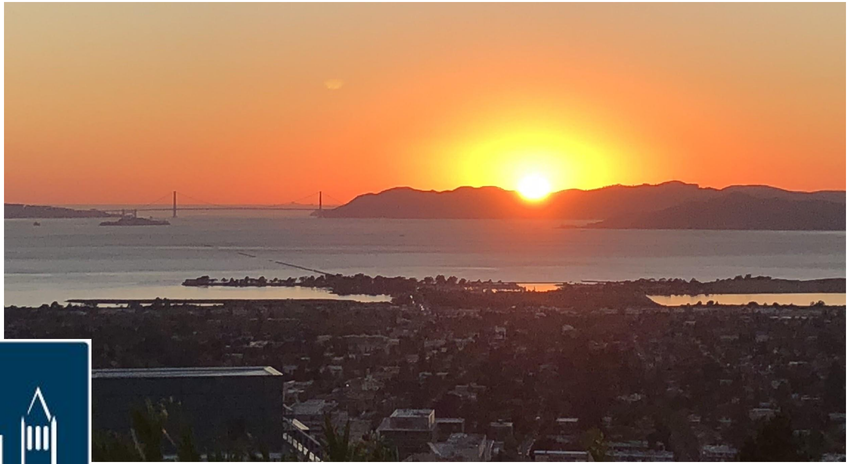
Euclidean Neural Networks...

rotation-, translation-, and permutation-equivariant convolutional neural networks for 3D point clouds

From Passive to Active: Generative and Reinforcement Learning with Physics
Machine Learning for Physics at IPAM
2019.09.27

...for emulating ab initio calculations and generating atomic geometries.



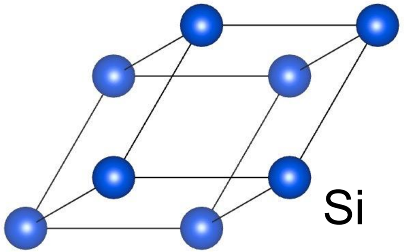
Tess Smidt
2018 Alvarez Postdoctoral Fellow in Computing Sciences

BERKELEY LAB

Research at Google

What a computational materials physicist does:

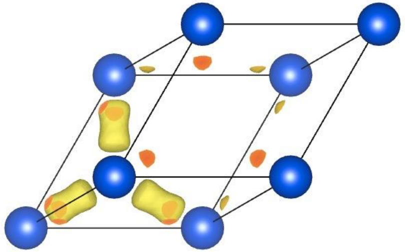
Given an atomic structure,



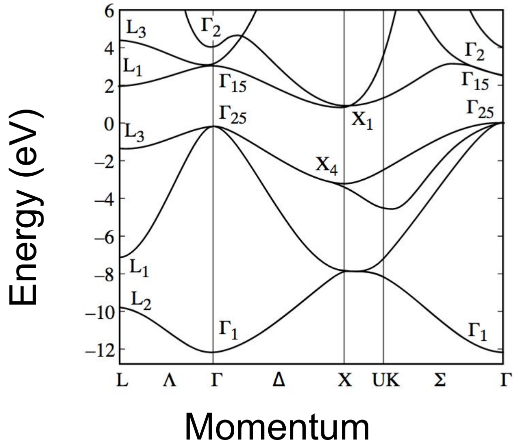
...use quantum theory and supercomputers to determine...

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

...where the electrons are...



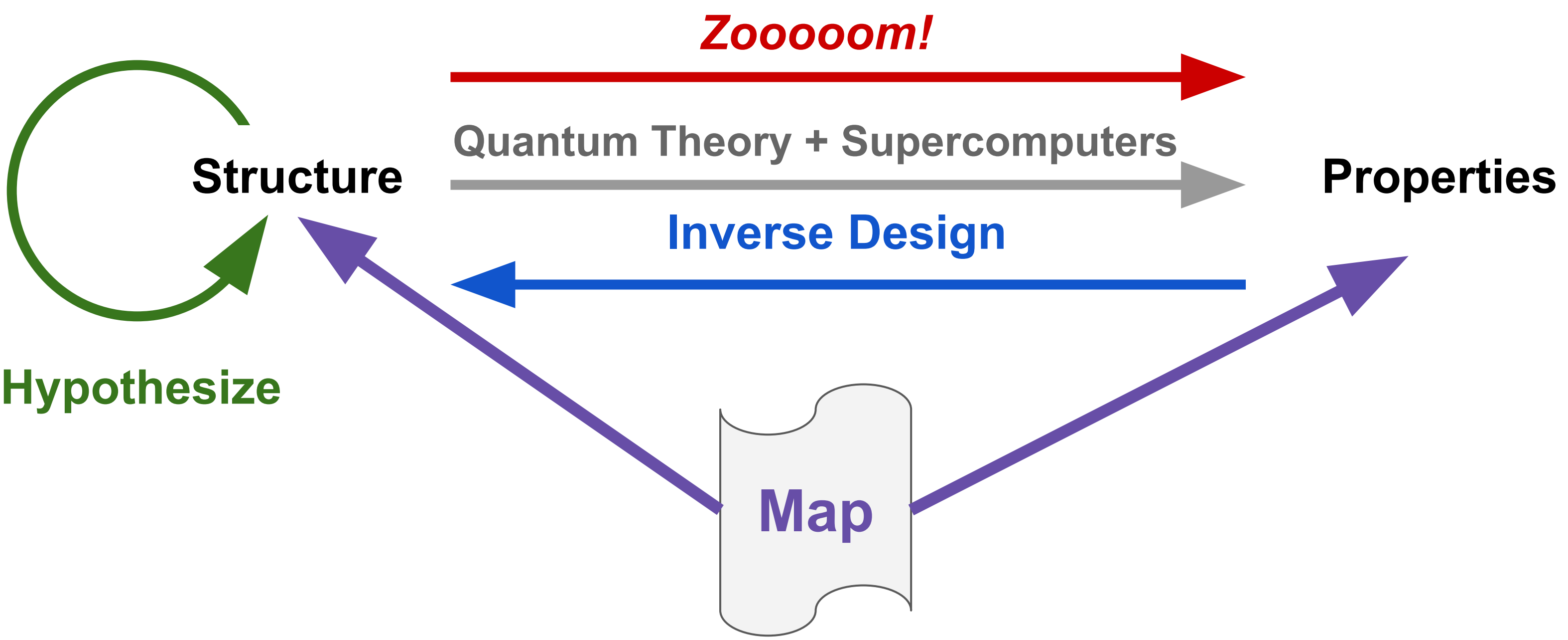
...and what the electrons are doing.



