)

$\begin{bmatrix} 1, 0 \\ 0, 1 \end{bmatrix}$

$\begin{bmatrix} 1, 0, 0, 1 \\ \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \\ 0, 1, 1, 0 \end{bmatrix}$
(3) Multiple outputs \(\Rightarrow\) Related by symmetry \(\Rightarrow\) Nonscalar sampling

An analogous problem to the square \(\Rightarrow\) rectangle for the permutation group.

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

\[
\begin{array}{c}
\text{subgraph 1} \\
\text{subgraph 2}
\end{array}
\]

\[
\begin{bmatrix}
1, 0 \\
0, 1
\end{bmatrix}
\]

**Trained**

\[
\begin{bmatrix}
1, 0, 0, 1
\end{bmatrix}
\]

**Learned**

\[
\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}
\]

Not particular helpful. Can't recover distinct partitions.

**Implied**

\[
\begin{bmatrix}
0, 1, 1, 0
\end{bmatrix}
\]
Goal: Partition this graph into two subgraphs

$\begin{align*}
\text{Trained} & : [1, 0, 0, 1] \\
\text{Learned} & : \left[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right] \\
\text{Implied} & : [0, 1, 1, 0]
\end{align*}$

$y \otimes y = \begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1
\end{pmatrix}$

An analogous problem to the square $\Rightarrow$ rectangle for the permutation group.
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

- **Trained**
  - Subgraph 1: [1, 0]
  - Subgraph 2: [0, 1]

- **Learned**
  - Subgraph 1: \([\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]\)

- **Implied**
  - Subgraph 1: [0, 1, 1, 0]

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Does not have unique decomposition! 
... but this (often) does!
(3) Multiple outputs ⇒ Related by symmetry ⇒ Nonscalar sampling

A physical example: Structural phase transitions of crystals

Predict distortion displacements

Octahedral tilting in perovskites

ABX$_3$

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

Target distorted structure

2c

$a + b$

$Imma$ (74)

6 different vectors in reciprocal space

$(+\frac{1}{2}, +\frac{1}{2}, 0)$

$(+\frac{1}{2}, -\frac{1}{2}, 0)$

$(0, +\frac{1}{2}, +\frac{1}{2})$

$(0, +\frac{1}{2}, -\frac{1}{2})$

$(+\frac{1}{2}, 0, +\frac{1}{2})$

$(+\frac{1}{2}, 0, -\frac{1}{2})$

Want to predict all possible order parameters and sample equally!

[References and links]

tsmidt@mit.edu | T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021) | e3nn.org
When can symmetry tell you something you don't know...

inputs x and outputs y

Sym(y) ≥ Sym(x)

single correct output

Sym(y) < Sym(x)

single correct output

multiple correct outputs

Not related by symmetry

Missing information

(1)

Related by symmetry

Missing information

(2)

Multiple correct outputs

Missing information

(3)

Related by symmetry

Nonscalar sampling

Scalar sampling

Direct prediction

Nonscalar sampling

Missing information

(2)

Multiple correct outputs

Missing information

(3)

Related by symmetry

Nonscalar sampling
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.

E(3)NNs have demonstrated accuracy on a wide range of atomistic systems. QM accurate MD on 100s of millions of atoms.

