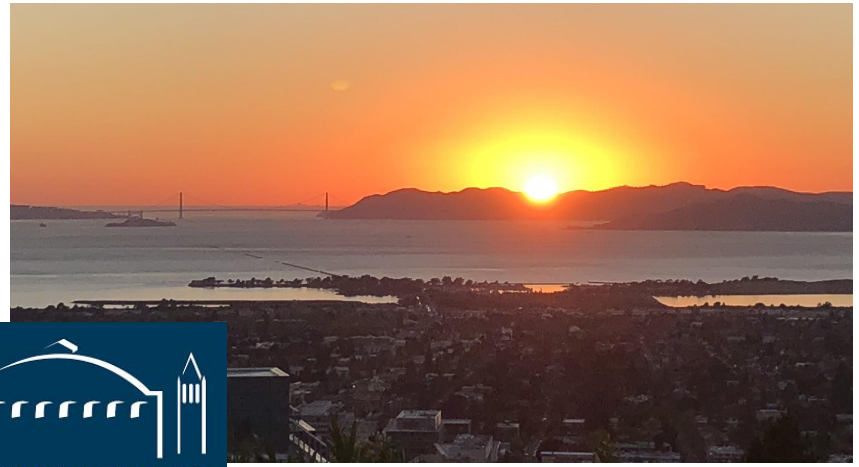
Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
2018 Luis W. Alvarez Fellow in Computing Sciences

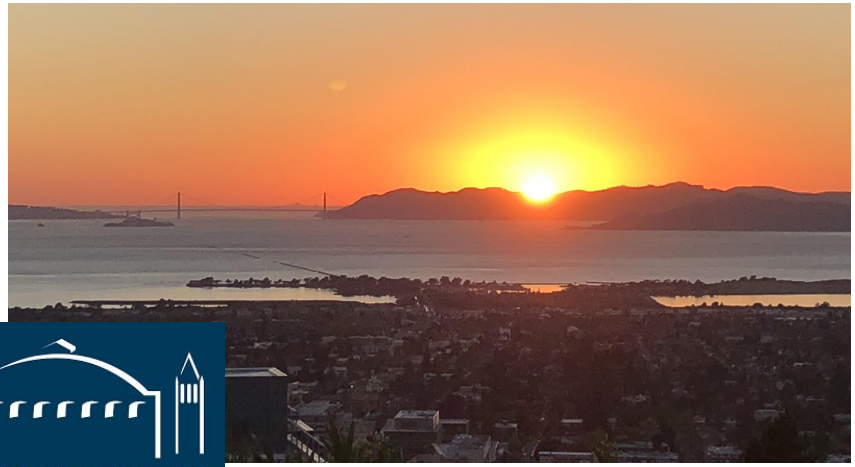


Slides: https://tinyurl.com/tess-cssp-2021

CSA Summer Series
2021.06.29

Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*

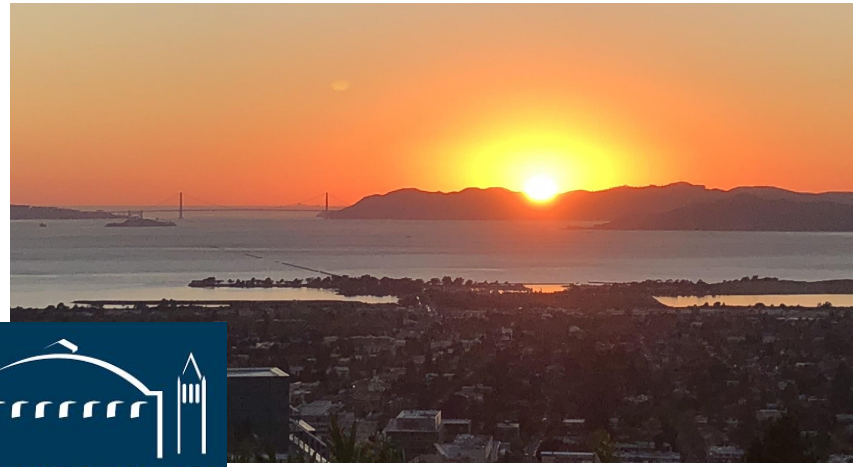


BERKELEY LAB

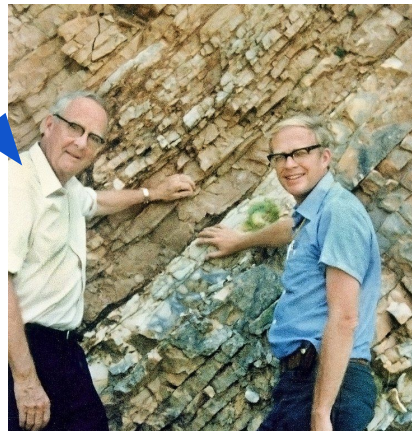
CSA Summer Series
2021.06.29

Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*

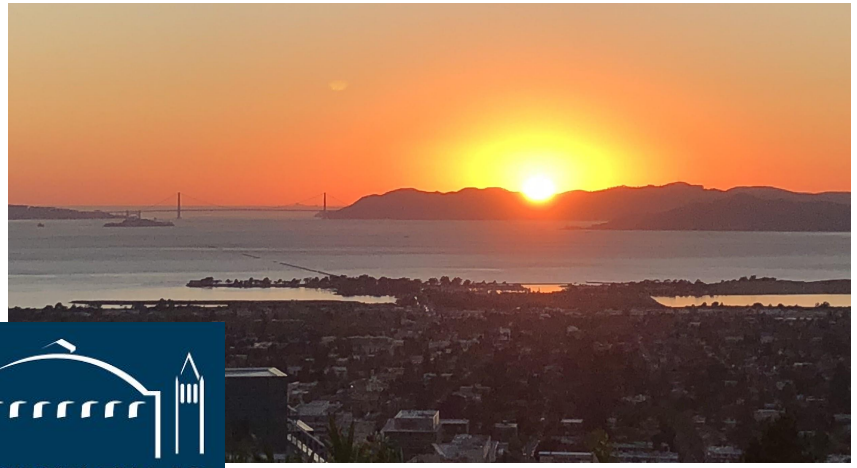


BERKELEY LAB



CSA Summer Series
2021.06.29

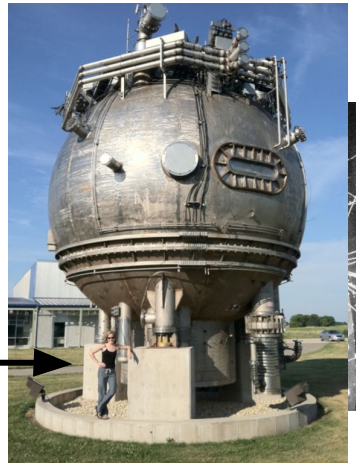
Neural Networks with Euclidean Symmetry for Physical Systems



Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*

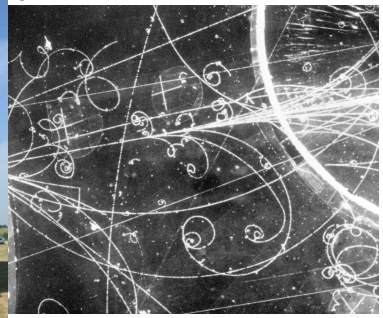
BERKELEY LAB

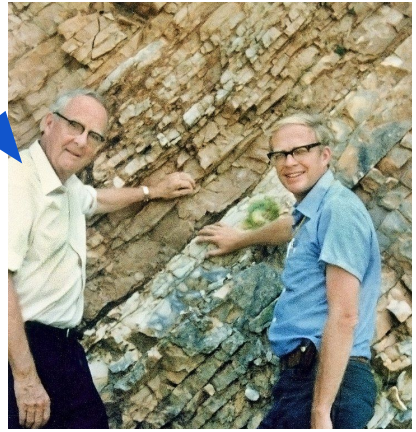
(liquid H_2) Bubble chamber



Me

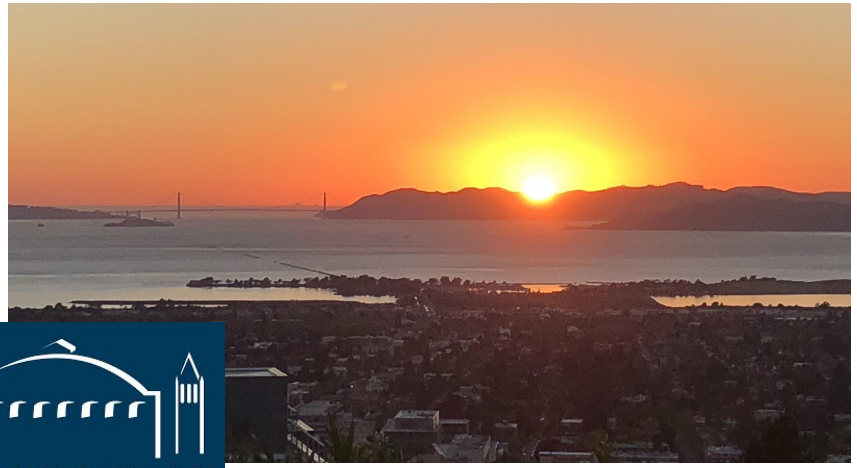
Subatomic particle tracks





CSA Summer Series
2021.06.29

Neural Networks with Euclidean Symmetry for Physical Systems



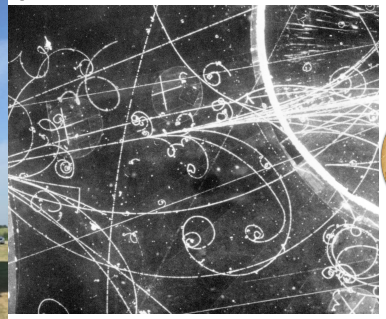
Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*

BERKELEY LAB

(liquid H_2) Bubble chamber

Me

Subatomic particle tracks

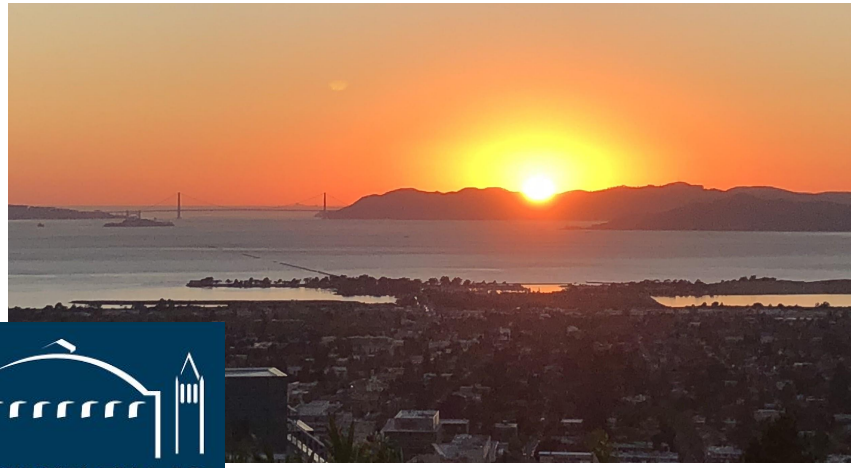


1968

CSA Summer Series
2021.06.29

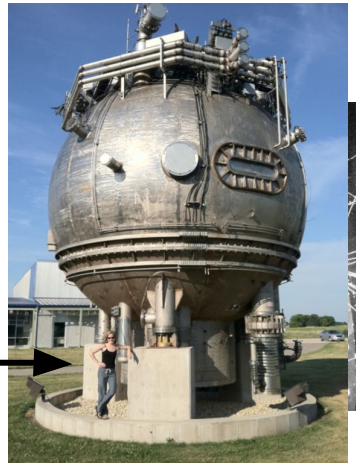
Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*



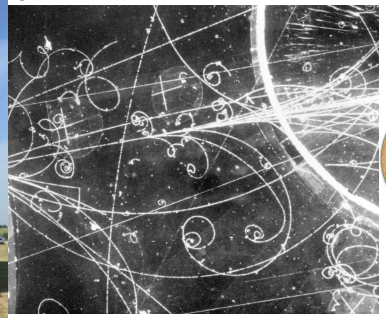
BERKELEY LAB

(liquid H_2) Bubble chamber

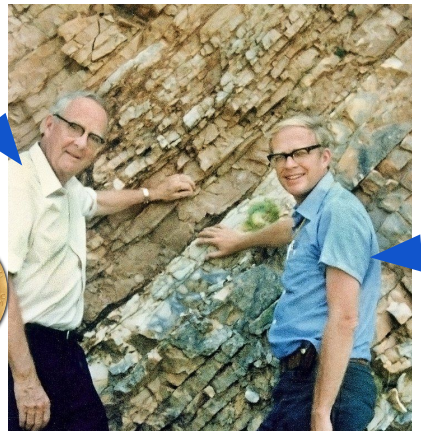


Me

Subatomic particle tracks



1968

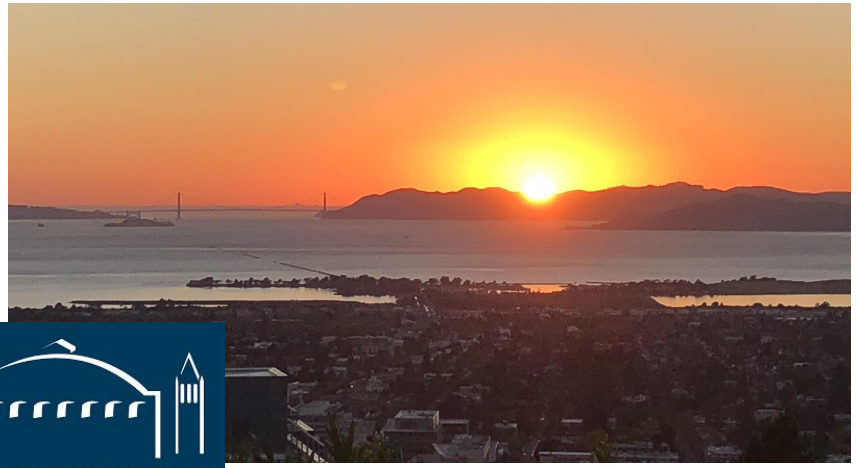


Walter Alvarez (son)

CSA Summer Series
2021.06.29

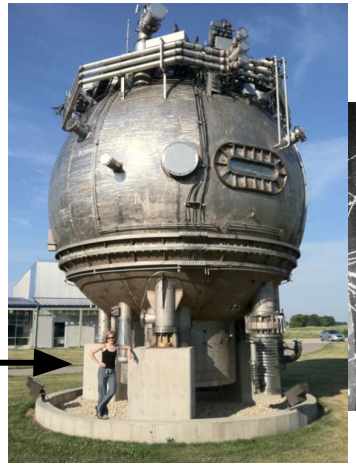
Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
*2018 **Luis W. Alvarez** Fellow in Computing Sciences*



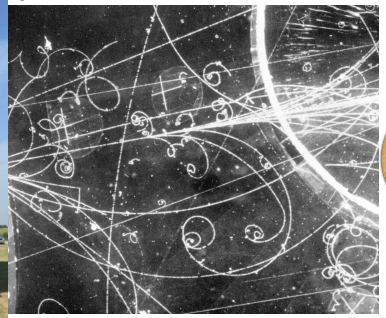
BERKELEY LAB

(liquid H_2) Bubble chamber



Me

Subatomic particle tracks



1968

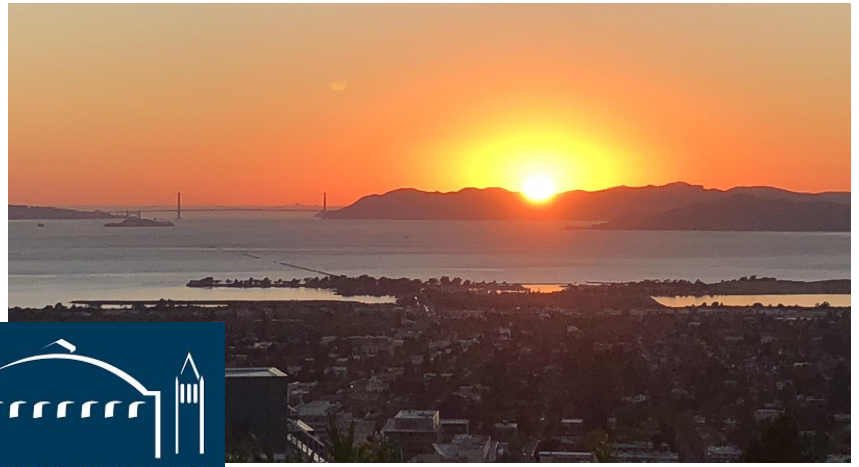
1980: Evidence that dinosaurs went extinct from a meteor (iridium rich deposit 66 million years ago).

Walter Alvarez (son)

CSA Summer Series
2021.06.29

Neural Networks with Euclidean Symmetry for Physical Systems

Tess Smidt
2018 Luis W. Alvarez Fellow in Computing Sciences



- **What is artificial intelligence vs. machine learning vs. neural networks vs. deep learning?**
- **What's different about neural networks for science vs. other data?**
- **Why are neural networks with Euclidean symmetry so good at learning from 3D data?**
- **How can these methods help us do science (especially materials physics)?**

CSA Summer Series
2021.06.29

An introduction to machine learning with emojis

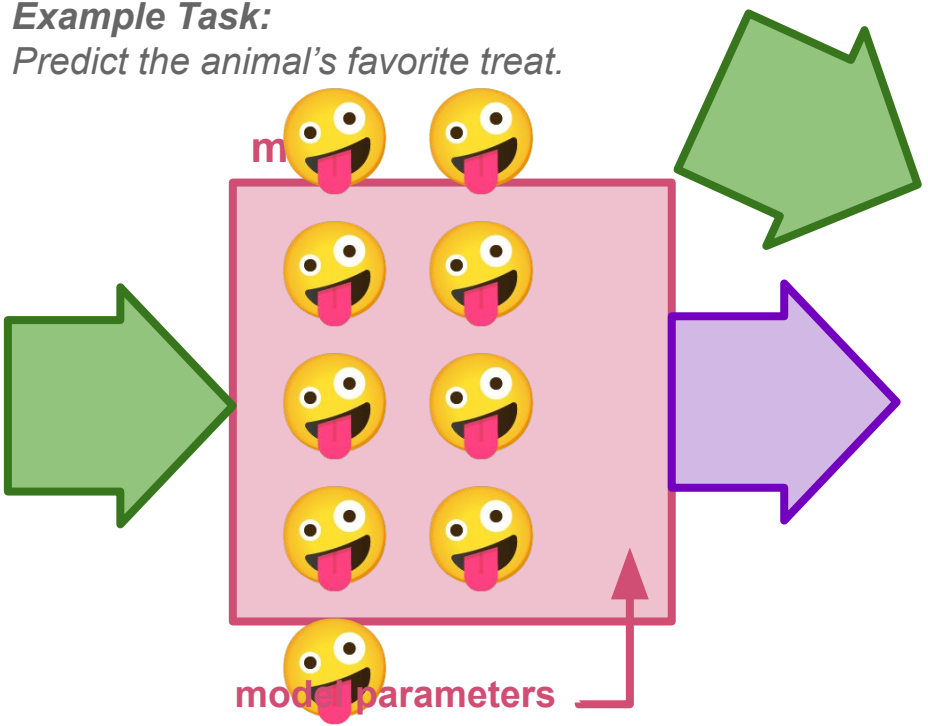
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

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.



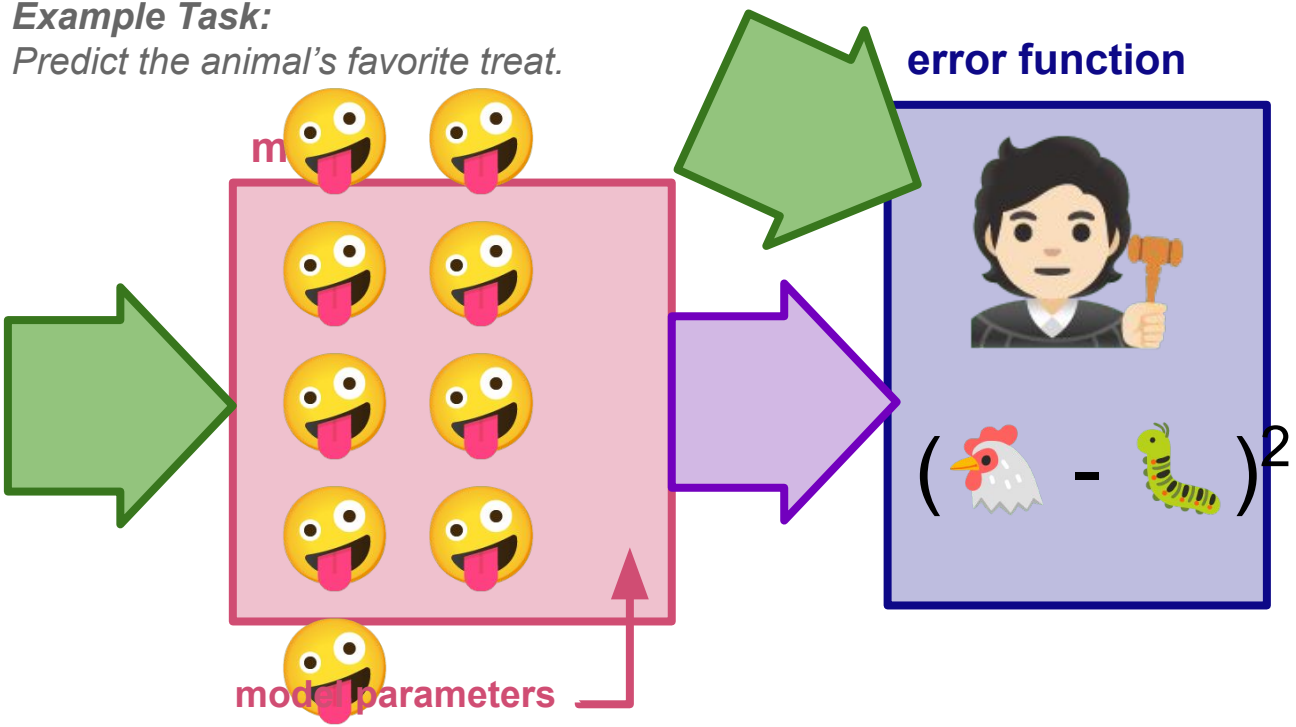
An introduction to machine learning with emojis

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

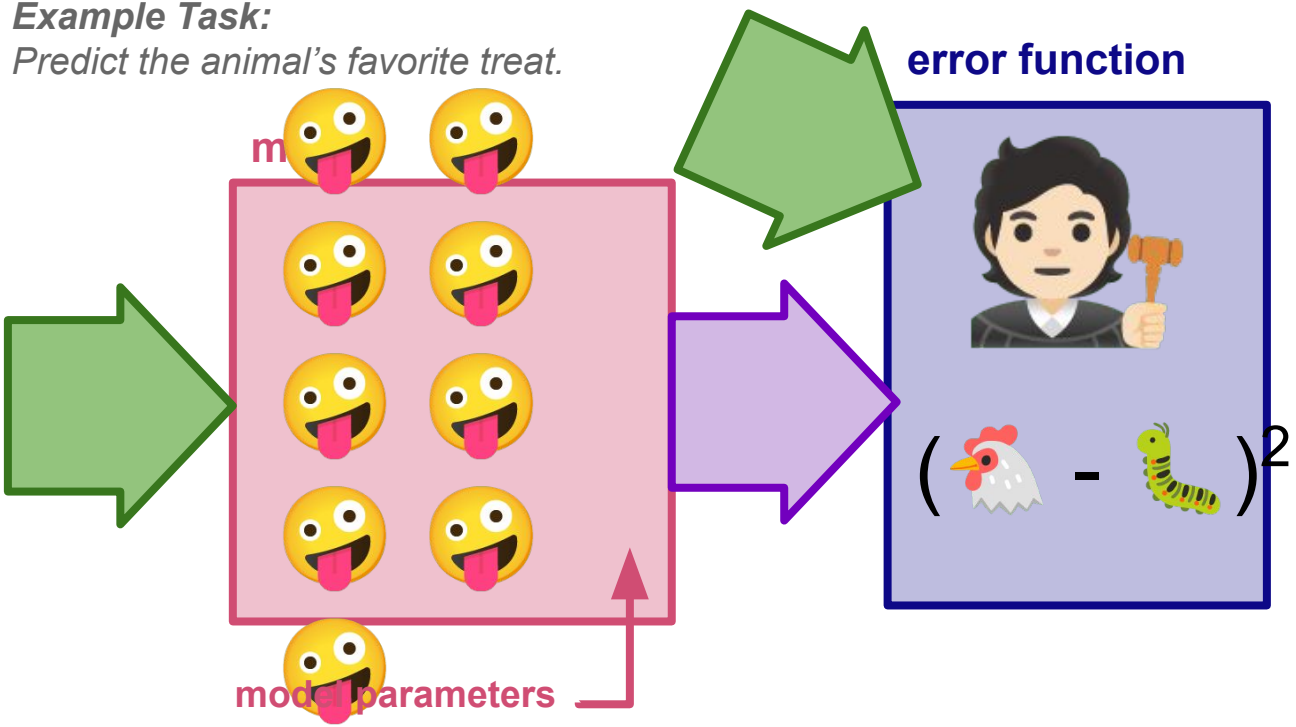
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

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.



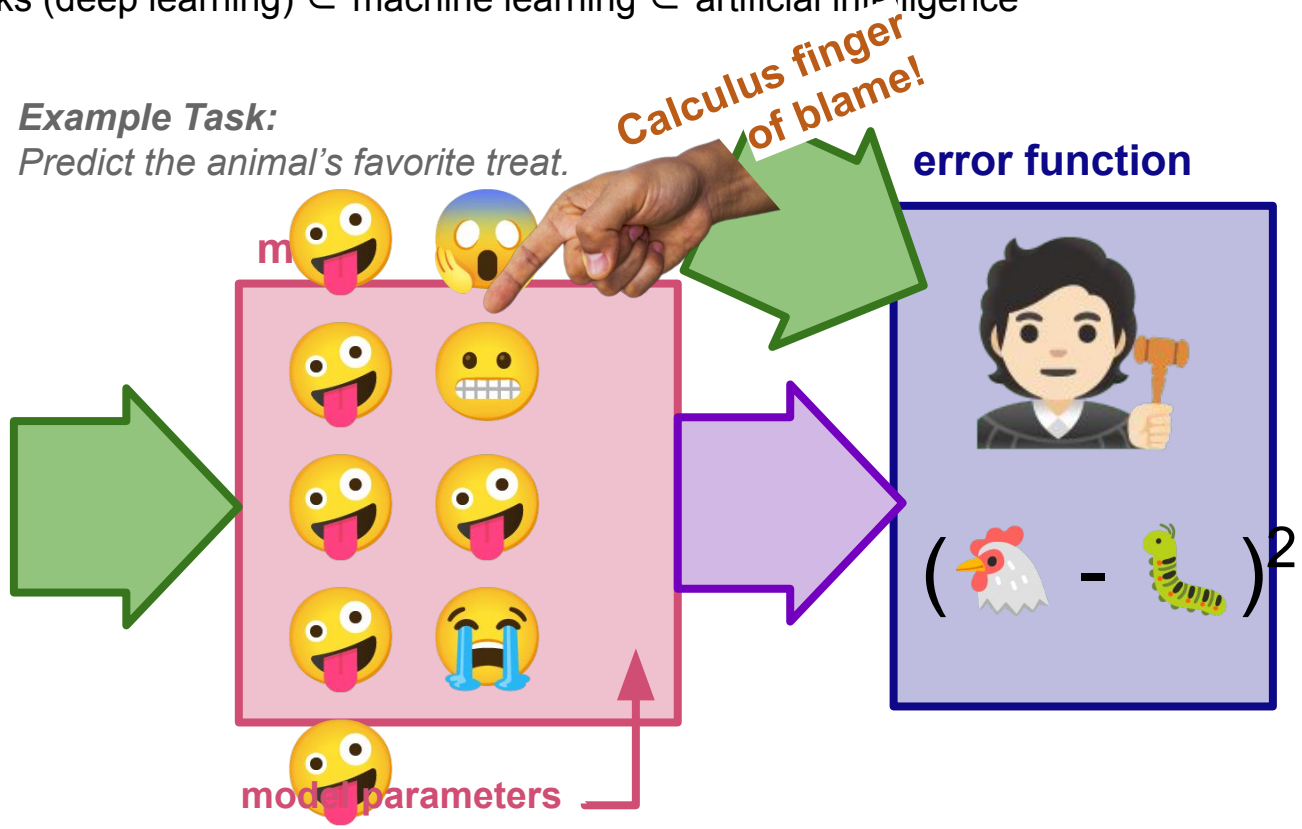
An introduction to machine learning with emojis

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**.



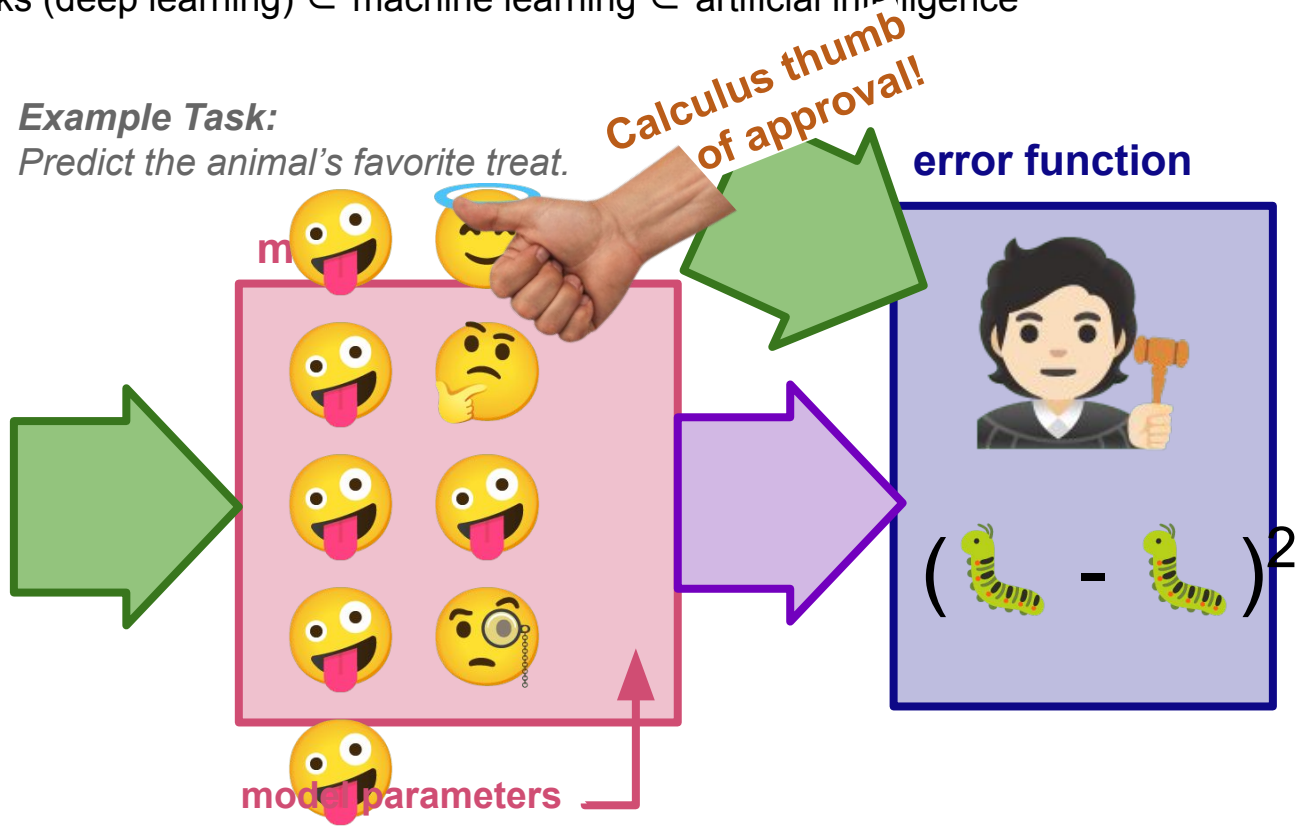
An introduction to machine learning with emojis

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

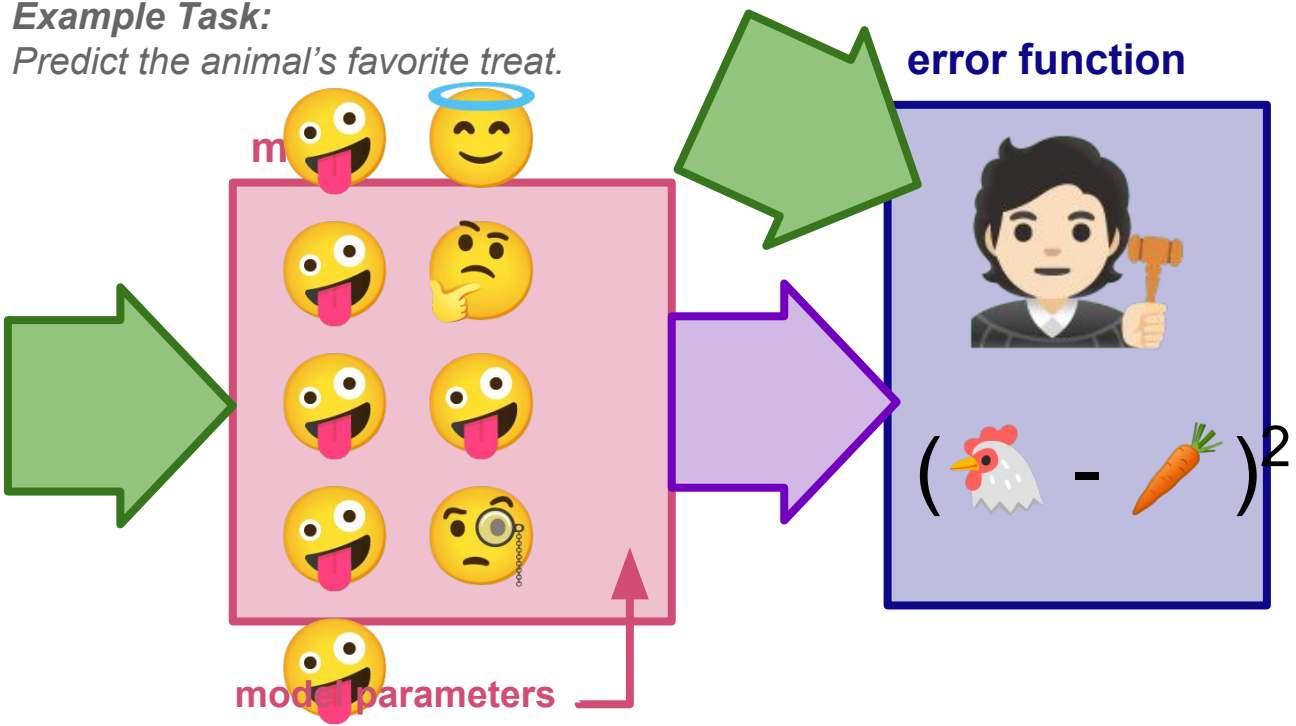
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

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.



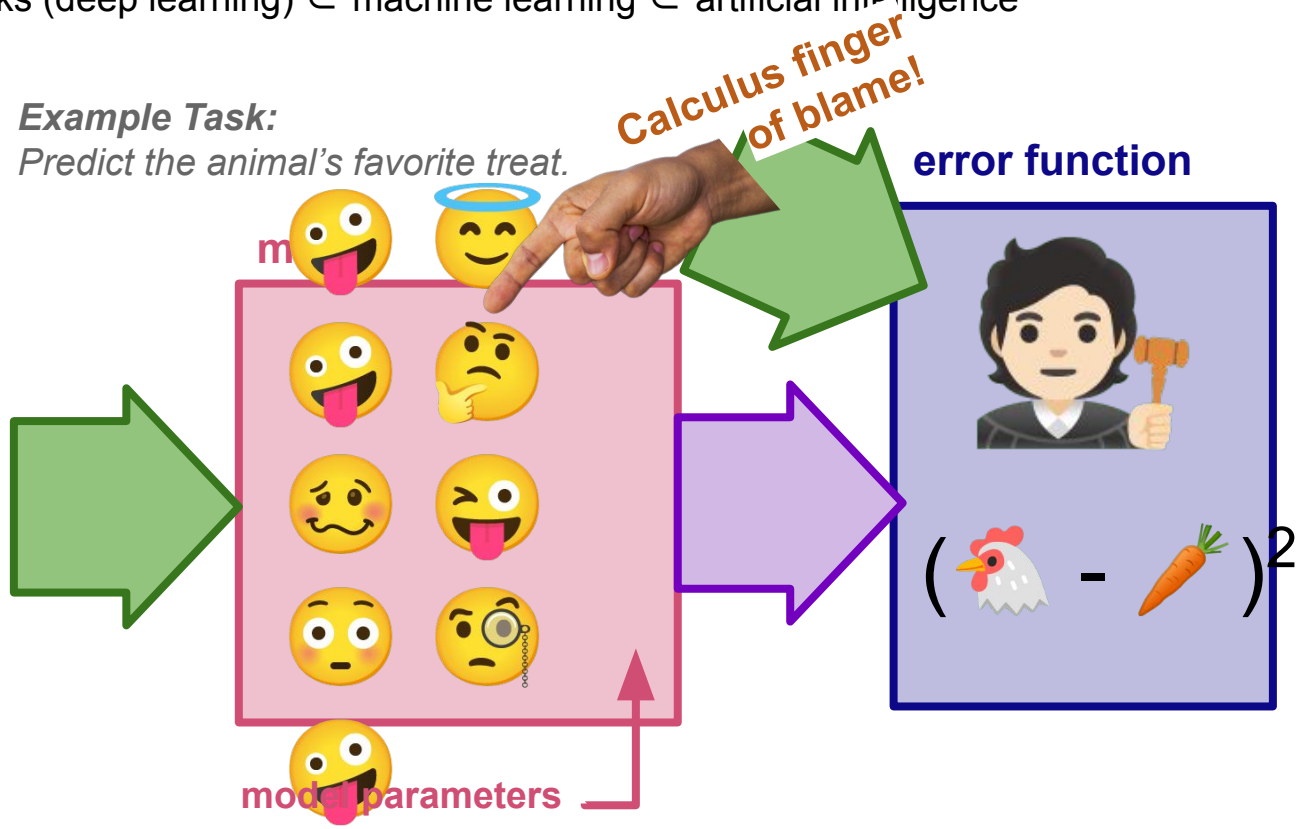
An introduction to machine learning with emojis

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**.