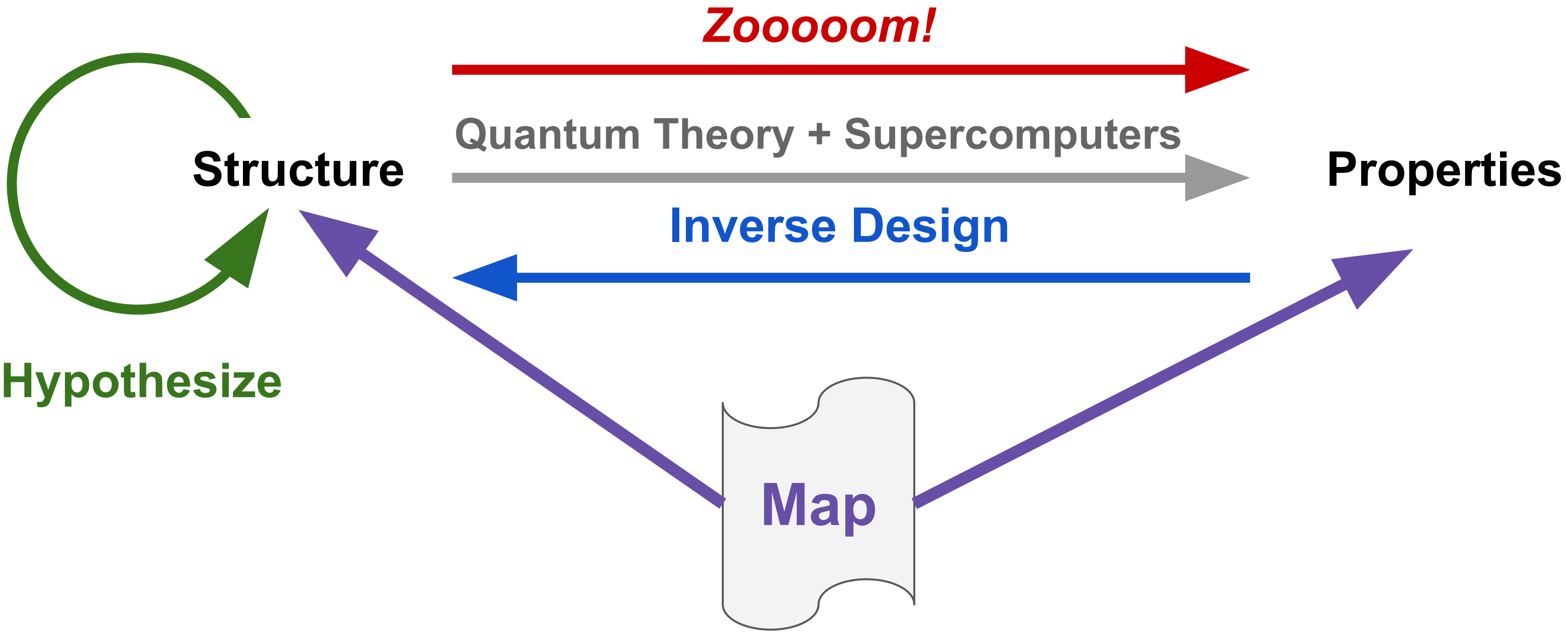
Structure

Properties

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



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

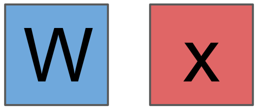


What types of neural networks are best suited for these tasks?

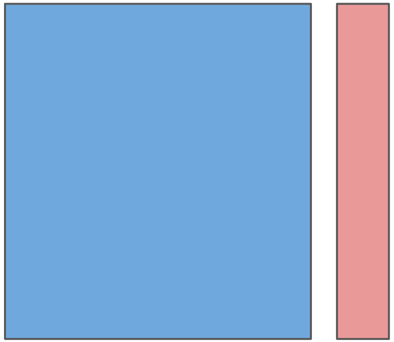
Neural networks are specially designed for different data types.
Assumptions about the data type are built into how the network operates.

W x

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

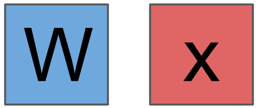


Vectors ➪ Dense NN

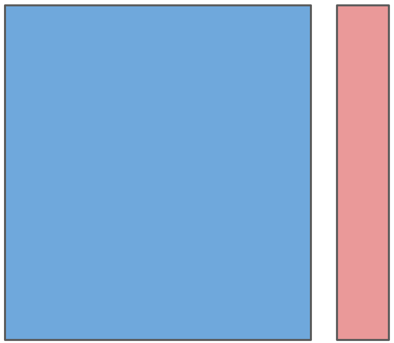


Components are independent.

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

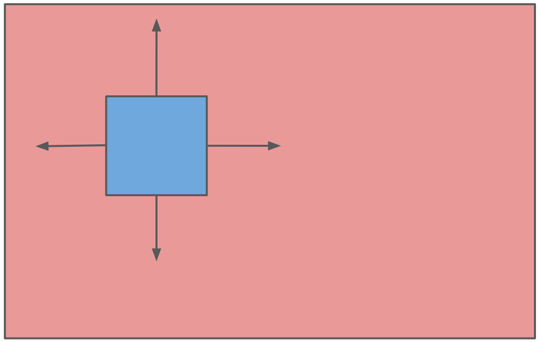


Vectors ➪ Dense NN



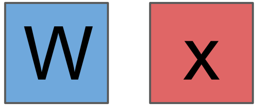
Components are independent.

2D images ➪ Convolutional NN

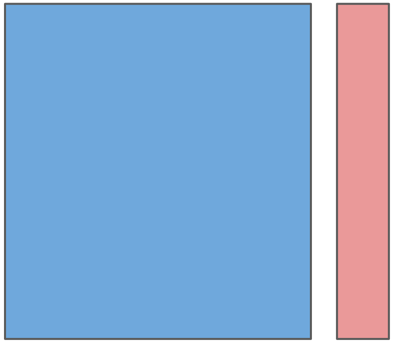


The same features can be found anywhere in an image. Locality.