ENN properties can uncover missing information and efficiently learn statistics of physical systems.
Calling in backup (slides)!
<table>
<thead>
<tr>
<th>Methods</th>
<th>Task Units</th>
<th>$\alpha$ bohr$^3$</th>
<th>$\Delta \varepsilon$ meV</th>
<th>$\varepsilon_{\text{HOMO}}$ meV</th>
<th>$\varepsilon_{\text{LUMO}}$ meV</th>
<th>$\mu$ D</th>
<th>$C_v$ cal/mol K</th>
<th>$G$ meV</th>
<th>$H$ meV</th>
<th>$R^2$ bohr$^3$</th>
<th>$U$ meV</th>
<th>$U_0$ meV</th>
<th>ZPVE meV</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMP [29]</td>
<td></td>
<td>.092</td>
<td>69</td>
<td>43</td>
<td>38</td>
<td>.030</td>
<td>.040</td>
<td>19</td>
<td>17</td>
<td>.180</td>
<td>20</td>
<td>20</td>
<td>1.50</td>
</tr>
<tr>
<td>SchNet [64]$^\dagger$</td>
<td></td>
<td>.235</td>
<td>63</td>
<td>41</td>
<td>34</td>
<td>.033</td>
<td>.033</td>
<td>14</td>
<td>14</td>
<td>.073</td>
<td>19</td>
<td>14</td>
<td>1.70</td>
</tr>
<tr>
<td>Cormorant [1]</td>
<td></td>
<td>.085</td>
<td>61</td>
<td>34</td>
<td>38</td>
<td>.038</td>
<td>.026</td>
<td>20</td>
<td>21</td>
<td>.961</td>
<td>21</td>
<td>22</td>
<td>2.03</td>
</tr>
<tr>
<td>LieConv [21]</td>
<td></td>
<td>.084</td>
<td>49</td>
<td>30</td>
<td>25</td>
<td>.032</td>
<td>.038</td>
<td>22</td>
<td>24</td>
<td>.800</td>
<td>19</td>
<td>19</td>
<td>2.28</td>
</tr>
<tr>
<td>DimeNet++ [25]$^\dagger$</td>
<td></td>
<td>.044</td>
<td>33</td>
<td>25</td>
<td>20</td>
<td>.030</td>
<td>.023</td>
<td>8</td>
<td>7</td>
<td>.331</td>
<td>6</td>
<td>6</td>
<td>1.21</td>
</tr>
<tr>
<td>TFN [72]$^\ddagger$</td>
<td></td>
<td>.223</td>
<td>58</td>
<td>40</td>
<td>38</td>
<td>.064</td>
<td>.101</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SE(3)-Transformer [23]</td>
<td></td>
<td>.142</td>
<td>53</td>
<td>35</td>
<td>33</td>
<td>.051</td>
<td>.054</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EGNN [63]</td>
<td></td>
<td>.071</td>
<td>48</td>
<td>29</td>
<td>25</td>
<td>.029</td>
<td>.031</td>
<td>12</td>
<td>12</td>
<td>.106</td>
<td>12</td>
<td>11</td>
<td>1.55</td>
</tr>
<tr>
<td>SphereNet [48]$^\dagger$</td>
<td></td>
<td>.046</td>
<td>32</td>
<td>23</td>
<td>18</td>
<td>.026</td>
<td>.021</td>
<td>8</td>
<td>6</td>
<td>.292</td>
<td>7</td>
<td>6</td>
<td>1.12</td>
</tr>
<tr>
<td>EQGAT [46]</td>
<td></td>
<td>.063</td>
<td>44</td>
<td>26</td>
<td>22</td>
<td>.014</td>
<td>.027</td>
<td>12</td>
<td>13</td>
<td>.257</td>
<td>13</td>
<td>13</td>
<td>1.50</td>
</tr>
<tr>
<td>Equiformer</td>
<td></td>
<td>.056</td>
<td>33</td>
<td>17</td>
<td>16</td>
<td>.014</td>
<td>.025</td>
<td>10</td>
<td>10</td>
<td>.227</td>
<td>11</td>
<td>10</td>
<td>1.32</td>
</tr>
</tbody>
</table>