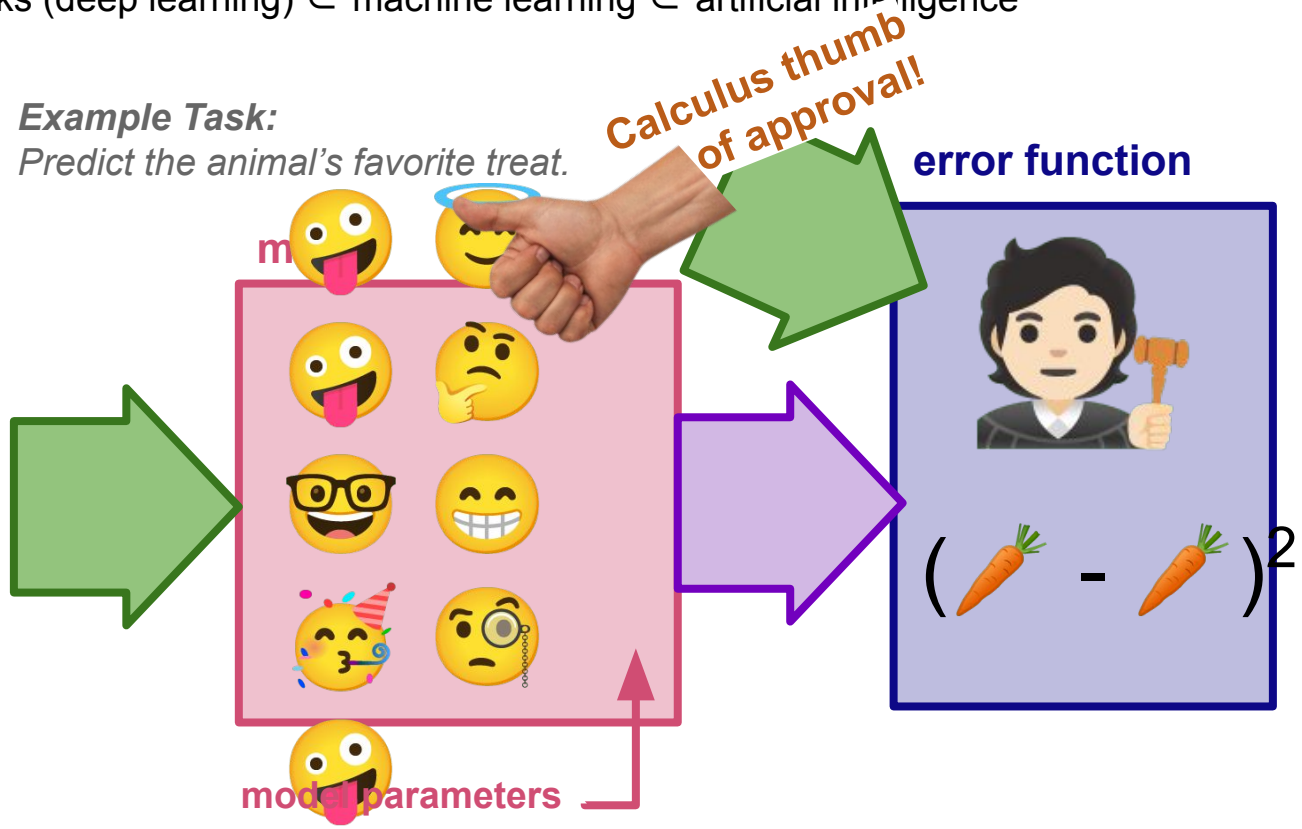
An introduction to machine learning with emojis

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

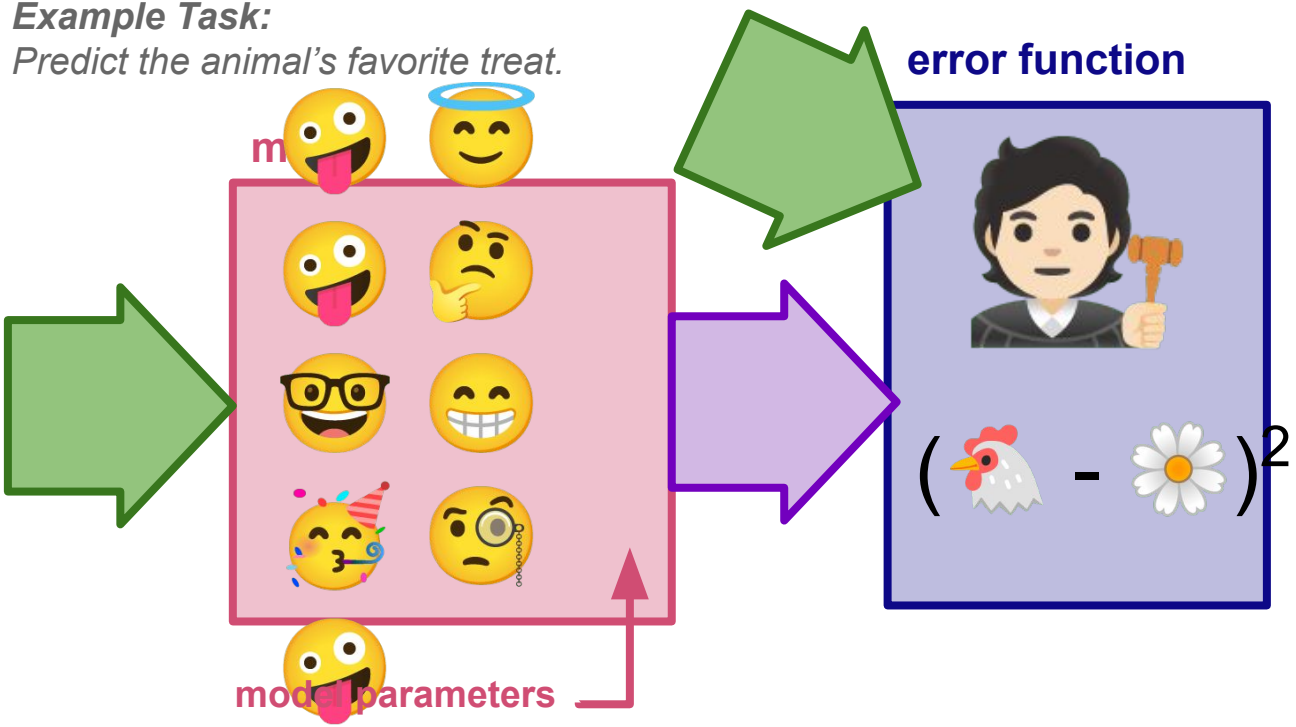
neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence

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.



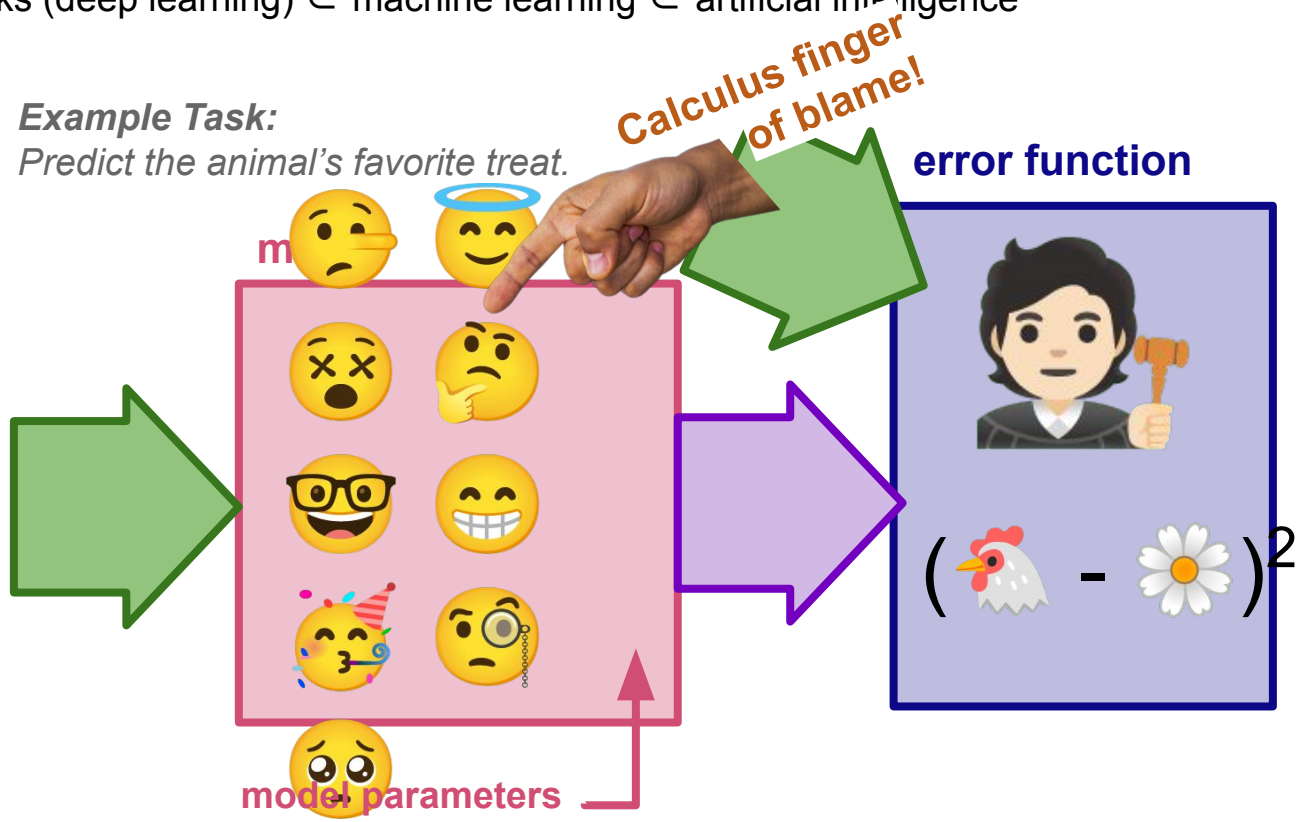
An introduction to machine learning with emojis

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**.



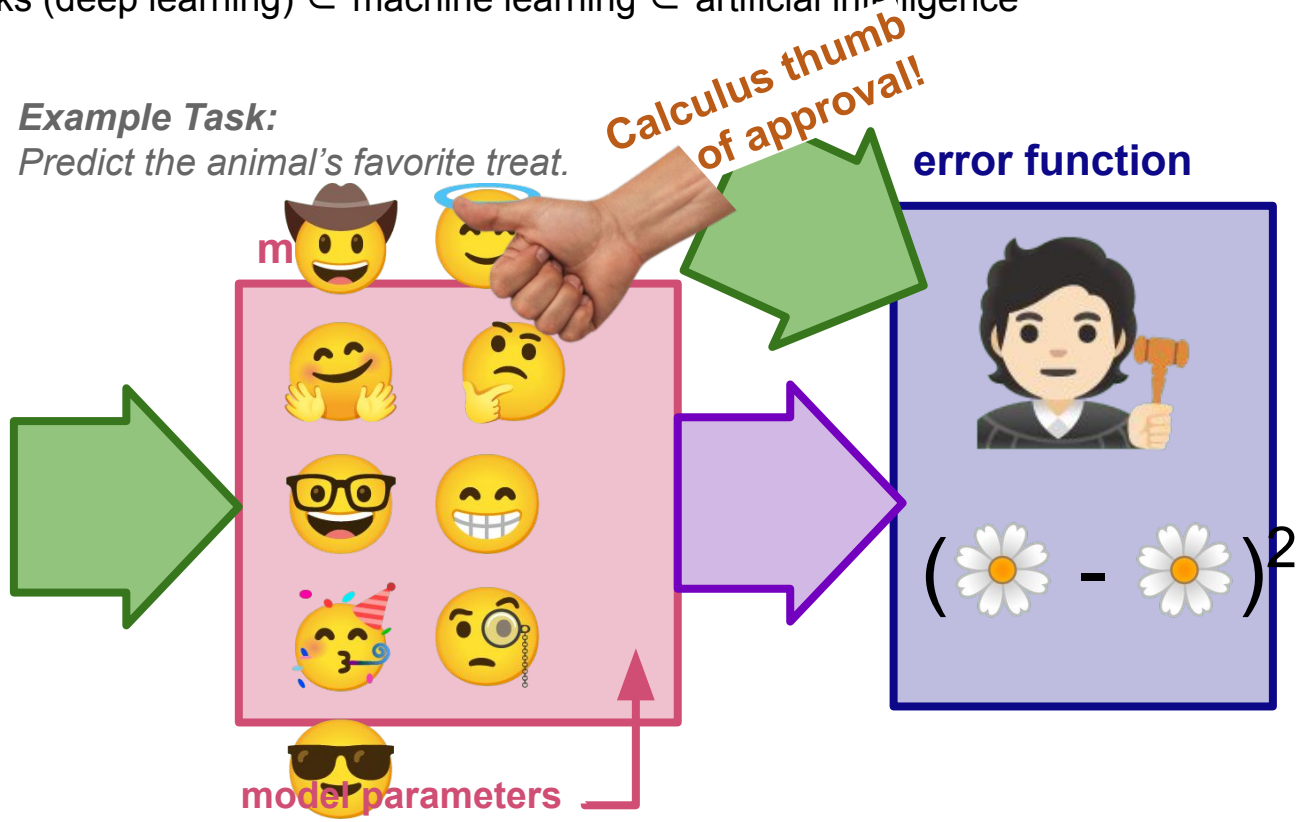
An introduction to machine learning with emojis

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

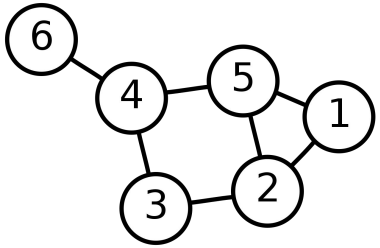
Goal is a model,
a program that **learns** to give **predictions** based on **examples** and **feedback** it gets during training.

Error function evaluates how well model is doing.

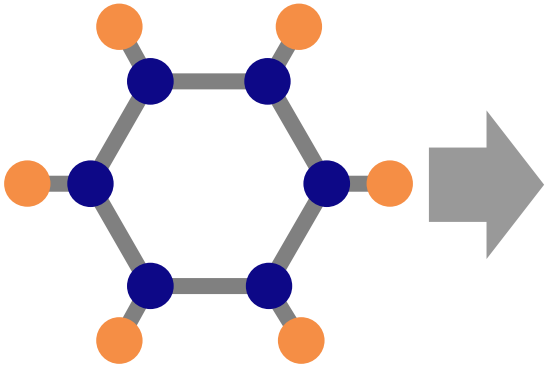
Neural networks use **derivatives (calculus)** to update **model parameters**.



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

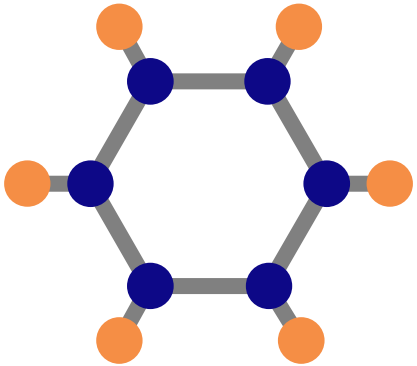
Data type	Images	Text	Graph	Science data in 3D
		Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ultricies, justo ac viverra euismod, justo odio eleifend dolor, a imperdiet quam nibh finibus mauris. Morbi lobortis a lorem id dapibus. Interdum et malesuada fames...		?
Type of neural network	**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	???

To describe physical systems in 3D, we use coordinates and coordinate systems.



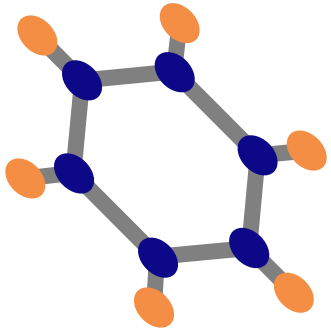
H	-0.21463	0.97837	0.33136
C	-0.38325	0.66317	-0.70334
C	-1.57552	0.03829	-1.05450
H	-2.34514	-0.13834	-0.29630
C	-1.78983	-0.36233	-2.36935
H	-2.72799	-0.85413	-2.64566
C	-0.81200	-0.13809	-3.33310
H	-0.98066	-0.45335	-4.36774
C	0.38026	0.48673	-2.98192
H	1.14976	0.66307	-3.74025
C	0.59460	0.88737	-1.66708
H	1.53276	1.37906	-1.39070

To describe physical systems in 3D, we use coordinates and coordinate systems.



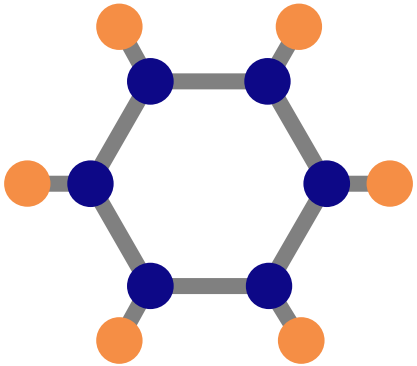
H	-0.21463	0.97837	0.33136
C	-0.38325	0.66317	-0.70334
C	-1.57552	0.03829	-1.05450
H	-2.34514	-0.13834	-0.29630
C	-1.78983	-0.36233	-2.36935
H	-2.72799	-0.85413	-2.64566
C	-0.81200	-0.13809	-3.33310
H	-0.98066	-0.45335	-4.36774
C	0.38026	0.48673	-2.98192
H	1.14976	0.66307	-3.74025
C	0.59460	0.88737	-1.66708
H	1.53276	1.37906	-1.39070

But coordinates are sensitive to translations, rotations, and inversion.



H	-0.17346	-0.64630	-0.81565
C	-0.28950	-0.97368	0.22250
C	-1.49229	-0.76124	0.88842
H	-2.32276	-0.26670	0.37450
C	-1.63978	-1.17719	2.20770
H	-2.58623	-1.00994	2.73174
C	-0.58461	-1.80562	2.86111
H	-0.70070	-2.13292	3.89924
C	0.61816	-2.01800	2.19520
H	1.44851	-2.51239	2.70939
C	0.76569	-1.60206	0.87592
H	1.71213	-1.76918	0.35188

To describe physical systems in 3D, we use coordinates and coordinate systems.



H	-0.21463	0.97837	0.33136
C	-0.38325	0.66317	-0.70334
C	-1.57552	0.03829	-1.05450
H	-2.34514	-0.13834	-0.29630
C	-1.78983	-0.36233	-2.36935
H	-2.72799	-0.85413	-2.64566
C	-0.81200	-0.13809	-3.33310
H	-0.98066	-0.45335	-4.36774
C	0.38026	0.48673	-2.98192
H	1.14976	0.66307	-3.74025
C	0.59460	0.88737	-1.66708
H	1.53276	1.37906	-1.39070

But coordinates are sensitive to translations, rotations, and inversion.

Three ways to make models "symmetry-aware" for 3D data

e.g. How to make a model that "understands" the geometry of atomic structures?

Approach 1:
Data Augmentation

Brute-force

Approach 2: 👌
Invariant Models

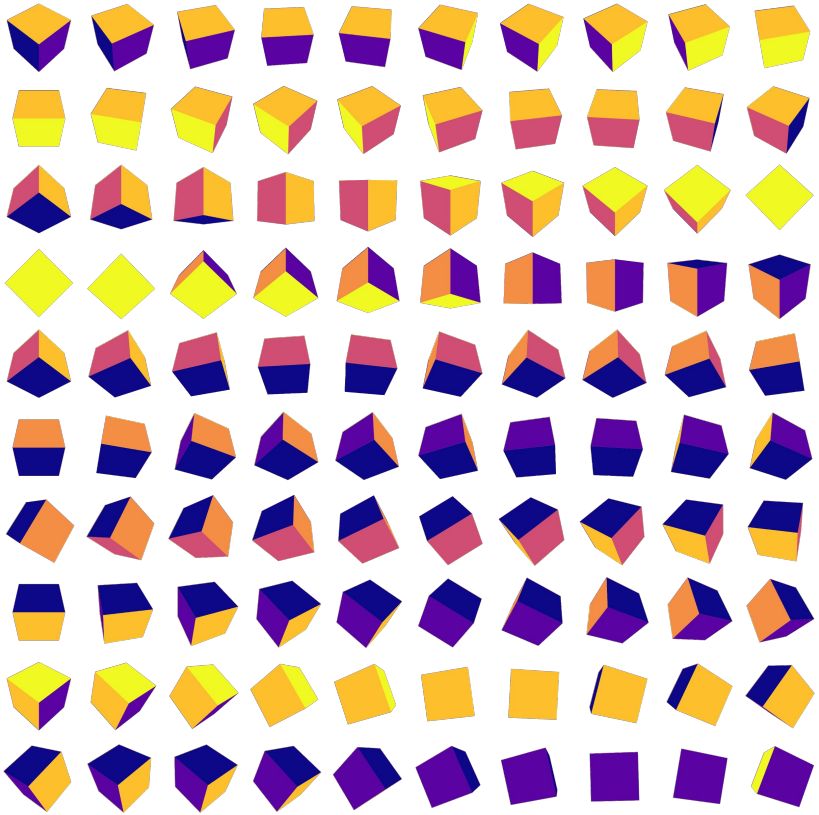
Most current models

Approach 3:
Equivariant models

What I'll talk about today

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



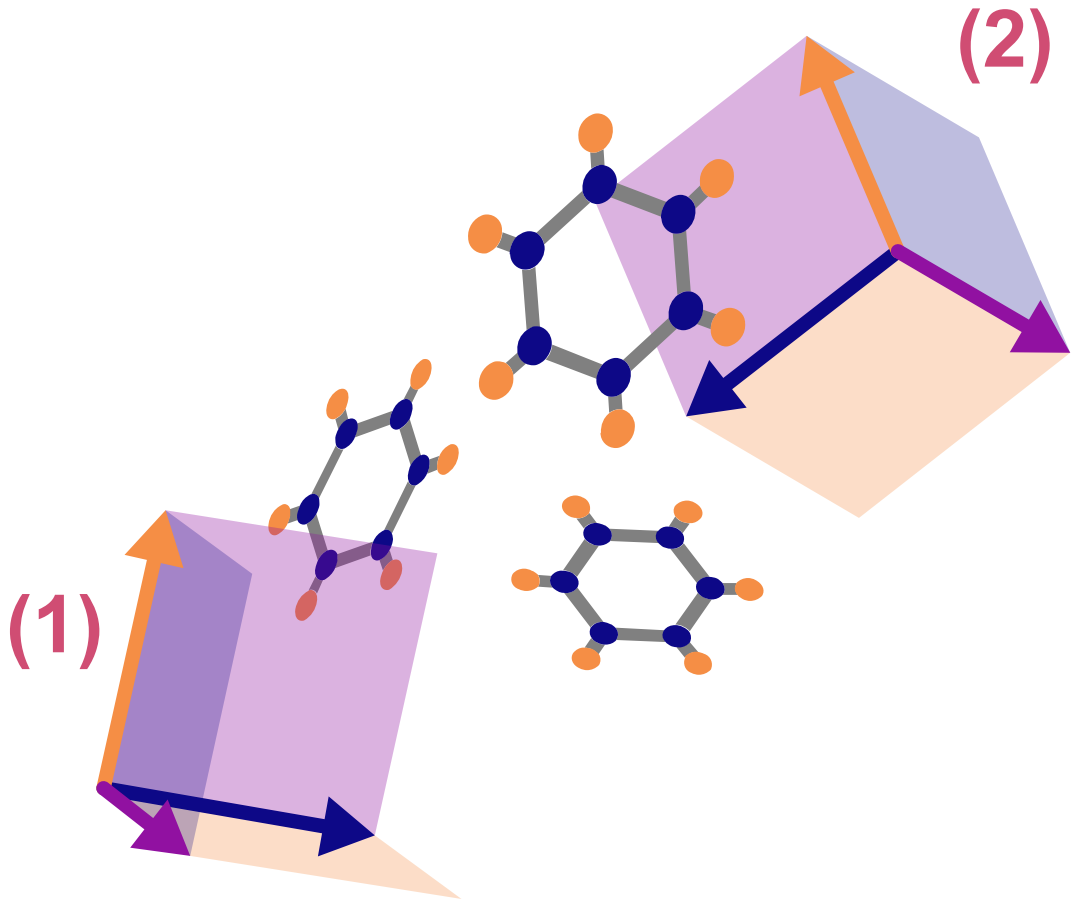
training with symmetry

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

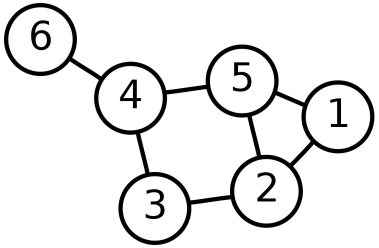
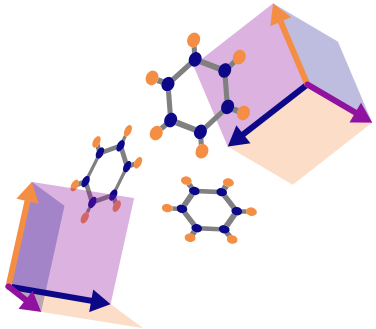
Traditional machine learning see (1) and (2) as completely different!

Invariant models can't distinguish between (1) and (2).

***Euclidean neural networks* see (1) and (2) as the same system just rotated and translated.**

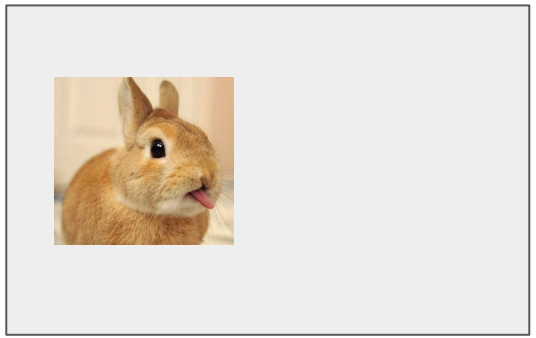


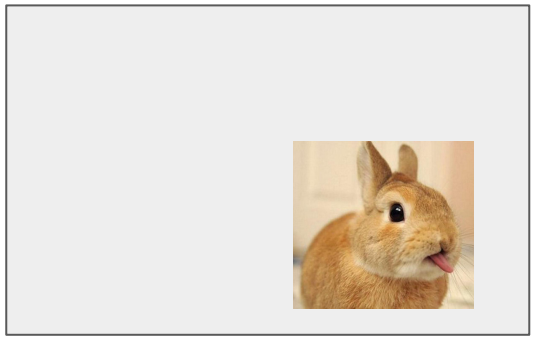
Neural networks are specially designed for different **data types** in order to make use of special features of the data.

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

Euclidean neural networks can identify an instance of a *local* or *global* pattern in any location / orientation / mirror (change of coordinate system).

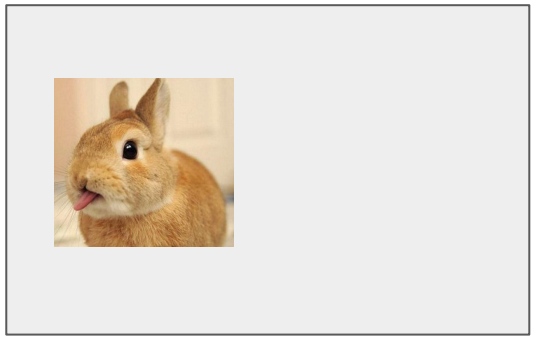
if ✓
then...

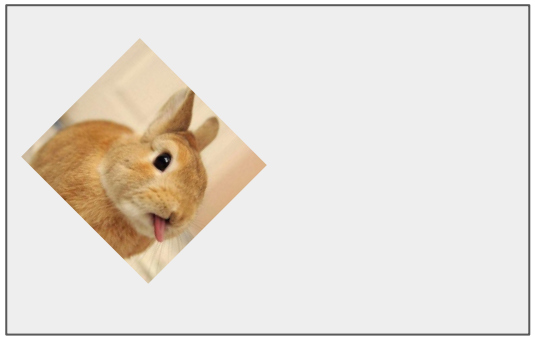




✓translations

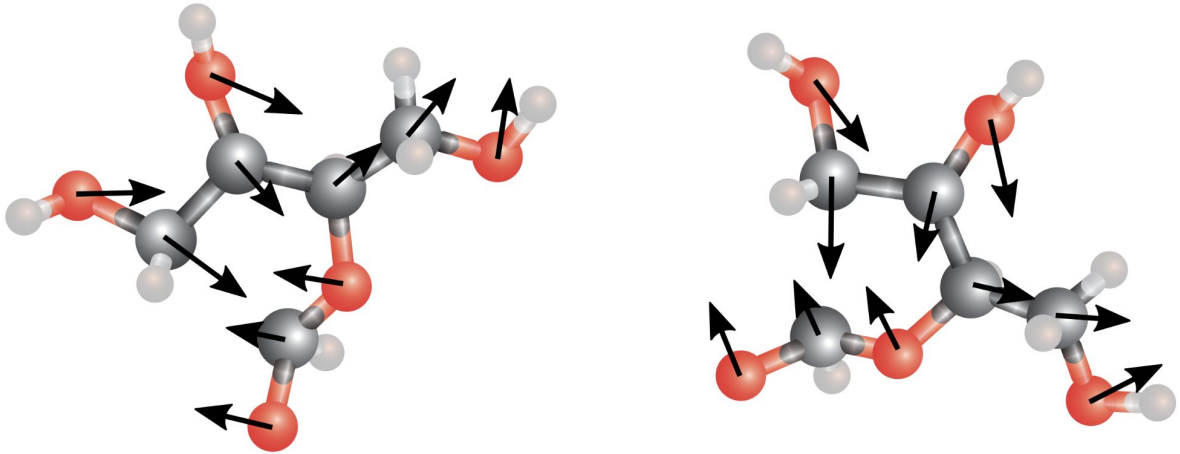
✓mirrors
(rotation + inversion)



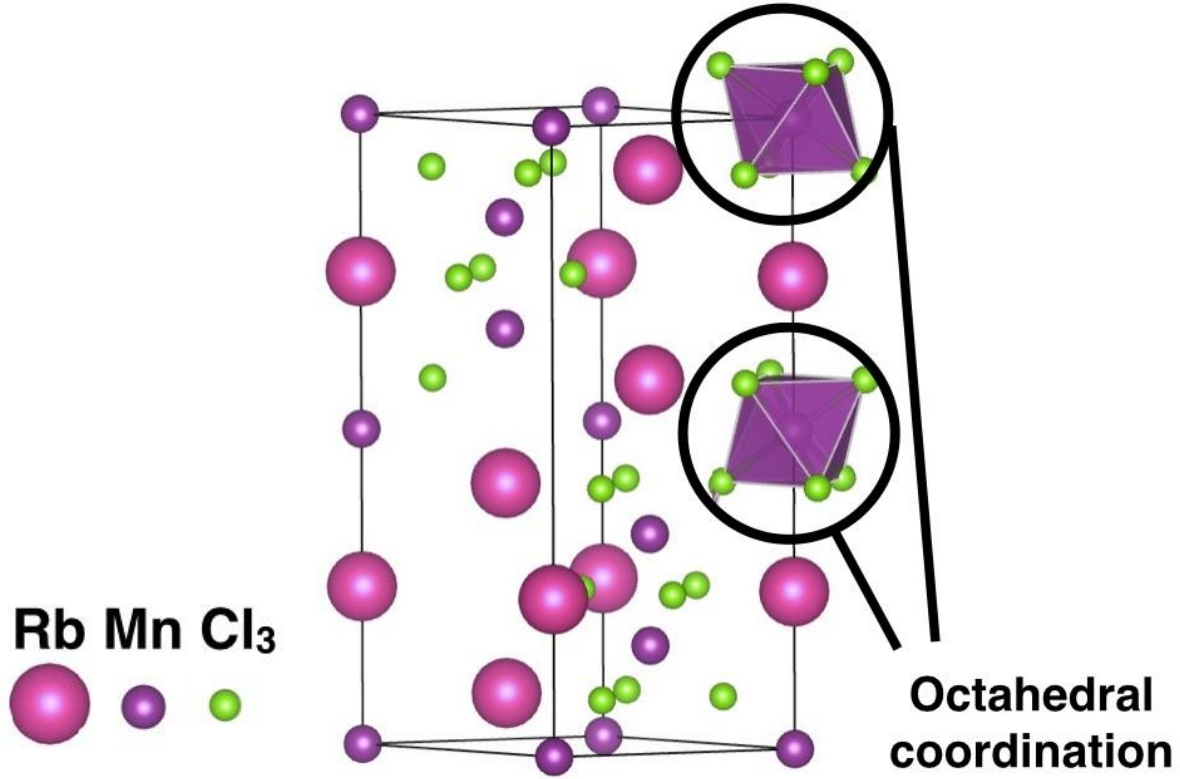


✓rotations

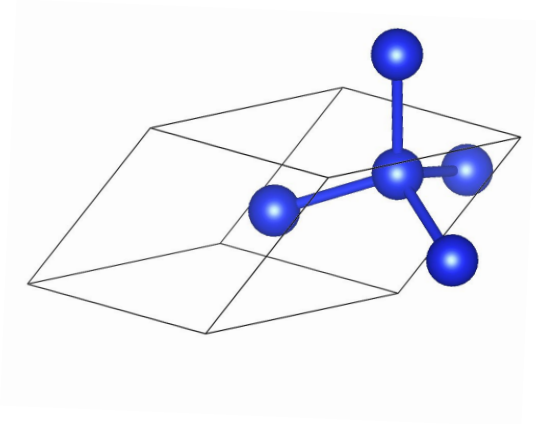
Given a molecule and a rotated copy,
predicted forces are the same up to rotation.
(Predicted forces are equivariant to rotation.)
Additionally, networks generalize to molecules with similar motifs.

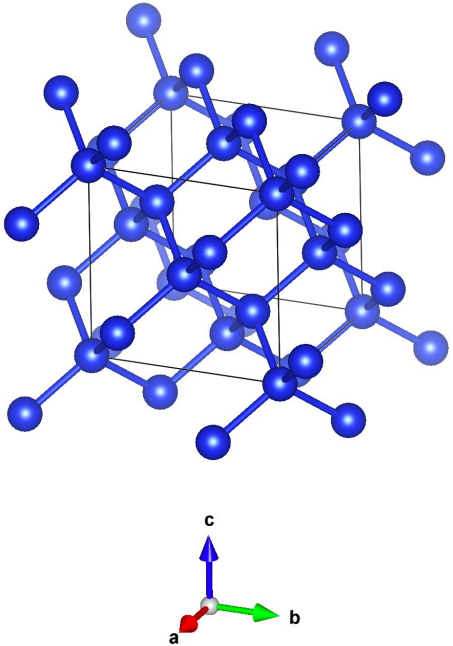


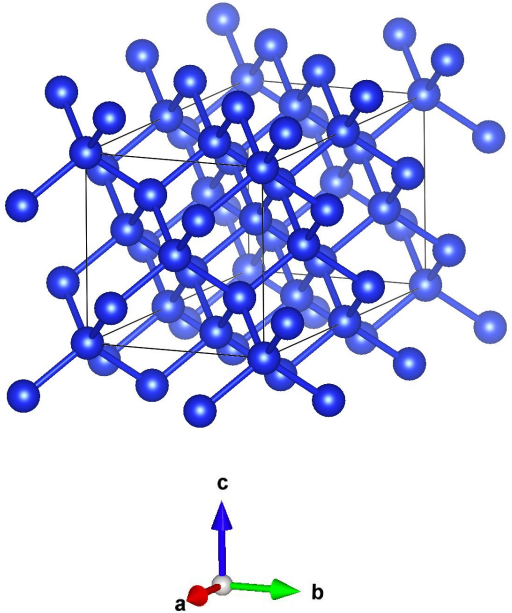
These networks can recognize equivalent recurring geometric patterns that appear in different locations and orientations.