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

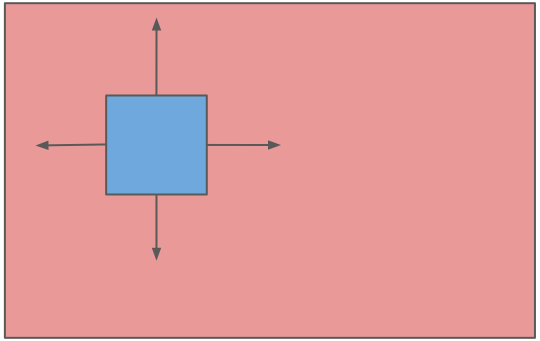


Vectors ➪ Dense NN



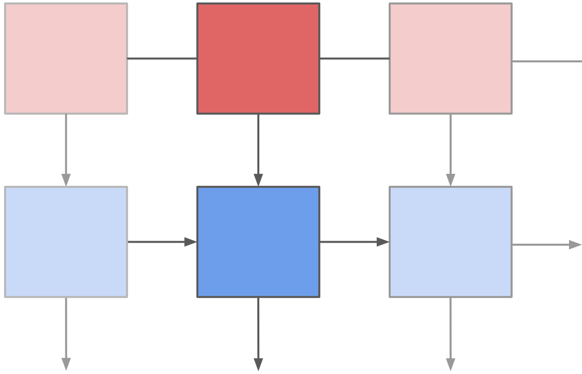
Components are independent.

2D images ➪ Convolutional NN



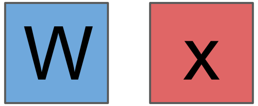
The same features can be found anywhere in an image. Locality.

Text ➪ Recurrent NN

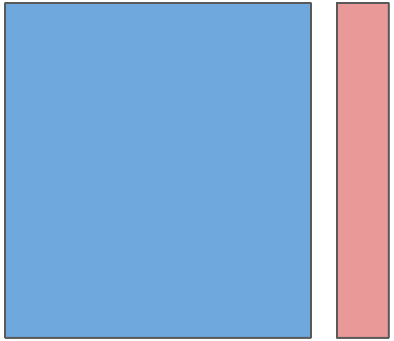


Sequential data. Next input/output depends on input/output that has come before.

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

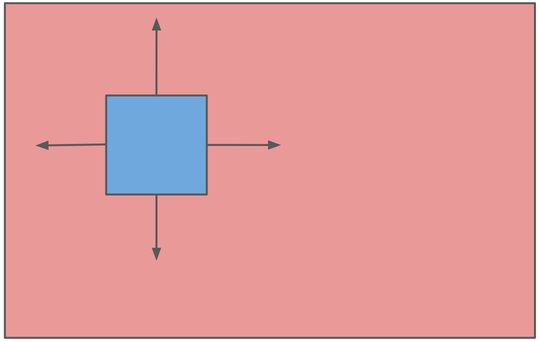


Vectors ➪ Dense NN



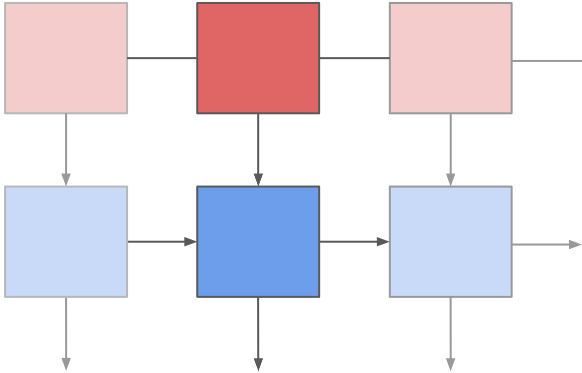
Components are independent.

2D images ➪ Convolutional NN



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

Text ➪ Recurrent NN

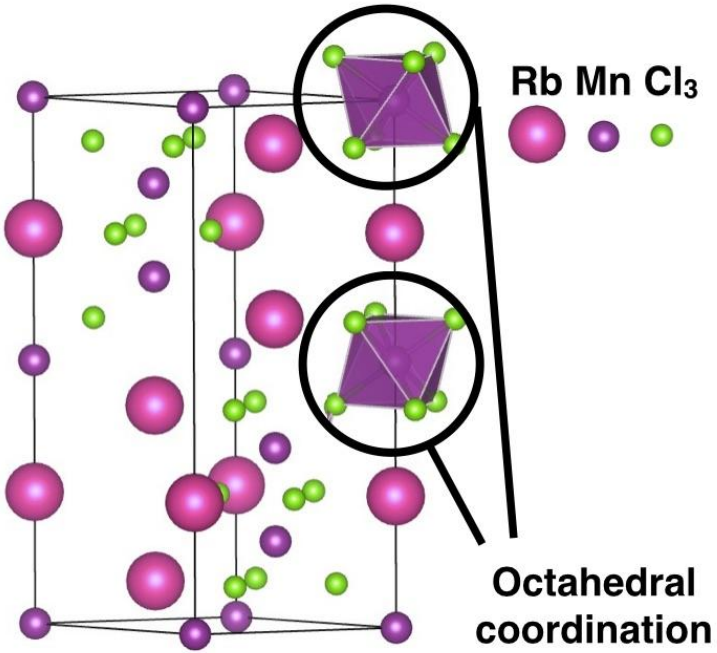


Sequential data. Next input/output depends on input/output that has come before.

What are our data types in materials physics?
How do we build neural networks for these data types?

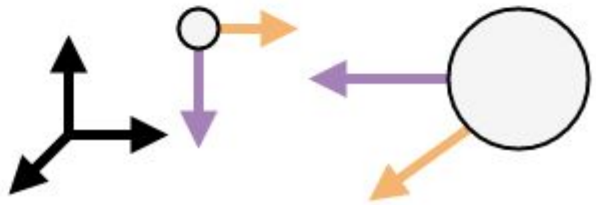
What assumptions do we want "built in" to our neural networks (for materials data)?

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

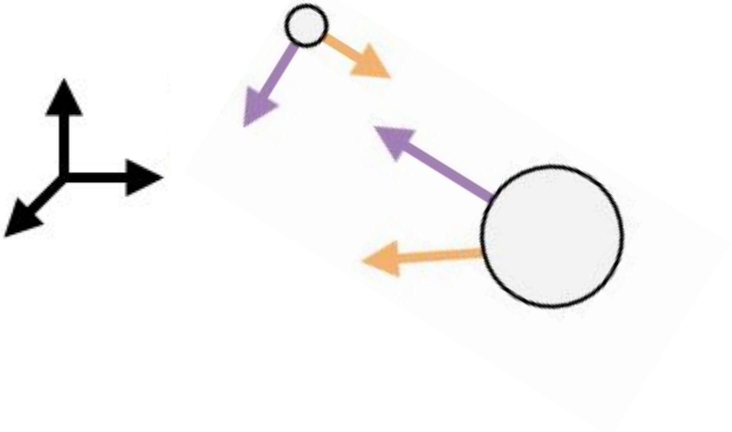


The properties of physical systems transform predictably under rotation.