Table 1: **MAE results on QM9 testing set.** $^\dagger$ denotes using different training, validation, testing data partitions as mentioned in SEGNN [5]. $^\ddagger$ denotes results from SE(3)-Transformer [23].
<table>
<thead>
<tr>
<th>Methods</th>
<th>ID</th>
<th>OOD Ads</th>
<th>OOD Cat</th>
<th>OOD Both</th>
<th>Average</th>
<th>ID</th>
<th>OOD Ads</th>
<th>OOD Cat</th>
<th>OOD Both</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchNet [64]†</td>
<td>0.6465</td>
<td>0.7074</td>
<td>0.6475</td>
<td>0.6626</td>
<td>0.6660</td>
<td>2.96</td>
<td>2.22</td>
<td>3.03</td>
<td>2.38</td>
<td>2.65</td>
</tr>
<tr>
<td>DimeNet++ [25]†</td>
<td>0.5636</td>
<td>0.7127</td>
<td>0.5612</td>
<td>0.6492</td>
<td>0.6217</td>
<td>4.25</td>
<td>2.48</td>
<td>4.40</td>
<td>2.56</td>
<td>3.42</td>
</tr>
<tr>
<td>GemNet-T [42]†</td>
<td>0.5561</td>
<td>0.7342</td>
<td>0.5659</td>
<td>0.6964</td>
<td>0.6382</td>
<td>4.51</td>
<td>2.24</td>
<td>4.37</td>
<td>2.38</td>
<td>3.38</td>
</tr>
<tr>
<td>SphereNet [48]</td>
<td>0.5632</td>
<td>0.6682</td>
<td>0.5590</td>
<td>0.6190</td>
<td>0.6024</td>
<td>4.56</td>
<td>2.70</td>
<td>4.59</td>
<td>2.70</td>
<td>3.64</td>
</tr>
<tr>
<td>(S)EGNN [5]</td>
<td>0.5497</td>
<td>0.6851</td>
<td>0.5519</td>
<td>0.6102</td>
<td>0.5992</td>
<td>4.99</td>
<td>2.50</td>
<td>4.71</td>
<td>2.88</td>
<td>3.77</td>
</tr>
<tr>
<td>SEGNN [5]</td>
<td>0.5310</td>
<td>0.6432</td>
<td>0.5341</td>
<td>0.5777</td>
<td>0.5715</td>
<td>5.32</td>
<td>2.80</td>
<td>4.89</td>
<td>3.09</td>
<td>4.03</td>
</tr>
<tr>
<td>Equiformer</td>
<td>0.5088</td>
<td>0.6271</td>
<td>0.5051</td>
<td>0.5545</td>
<td>0.5489</td>
<td>4.88</td>
<td>2.93</td>
<td>4.92</td>
<td>2.98</td>
<td>3.93</td>
</tr>
</tbody>
</table>

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

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

Table 3: Results on OC20 IS2RE testing set.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Energy MAE (eV) ↓</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID</td>
<td>OOD Ads</td>
<td>OOD Cat</td>
<td>OOD Both</td>
<td>Average</td>
<td>ID</td>
<td>OOD Ads</td>
<td>OOD Cat</td>
<td>OOD Both</td>
<td>Average</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNS [30]</td>
<td>0.54</td>
<td>0.65</td>
<td>0.55</td>
<td>0.59</td>
<td>0.5825</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noisy Nodes [30]</td>
<td>0.47</td>
<td>0.51</td>
<td>0.48</td>
<td>0.46</td>
<td>0.4800</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphormer [66]</td>
<td>0.4329</td>
<td>0.5850</td>
<td>0.4441</td>
<td>0.5299</td>
<td>0.4980</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equiformer</td>
<td>0.4222</td>
<td>0.5420</td>
<td>0.4231</td>
<td>0.4754</td>
<td>0.4657</td>
<td>7.23</td>
<td>3.77</td>
<td>7.13</td>
<td>4.10</td>
<td>5.56</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Noisy Nodes</td>
<td>0.4156</td>
<td>0.4976</td>
<td>0.4165</td>
<td>0.4344</td>
<td>0.4410</td>
<td>7.47</td>
<td>4.64</td>
<td>7.19</td>
<td>4.84</td>
<td>6.04</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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].

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

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.

\[
\text{geometry} = [[[x_0, y_0, z_0], [x_1, y_1, z_1]]
\]

\[
\text{features} = [
\begin{array}{c}
[ m_0, v_{0y}, v_{0z}, v_{0x}, a_{0y}, a_{0z}, a_{0x}] \\
[ m_1, v_{1y}, v_{1z}, v_{1x}, a_{1y}, a_{1z}, a_{1x}] \\
\end{array}
\]
\]

...
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 $O(3)$.

g = \begin{bmatrix} [x_0, y_0, z_0], [x_1, y_1, z_1] \end{bmatrix}

f = \begin{bmatrix}
  \begin{bmatrix}
    m_0, v_{0y}, v_{0z}, v_{0x}, a_{0y}, a_{0z}, a_{0x} \\
    m_1, v_{1y}, v_{1z}, v_{1x}, a_{1y}, a_{1z}, a_{1x}
  \end{bmatrix}
\end{bmatrix}

s = \text{e3nn.o3.Irrep("0e")} \quad \# \text{L=0, even}

v = \text{e3nn.o3.Irrep("1o")} \quad \# \text{L=1, odd}

irreps = 1 * s + 1 * v + 1 * v

---

tsmidt@mit.edu | T. E. Smidt. Trends in Chemistry (2021) | e3nn.org
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 $O(3)$.