Primitive unit cells, conventional unit cells, and supercells of the same crystal produce the same output (assuming periodic boundary conditions).

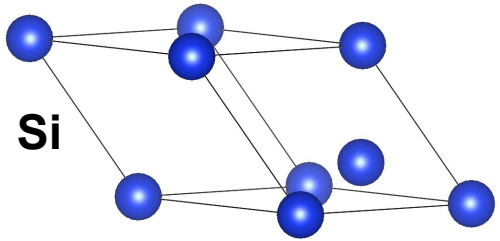




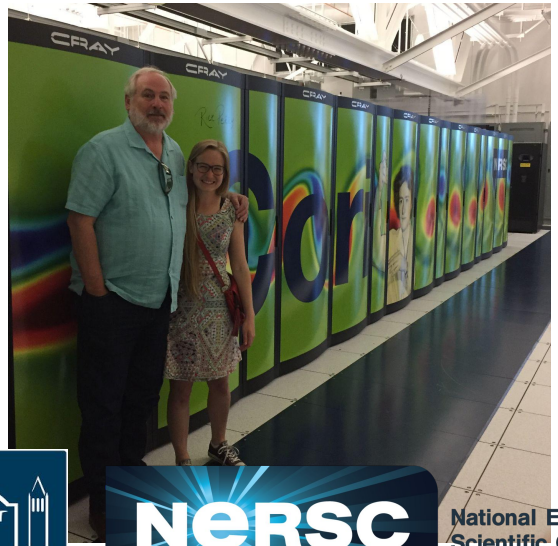


I'm a computational materials physicist. I take atomic structures and calculate their properties to see if they'd be suitable for a variety of applications.

Structure



Quantum Theory + Supercomputers

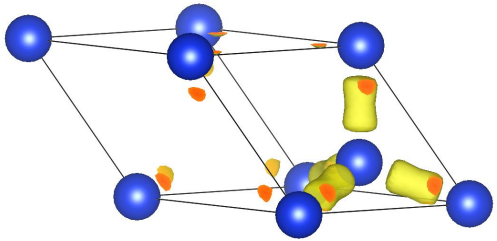


BERKELEY LAB

NERSC

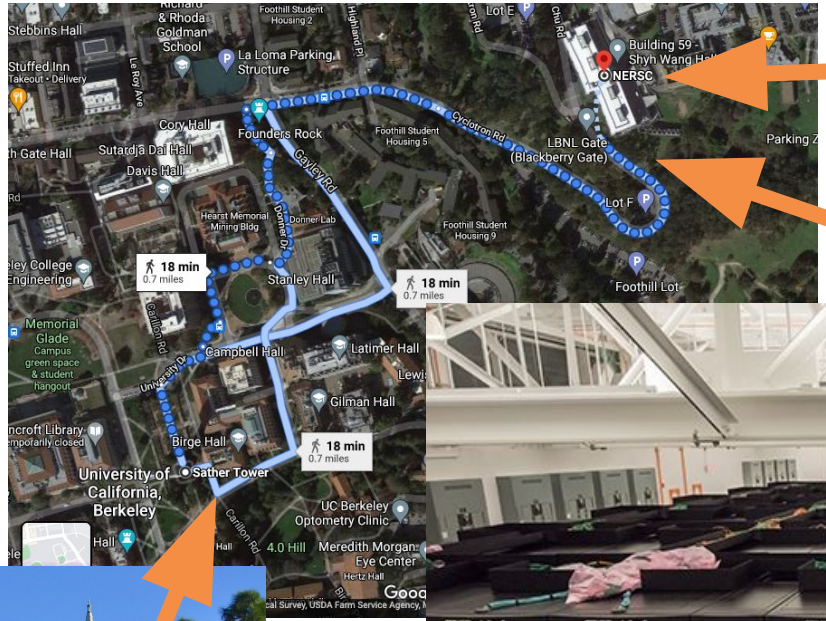
National Energy Research Scientific Computing Center

Properties



Photovoltaics?
Quantum computing?
Energy storage?

Quick aside: Fantastic Supercomputers and Where to Find Them!



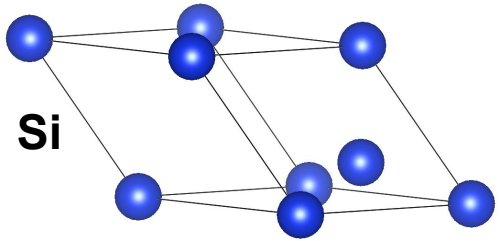
NERSC National Energy Research Scientific Computing Center

BERKELEY LAB

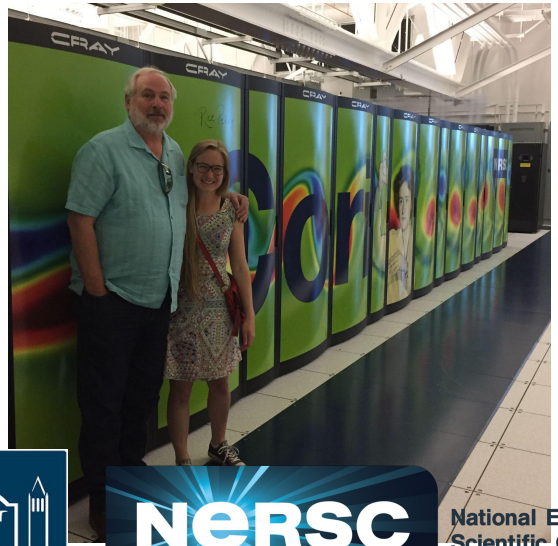
My dad and me for scale

I'm a computational materials physicist. I take atomic structures and calculate their properties to see if they'd be suitable for a variety of applications.

Structure



Quantum Theory + Supercomputers

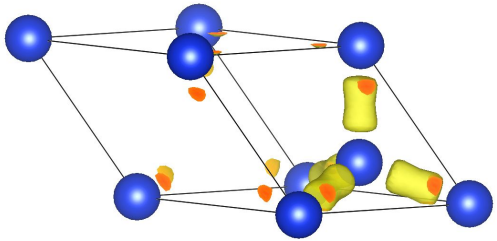


BERKELEY LAB

NERSC

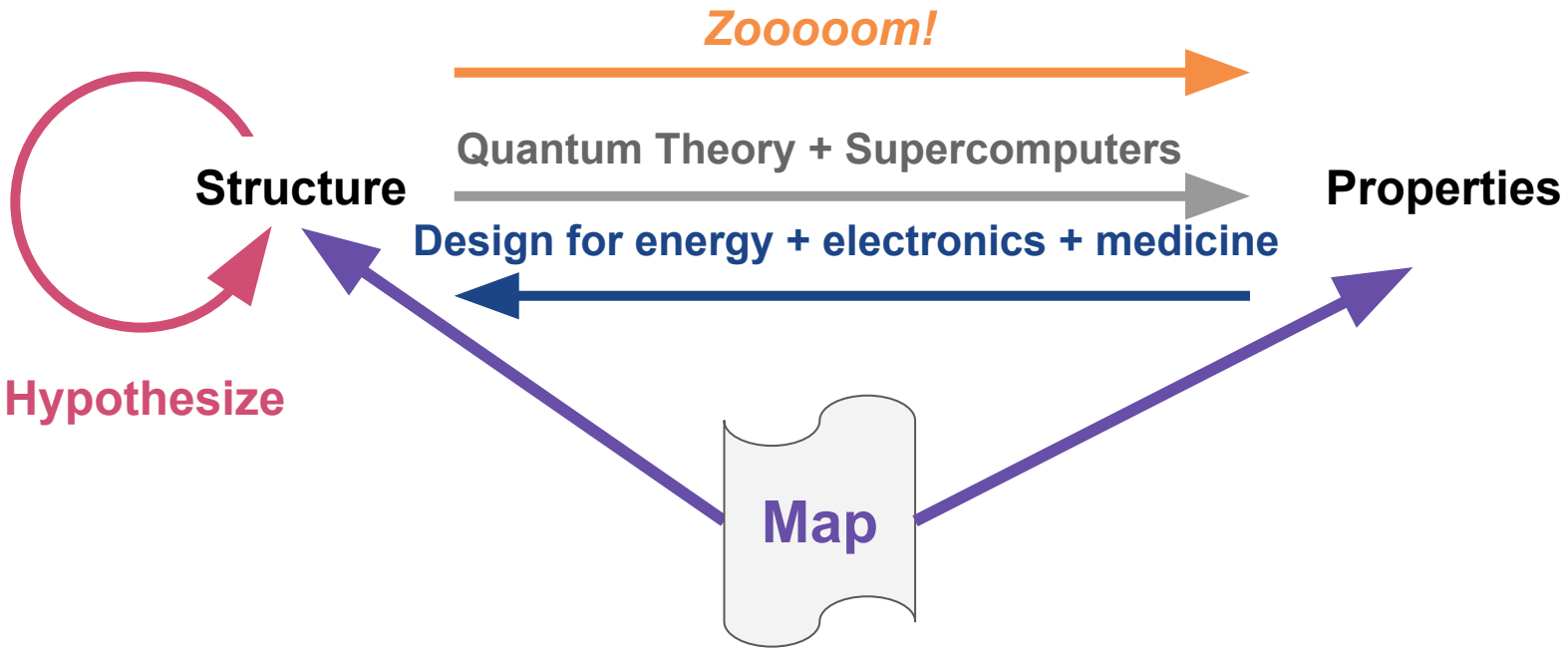
National Energy Research Scientific Computing Center

Properties



Photovoltaics?
Quantum computing?
Energy storage?

I want to use machine learning to speed up calculations, hypothesize new structures, design custom molecules and materials, and organize these relations.



Euclidean neural networks and related methods are helping us scale expensive physics simulations so we can understand and design atomic systems for applications in medicines, energy technologies, new electronics, etc.

Protein folding (sequence ⇨ 3D structure)

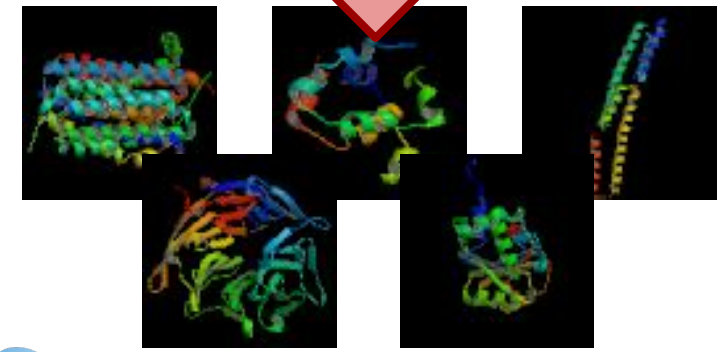
MKEFWNLDKNLQLRLGIVFLGAFSYGTVFSSMTIYYNQYLGSA...

MTDISQMYDQLSDPFAGLGAGNIHLGYFDGPDDAATLAEAADR...

MVSNWNWSGKKGRRTPRRGYTRPFKSAVPTTRVVVHQSAVLKK...

KPTENNEDFNIVAVASNFATTDLDADRGKLPGKKLPLEVLKEM...

MARIGDLDAARPAPEAVPGDMVRIPGGTFLQGSPERTLDWLDR...

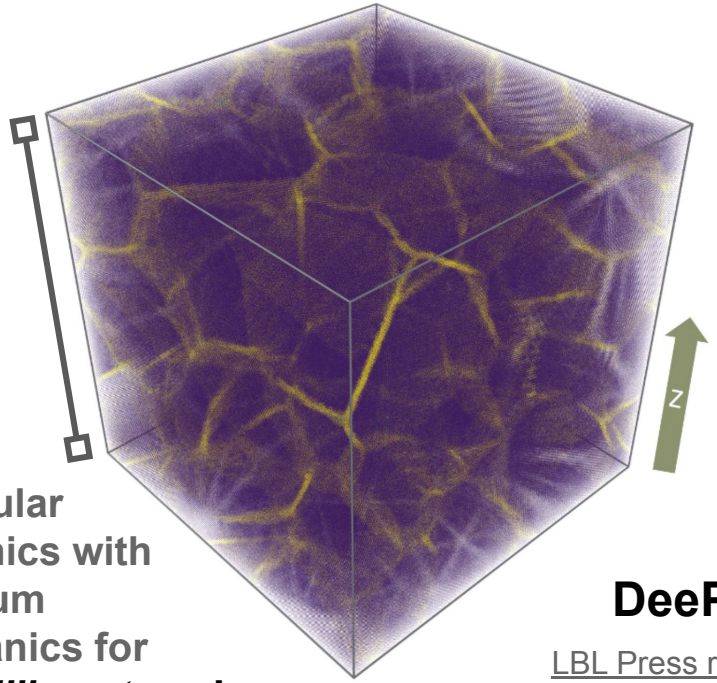


DeepMind **AlphaFold2**

Large molecular dynamics simulations of materials

Cu

50 nm



Molecular dynamics with quantum mechanics for *100 million atoms*!

DeePMD

LBL Press release

ACM Press release

Paper: https://arxiv.org/pdf/2005.00223.pdf

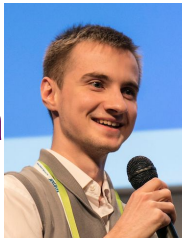
Collaborators!

Science is most fun with friends!


Mario Geiger

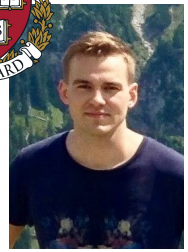

Martin Uhrin


Ben Miller


Kostiantyn Lapchevskyi


Thomas Hardin


Josh Rackers


Simon Batzner

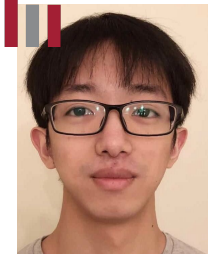

Alby Musaelian

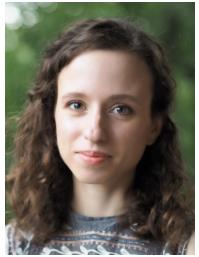

Boris Kozinsky

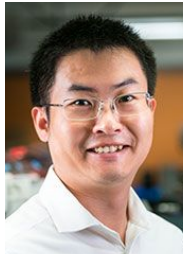

Eugene Kwan


Frank Noé


Claire West


Zhantao Chen


Nina Andrejevic


Mingda Li


Bradley Dice

A Quick Recap!

Slides: https://tinyurl.com/tess-cssp-2021

Feel free to reach out if you have any questions!

Tess Smidt
tsmidt@lbl.gov

My website
http://blondegeek.github.io/

Euclidean neural networks
e3nn.org

A Quick Recap!

Slides: https://tinyurl.com/tess-cssp-2021

- What is artificial intelligence vs. machine learning vs. neural networks vs. deep learning?
  - neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence
  - All give “models” which learn to give predictions based on examples

Feel free to reach out if you have any questions!

Tess Smidt
tsmidt@lbl.gov

My website
http://blondegeek.github.io/

Euclidean neural networks
e3nn.org

A Quick Recap!

Slides: https://tinyurl.com/tess-cssp-2021

- What is artificial intelligence vs. machine learning vs. neural networks vs. deep learning?
  - neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence
  - All give "models" which learn to give predictions based on examples

- What's different about neural networks for science vs. other data?
  - Neural networks are custom built for specific data "types".
  - Scientific data has special features that other data (images, text, graphs, etc.) doesn't.

Feel free to reach out if you have any questions!

Tess Smidt
tsmidt@lbl.gov

My website
http://blondegeek.github.io/

Euclidean neural networks
e3nn.org

A Quick Recap!

Slides: https://tinyurl.com/tess-cssp-2021

- What is artificial intelligence vs. machine learning vs. neural networks vs. deep learning?
  - neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence
  - All give "models" which learn to give predictions based on examples
- What's different about neural networks for science vs. other data?
  - Neural networks are custom built for specific data "types".
  - Scientific data has special features that other data (images, text, graphs, etc.) doesn't.
- Why are neural networks with Euclidean symmetry so good at learning from 3D data?
  - They understand 3D coordinate systems and that the same pattern can appear in different locations, rotations, and mirrors.

Feel free to reach out if you have any questions!

Tess Smidt
tsmidt@lbl.gov

My website
http://blondegeek.github.io/

Euclidean neural networks
e3nn.org

A Quick Recap!

Slides: https://tinyurl.com/tess-cssp-2021

- What is artificial intelligence vs. machine learning vs. neural networks vs. deep learning?
  - neural networks (deep learning) ⊂ machine learning ⊂ artificial intelligence
  - All give "models" which learn to give predictions based on examples
- What's different about neural networks for science vs. other data?
  - Neural networks are custom built for specific data "types".
  - Scientific data has special features that other data (images, text, graphs, etc.) doesn't.
- Why are neural networks with Euclidean symmetry so good at learning from 3D data?
  - They understand 3D coordinate systems and that the same pattern can appear in different locations, rotations, and mirrors.
- How can these methods help us do science (especially materials physics)?
  - Learn faster models of slow calculations (run on supercomputers).
  - Understand and design molecules and crystals for e.g. medicine and energy!

Feel free to reach out if you have any questions!

Tess Smidt
tsmidt@lbl.gov

My website
http://blondegeek.github.io/

Euclidean neural networks
e3nn.org

Calling in backup (slides)!

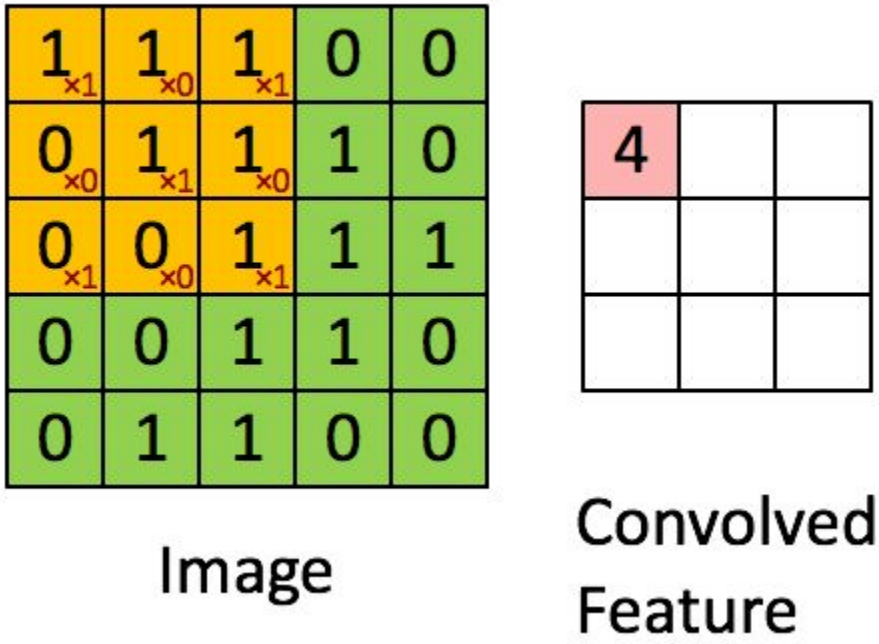
The nitty gritty details :)

Euclidean Neural Networks are similar to convolutional neural networks...

Euclidean Neural Networks are similar to convolutional neural networks...

convolutional neural networks:

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



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

Euclidean Neural Networks are similar to convolutional neural networks...

convolutional neural networks:

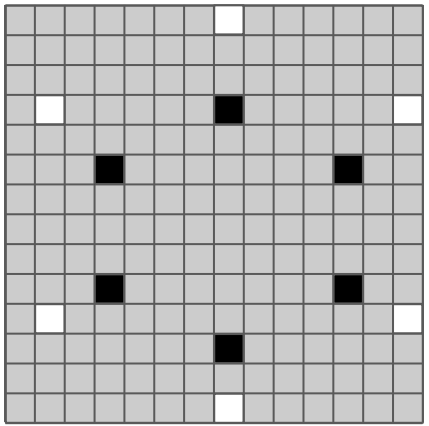
Used for images. In each layer, scan over image with learned filters.

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

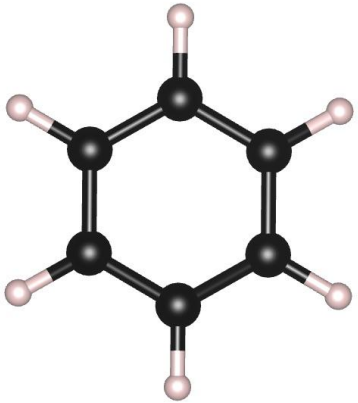
Euclidean Neural Networks are similar to convolutional neural networks...

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

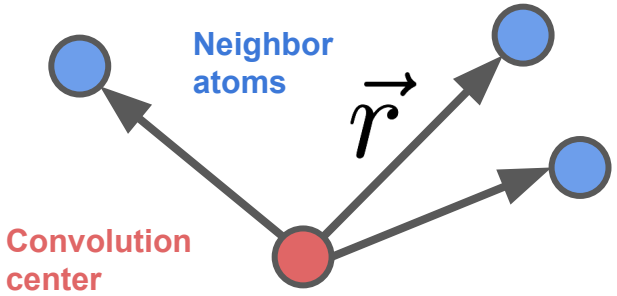
We use points. Images of atomic systems are sparse and imprecise.



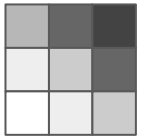
vs.



We use continuous convolutions with atoms as convolution centers.



filter



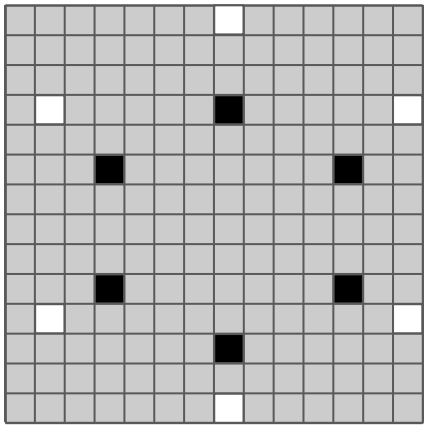
filter function

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

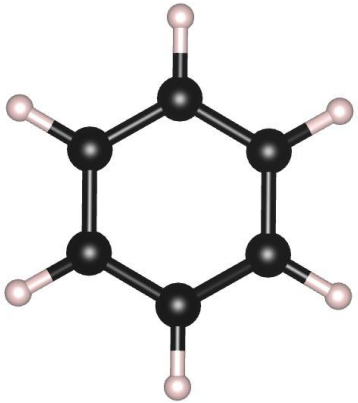
Euclidean Neural Networks are similar to convolutional neural networks...

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

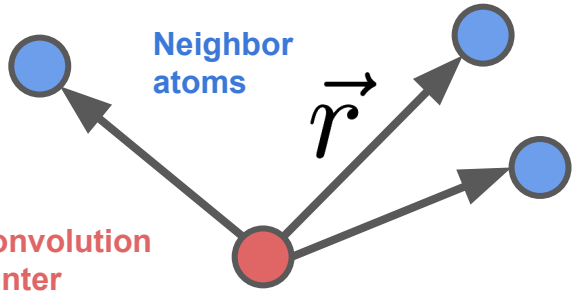
We use points. Images of atomic systems are sparse and imprecise.



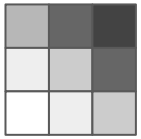
vs.



We use continuous convolutions with atoms as convolution centers.



filter



filter function

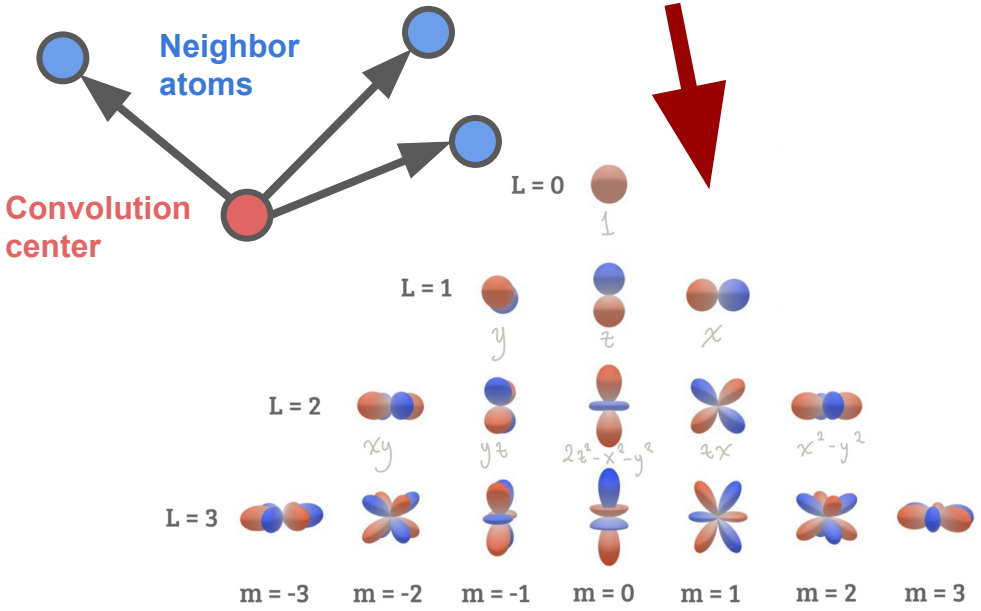
$$W(\vec{r})$$

Using convolutions gives us translation equivariance!

Euclidean Neural Networks are similar to convolutional neural networks...

Rotation equivariant convolutional filters are based on learned radial functions and spherical harmonics...

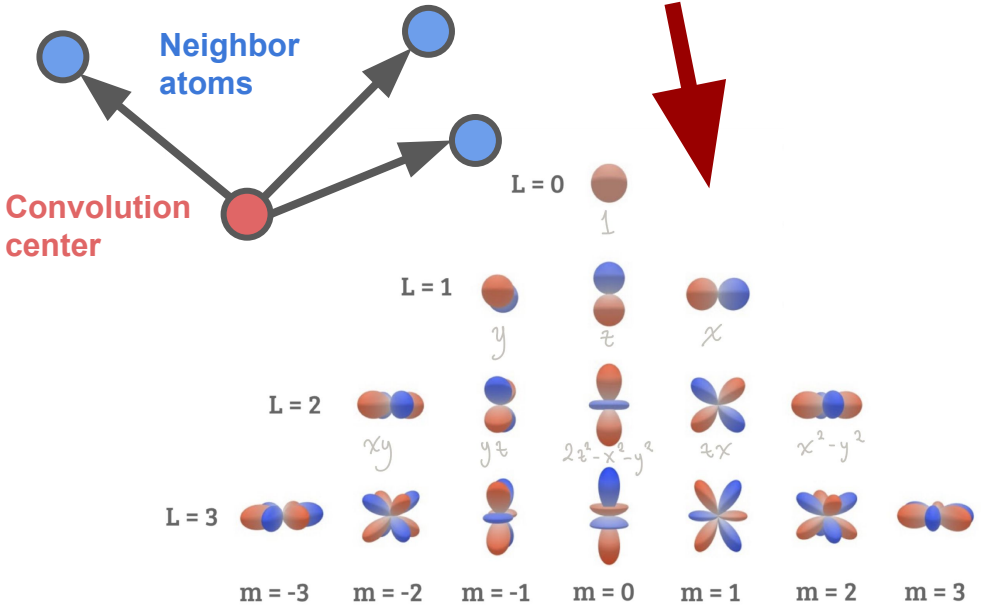
$$W(\vec{r}) = R(r)Y_l^m(\hat{r})$$



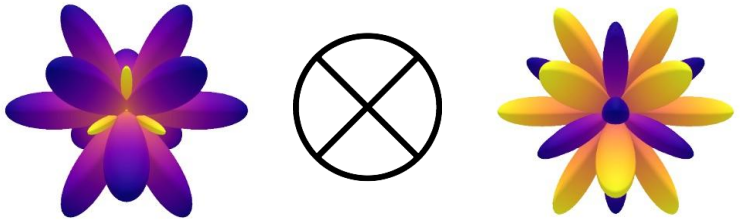
Euclidean Neural Networks are similar to convolutional neural networks...

Rotation equivariant convolutional filters are based on learned radial functions and spherical harmonics...

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



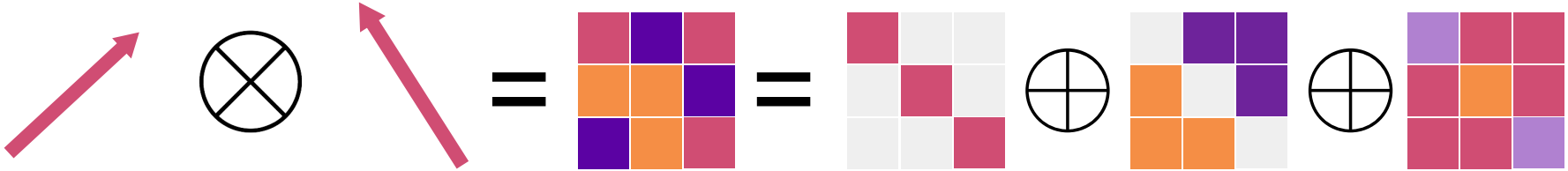
...and in order to interact our filters with our inputs we need geometric tensor algebra.



Interactions in equivariant models are more complex.

How do we interact invariant objects? Scalar multiplication.

How do we interact equivariant objects? Tensor products!



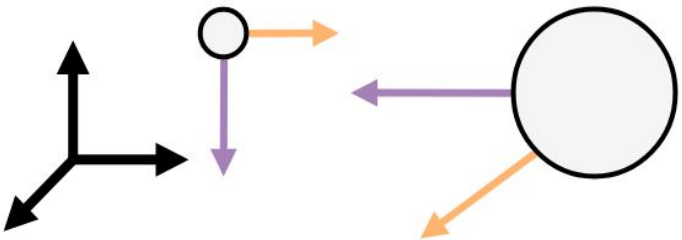
dot product
trace
invariant
L=0
1 degree of freedom

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

symmetric
traceless
equivariant
L=2
5 degrees of freedom

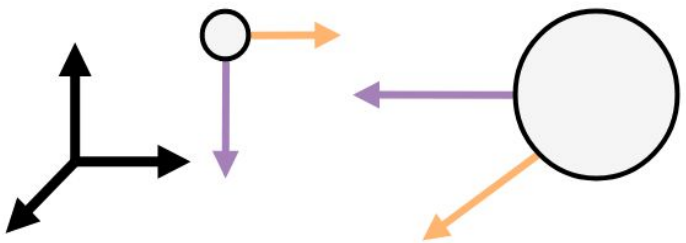
Generalizes to higher orders. Same mathematics that describes atomic interactions, e.g. selection rules in spectroscopy.

The input to our network is geometry and 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]
]
...
```

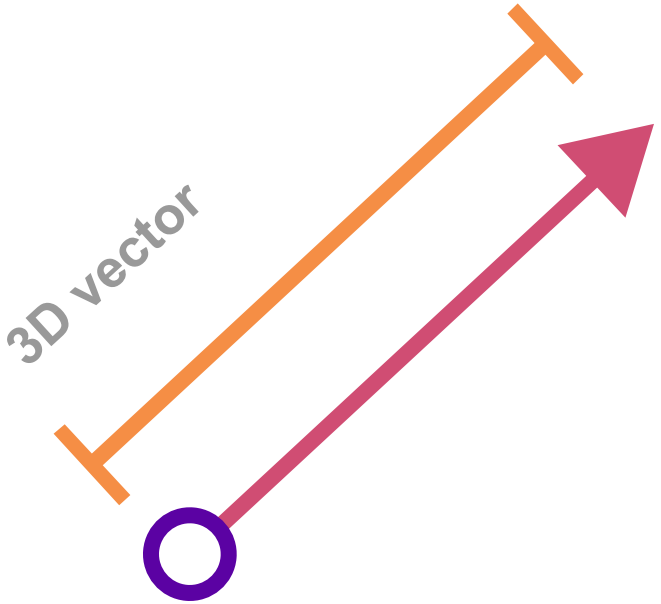
The input to our network is geometry and features on that geometry.
We categorize our features by how they transform under rotation and parity
as *irreducible representations of O(3).*

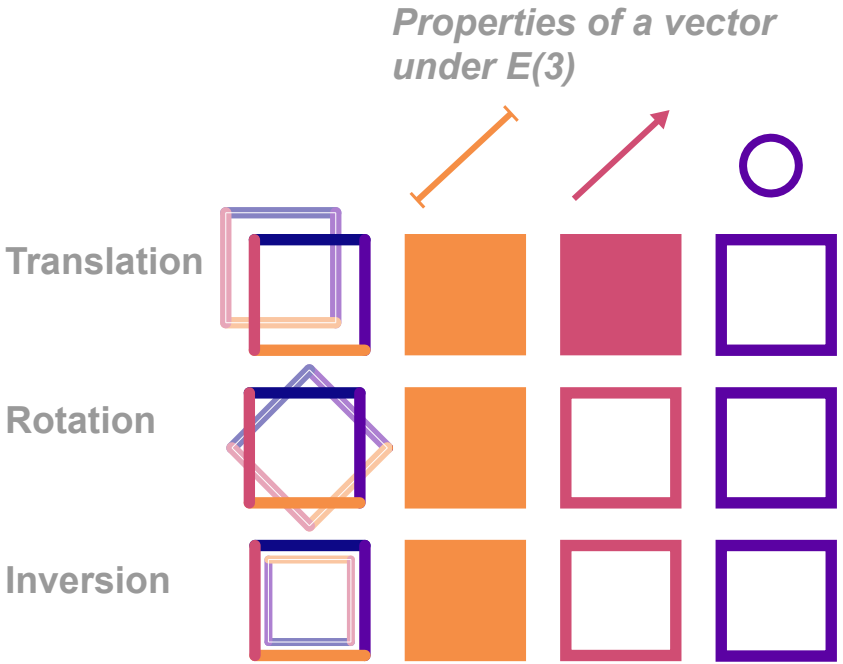