Two point **masses** with **velocity** and **acceleration.**

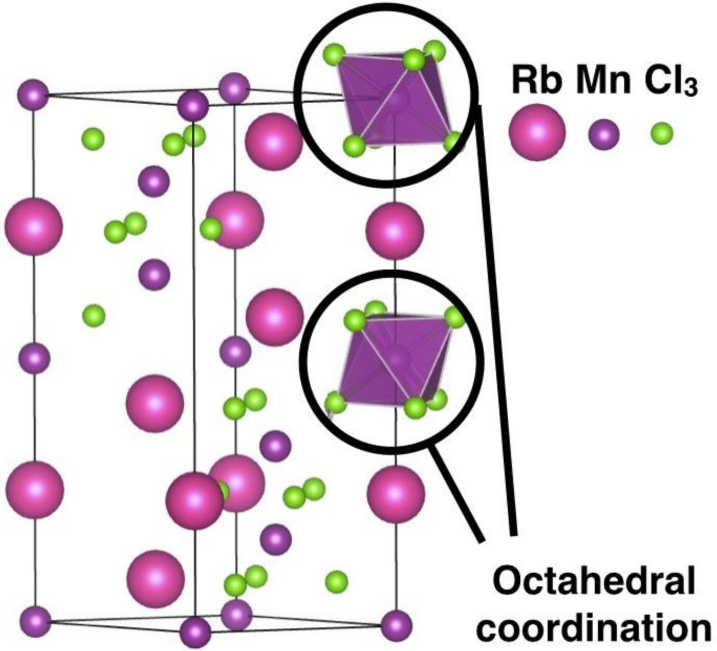


Same system, with rotated coordinates.



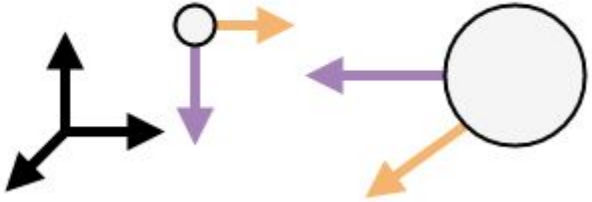
What assumptions do we want “built in” to our neural networks (for materials data)?

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

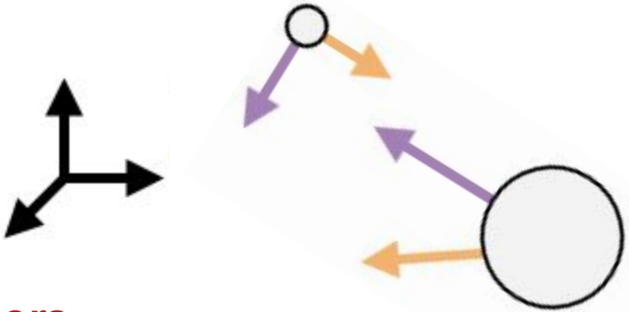


The properties of physical systems transform predictably under rotation.

Two point **masses** with **velocity** and **acceleration.**



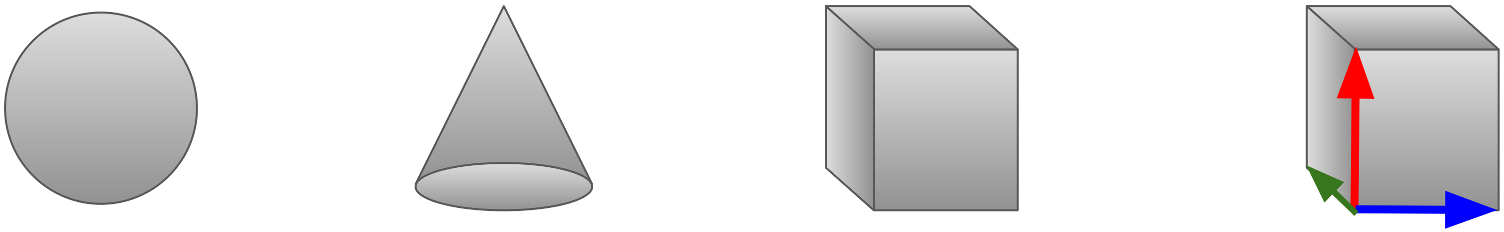
Same system, with rotated coordinates.



Our data types are geometry and geometric tensors.

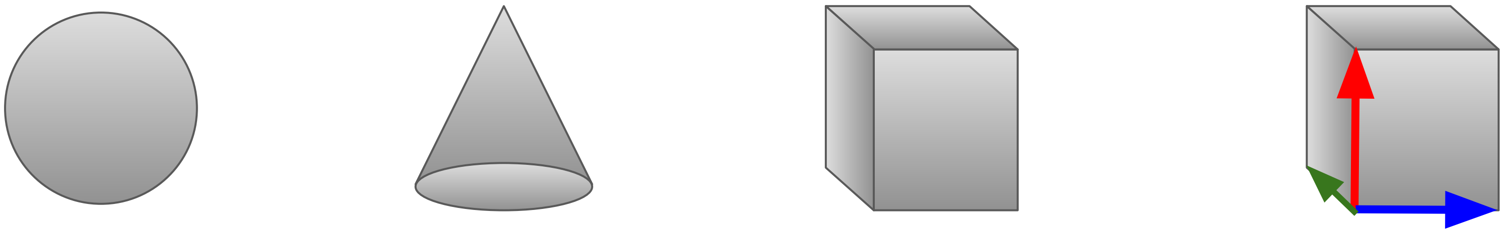
These data types assume Euclidean symmetry (3D translations, 3D rotations, and inversion).

Space has Euclidean symmetry, E(3). Objects break that symmetry.
The broken symmetry is a subgroup of E(3).



Space has Euclidean symmetry, E(3). Objects break that symmetry.
The broken symmetry is a subgroup of E(3).