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

```
geometry = [[x0, y0, z0],[x1, y1, z1]]
geatures = [
    [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)

Spherical harmonics

\[ Y^m_l \]

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")
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.

We use continuous convolutions with atoms as convolution centers.
... 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...

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

\[ W(\vec{r}) = R(r) Y_{l}^{m}(\hat{r}) \]
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 $\Rightarrow$ Related by symmetry $\Rightarrow$ Nonscalar sampling

- Trained
- Learned
- Implied
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

Predicted

ABX₃

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

Target

distorted
structure

$Imma$ (74)
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)

6 different vectors in reciprocal space

\(+\frac{1}{2}, +\frac{1}{2}, 0\)
\(+\frac{1}{2}, -\frac{1}{2}, 0\)
\(0, +\frac{1}{2}, +\frac{1}{2}\)
\(0, +\frac{1}{2}, -\frac{1}{2}\)
\(+\frac{1}{2}, 0, +\frac{1}{2}\)
\(+\frac{1}{2}, 0, -\frac{1}{2}\)

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

tsmidt@mit.edu | T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021) | e3nn.org
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.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Images</th>
<th>Text</th>
<th>Graph</th>
<th>Science data in 3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional</td>
<td><img src="image" alt="Convolutional image" /></td>
<td><strong>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...</strong></td>
<td><img src="image" alt="Graph" /></td>
<td><strong>Euclidean</strong>&lt;br&gt;Physical data “means” the same thing even when we use different coordinate systems</td>
</tr>
<tr>
<td>Recurrent</td>
<td></td>
<td><strong>Graph</strong>&lt;br&gt;Data on nodes interacts via edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spatial translation symmetry</td>
<td>Recurrent&lt;br&gt;The meaning of a current word depends on what came before.</td>
<td>Graph&lt;br&gt;Data on nodes interacts via edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Euclidean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Euclidean symmetry</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

if ✓ then...

✓ translations

✓ mirrors (rotation + inversion)

✓ rotations

tsmidt@mit.edu | T. E. Smidt. Trends in Chemistry (2021) | e3nn.org
When can symmetry tell you something you don't know...

given inputs $x$...

single output $y$

- $\text{Sym}(y) \geq \text{Sym}(x)$
  - direct prediction

- $\text{Sym}(y) < \text{Sym}(x)$
  - Missing information

multiple outputs $y_i \in \{y\}$

- $\text{Sym}(y_i) \geq \text{Sym}(x)$
  - Scalar sampling
    - (2) Missing information

- $\text{Sym}(y_i) < \text{Sym}(x)$
  - Not related by symmetry
    - Related by symmetry $D(m)y \in \{y\}, \forall m \in G/H$

- $\text{Sym}(y) \geq \text{Sym}(x)$
  - (1) Missing information

- $\text{Sym}(y) < \text{Sym}(x)$
  - Missing information

- $\text{Sym}(y) \geq \text{Sym}(x)$
  - (3) Nonscalar sampling
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.*

```latex
\text{error function} = (\text{model}(\text{animal}) - \text{true preference})^2
```
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.

Error function

```
(🐔 - 🐒)^2
```
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.

Calculus finger of blame!

(error function)

((chicken - worm)^2)
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.

\[
\text{error function} = (\text{model parameters} - \text{true parameters})^2
\]
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.

$\text{(\text{chicken} - \text{carrot})}^2$
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.*

```
(किरदार - तराई)²
```
An introduction to machine learning with emojis

**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.

\[(\text{carrot} - \text{carrot})^2\]
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.

`error function = (\text{model output} - \text{actual output})^2`
An introduction to machine learning with emojis

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.

$$((\text{chicken} - \text{daisy})^2$$

error function

Calculus finger of blame!
An introduction to machine learning with emojis

**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.

(needs example task with data)

Calculus finger of blame!

Calculus thumb of approval!

error function

$(\frac{\text{flower}}{-\text{flower}})^2$