```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
scalar = e3nn.o3.Irrep("0e")  # L=0, even p
vector = e3nn.o3.Irrep("1o")  # L=1, odd p
irreps = 1 * scalar + 1 * vector + 1 * vector
```

- All input, intermediate, and output data is “typed” by its transformation properties.
- All operations must respect this “typing”.

Invariant (scalar) quantities **don't change** under a change of coordinate systems while **equivariant (tensor)** quantities change ***deterministically***.



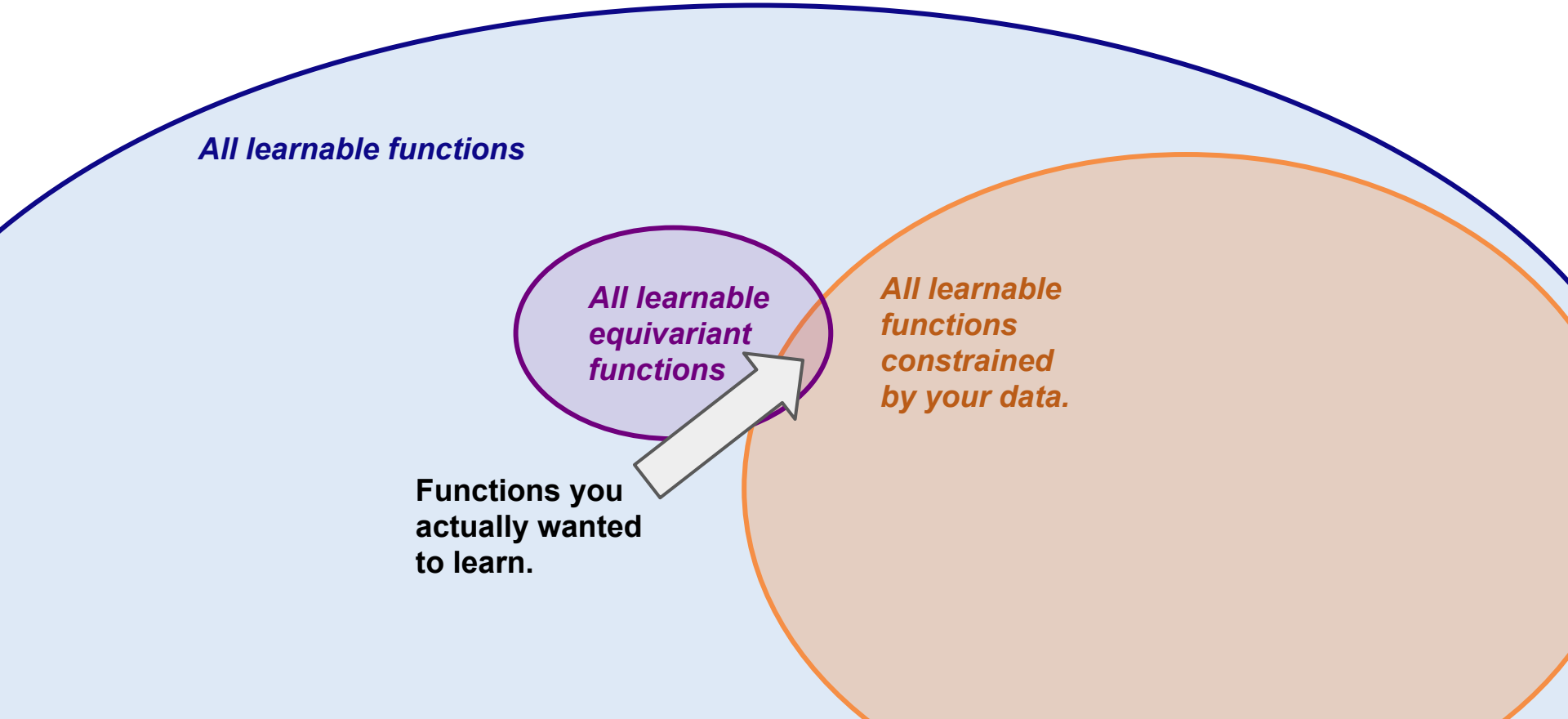


Invariant models operate on **invariant** quantities.
Equivariant models operate on **invariant** AND **equivariant** quantities.

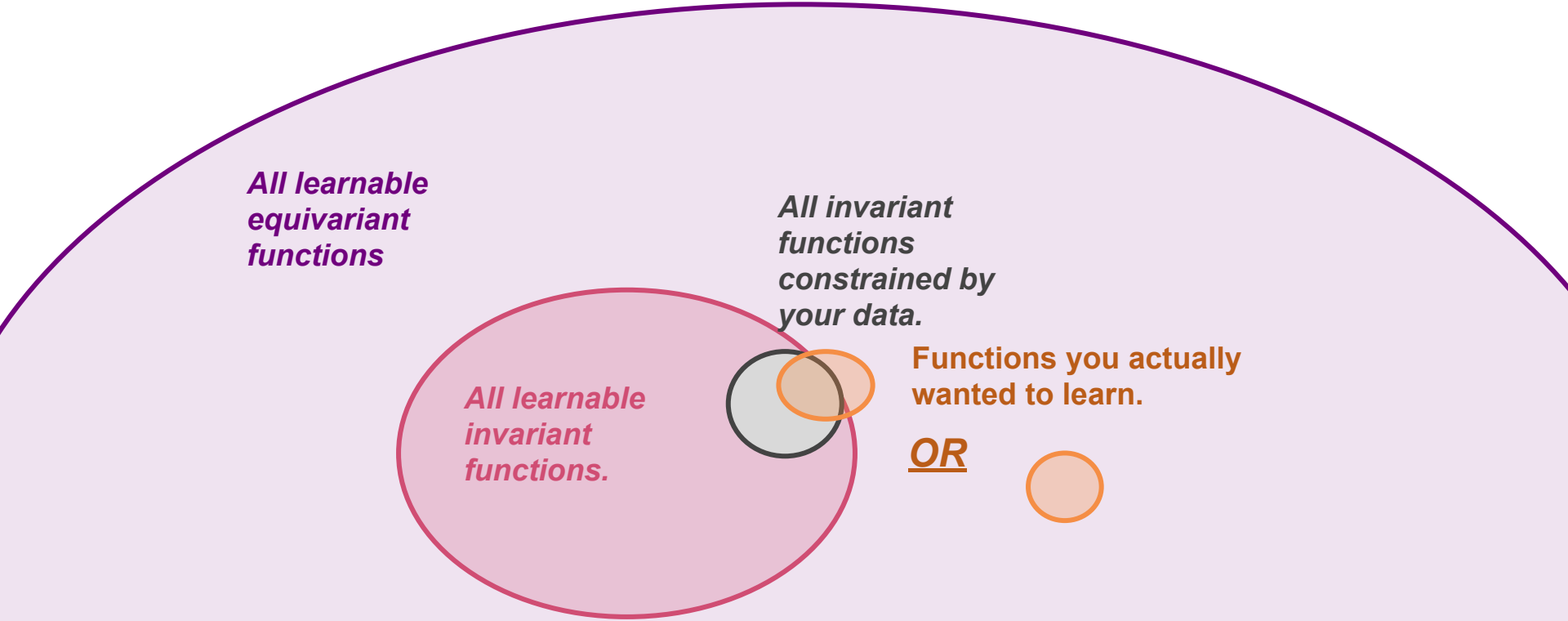
Why invariant models? Mathematically simpler and *so far* effective.
But equivariant models are starting to show that they're more expressive and efficient.

Afterall, physics is equivariant.

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



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



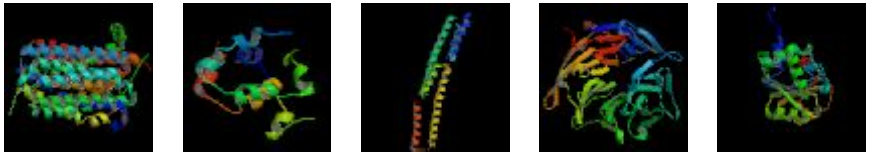
News item 1 of 2:

Deepmind's AlphaFold2 achieved an accuracy of >90% on the CASP14 (2020) challenge.

CASP14

Critical **A**ssessment of Techniques for Protein **S**tructure **P**rediction

Goal: Amino Acid Sequence ⇨ 3D Folded Structure



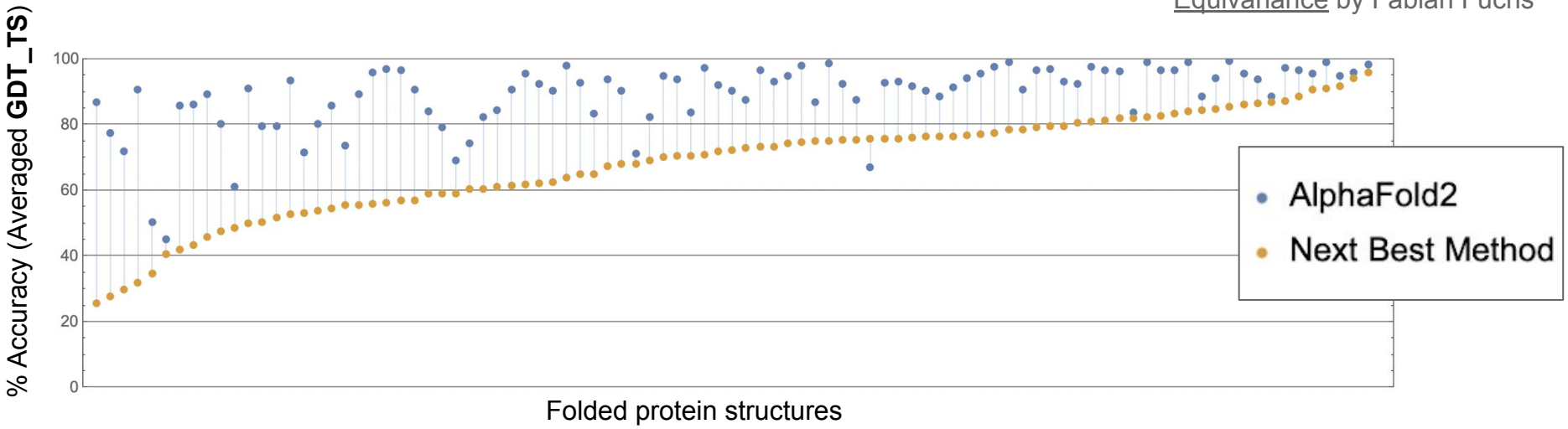
Images from CASP.

Plot and quote from AlphaFold2 @ CASP14 by Mohammed AlQurashi

Also see: AlphaFold 2 & Equivariance by Fabian Fuchs

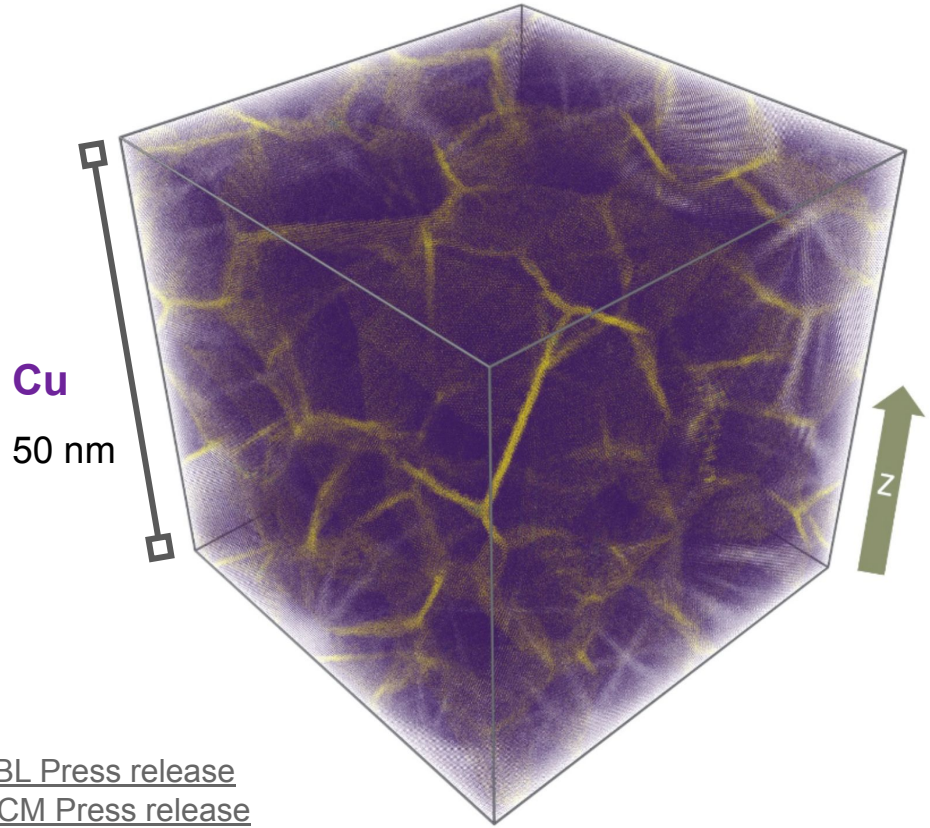
***One thing that AlphaFold2 had that others didn't*:**
neural networks with Euclidean symmetry

"Iterative refinement using SE(3)-equivariant transformers"



News item 2 of 2:

The 2020 Gordon Bell Prize (the Nobel Prize of Supercomputing) **was awarded to DeePMD for** "pushing molecular dynamics with quantum mechanics accuracy to **100 million atoms** with machine learning."

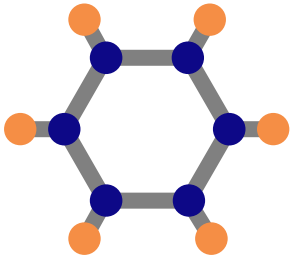


Our methods are *1000x more data efficient* than DeePMD (more accurate with less data) because we have Euclidean symmetry built in.

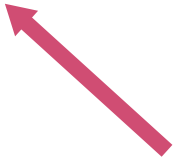
This is the difference between being able to scale density functional theory $O(n^3)$ vs. coupled cluster $O(n^6)$ - $O(n^8)$.

LBL Press release
ACM Press release
Paper: https://arxiv.org/pdf/2005.00223.pdf

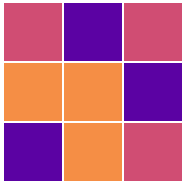
To describe physical systems in 3D, we use coordinates and coordinate systems.



- atomic coordinates
- atomic orbitals



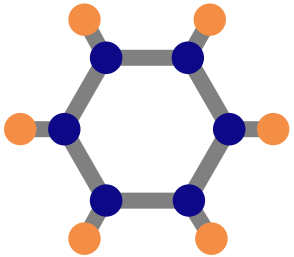
- dipole moments
- forces



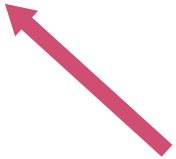
- polarizabilities
- susceptibilities

...

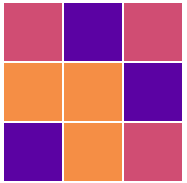
To describe physical systems in 3D, we use coordinates and coordinate systems.



- atomic coordinates
- atomic orbitals



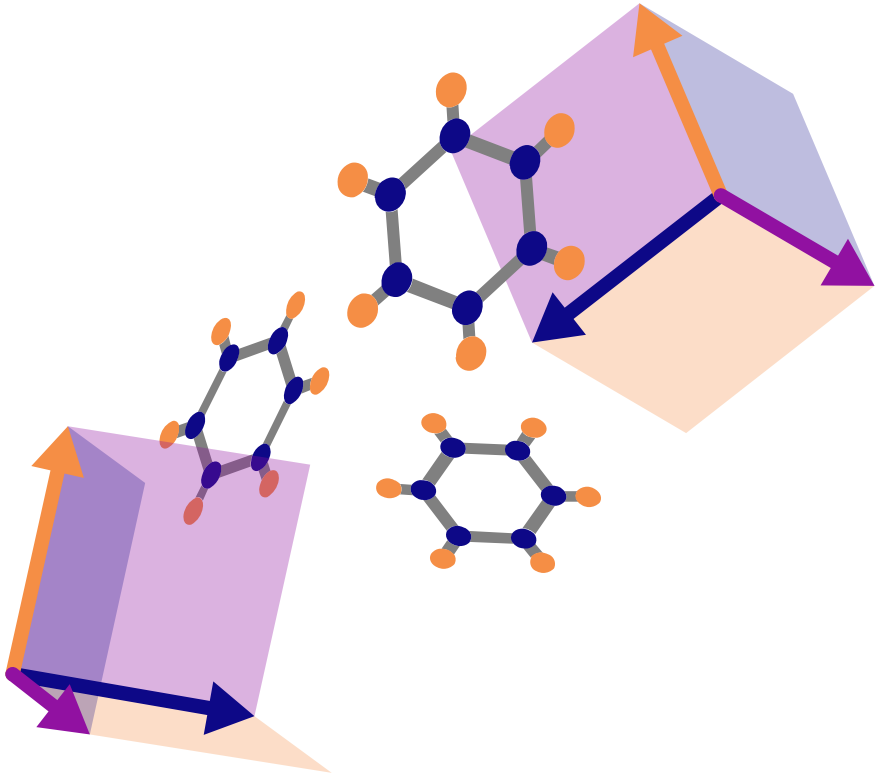
- dipole moments
- forces



- polarizabilities
- susceptibilities

...

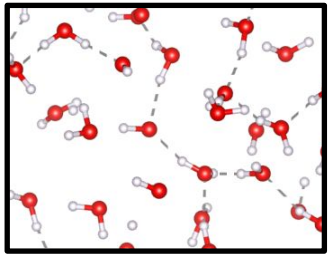
We transform coordinate systems using 3D rotations, translations, and inversion -- the symmetries of 3D Euclidean space.

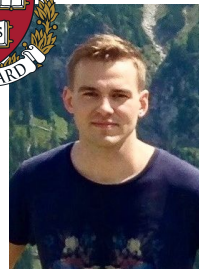


Ab initio Molecular dynamics with E(3)NNs (Nequip)

Molecular, liquid, reaction, amorphous and crystalline datasets.
Out-performs kernel models with the same amount of data!
Extremely data-efficient!

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials (arXiv:2101.03164)





Simon Batzner

Boris Kozinsky

1000x less data!

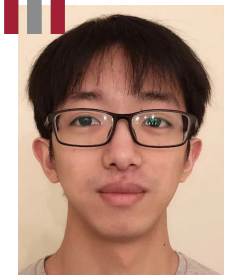
News item 2 of 2

System	NequIP, 133 data points	DeepMD, 133,500 data points
Liquid Water	35.9	40.4
Ice Ih (b)	25.9	43.3
Ice Ih (c)	16.6	26.8
Ice Ih (d)	13.5	25.4

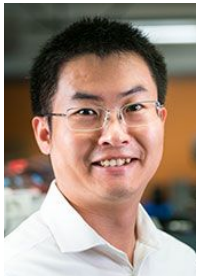
TABLE III: Root mean square error (RMSE) of force components on liquid water and the three ices in units of [meV/A]. Note that the NequIP model was trained on $< 0.1\%$ of the training data of DeepMD.

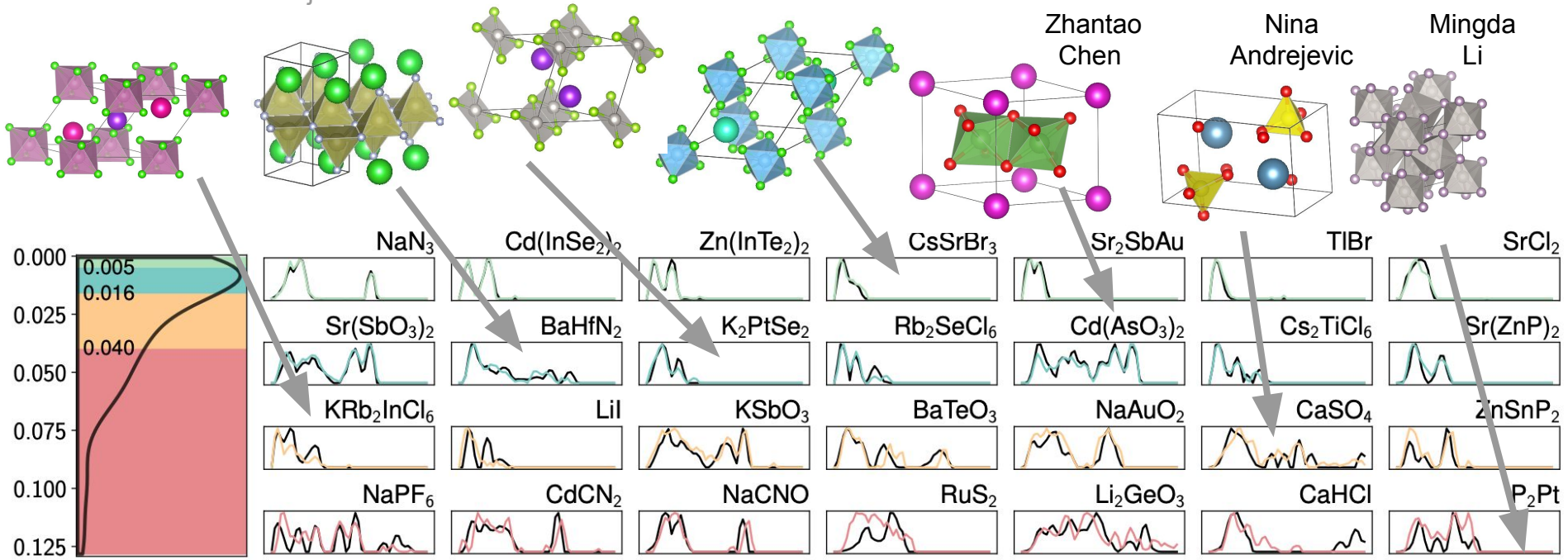
Thermal properties of crystals

E(3)NNs for phonon density of states (DOS)
Accepted to Advanced Science (arXiv:2009.05163)
Training set of 1,200 crystal structures with 64 atom types.
Test set includes atom types never seen. Used to find high C_v.
Data from Materials Project


Zhantao Chen


Nina Andrejevic


Mingda Li



Geometric tensors are the “data types” of 3D space and have <u>many</u> forms.

Examples of “irreps” used by our networks.

Vector

```
Rs_vector = o3.Irrep("1o")
```

Pseudo-vector

```
Rs_pseudovector = o3.Irrep("1e")
```

Double-Headed Ray

```
Rs_doubleray = o3.Irrep("2e")
```

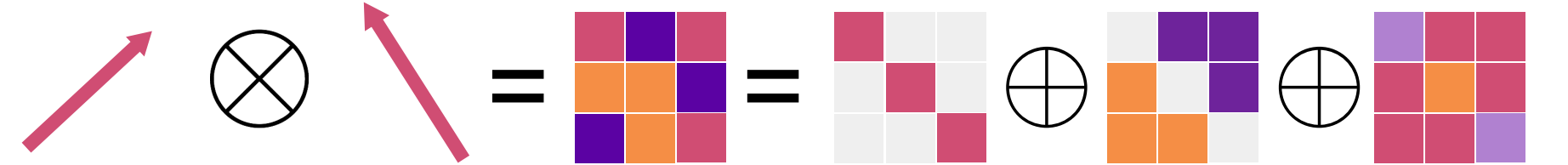
Spiral

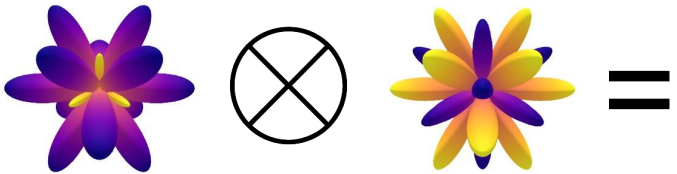
```
Rs_spiral = o3.Irrep("2o")
```

Euclidean neural networks have special operations for interacting geometric tensors (irreps).

How do we interact two numbers? Scalar multiplication.

How do we interact two 3D vectors and more complicated geometric tensors? Tensor products!

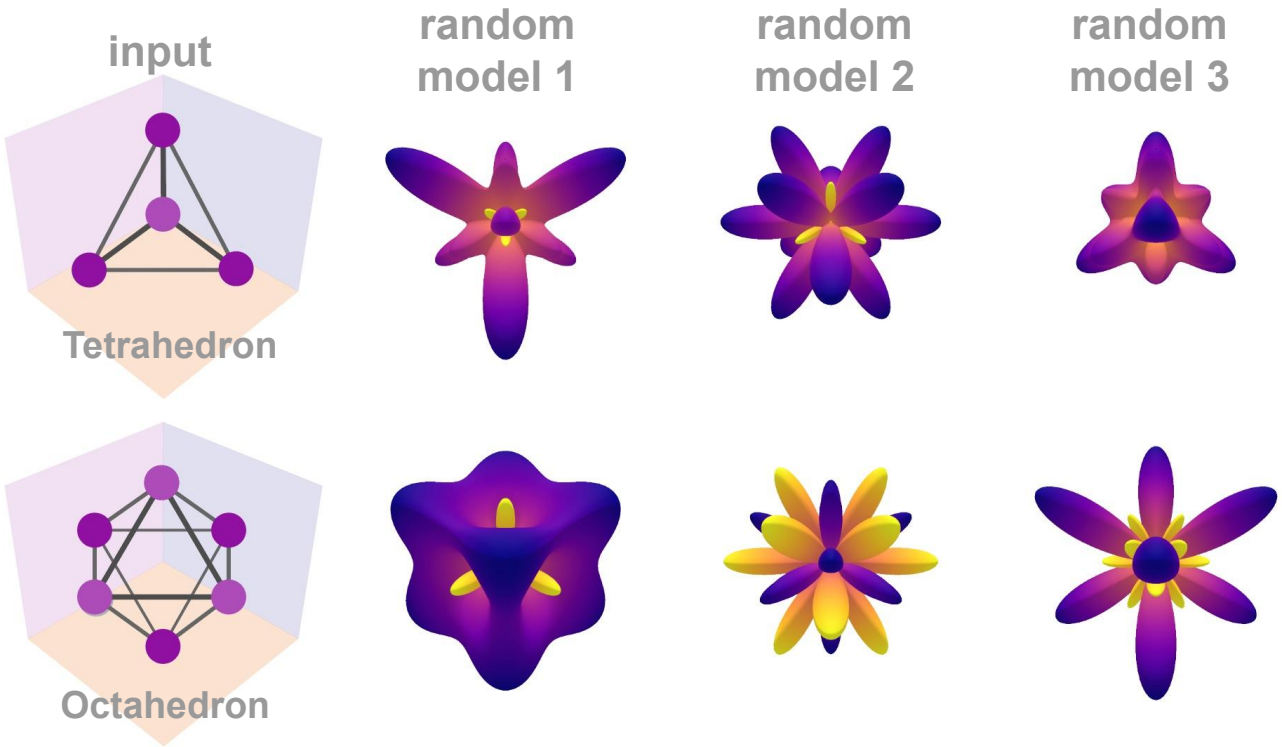




. . . (a bit more complicated)

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."**



Euclidean neural networks: neural networks + representation theory

Euclidean neural networks: neural networks + representation theory

neural networks (in 1 slide):

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

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

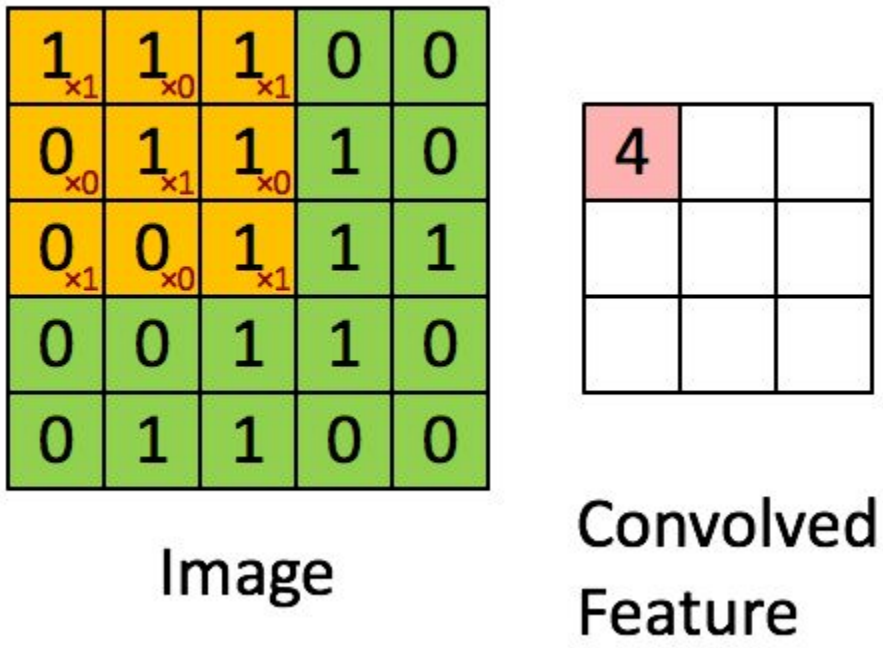
Neural networks must be differentiable so we can update the weights with...

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

Euclidean neural networks: neural networks + representation theory

convolutional neural networks:

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



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

Euclidean neural networks: neural networks + representation theory

convolutional neural networks:

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

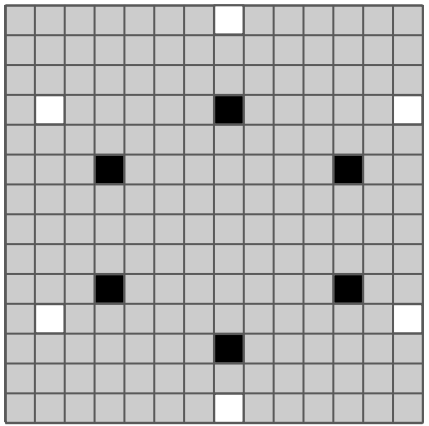
http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Euclidean neural networks: neural networks + representation theory

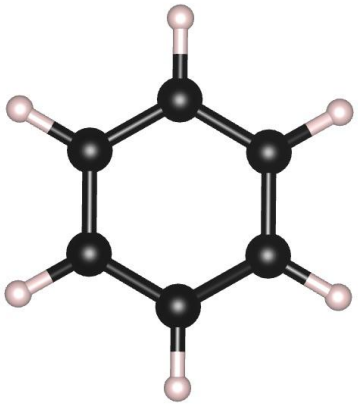
convolutional neural networks:

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

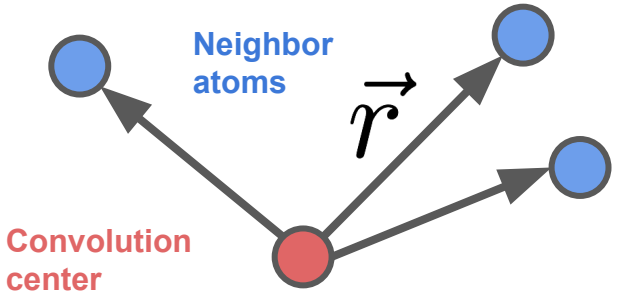
We use points. Images of atomic systems are sparse and imprecise.



vs.

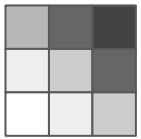


We use continuous convolutions with atoms as convolution centers.



Using convolutions gives us translation equivariance!

filter



filter function

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

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

In this case, g ∈ E(3) and weights are scalars, D(g) = I.

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

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

In this case, g ∈ E(3) and weights are scalars, D(g) = I.

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

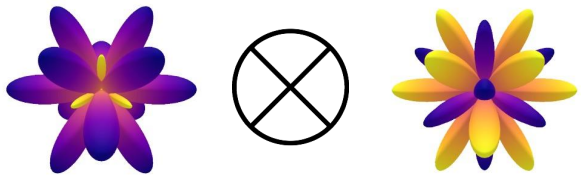
Convolutional filters have a specific functional form.

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

learnable radial functions

spherical harmonics

Scalar multiplication is replaced by geometric tensor products.



We've implemented the math so you don't have to!

e3nn: a modular PyTorch framework for Euclidean neural networks

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

Making Euclidean neural networks practical and efficient

Utilities and classes for

- building E(3) equivariant neural networks
- manipulating geometric tensors
- visualizing spherical harmonics

core-developers of e3nn

Mario Geiger

EPFL

Ben Miller

Kostiantyn Lapchevskyi

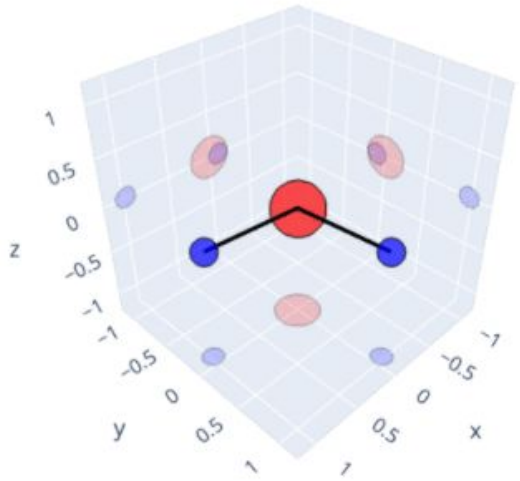
Mila

Tess Smidt

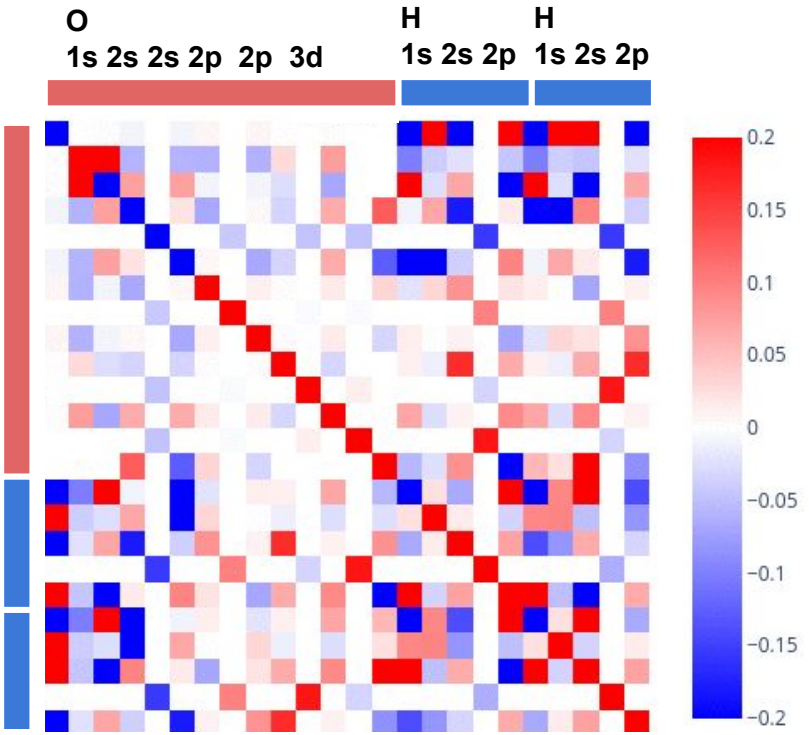
BERKELEY LAB

E(3)NNs can express complex tensors of atomic orbitals and predict molecular Hamiltonians in any orientation from seeing a single example.

$$R\vec{v}$$

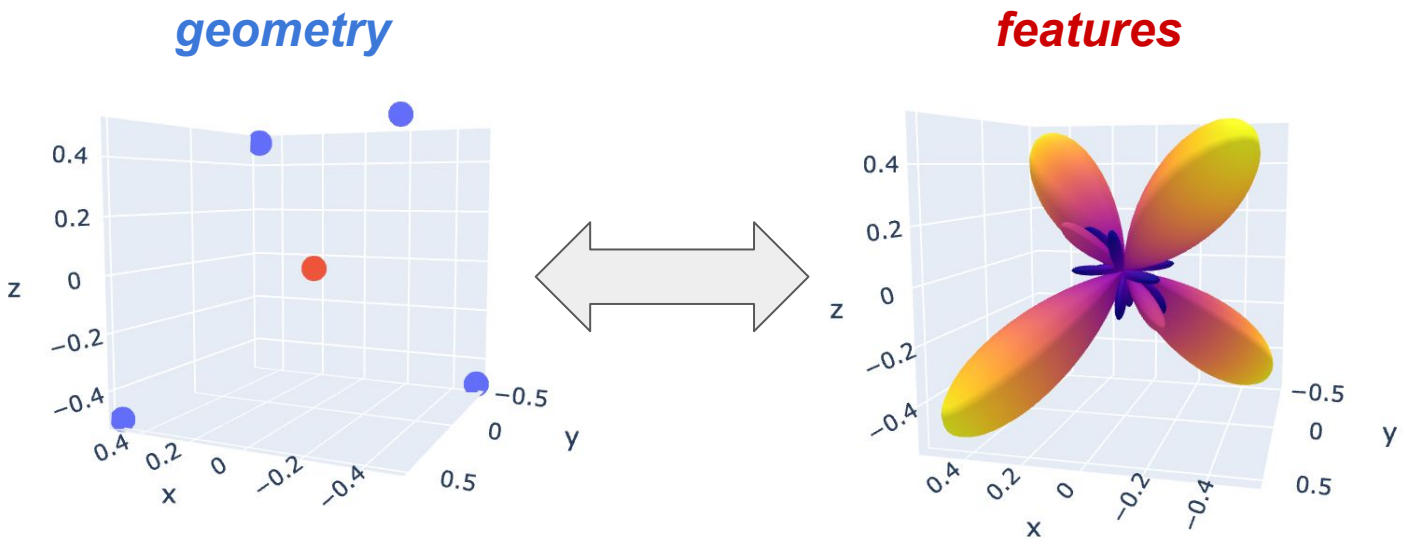


$$RHR^T$$



E(3)NNs can manipulate geometry,
which means they can be used for generative models of e.g. point geometry.

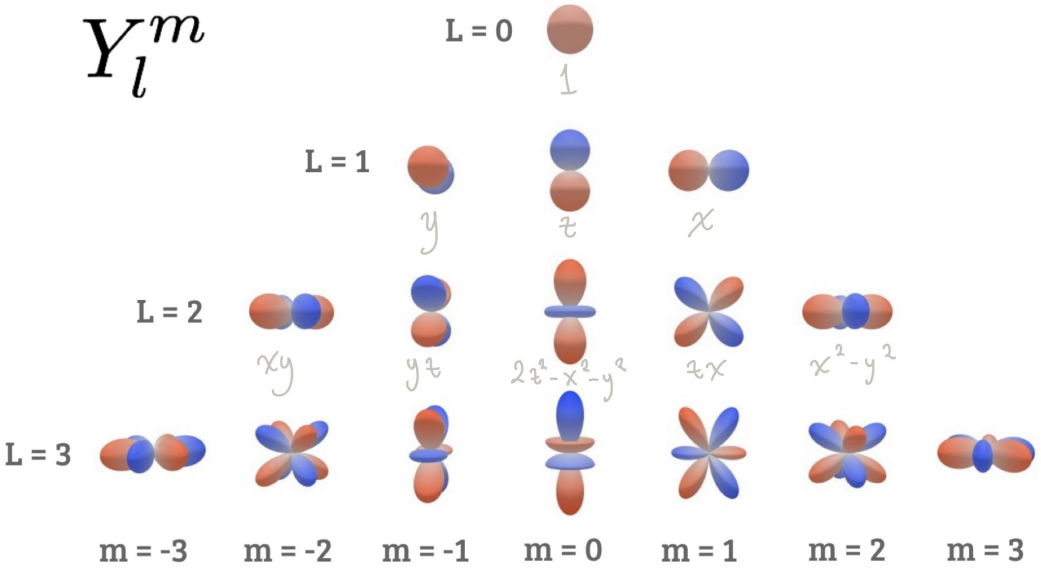
We can convert local *geometry* into *features* and *vice versa*
via spherical harmonic projections.



Geometric tensors are the “data types” of 3D space and have <u>many</u> forms.
Examples of irreps used by our networks.

Spherical harmonics

$$Y_l^m$$

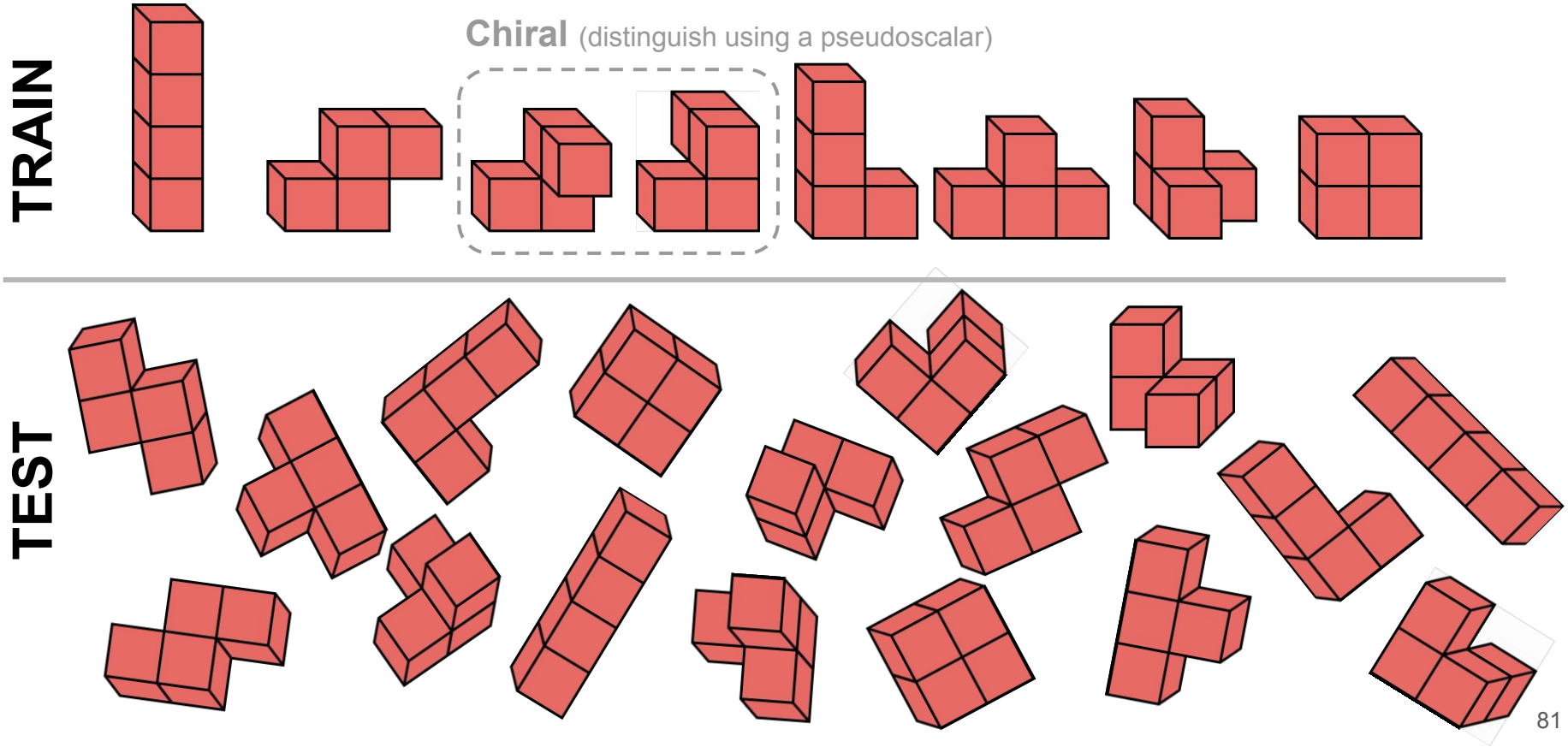



```
Rs_s_orbital = o3.Irrep("0e")

Rs_p_orbital = o3.Irrep("1o")

Rs_d_orbital = o3.Irrep("2e")

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

What does equivariance get you?