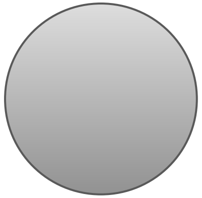
3D rotations and
inversions



O(3)

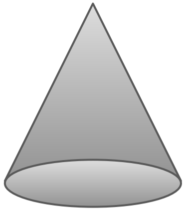
Space has Euclidean symmetry, E(3). Objects break that symmetry.
The broken symmetry is a subgroup of E(3).

3D rotations and inversions



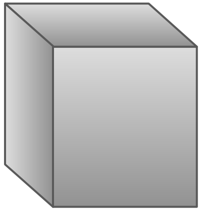
O(3)

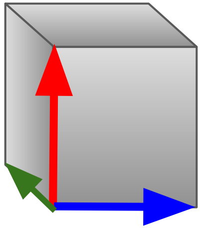
2D rotation and mirrors along cone axis



SO(2) + mirrors

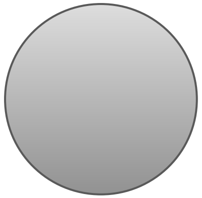
($C_{\infty v}$)





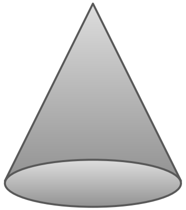
Space has Euclidean symmetry, E(3). Objects break that symmetry.
The broken symmetry is a subgroup of E(3).

3D rotations and inversions



O(3)

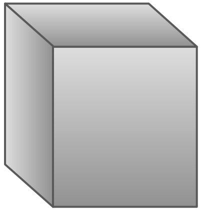
2D rotation and mirrors along cone axis



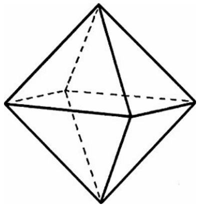
SO(2) + mirrors

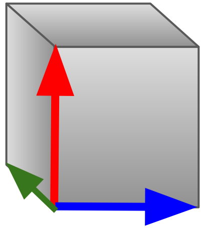
($C_{\infty v}$)

Discrete rotations and mirrors



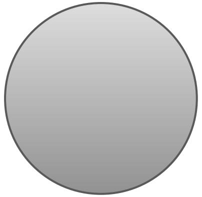
O_h





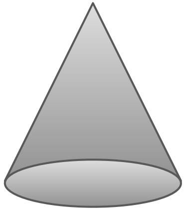
Space has Euclidean symmetry, E(3). Objects break that symmetry.
The broken symmetry is a subgroup of E(3).

3D rotations and inversions



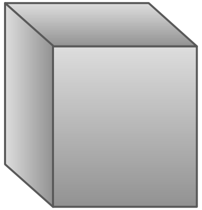
O(3)

2D rotation and mirrors along cone axis

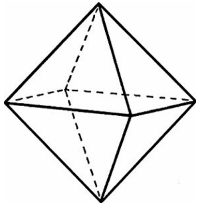


SO(2) + mirrors
($C_{\infty v}$)

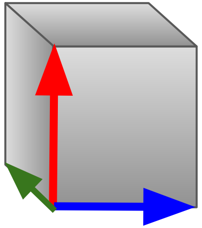
Discrete rotations and mirrors



O_h

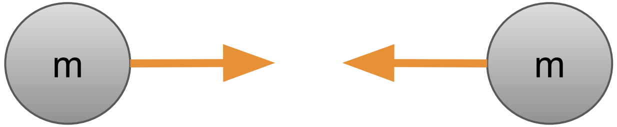


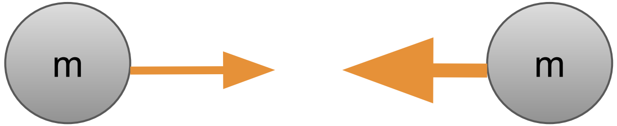
Discrete rotations, mirrors, and translations



Pm-3m (221)

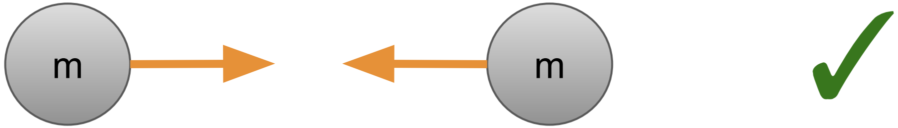
Properties of a system must be compatible with symmetry.
Which of these situations (inputs / outputs) are symmetrically allowed / forbidden?

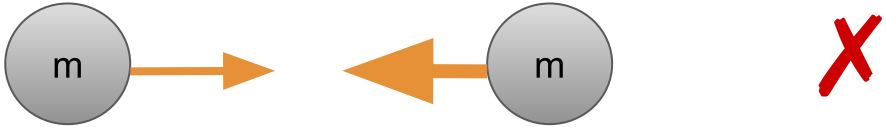
a. 

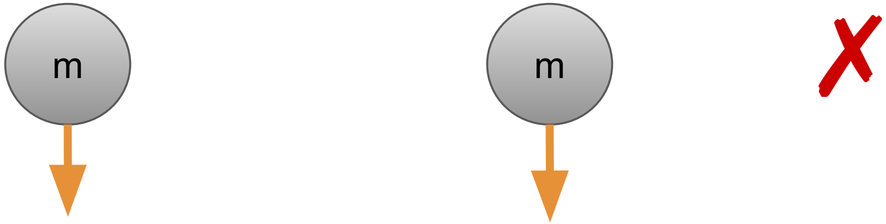
b. 

c. 

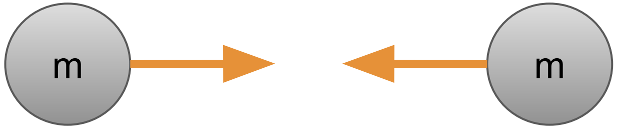
Properties of a system must be compatible with symmetry.
Which of these situations (inputs / outputs) are symmetrically allowed / forbidden?

a.


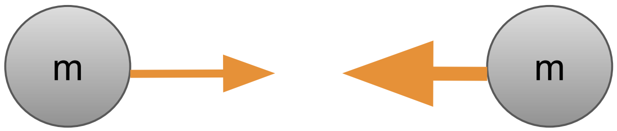
b.


c.


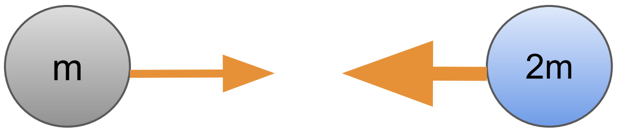
Properties of a system must be compatible with symmetry.
Which of these situations (inputs / outputs) are symmetrically allowed / forbidden?

a. 

✓

b. 

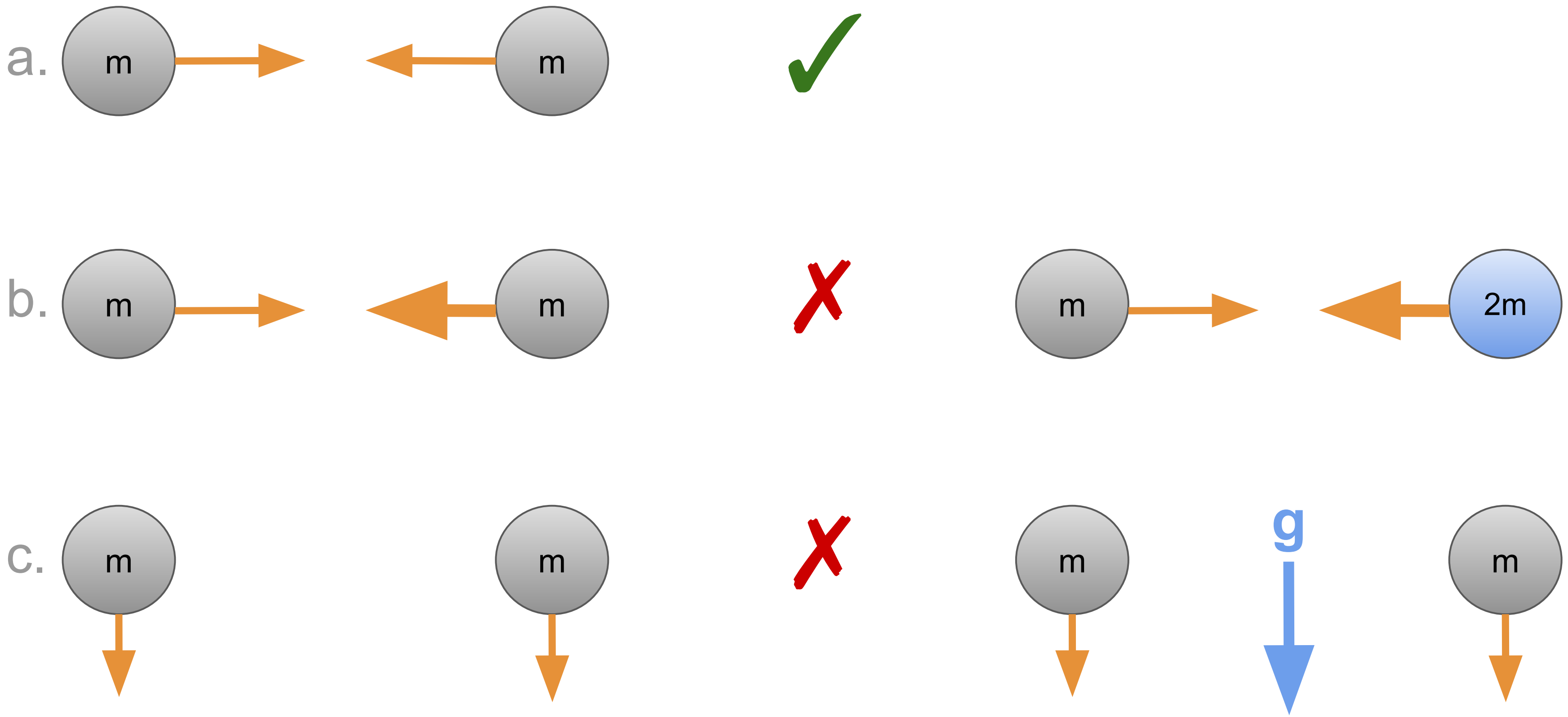
✗



c. 

✗

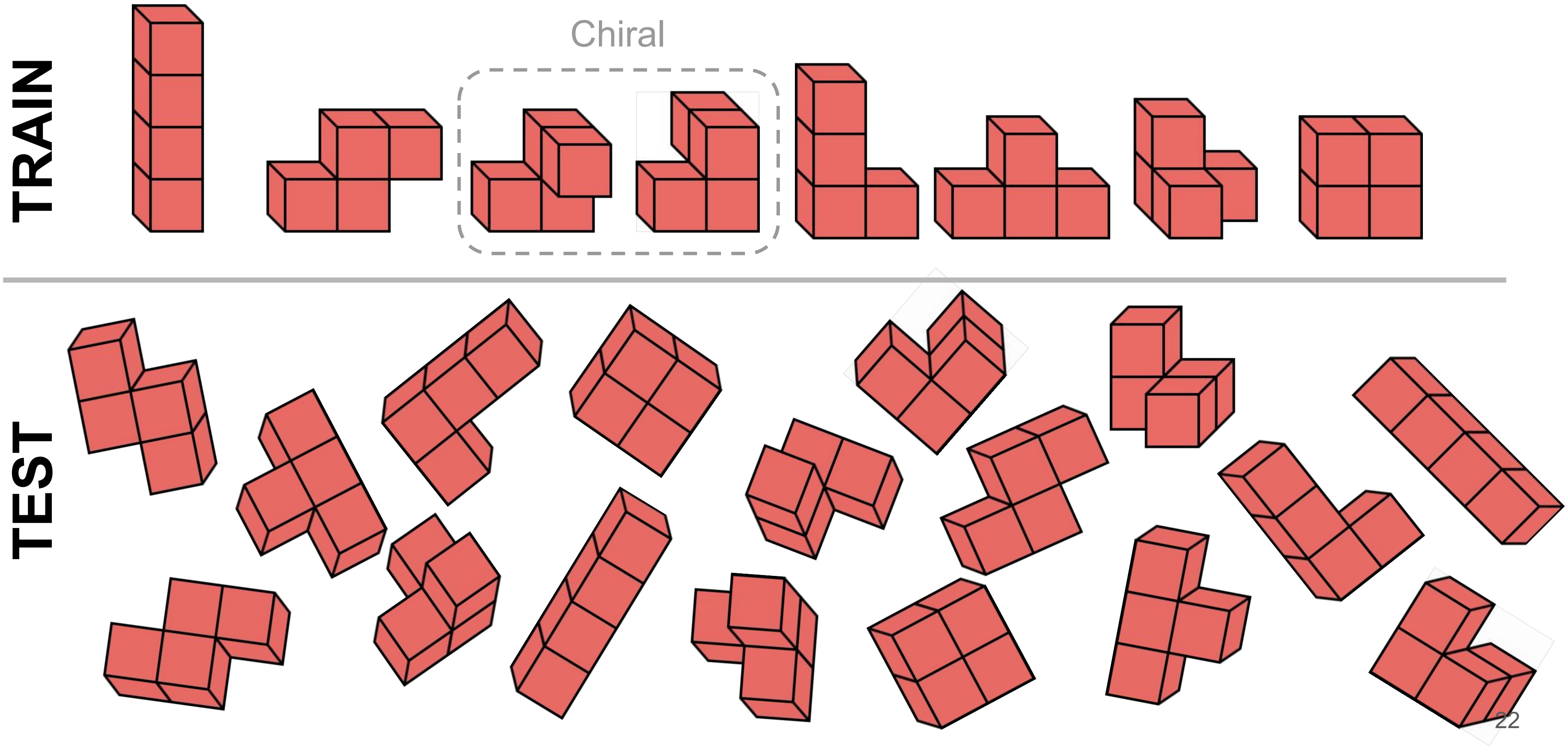
Properties of a system must be compatible with symmetry.
Which of these situations (inputs / outputs) are symmetrically allowed / forbidden?