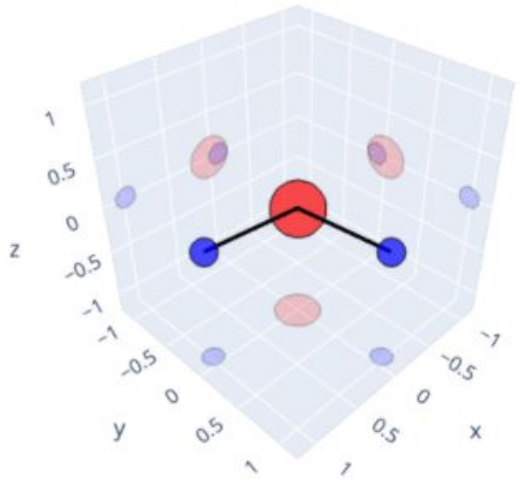
Examples of what my methods, Euclidean neural networks (E3NNs), can do because they are equivariant.

Our unit test: Trained on 3D Tetris shapes in one orientation, these network can perfectly identify these shapes in any orientation.

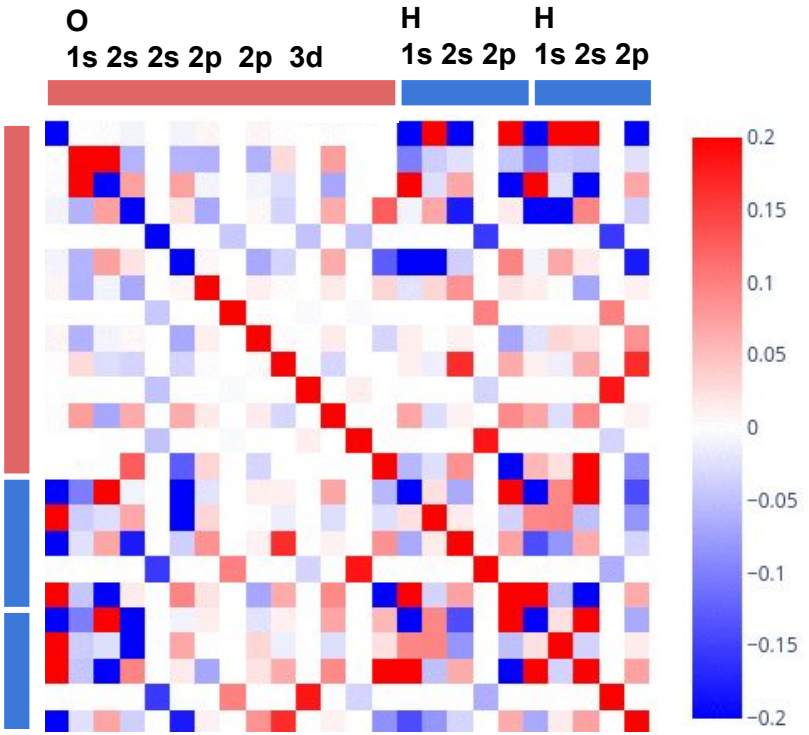


E(3)NNs can express tensors of atomic orbitals and predict molecular Hamiltonians in any orientation from seeing a single example.

$R\vec{v}$

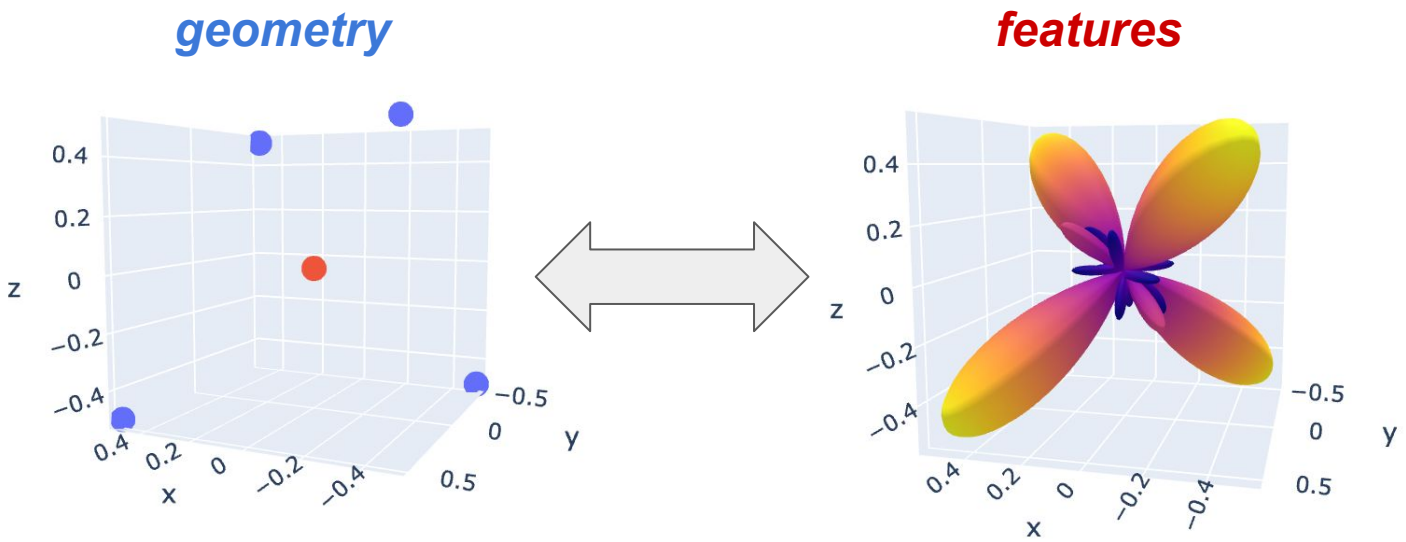


RHR^T



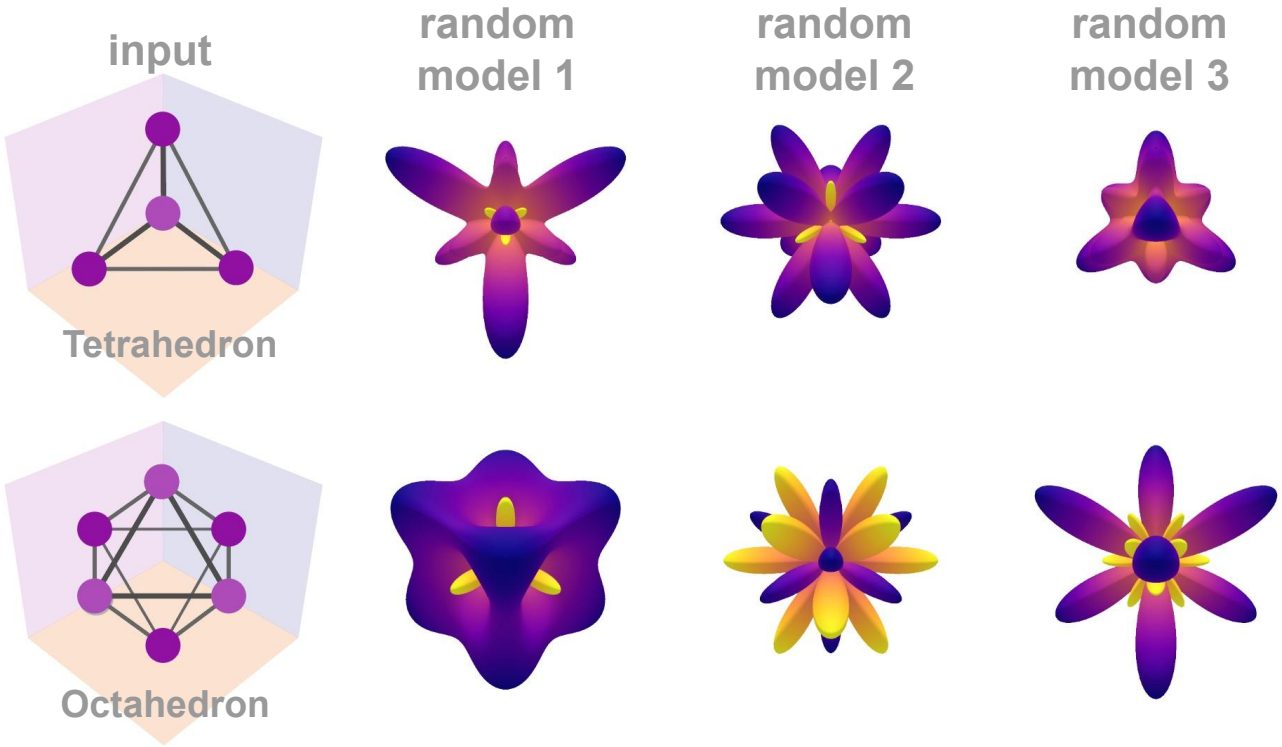
E(3)NNs can manipulate geometry,
which means they can be used for generative models of e.g. point geometry.

We can convert local *geometry* into *features* and *vice versa*
via spherical harmonic projections.



Understanding implications of symmetry: **A E(3)NNs act as a symmetry "compiler". The outputs 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."



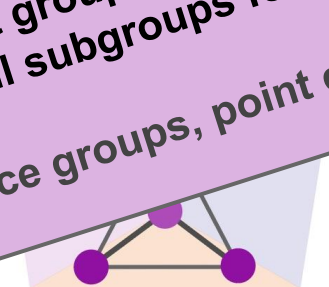
T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

Understanding implications of symmetry: A E(3)NNs act as a symmetry "compiler". The outputs have equal or hig… mmetry than the inputs.

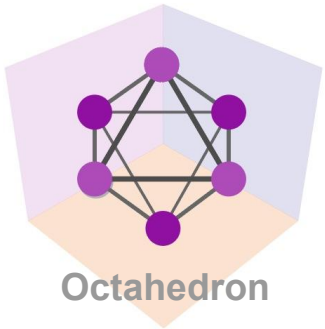
Curie's principle: … show certain asymmetry, this asymmetry must be found … that gave rise to them."

Implement group equivariance and get all subgroups for FREE!

e.g. space groups, point groups

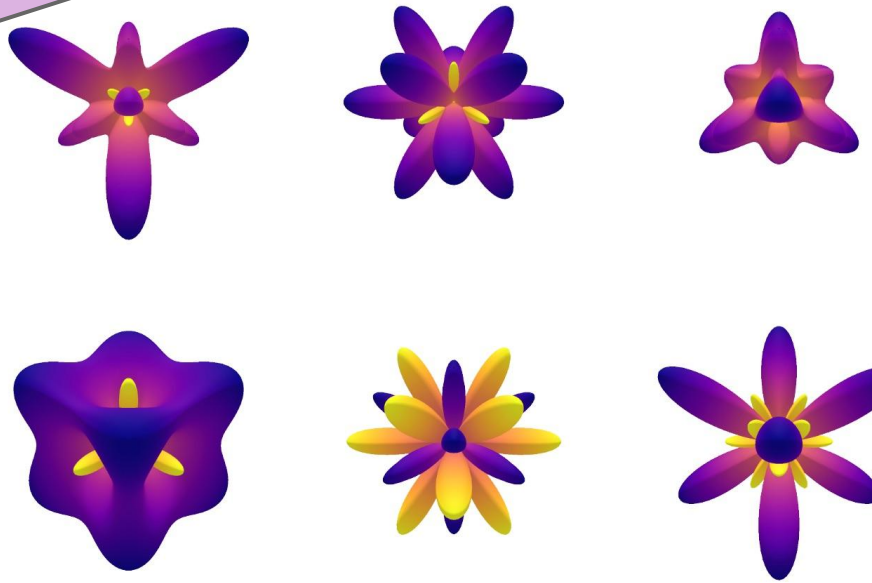


Tetrahedron



Octahedron

m 1 | random model 2 | random model 3

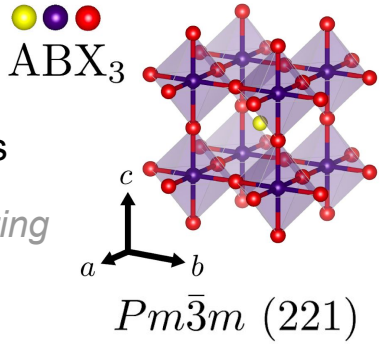


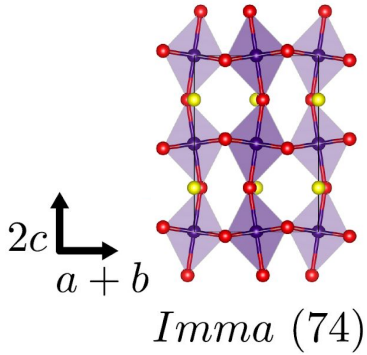
T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

Understanding implications of symmetry: Using the training procedure itself , we can uncover missing symmetry-implied data, unbeknownst to the researcher.

Predict distortion displacements

Octahedral tilting in perovskites



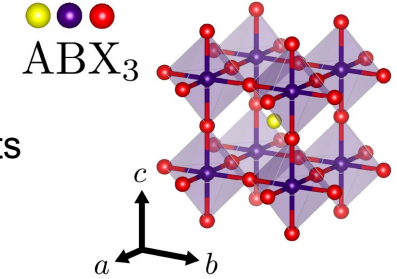


Target distorted structure

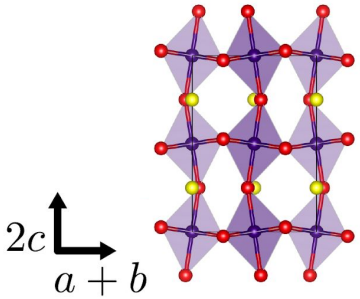
T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

Understanding implications of symmetry: **Using the training procedure itself , we can uncover missing symmetry-implied data, unbeknownst to the researcher.**

Predict distortion displacements

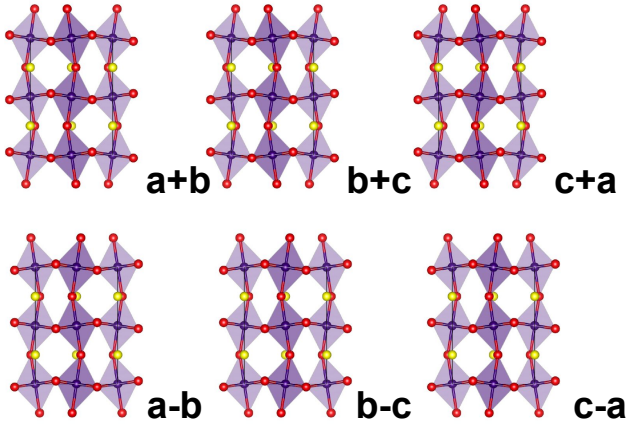


$Pm\bar{3}m$ (221)



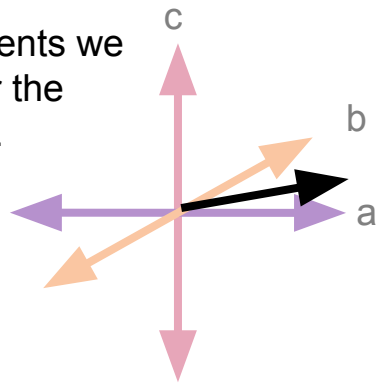
$Imma$ (74)

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 correct one.



T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

more results

NMR

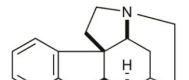
In progress, but **already competitive with current state of the art** with kernel methods and invariant neural networks.

Michael Bailey (University of Toronto),
Eugene Kwan (Merck), Richard Liu (MIT),
Corin Wagen (Harvard)

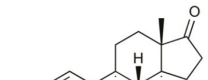
MERCK

Eugene Kwan

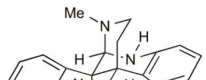
Train models on specific density functional theory calculations



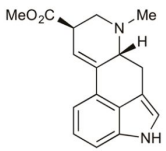
strychnine



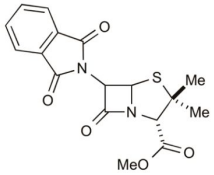
estrone



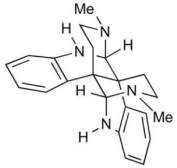
meso-calycanthine



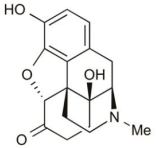
lysergic acid methyl ester



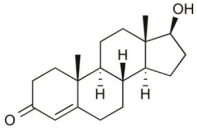
penam β-lactams (4 diastereomers)



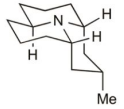
(–)-calycanthine



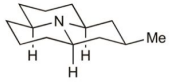
oxymorphone



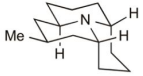
testosterone



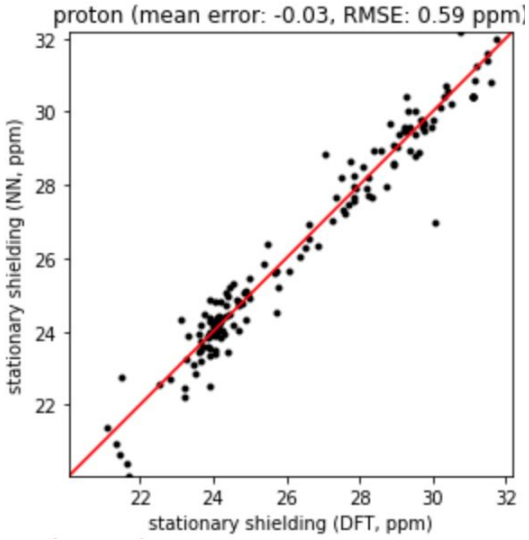
precoccinelline

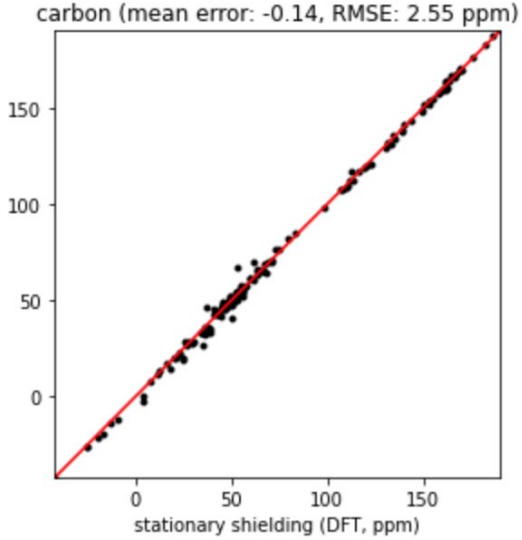


myrrhine



hippodamine

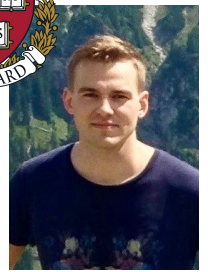




Ab initio Molecular dynamics with E(3)NNs (Nequip)

Molecular, liquid, reaction, amorphous and crystalline datasets.
Out-performs kernel models with the same amount of data!
Extremely data-efficient!

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials (arXiv:2101.03164)



Simon Batzner

Boris Kozinsky

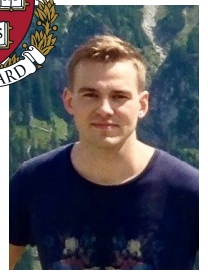
Datasets

- MD-17 small molecule dynamics
  - DFT
  - CCSD(T)
- Liquid Water and Ice Dynamics
  - 1000x more data-efficient than DeePMD
- Heterogeneous catalysis of formate dehydrogenation
- Lithium Phosphate Amorphous Glass Formation
- Superionic Kinetic Transport of Li in Lithium Thiophosphate (LiPS)

Molecular dynamics with E(3)NNs (Nequip)

Molecular, liquid, reaction, amorphous and crystalline datasets.
Out-performs kernel models with the same amount of data!
Extremely data-efficient!

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials (arXiv:2101.03164)



Simon Batzner

Boris Kozinsky

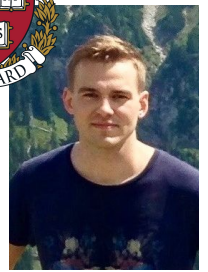
	Equivariant nn	Kernel
Molecule	NequIP	sGDML
Aspirin	14.7	33.0
Benzene	0.8	1.7
Ethanol	9.4	15.2
Malonaldehyde	16.0	16.0
Toluene	4.4	9.1

TABLE II: Force MAE for molecules at CCSD/CCSD(T) accuracy, reported in units of [meV/Å], with 1,000 reference configurations).

Molecular dynamics with E(3)NNs (Nequip)

Molecular, liquid, reaction, amorphous and crystalline datasets.
Out-performs kernel models with the same amount of data!
Extremely data-efficient!

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials (arXiv:2101.03164)



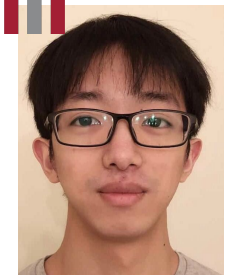
Simon Batzner

Boris Kozinsky

System	Number of atoms	NequIP	Ab-initio	Speed-up
Toluene	15	16 ms	4 hours*	900,000
Formate on Cu	52	58 ms	1045.6 s	18,028

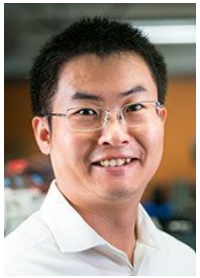
TABLE VI: Time required for a single force call for NequIP in comparison to CCSD(T) for Toluene and DFT for Formate on Cu; * personal communication with Stefan Chmiela and Alexandre Tkatchenko.

Thermal properties of crystals



Zhantao Chen

Nina Andrejevic

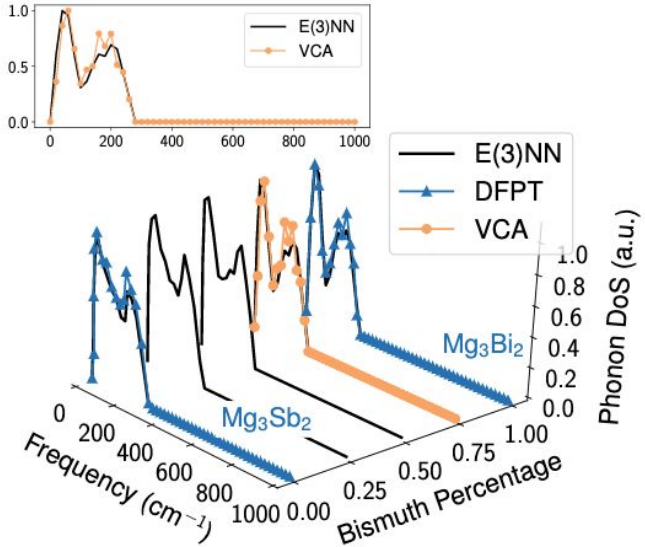


Mingda Li

E(3)NNs for phonon density of states
Accepted to Advanced Science (arXiv:2009.05163)
Training set of 1,200 crystal structures with 64 atom types.
Test set includes atom types never seen. Used to find high C_v.
Data from Materials Project

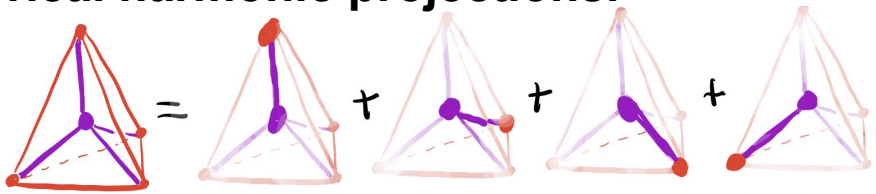
Applications to alloys!

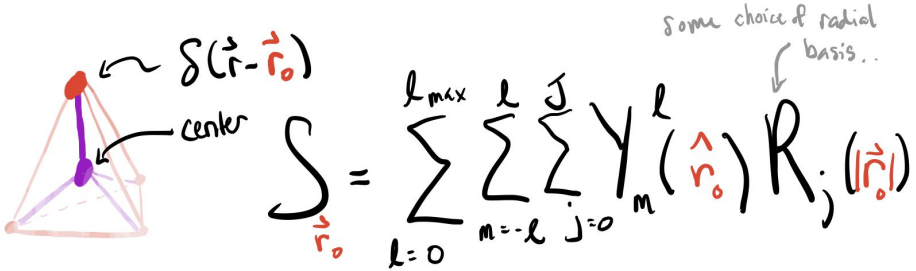
Figure 4. Comparison between E(3)NN model predictions and virtual crystal approximation (VCA) calculations. E(3)NN model can predict phonon DoS for the alloy $Mg_3Sb_{2(1-p)}Bi_{2p}$ of continuous p with two-hot weighted encoding. The triangle-marked curves indicate DFPT results for Mg_3Sb_2 and Mg_3Bi_2 curated from [42] and [51], respectively. The circle-marked curve represent VCA calculations for $Mg_3Sb_{0.5}Bi_{1.5}$ ($p = 0.75$). The inset figure shows the front view for E(3)NN and VCA comparison.



geo

To autoencode, we have to be able to convert *geometry* into *features* and *vice versa*.
We do this via spherical harmonic projections.



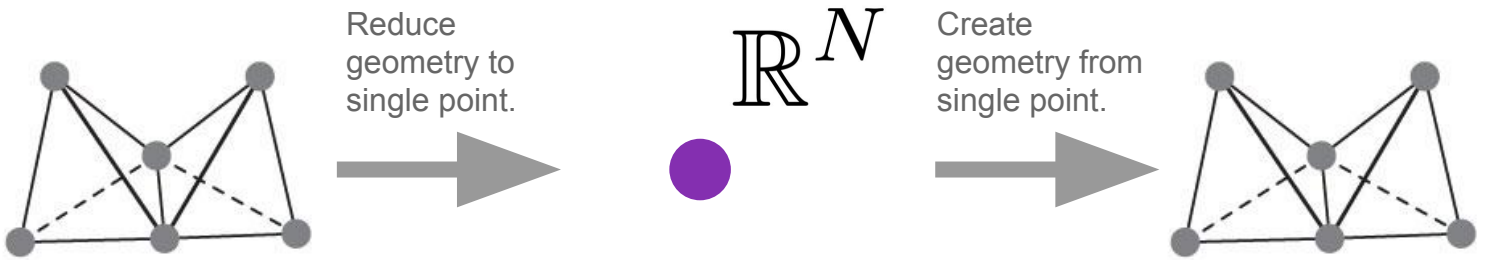


vector size $\left[\sum_{l=0}^{l_{max}} (2l+1) \times J\right]$

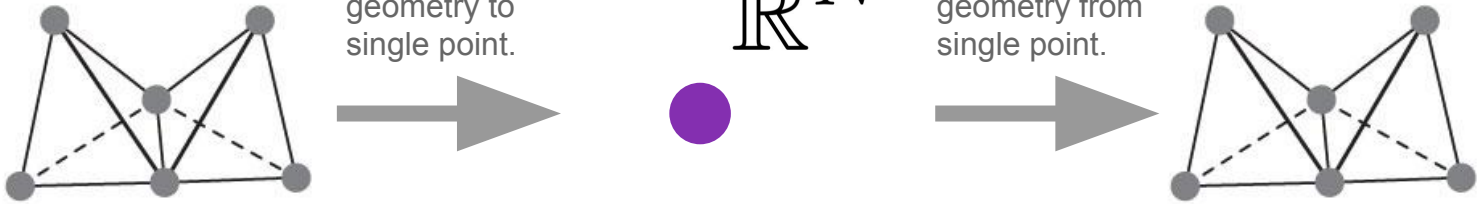
Yay! Linearity!

$$S_{\text{tetrahedron}} = S_{\vec{r}_0} + S_{\vec{r}_1} + S_{\vec{r}_2} + S_{\vec{r}_3}$$

We want to convert geometric information (3D coordinates of atomic positions) **into features on a trivial geometry** (a single point) **and back again.**



We want to convert geometric information (3D coordinates of atomic positions) **into features on a trivial geometry** (a single point) **and back again.**

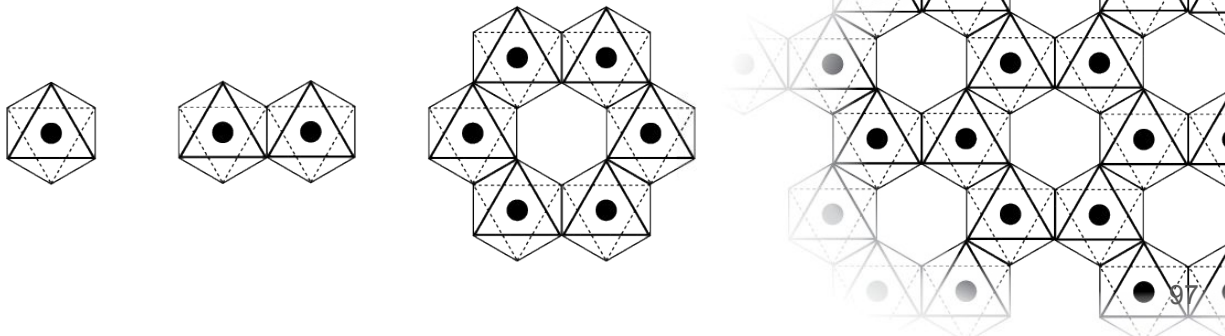


Discrete geometry

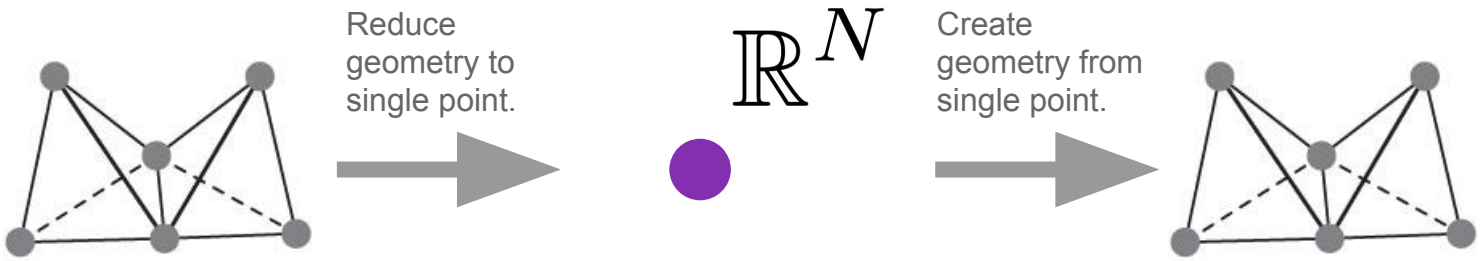
Single point with continuous latent representation (N dimensional vector)

Discrete geometry

Atomic structures are <u>hierarchical</u> and can be constructed from <u>recurring geometric motifs</u>.



We want to convert geometric information (3D coordinates of atomic positions) into features on a trivial geometry (a single point) and back again.



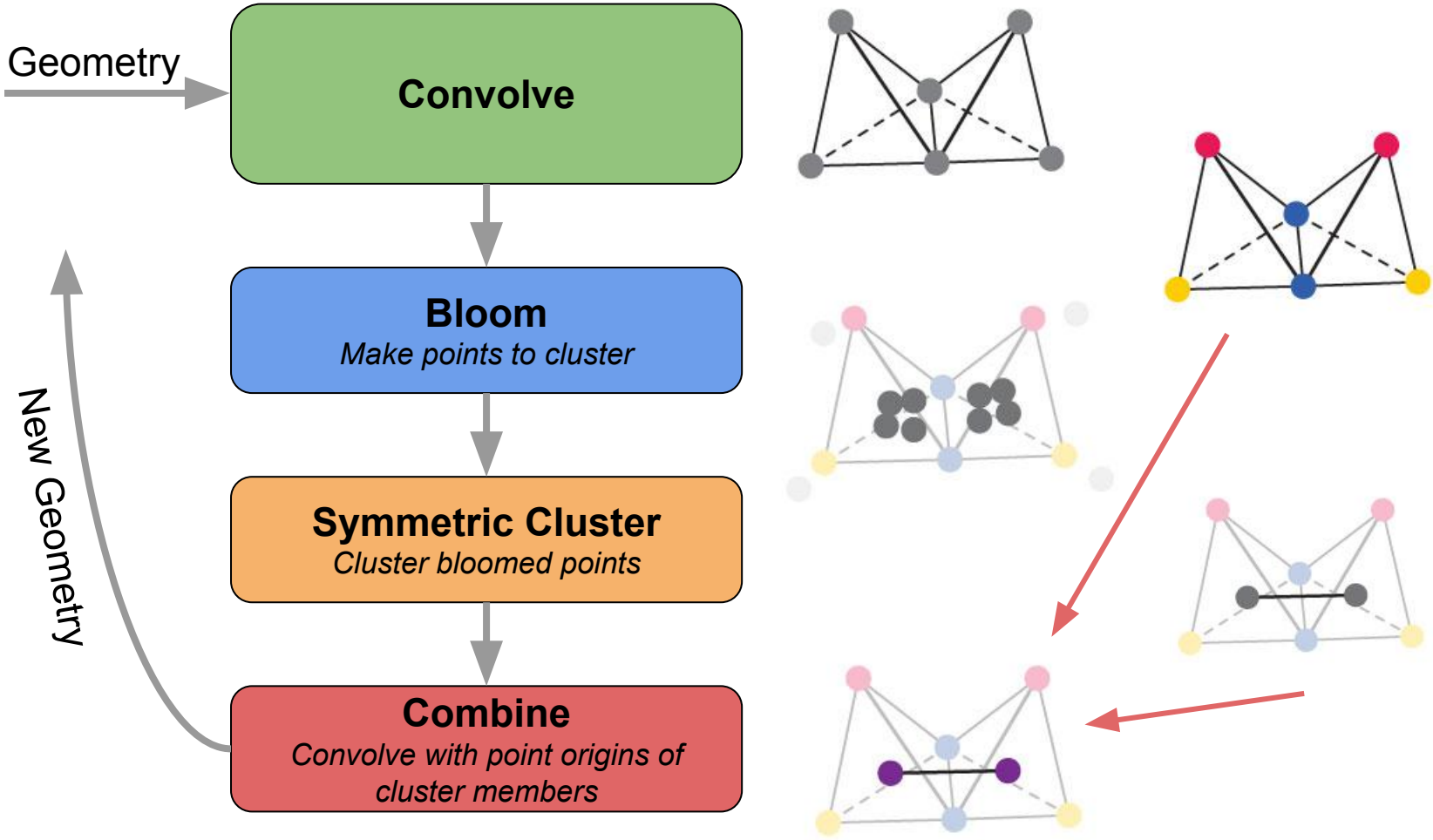
Atomic structures are <u>hierarchical</u> and can be constructed from <u>recurring geometric motifs</u>.