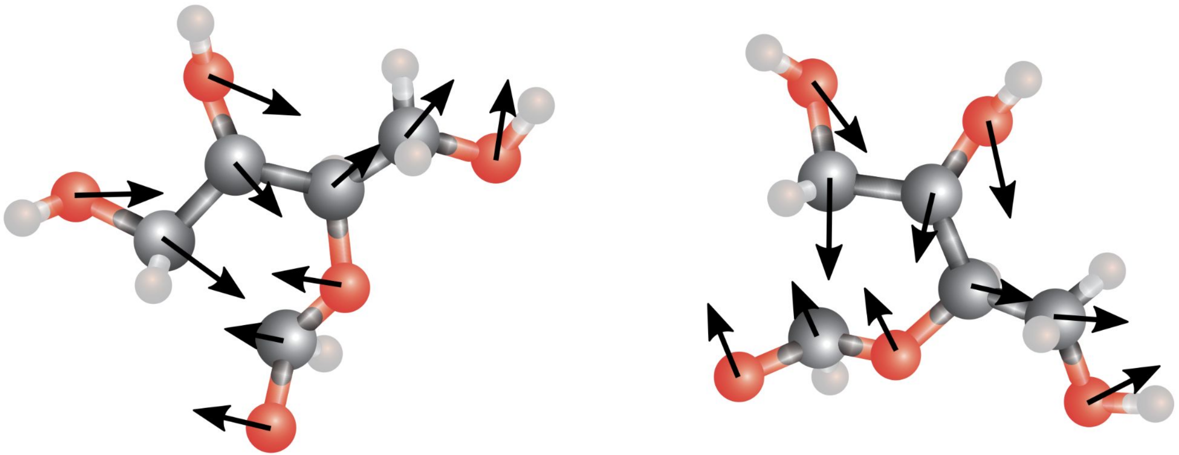
We build neural networks with Euclidean symmetry, E(3) and SE(3).

- What neural networks with Euclidean symmetry can do.
- How Euclidean Neural Networks work.
- Applications of Euclidean Neural Networks.

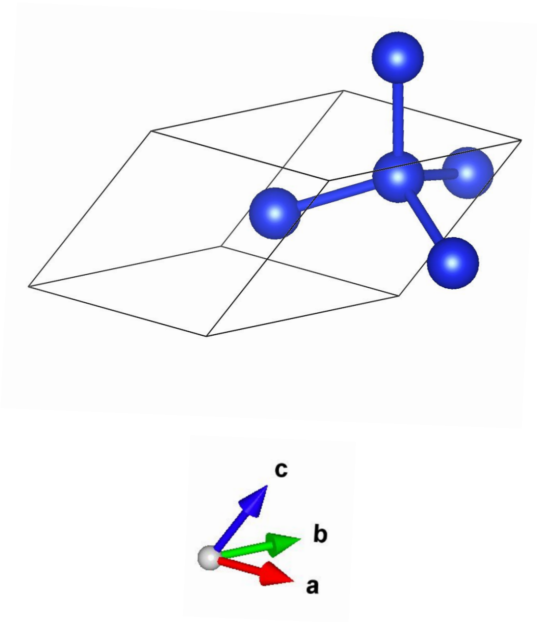
Trained on 3D Tetris shapes in one orientation, these network can perfectly identify these shapes in any orientation.

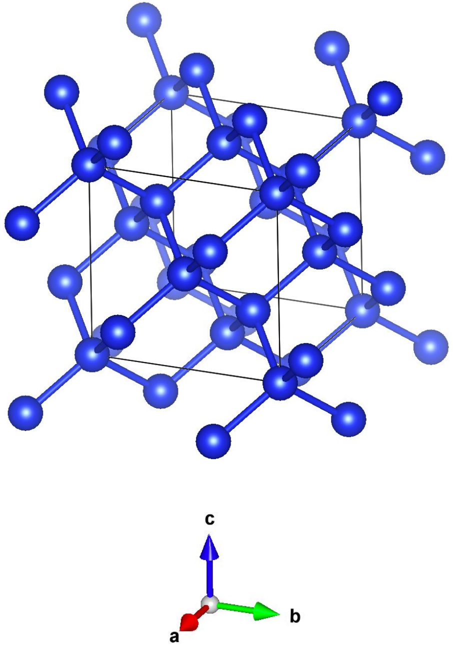


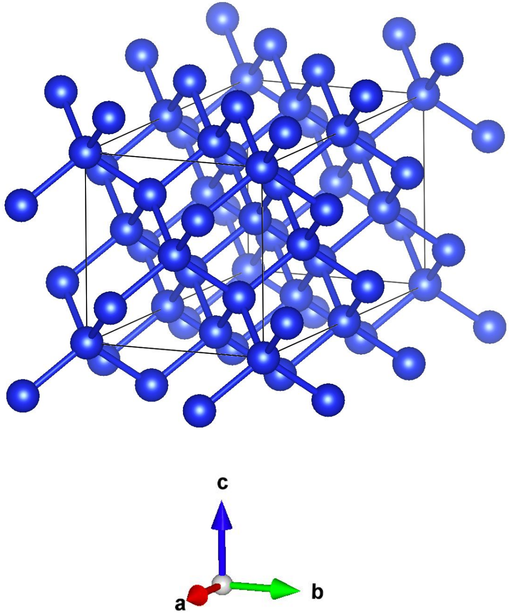
Given a molecule and a rotated copy,
the predicted forces are the same up to rotation.
(Predicted forces are equivariant to rotation.)