\+ Encode geometry
\+ Encode hierarchy

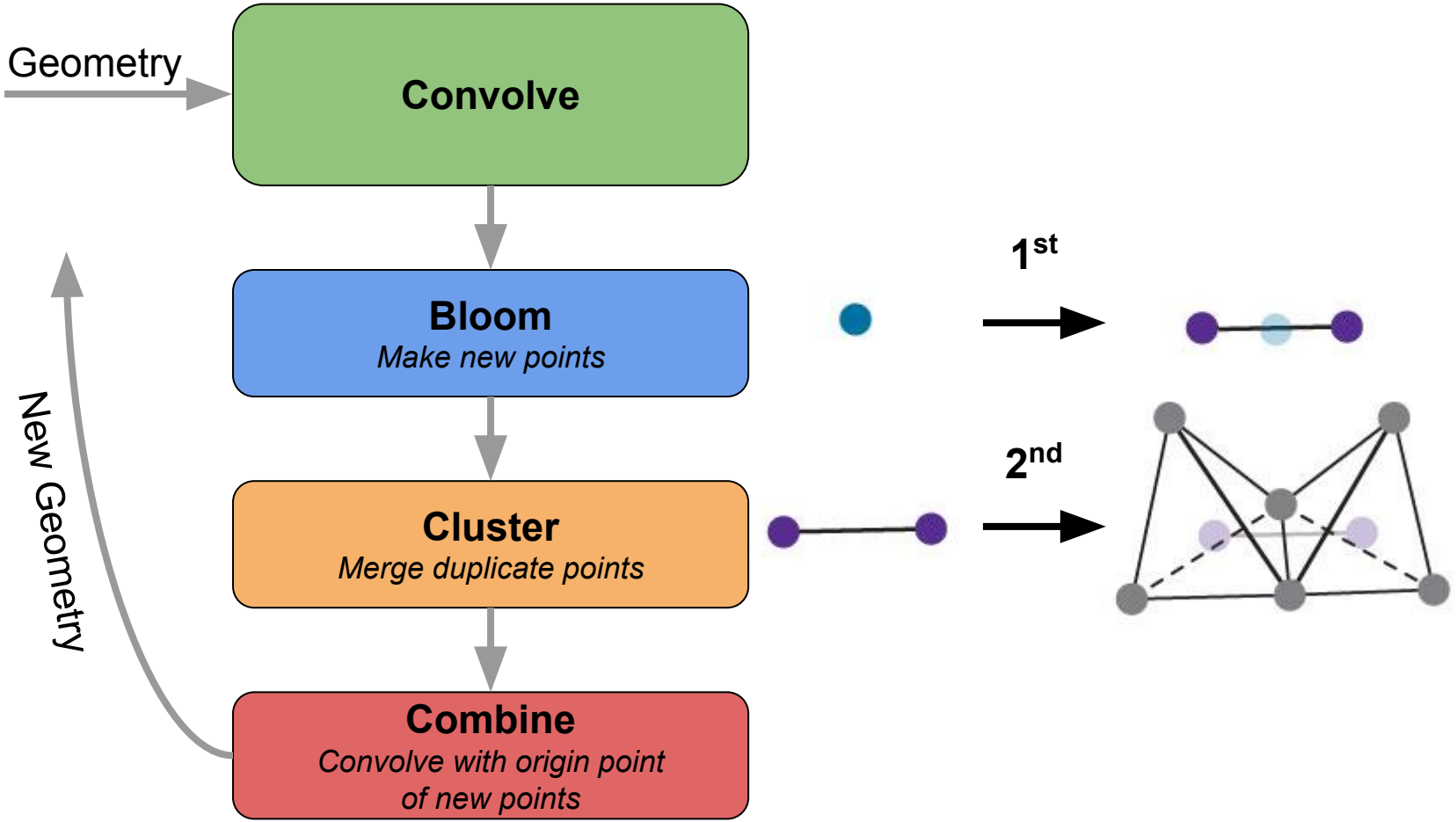
\+ Decode geometry
\+ Decode hierarchy

(Need to do this in a recursive manner)

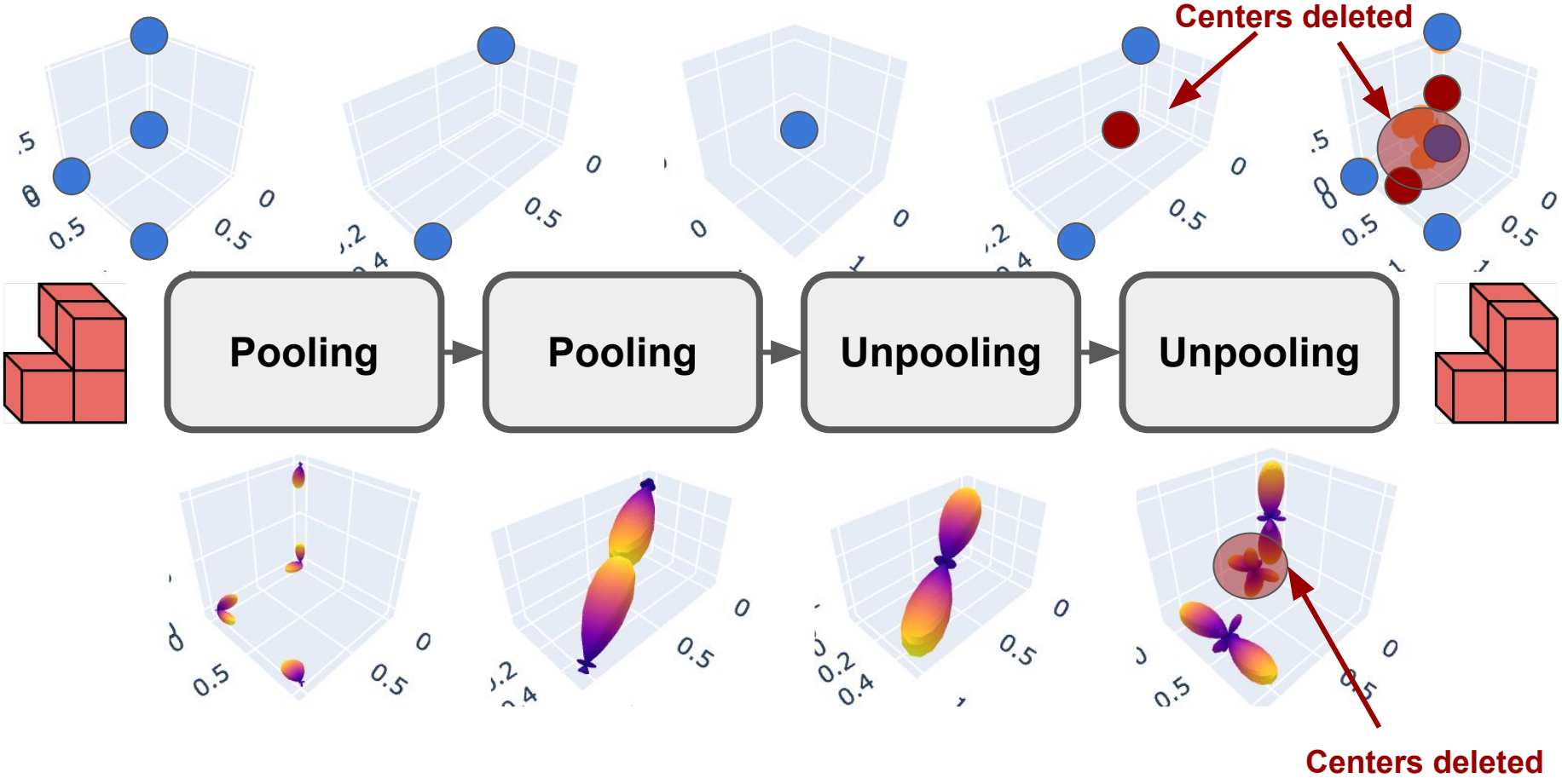
How to encode *(Pooling layer).* Recursively convert geometry to features.



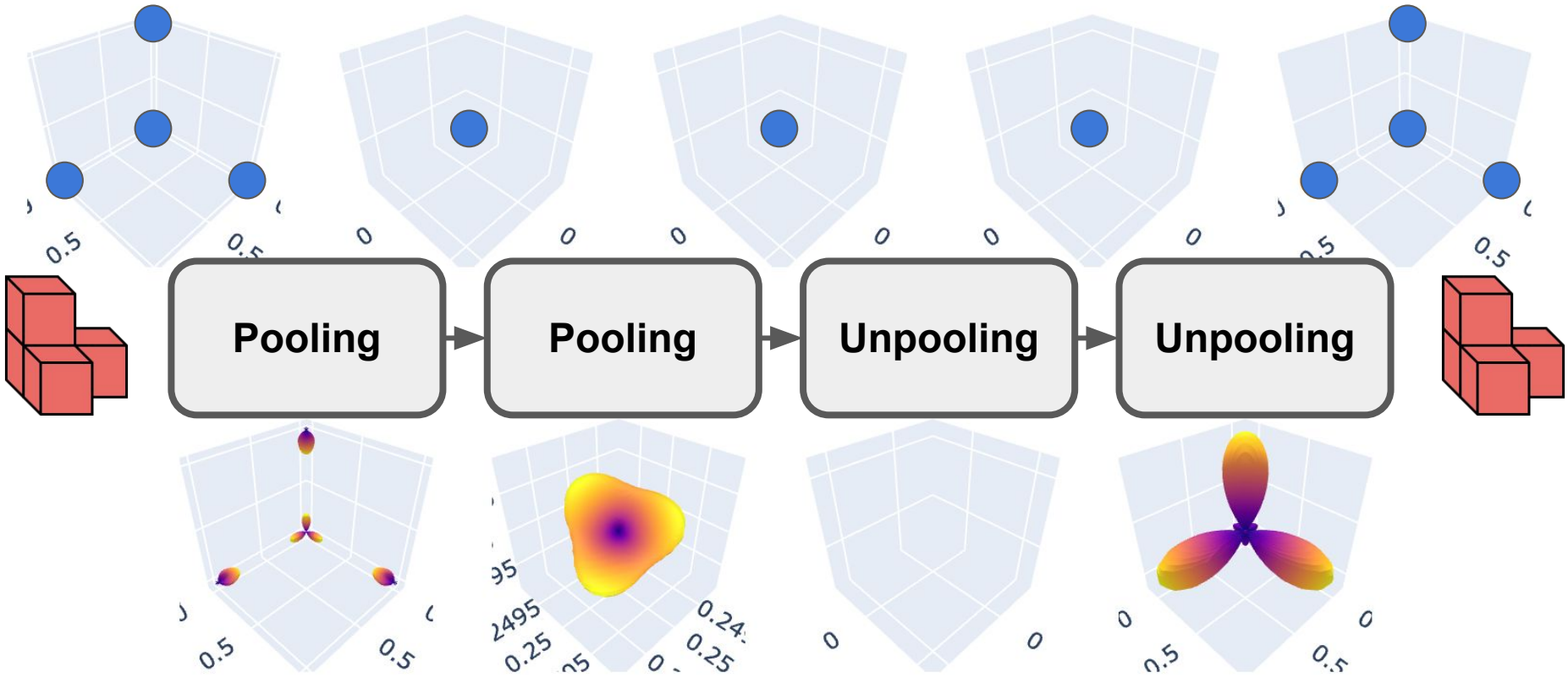
How to decode *(Unpooling layer).* Recursively convert features to geometry.



We can also build an autoencoder for geometry: e.g. Autoencoder on 3D Tetris



We can also build an autoencoder for geometry: e.g. Autoencoder on 3D Tetris

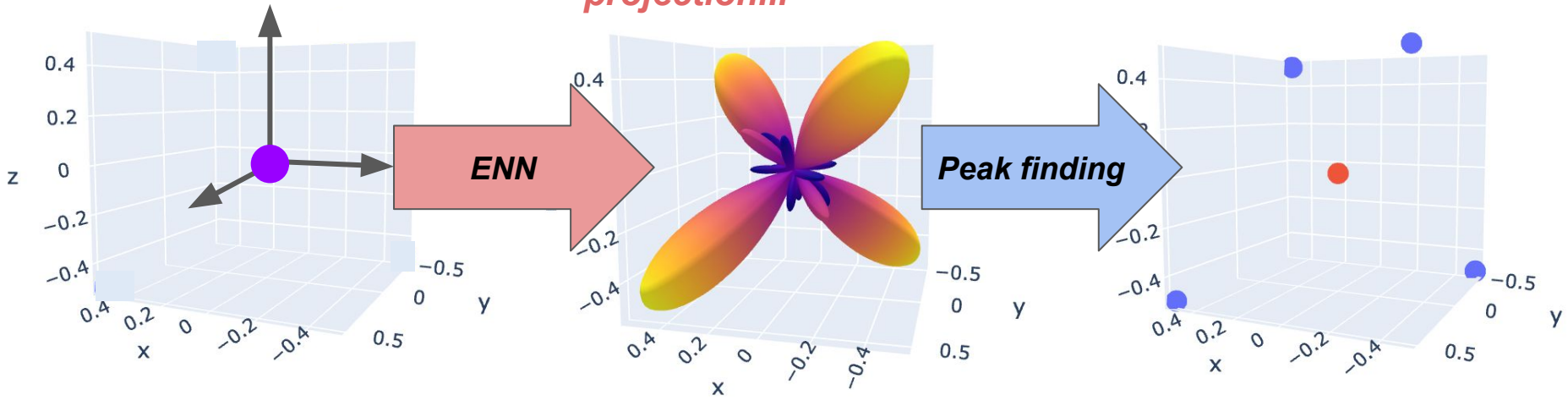


Equivariant neural networks can learn to invert invariant representations.

Invariant features + coordinate frame

Network can predict spherical harmonic projection...

Which can be used to recover geometry.



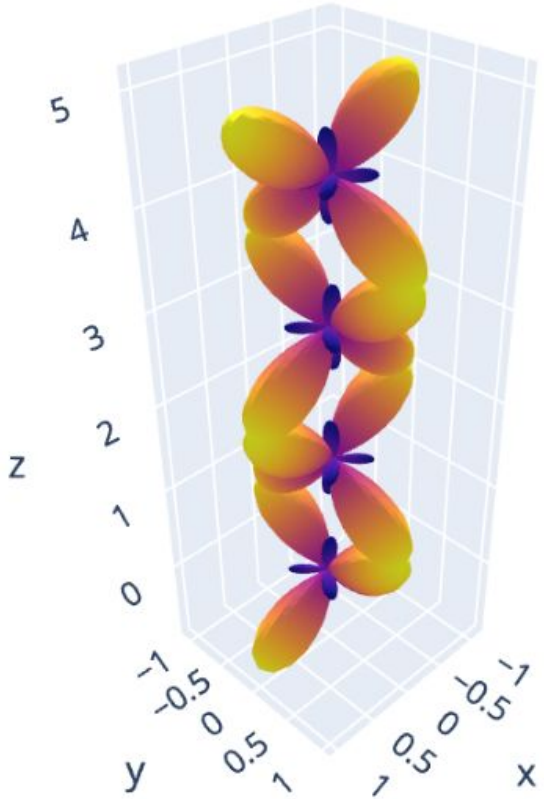
Thomas Hardin

Josh Rackers

We can also use gradients to invert equivariant functions that produce invariants, like the bispectrum

Gradients of invariants from the bispectrum

$$x \otimes x \otimes x$$



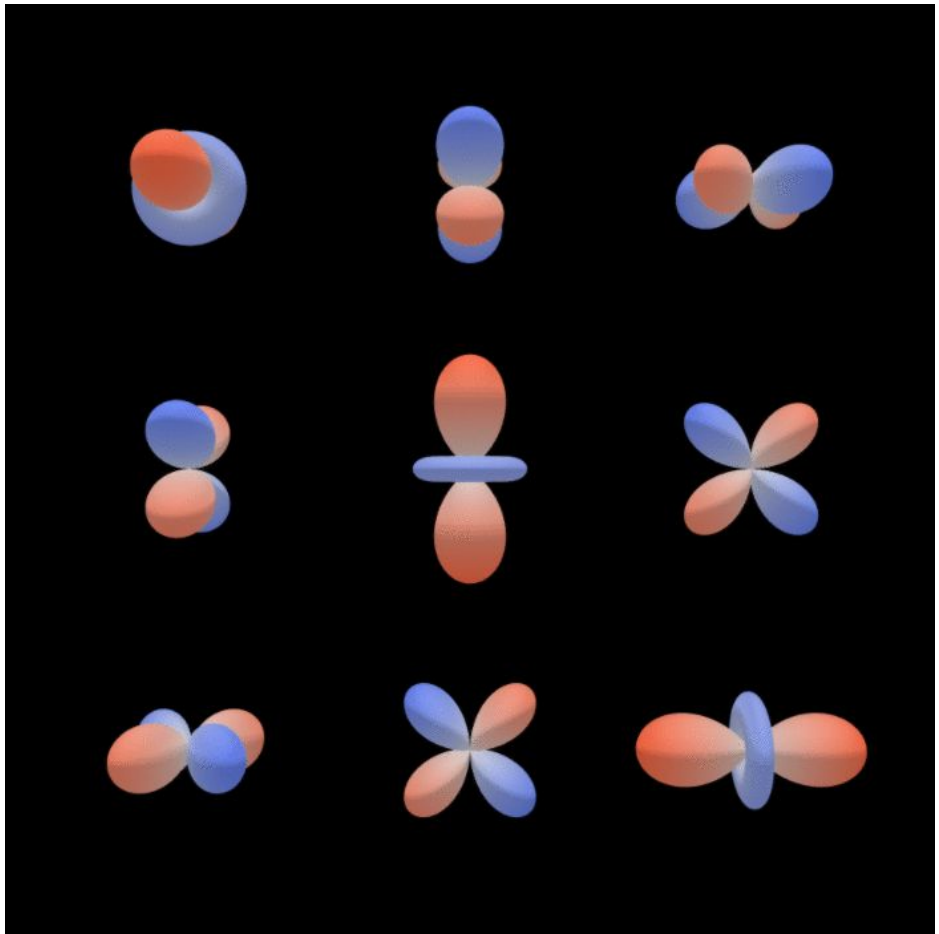
Martin Uhrin

Thomas Hardin

Josh Rackers

e3nn

For L=1 ⇨ L=1, the filters will be a learned, radially-dependent linear combinations of the L = 0, 1, and 2 spherical harmonics.



Random filters for L=1 ⇨ L=1…

(3 in L=1 channels by 3 out L=1 channels)

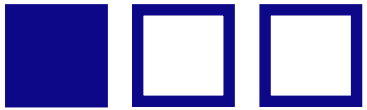
… as a function of increasing r.

Time showing filter for varying r, where $0 \leq r \leq r_{max}$.

(+ / –) Radial distance is magnitude as a function of angle

Feature 1: 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.

Vector



Pseudo-vector

Double-Headed Ray

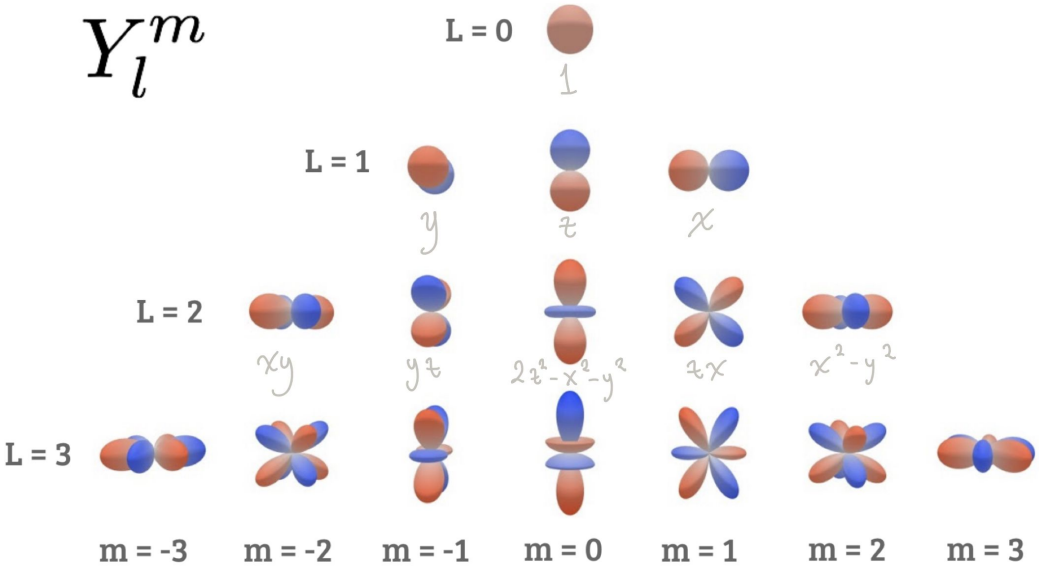
Spiral

```
Rs_vector = o3.Irrep("1o")

Rs_pseudovector = o3.Irrep("1e")

Rs_doubleray = o3.Irrep("2e")

Rs_spiral = o3.Irrep("2o")
```

Feature 1: 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.

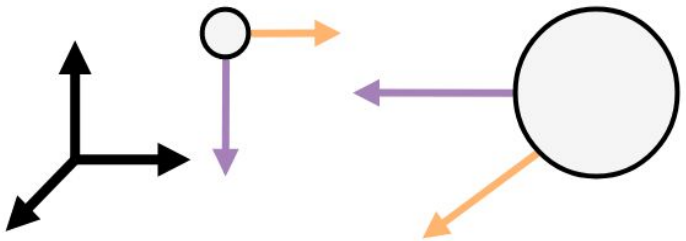
Spherical harmonics

Y_l^m




```
Rs_s_orbital = o3.Irrep("0e")

Rs_p_orbital = o3.Irrep("1o")

Rs_d_orbital = o3.Irrep("2e")

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

The input to our network is geometry and features on that geometry.

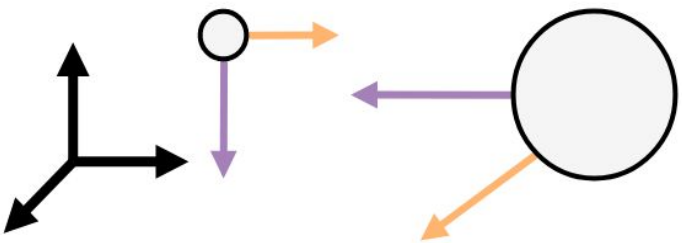


```
geometry = [[x0, y0, z0],[x1, y1, z1]]
features = [
    [m0, v0y, v0z, v0x, a0y, a0z, a0x]
    [m1, v1y, v1z, v1x, a1y, a1z, a1x]
]
...
```

The input to our network is geometry and 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
```

e3nn: a modular PyTorch framework for Euclidean neural networks

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

Creating a basic Euclidean neural network

```
from e3nn import o3
from e3nn.nn.models.gate_points_2101 import Network

model_kwargs = {
    'irreps_in': o3.Irreps("3x0e + 2x1o"),
    'irreps_hidden': o3.Irreps("5x0e + 5x0o + 5x1e + 5x1o"),
    'irreps_out': o3.Irreps("2x0o + 2x1o + 2x2e"),
    'irreps_node_attr': o3.Irreps("10x0e"),
    `irreps_edge_attr`: o3.Irreps.spherical_harmonics(3),
    'layers': 3, 'max_radius': r_max, 'number_of_basis': 10, 'radial_layers': 2,
    'radial_neurons': 100, `num_nodes`: 5, `num_neighbors`: 5
}

model = Network(**model_kwargs)
```

e3nn: a modular PyTorch framework for Euclidean neural networks

https://github.com/e3nn/e3nn

Convert between Cartesian tensors (with symmetric indices) and Irrep tensors and calculate degrees of freedom (e.g. elasticity tensor)

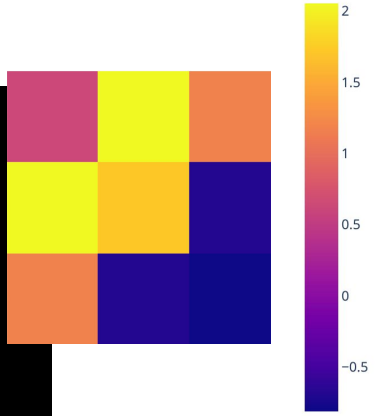
```
from e3nn.io import CartesianTensor

rank4_symmetric = CartesianTensor('ijkl=jikl=klij')
print("Representations: ", rank4_symmetric)
print("Degrees of freedom: ", rank4_symmetric.dim())

>> Representations: "2x0e + 2x2e + 1x4e"
>> Degrees of freedom: 21
```

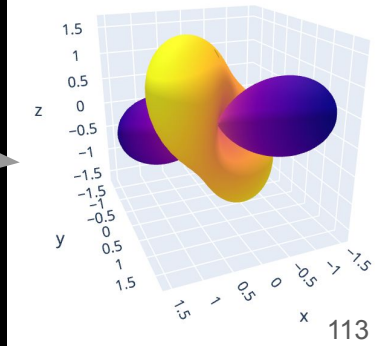
e3nn: a modular PyTorch framework for Euclidean neural networks

https://github.com/e3nn/e3nn

Plot 3x3 matrix as linear combination of spherical harmonics.

```
...

# Symmetric Matrix
M = torch.randn(3,3)
M = M + M.transpose(0, 1)

# Plot matrix
px.imshow(M)

symm_matrix = CartesianTensor('ij=ji')
symm_matrix.change_of_basis(M)
data = SphericalTensor(symm_matrix.lmax, 1, -1).plotly_surface()

# Plot SH signal
surface_plot = go.Surface(**data)
go.Figure([trace])
```



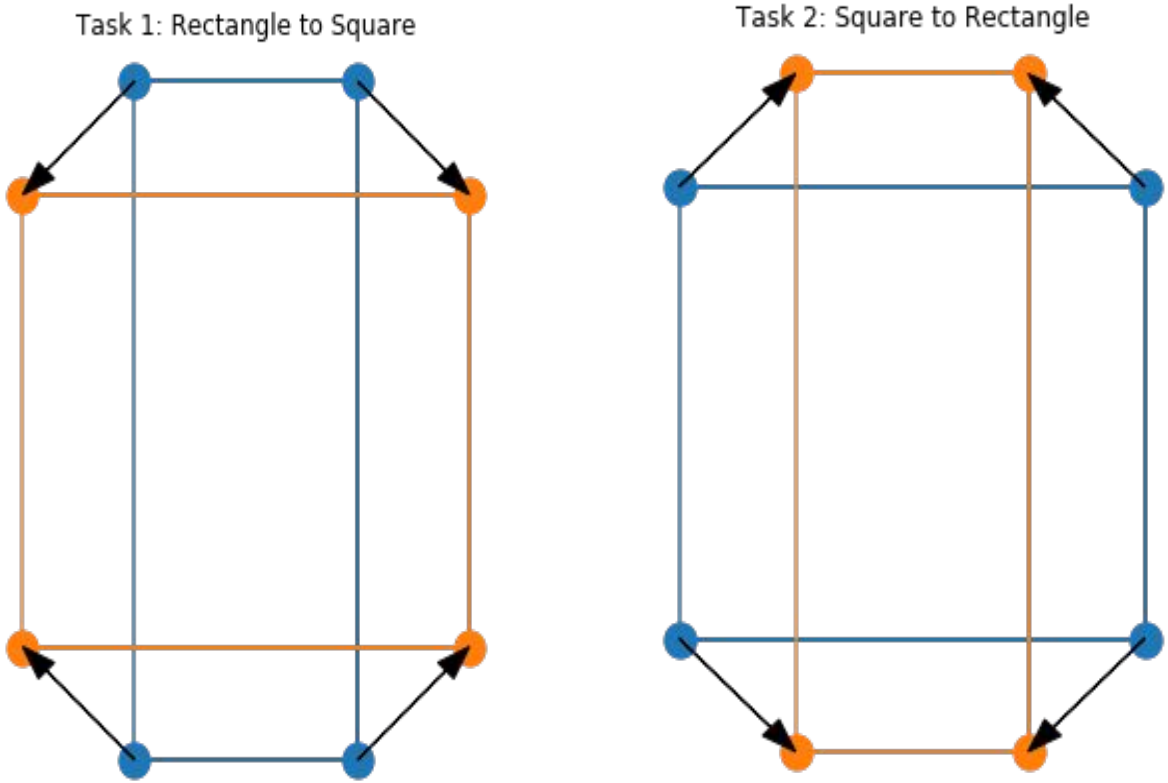


rectangle

square

The outputs have equal or higher symmetry than the inputs.

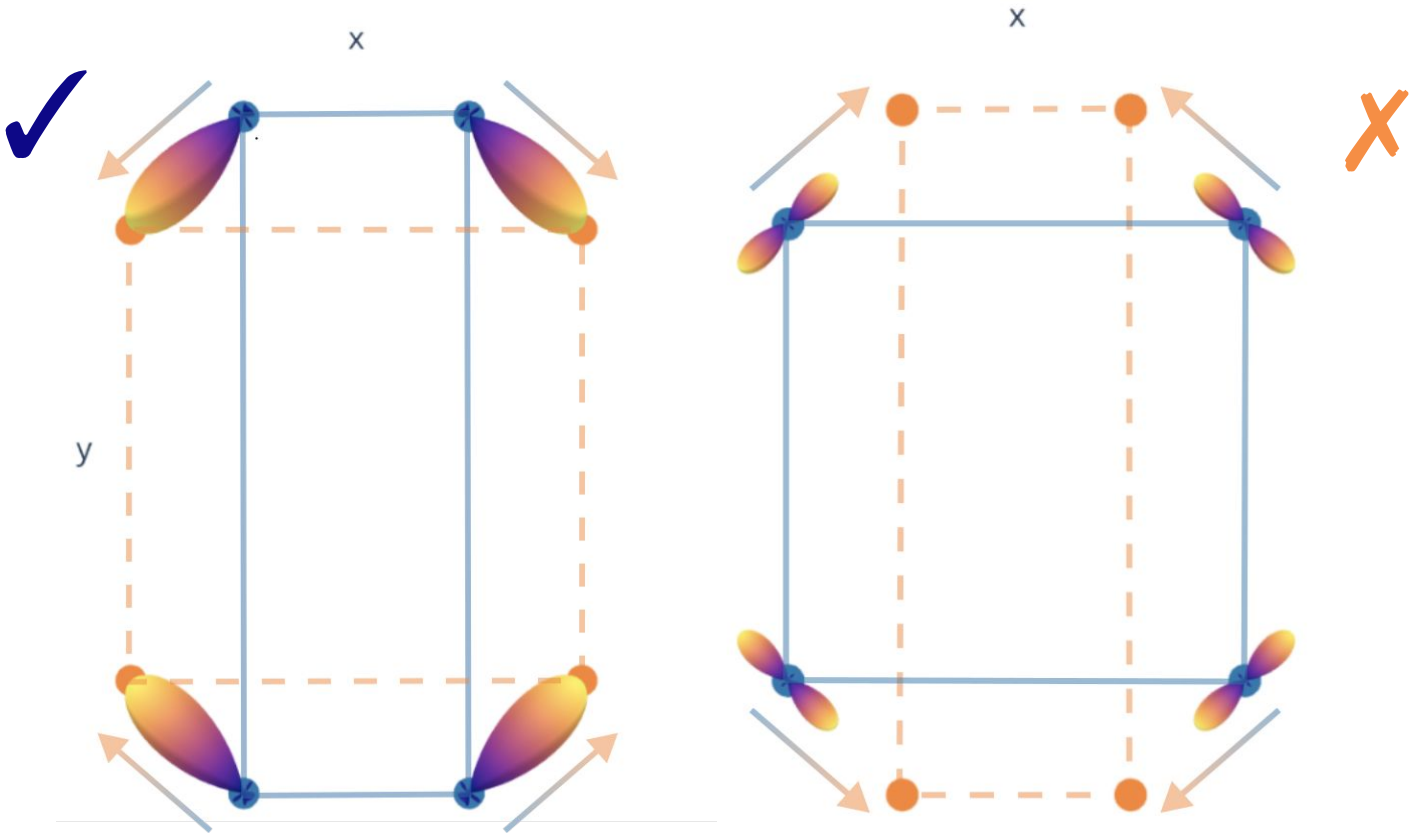
Symmetry compiler -- can't fit a model that does not symmetrically make sense



T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

The outputs have equal or higher symmetry than the inputs.

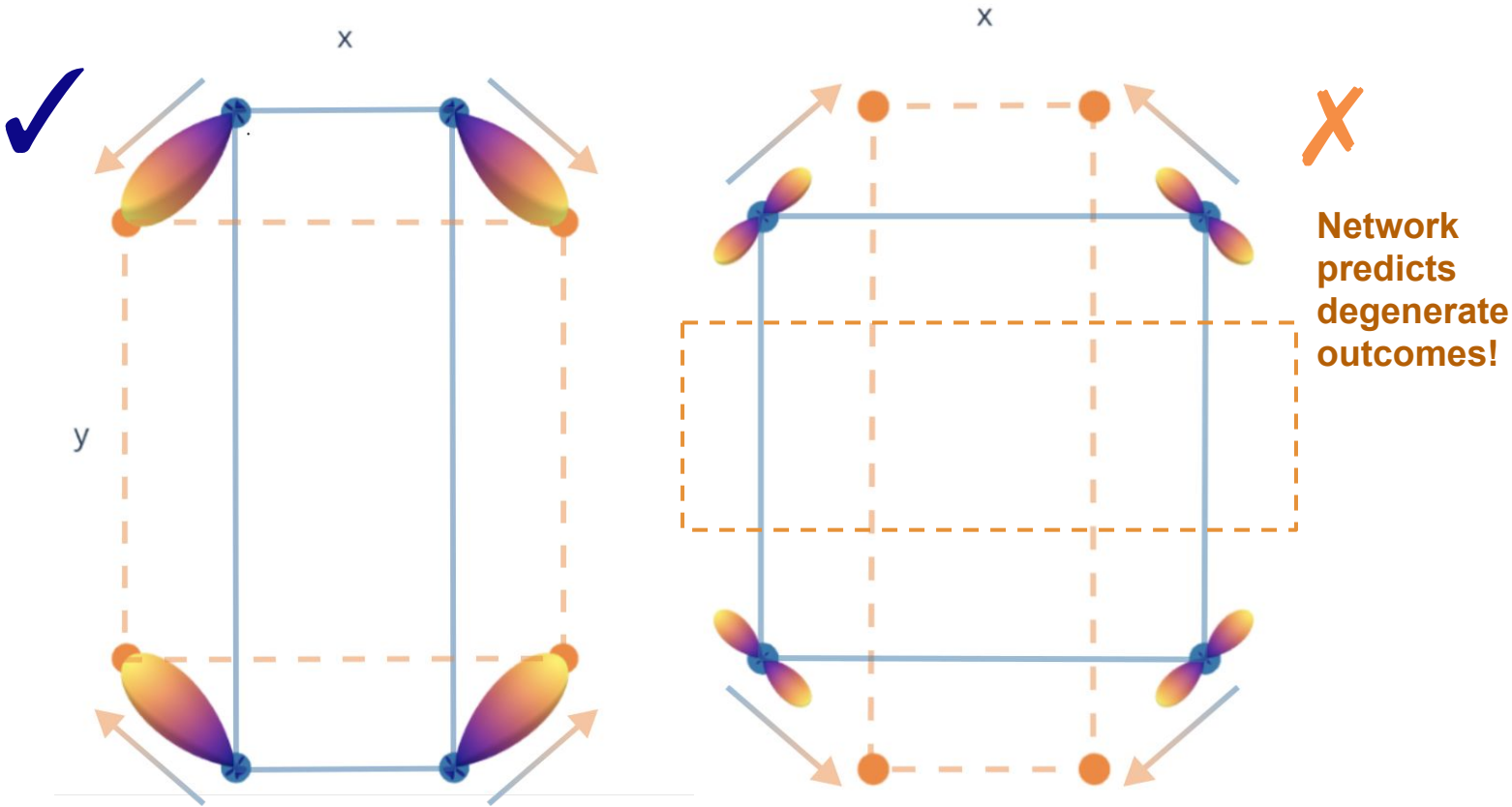
Symmetry compiler -- can't fit a model that does not symmetrically make sense



T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

The outputs have equal or higher symmetry than the inputs.

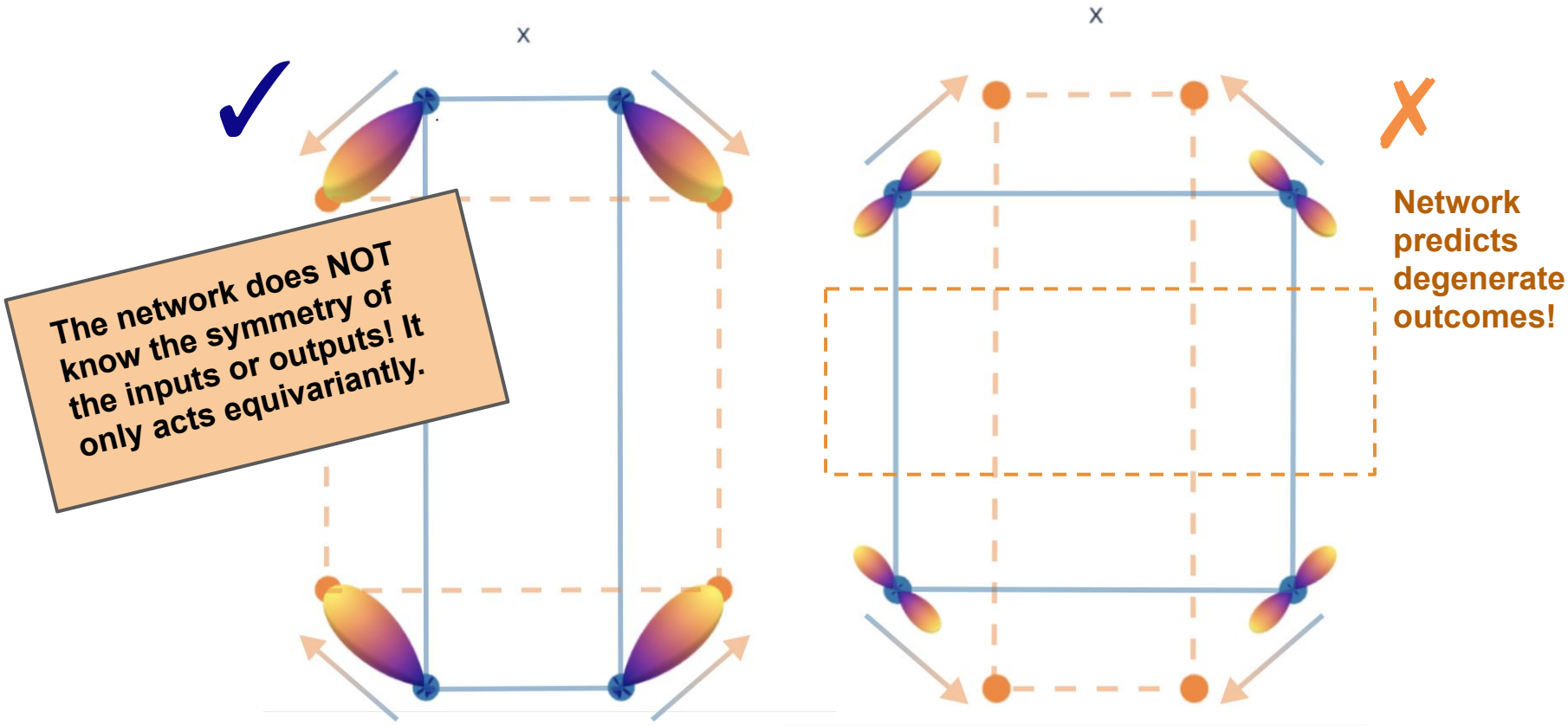
Symmetry compiler -- can't fit a model that does not symmetrically make sense



T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

The outputs have equal or higher symmetry than the inputs.

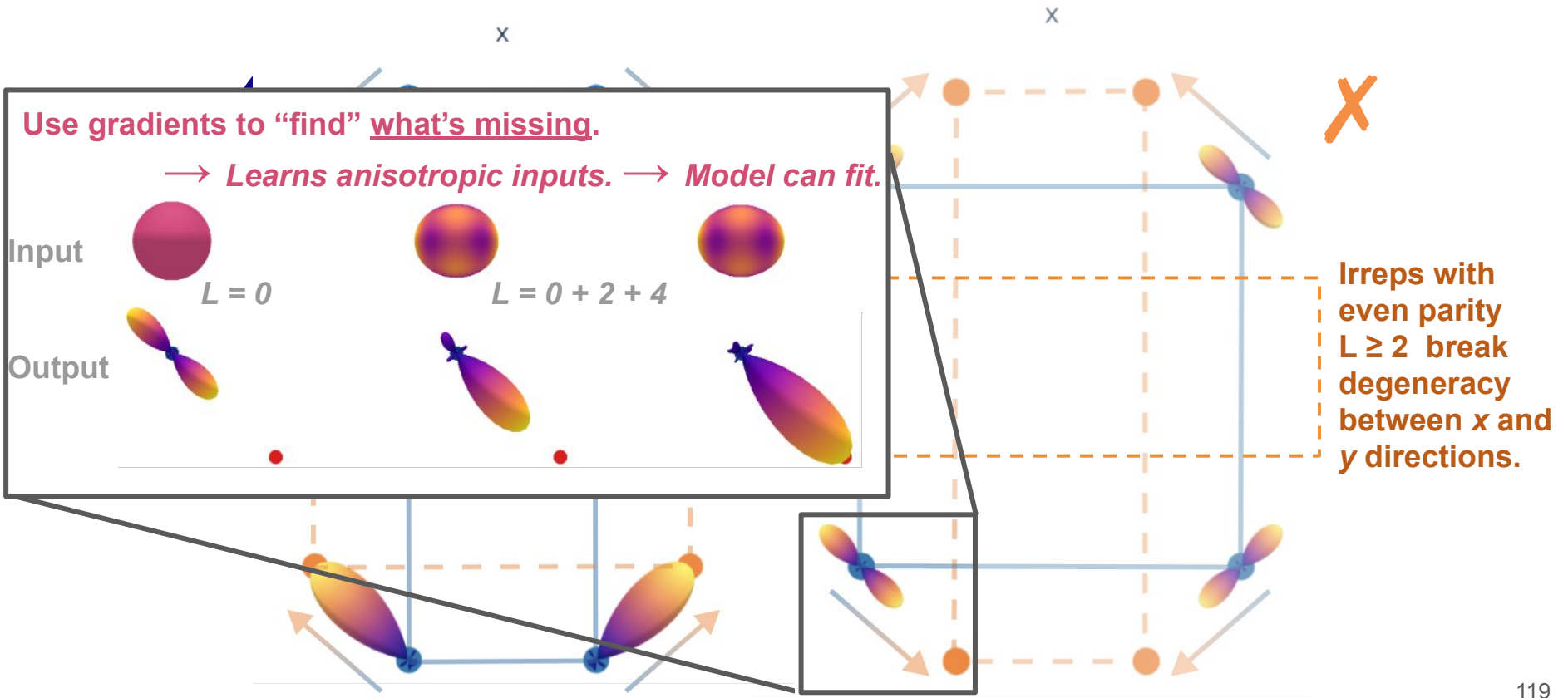
Symmetry compiler -- can't fit a model that does not symmetrically make sense



T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

We can find data that is implied by symmetry.

Using gradients of loss wrt input we can find symmetry breaking "order parameters"



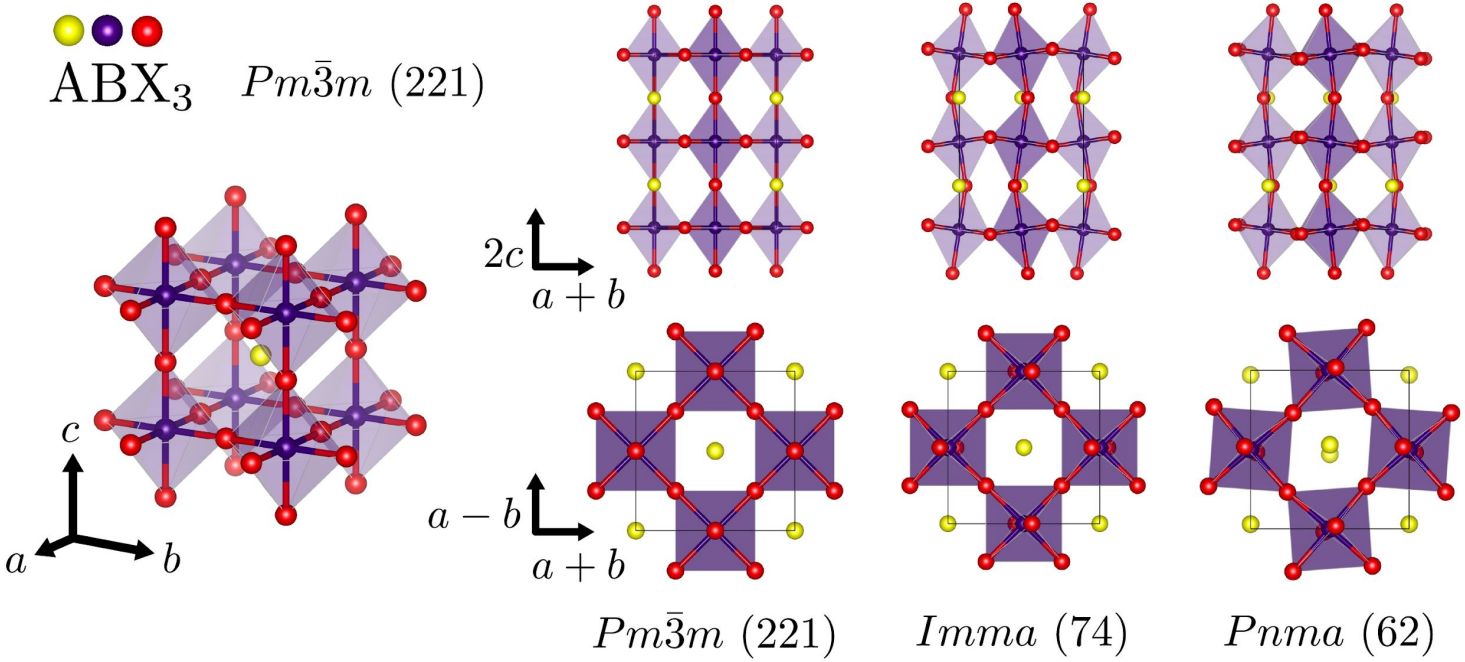
T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

We can find data that is implied by symmetry.

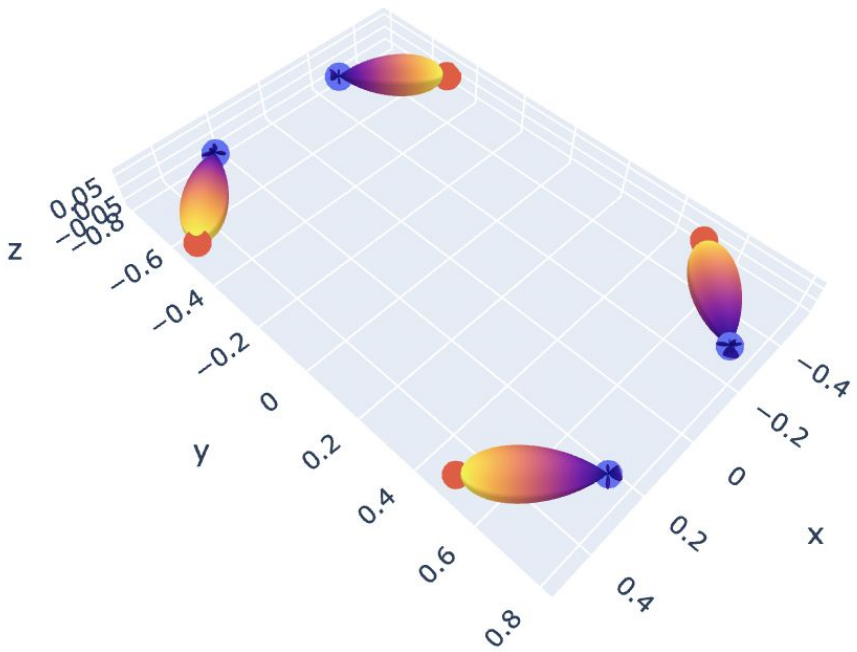
Using gradients of loss wrt input we can find symmetry breaking "order parameters"

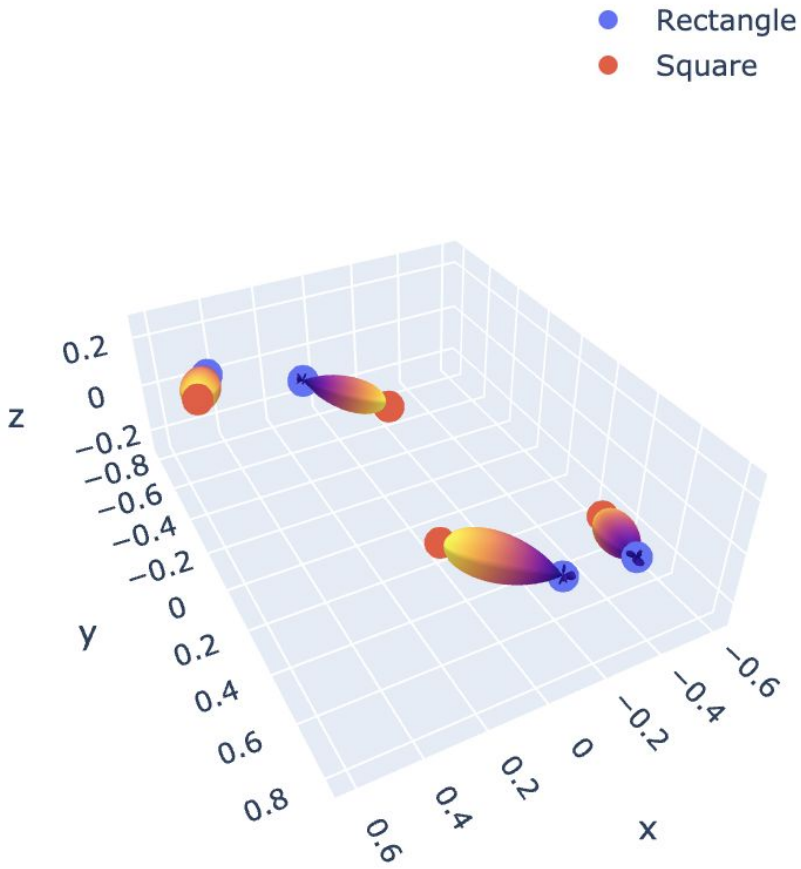
Octahedral tilting in perovskites ($M^{3+} \oplus R^{4+}$) ⇨

Network learns equal magnitude pseudovector order parameters on B site with proper spatial patterning.



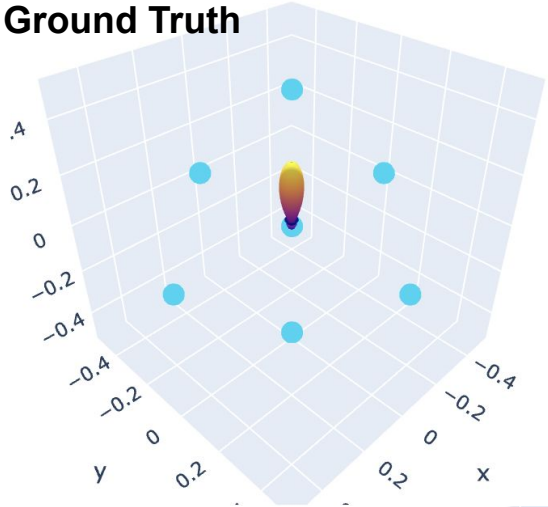
T. E. Smidt, M. Geiger, B. K. Miller. Physical Review Research (2021)

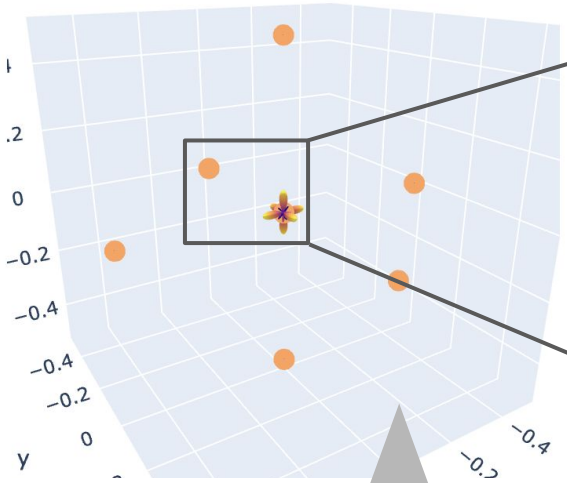

z
0.05
−0.05
−0.8
−0.6
−0.4
−0.2
y
0
0.2
0.4
0.6
0.8
−0.4
−0.2
0
0.2
0.4
x

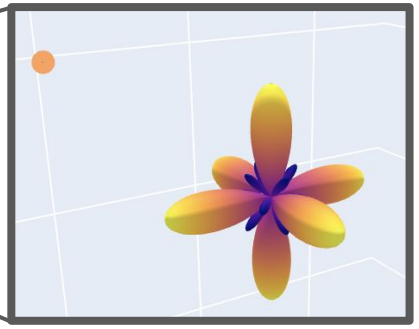

Rectangle
Square
z
0.2
0
−0.2
−0.8
−0.6
−0.4
−0.2
y
0
0.2
0.4
0.6
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
x

Predictions for O_h symmetry

Ground Truth

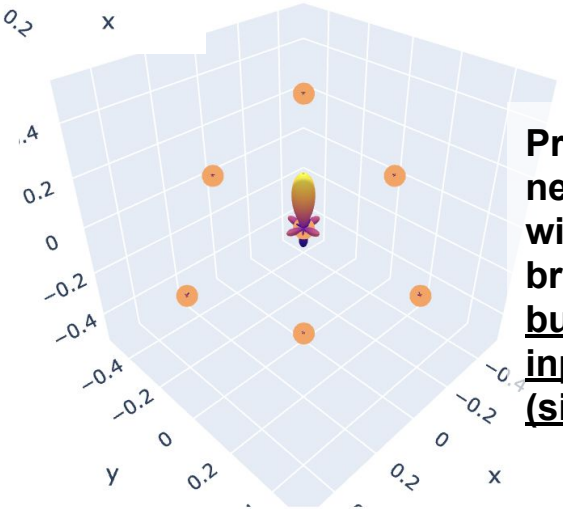




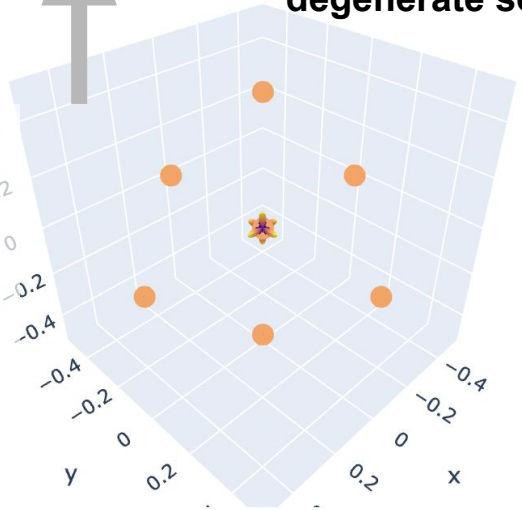


Superposition of 6 rotationally degenerate solutions.

Prediction of network trained with symmetry breaking input <u>and given symmetry breaking input along z.</u>



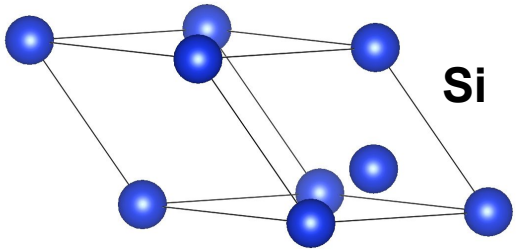
Prediction of network trained with symmetry breaking input <u>but given trivial input (single scalar)</u>.



misc.
intro slides

What a computational materials physicist does:

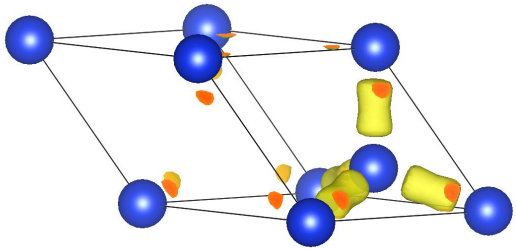
Given an atomic structure,



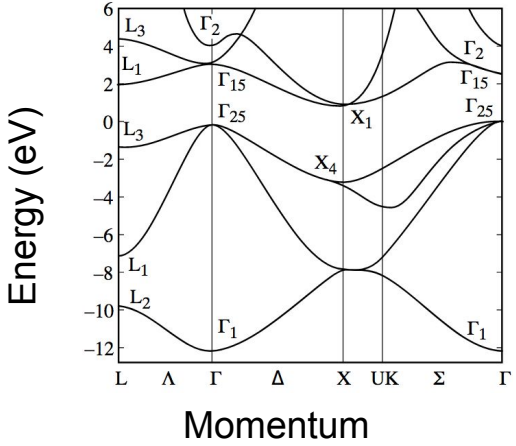
...use quantum theory and supercomputers to determine...

$$\hat{H}\,|\psi\rangle = E\,|\psi\rangle$$

...where the electrons are...



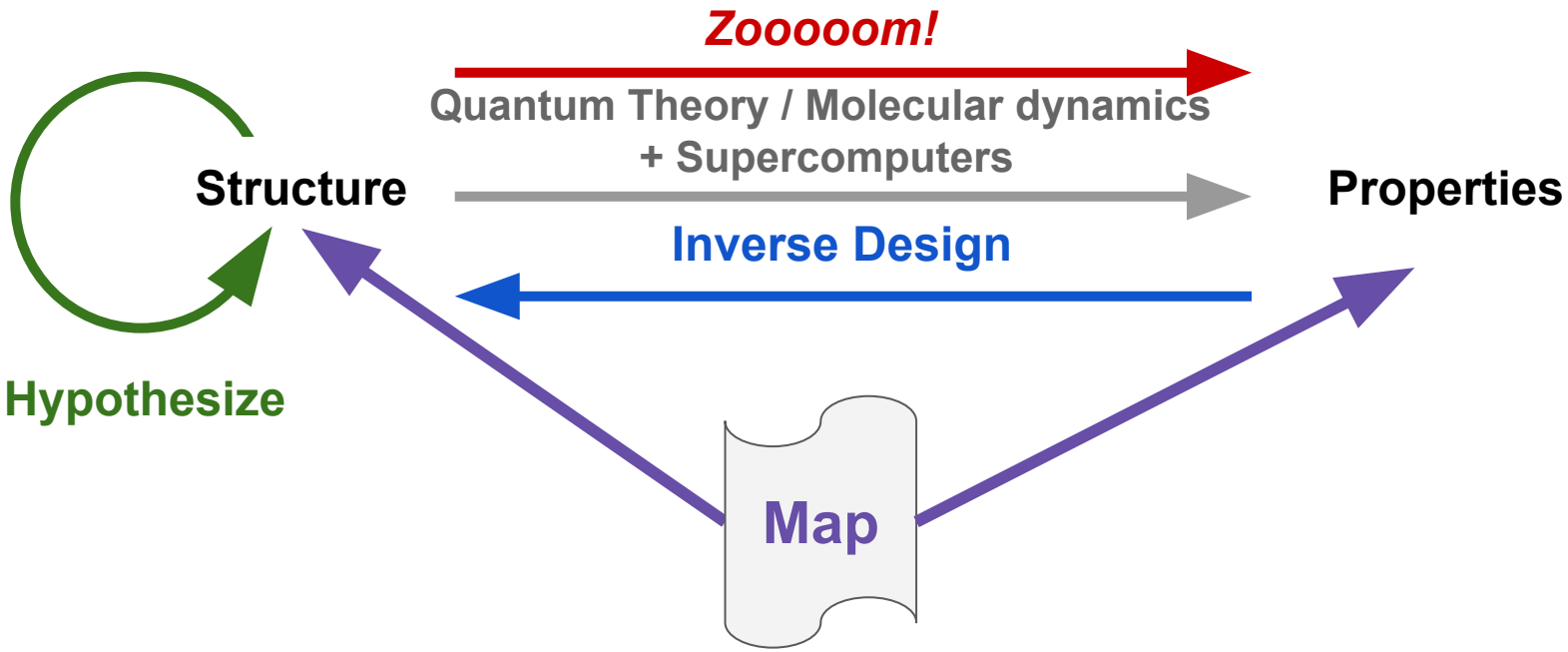
...and what the electrons are doing.



Structure

Properties

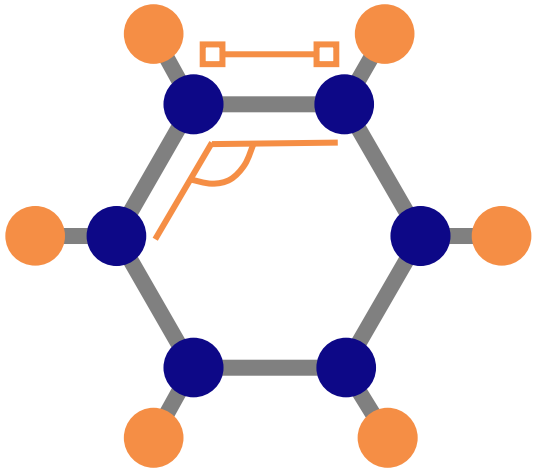
We want to use deep learning to speed up calculations, hypothesize new structures, perform inverse design, and organize these relations.



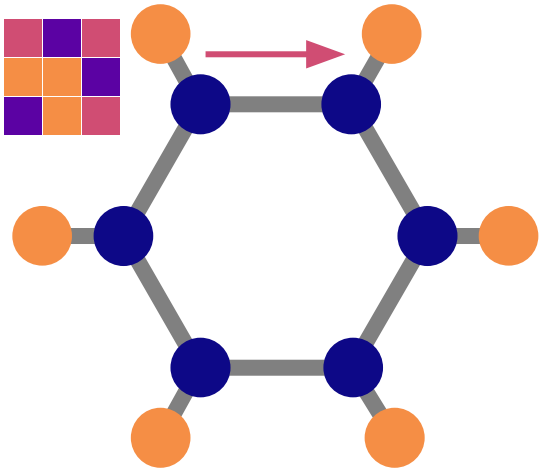
Primer?

Why invariant models? Mathematically simpler and *so far* effective.
But equivariant models are starting to show that they're more expressive and efficient.

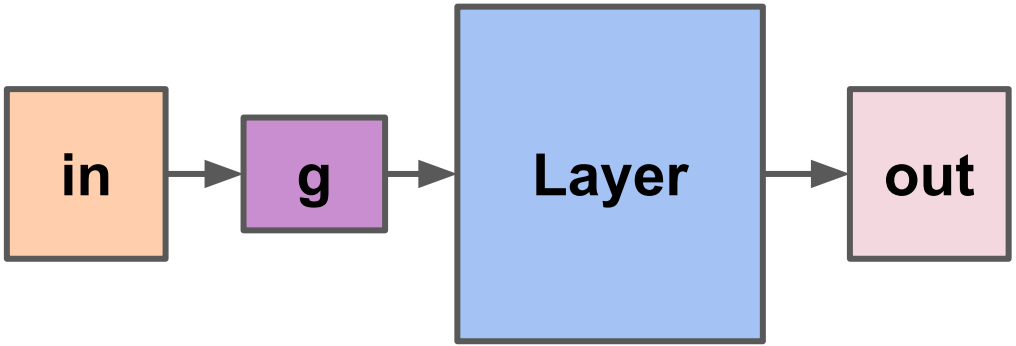
Invariant models can only operate on invariant features: radial distances, pairwise angles...

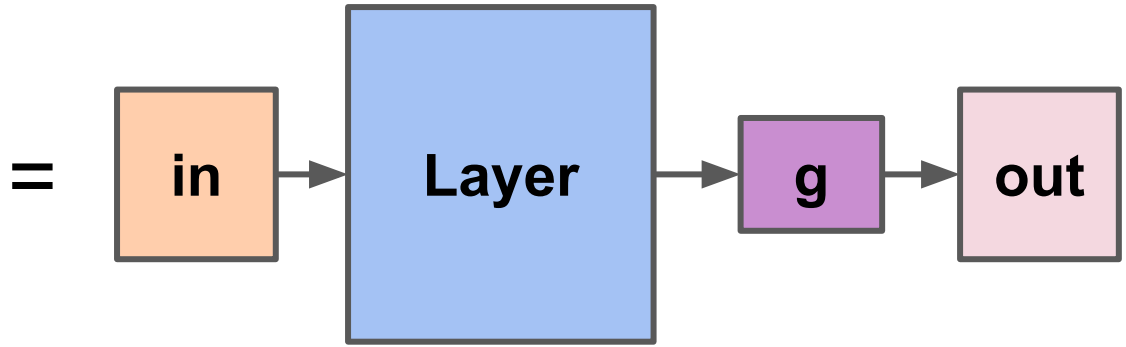


Equivariant models can operate on invariant AND equivariant features: relative distance vectors and other tensor quantities



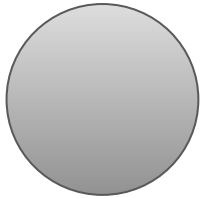
For a function to be *equivariant* means that we can act on our inputs with g OR act our outputs with g and we get the same answer *(for every operation)*.
For a function to be *invariant* means g is the identity (no change).





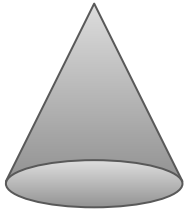
A Euclidean symmetry preserving network produces *outputs* that preserve the subset of symmetries induced by the *input*.

3D rotations and inversions



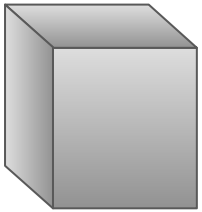
O(3)

2D rotation and mirrors along cone axis

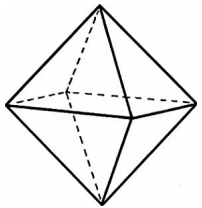


SO(2) + mirrors ($C_{\infty v}$)

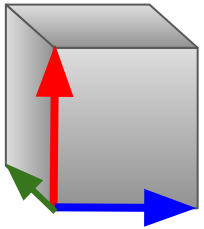
Discrete rotations and mirrors



O_h



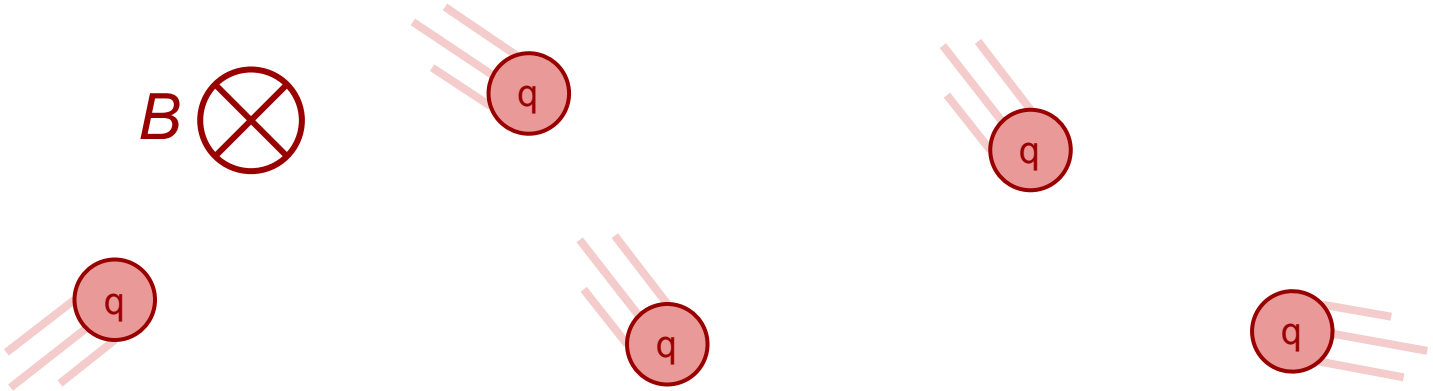
Discrete rotations, mirrors, and translations



Pm-3m (221)

The laws of (nonrelativistic) physics have Euclidean Symmetry.
Thus, machine learning models used to emulate physics should also have this symmetry.

Symmetry of model vs. symmetry of system:
The **parameters** of a model describes the **"physics"**. The **input** to the model is our **system**.



$$\vec{F}(q, \vec{r}, \vec{v}, \vec{B}) = \vec{F}_i = \sum_i q_i(\vec{v_i} \times \vec{B}) + \sum_{i \neq j} \frac{q_i q_j}{r_{ij}^2} \hat{r_{ij}}$$

nitty gritty

Invariant featurizations can be very expressive if well-crafted

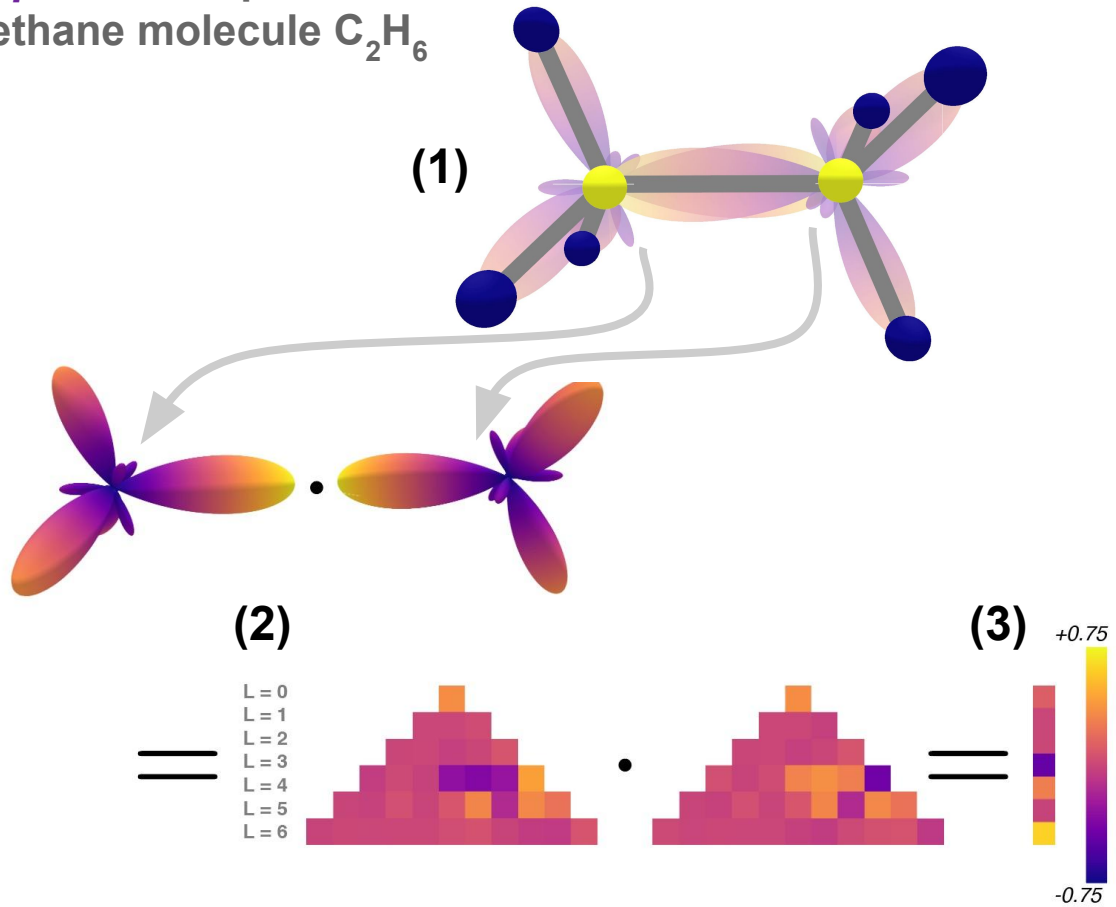
Many *invariant* featurizations use *equivariant* operations

e.g. a (simplified) SOAP kernel for ethane molecule C_2H_6

(1) **Project neighbors of given atom onto spherical harmonics (*equivariant* quantity).**

(2) **Interact signals from different atoms via tensor dot product (*equivariant* operation) to produce scalars (*invariant* quantity).**

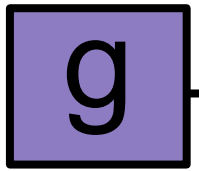
(3) **Give scalars to model.**



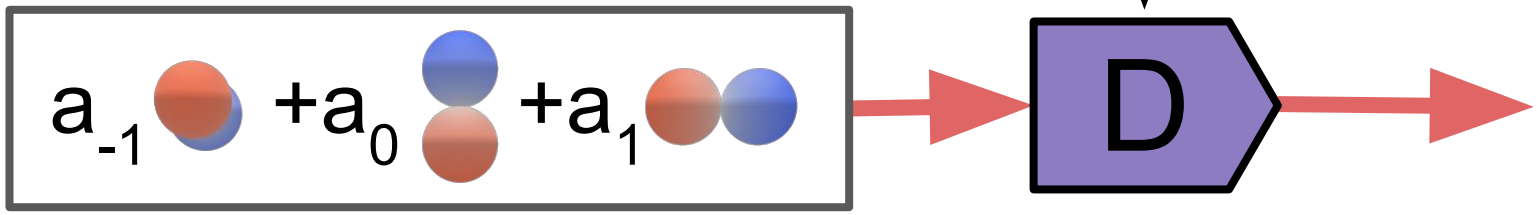
(Favored for kernel methods)

Spherical harmonics of a given L transform together under rotation.

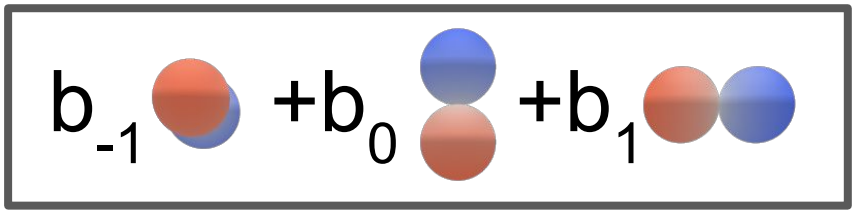
Let **g** be an element of SO(3)



D is the Wigner-D matrix. It has shape $[2l+1, 2l+1]$ and is a function of **g**.

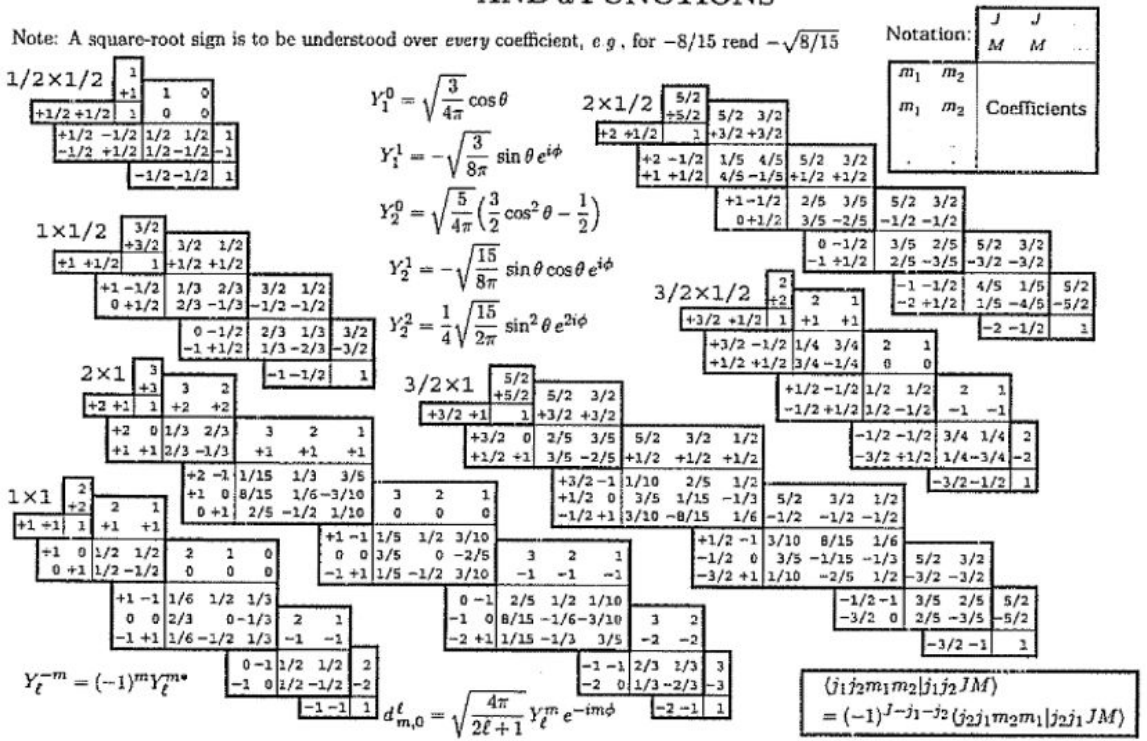


=



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 $-\sqrt{8/15}$

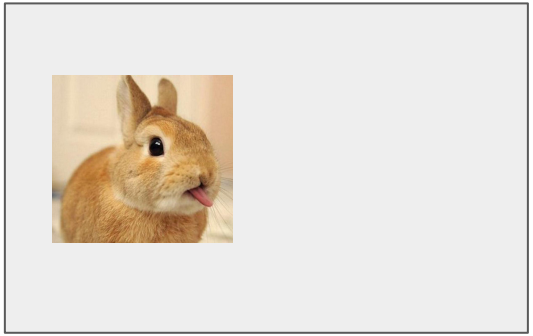


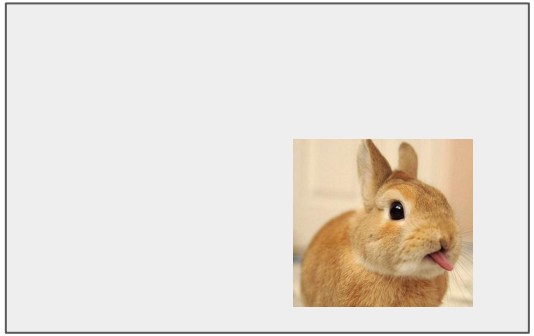
Applications so far...

- Finding order parameters of 2nd order structural phase transitions
- Molecular dynamics *(Harvard)*
- Molecule and crystal property prediction *(FU Berlin)*
- Inverting invariant representations of atomic geometries *(Sandia)*
- Autoencoding Geometry
- Predicting molecular Hamiltonians *(TU Berlin)*
- Long range interactions *(FU Berlin, TU Berlin)*
- Electron density prediction for large molecules *(Sandia)*
- Predicting chemical shifts for NMR *(Merck, MIT)*
- Conditional protein design *(UW)*
- Inverse design of optical properties of nanoparticle assemblies *(LBL and UW)*
- Phonon properties of crystal structures *(MIT)*
- Anharmonic elastic properties of crystal structures *(UTEP)*
- ...

neural nets intros

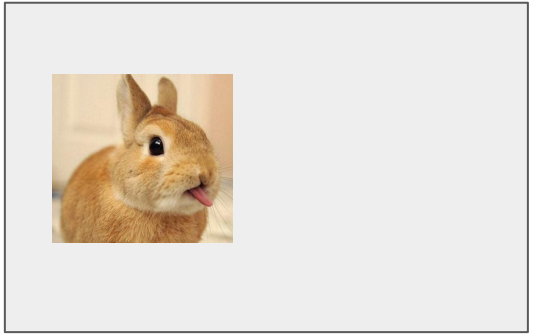

MultidimensionalArray
TensorFlow

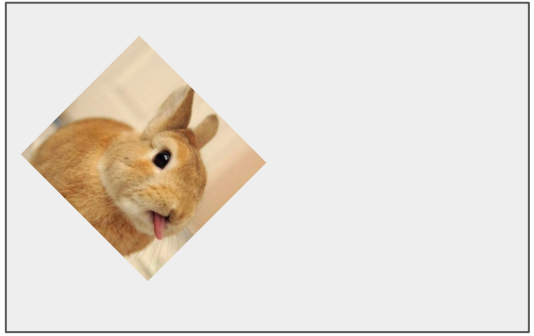




Translation equivariance

Convolutional neural network ✓





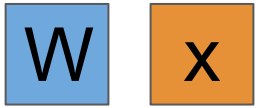
Rotation equivariance

~~Data augmentation~~

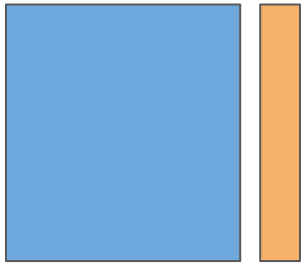
~~Radial functions (invariant)~~

Want a network that both preserves geometry and exploits symmetry.

Neural networks are specially designed for different data types.
Assumptions about the data type are built into how the network operates.

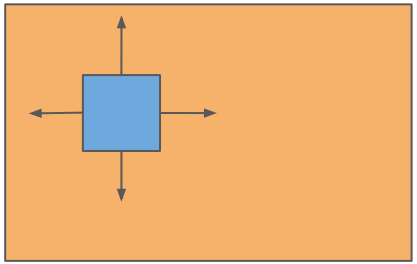


Arrays ⇨ Dense NN



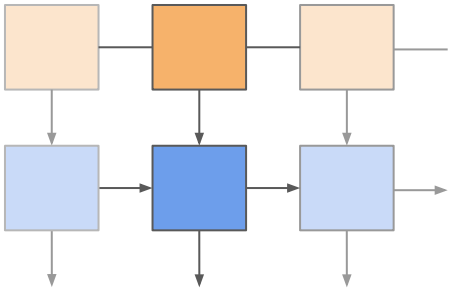
Components are independent.

2D images ⇨ Convolutional NN



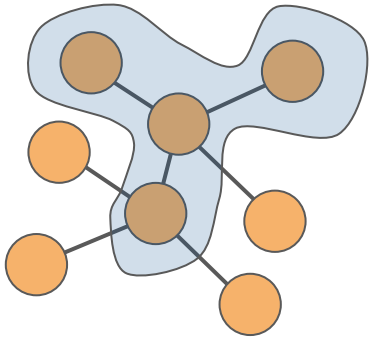
The same features can be found anywhere in an image. Locality.

Text ⇨ Recurrent NN



Sequential data. Next input/output depends on input/output that has come before.

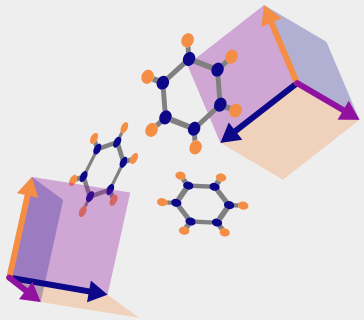
Graph ⇨ Graph (Conv.) NN



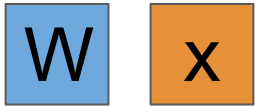
Topological data. Nodes have features and network passes messages between nodes connected via edges.

3D physical data ⇨ Euclidean NN

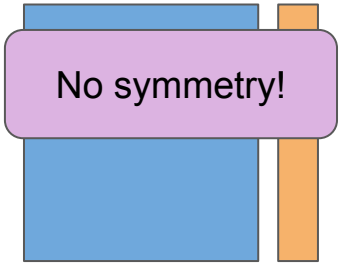
Data in 3D Euclidean space. Freedom to choose coordinate system.



Neural networks are specially designed for different data types.
Assumptions about the data type are built into how the network operates.
Symmetries emerge from these assumptions.

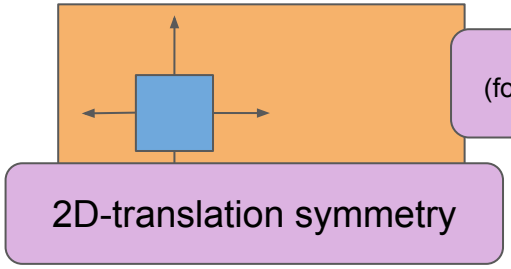


Arrays ⇨ Dense NN



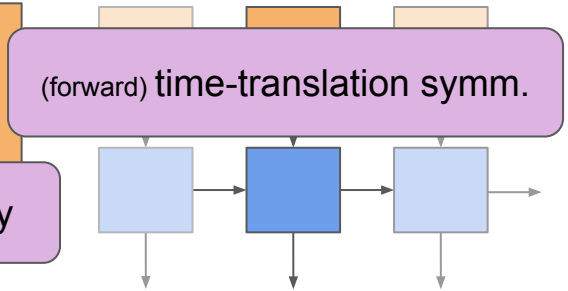
Components are independent.

2D images ⇨ Convolutional NN



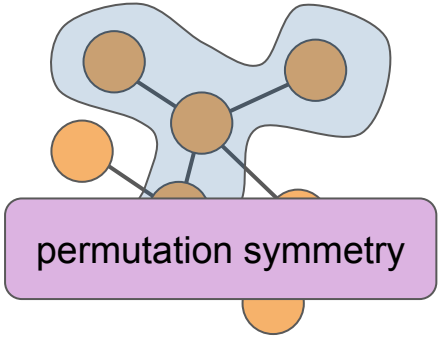
The same features can be found anywhere in an image. Locality.

Text ⇨ Recurrent NN



Sequential data. Next input/output depends on input/output that has come before.

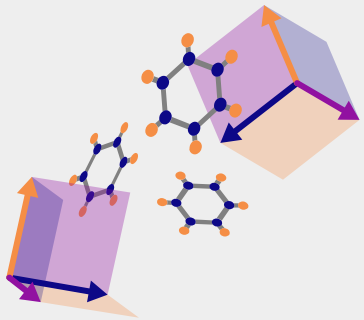
Graph ⇨ Graph (Conv.) NN



Topological data. Nodes have features and network passes messages between nodes connected via edges.

3D physical data ⇨ Euclidean NN

3D Euclidean symmetry **E(3)**: 3D rotations translations and inversion



A brief primer on deep learning

Skip?

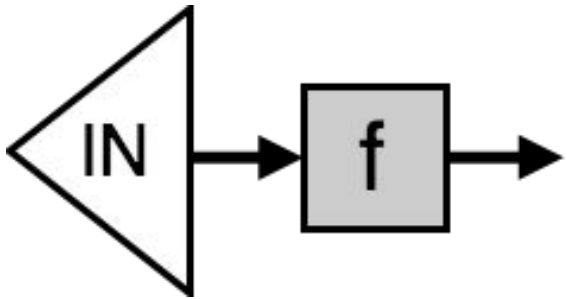
deep learning $\subset$ machine learning $\subset$ artificial intelligence

A brief primer on deep learning

model (“neural network”):

Function with learnable parameters.

$$y = f(x)$$



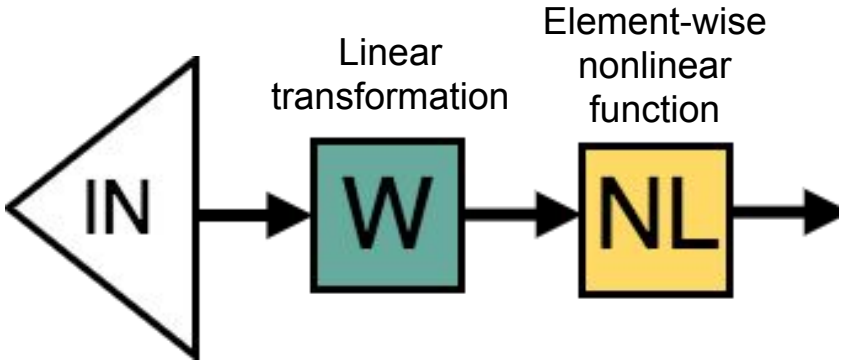
A brief primer on deep learning

model (“neural network”):
Function with learnable parameters.

Ex: "Fully-connected" network

$$y = \tanh(Wx + b)$$

Learned Parameters



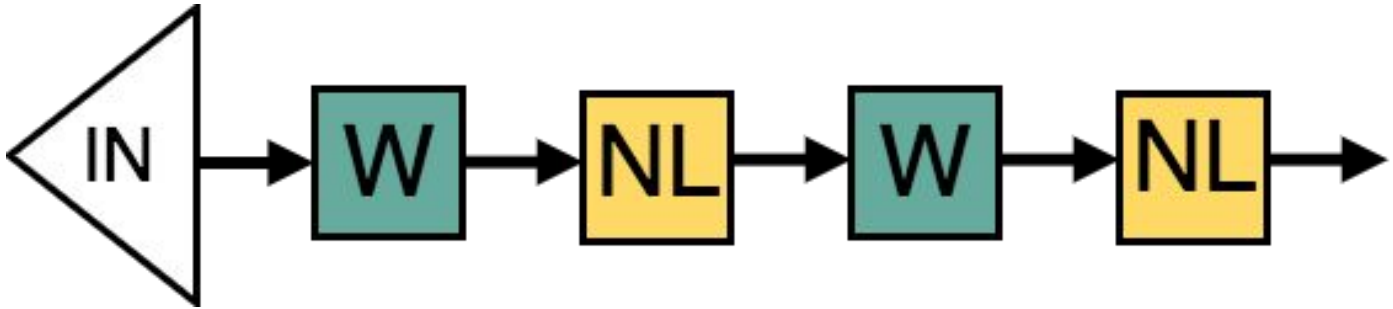
A brief primer on deep learning

model (“neural network”):
Function with learnable parameters.

Ex: "Fully-connected" network

$$y = \tanh(W_2 \tanh(W_1 x + b_1) + b_2)$$

Learned Parameters



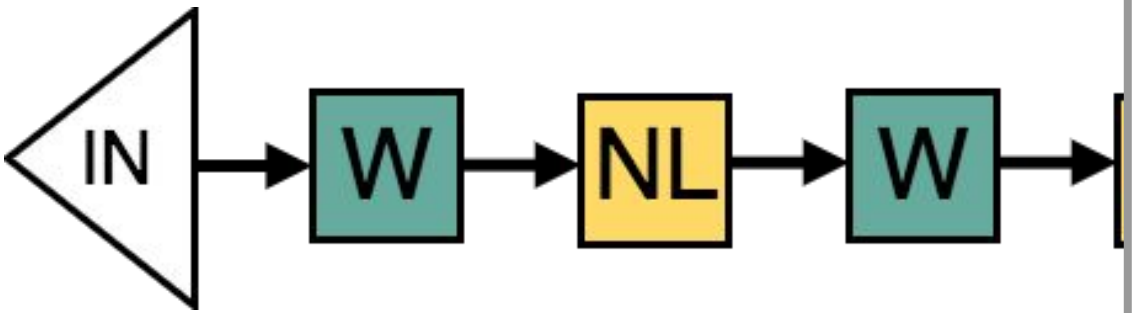
Neural networks with multiple layers can learn more complicated functions.

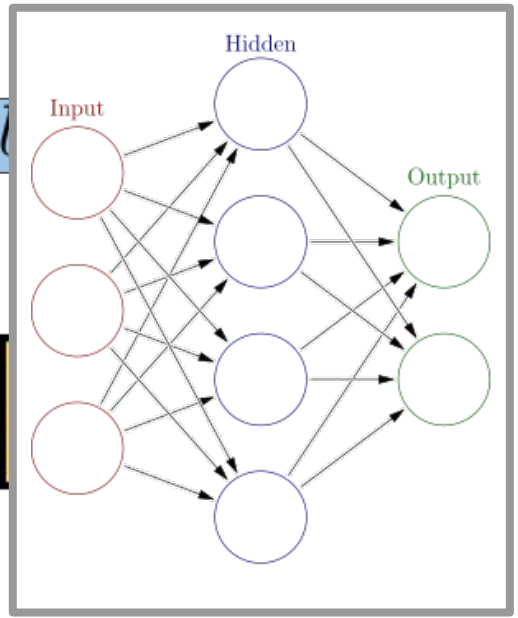
A brief primer on deep learning

model ("neural network"):
Function with learnable parameters.

Ex: "Fully-connected" network

$$y = \tanh(W_2 \tanh(W_1 x + b_1) + b$$



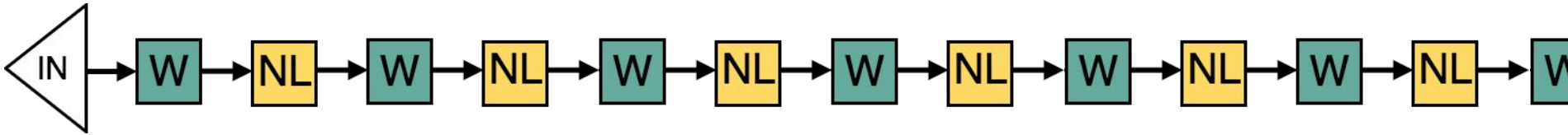


Neural networks with multiple layers can learn more complicated functions.

A brief primer on deep learning

deep learning:

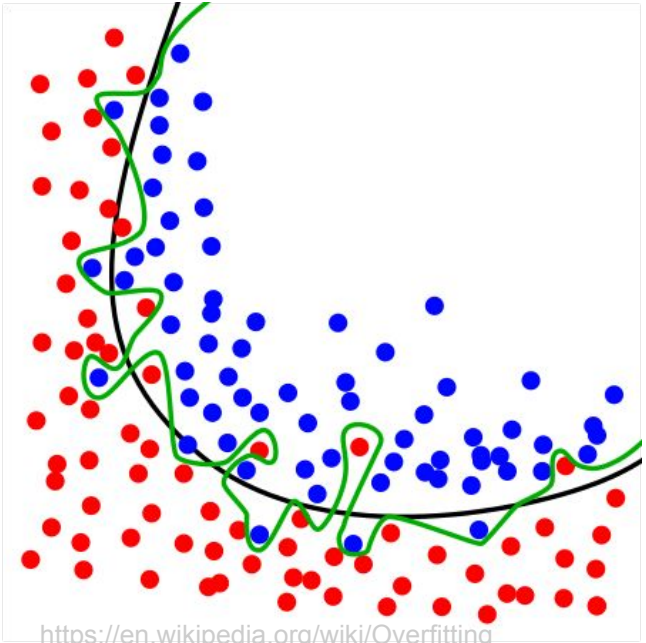
Add more layers.



A brief primer on deep learning

data:

Want lots of it. Model has many parameters. Don't want to easily overfit.



https://en.wikipedia.org/wiki/Overfitting

A brief primer on deep learning

cost function:

A metric to assess how well the model is performing.
The cost function is evaluated on the output of the model.
Also called the **loss** or **error**.

A brief primer on deep learning

way to update parameters:

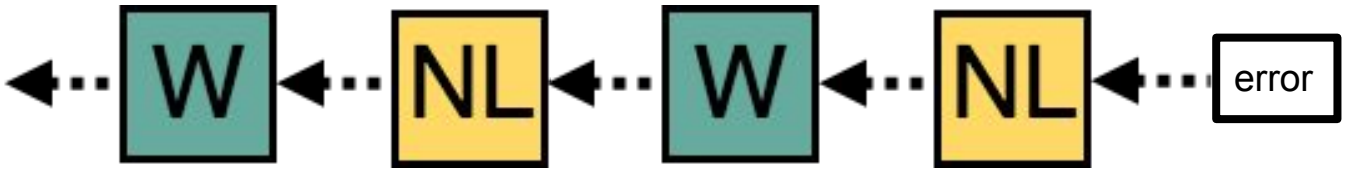
Construct a model that is differentiable

Easiest to do with differentiable programming frameworks: e.g. Torch, TensorFlow, JAX, ...

Take derivatives of the cost function (loss or error) wrt to learnable parameters.

This is called backpropogation (aka the chain rule).

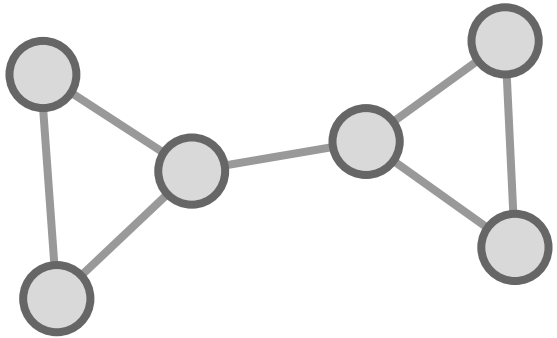
$$\Delta W_{ij} = -\eta \frac{\partial \text{ error}(f(W, x), y)}{\partial W_{ij}}$$



Equivariance can have unintuitive consequences.

Partition graph with permutation equivariant function into two sets using ordered labels.

Predict node labels
[0, 1] vs. [1, 0]

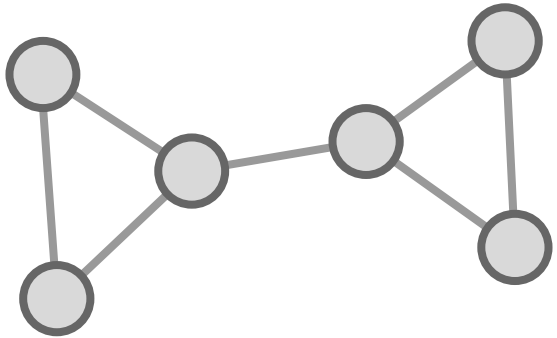


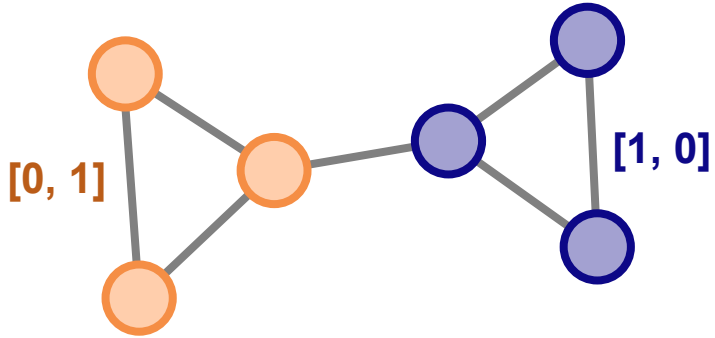
Equivariance can have unintuitive consequences.

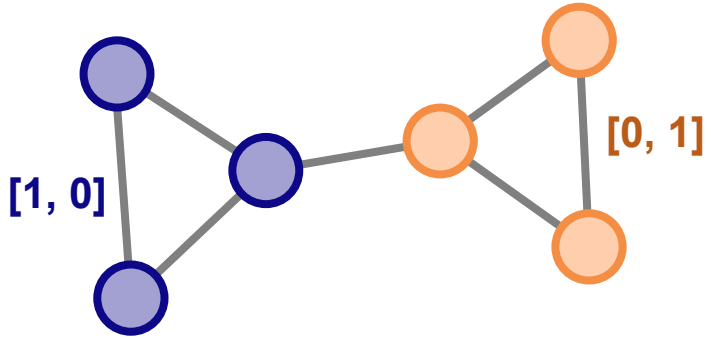
Partition graph with permutation equivariant function into two sets using ordered labels.

You can't *due to degeneracy*.

Predict node labels
[0, 1] vs. [1, 0]







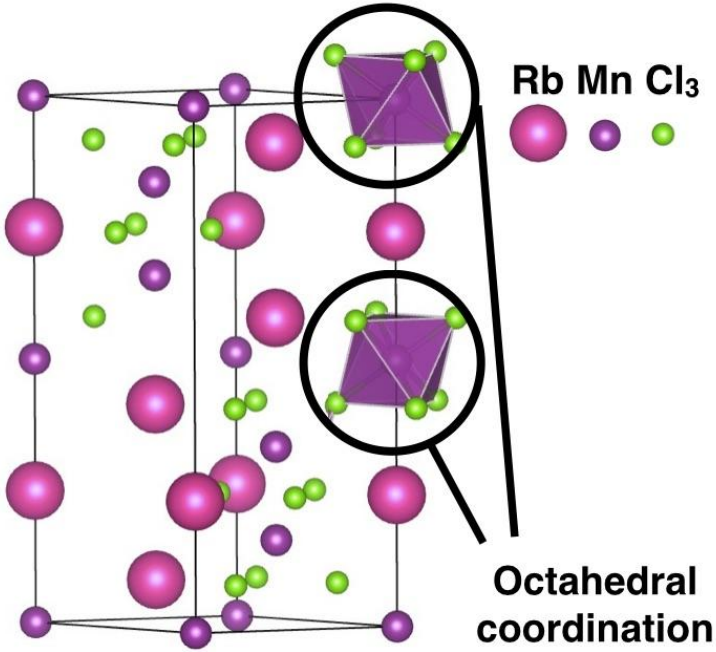
There's nothing to distinguish one partition to be "first" vs. "second".

Convolutions: Local vs. Global Symmetry

Convolutions capture local symmetry. Interaction of features in later layers yields global symmetry.
e.g. Coordination environments in crystals

Atomic systems form geometric motifs that can appear at multiple locations and orientations.

Space group:
Symmetry of unit cell
(Global symmetry)



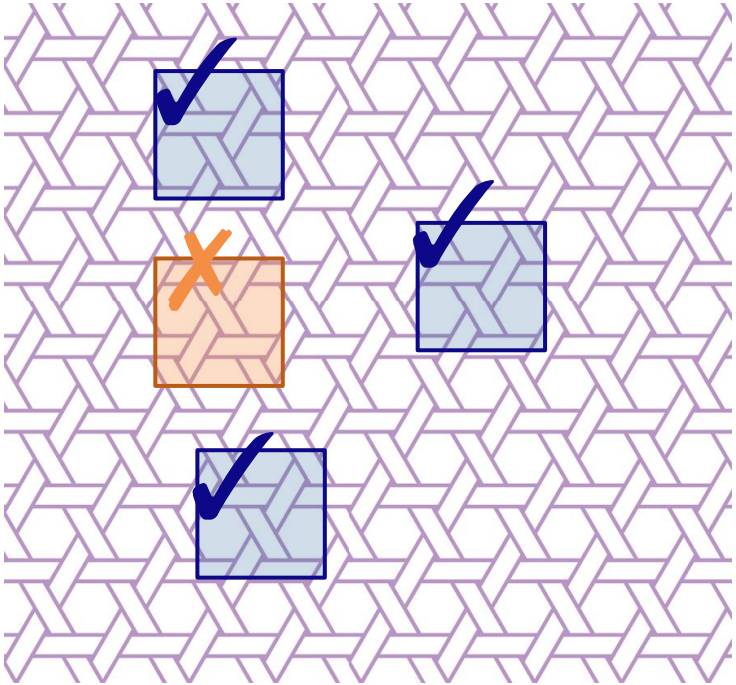
(Local symmetry)

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

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

Translation symmetry in 2D:

Features "mean" the same thing in any location.

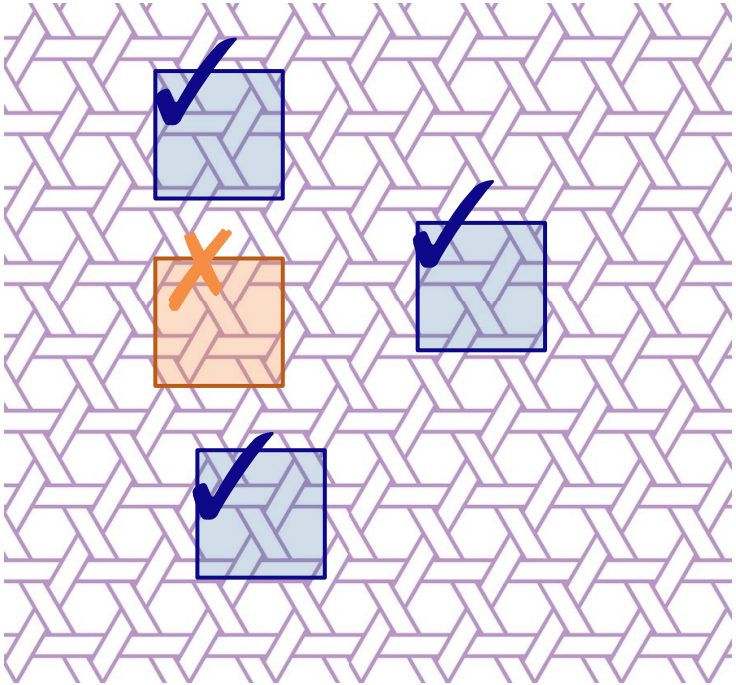


Symmetry emerges when different ways of representing something "mean" the same thing.

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

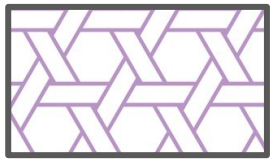
Translation symmetry in 2D:

Features "mean" the same thing in any location.

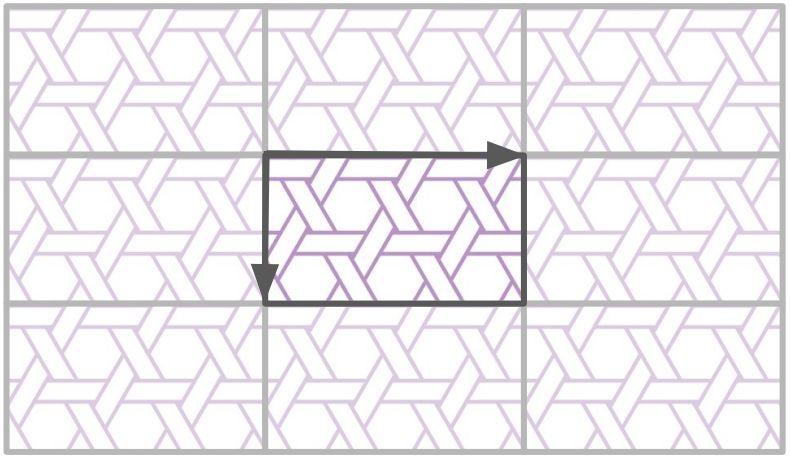


Symmetry of 2D objects

Boundaries "break" global translation symmetry.



Periodic boundary conditions preserve discrete translation symmetry.



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

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

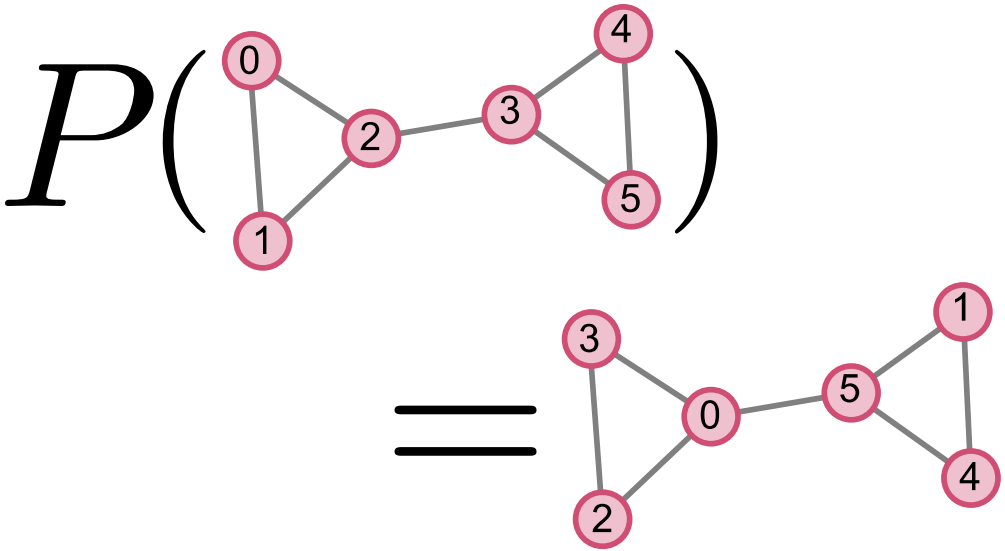
Permutation symmetry, S_N:

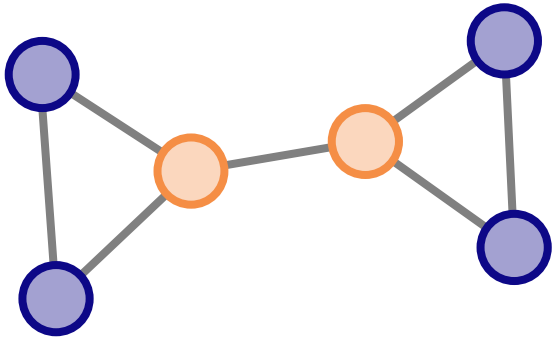
Symmetry of sets

The freedom to list things in any order

Symmetry of elements of a graph

Graph automorphism, specific nodes are indistinguishable (same global connectivity)



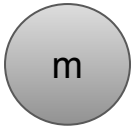


Geometric tensors take many forms. *They are a general data type beyond materials.*

Scalars

- Energy
- Mass
- Isotropic *

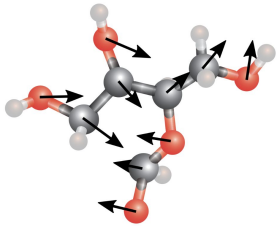
$$E$$



Vectors

- Force
- Velocity
- Acceleration
- Polarization

$$\vec{F}$$



Pseudovectors

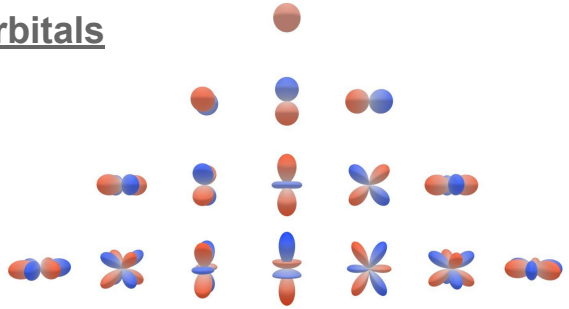
- Angular momentum
- Magnetic fields

Matrices, Tensors, …

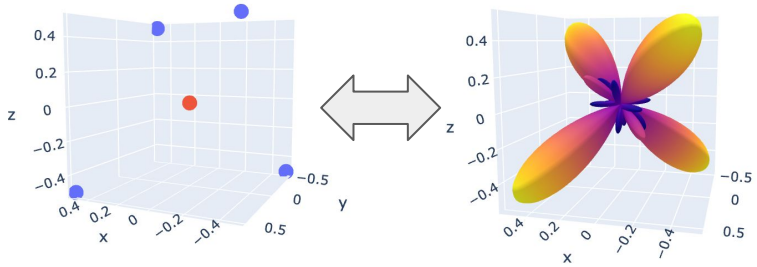
- Moment of Inertia
- Polarizability
- Interaction of multipoles
- Elasticity tensor (rank 4)

$$\alpha = \begin{bmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{bmatrix}$$

Atomic orbitals



Output of Angular Fourier Transforms



Vector fields on spheres (e.g. B-modes of the Cosmic Microwave Background)