To these networks, primitive unit cells, conventional unit cells, and supercells of the same crystal will produce the same output (assuming periodic boundary conditions)**.**







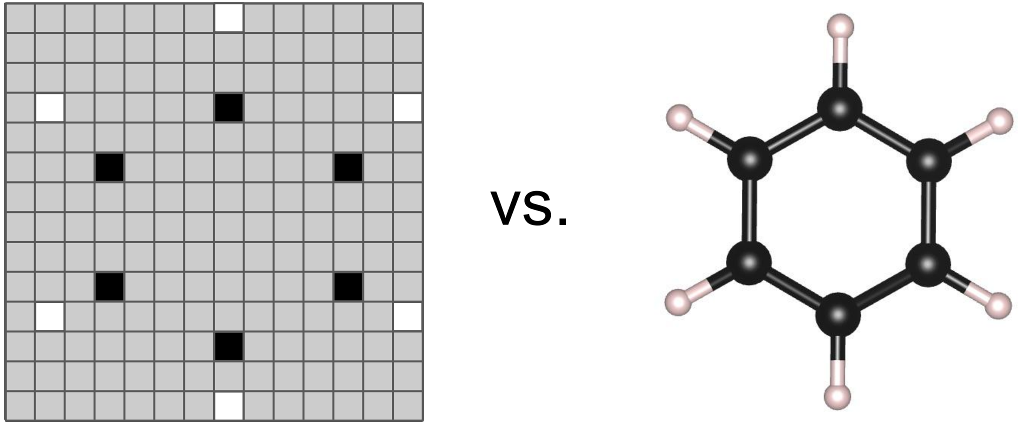
We build neural networks with Euclidean symmetry, E(3) and SE(3).

- What neural networks with Euclidean symmetry can do.

- How Euclidean Neural Networks work.
  - Overview
  - Input to network
  - Network operations
  - Visualizing kernels
  - Interpreting input / output

- Applications of Euclidean Neural Networks.

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

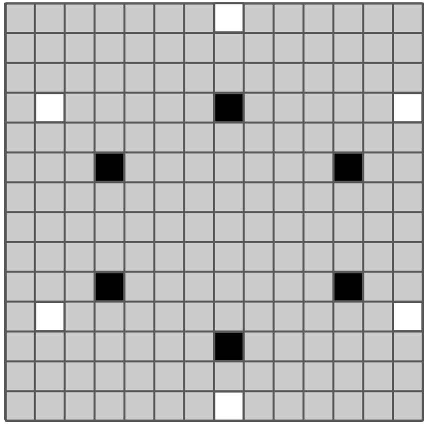


We use continuous convolutions with atoms as convolution centers.

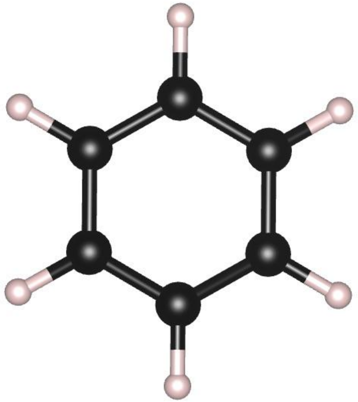
K. T. Schütt et al, NIPS 30 (2017).
(arXiv: 1706.08566)

We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).

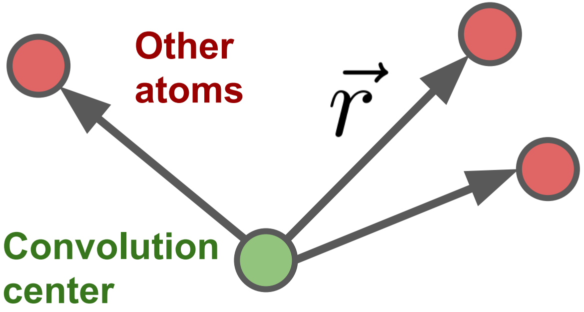
We use points. Images of atomic systems are sparse and imprecise.



vs.



We use continuous convolutions with atoms as convolution centers.

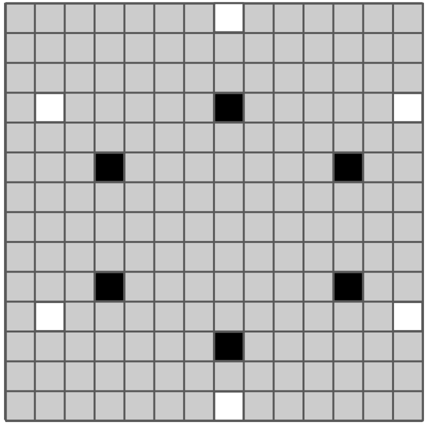


K. T. Schütt et al, NIPS 30 (2017). (arXiv: 1706.08566)

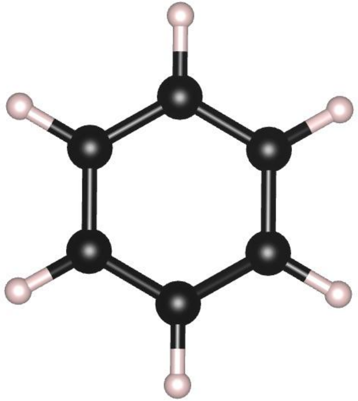
We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).

$g \in SE(3)$

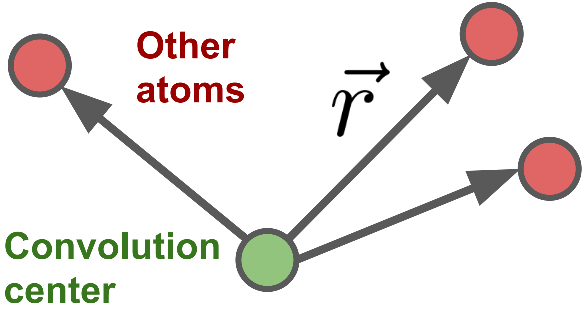
We use points. Images of atomic systems are sparse and imprecise.



vs.

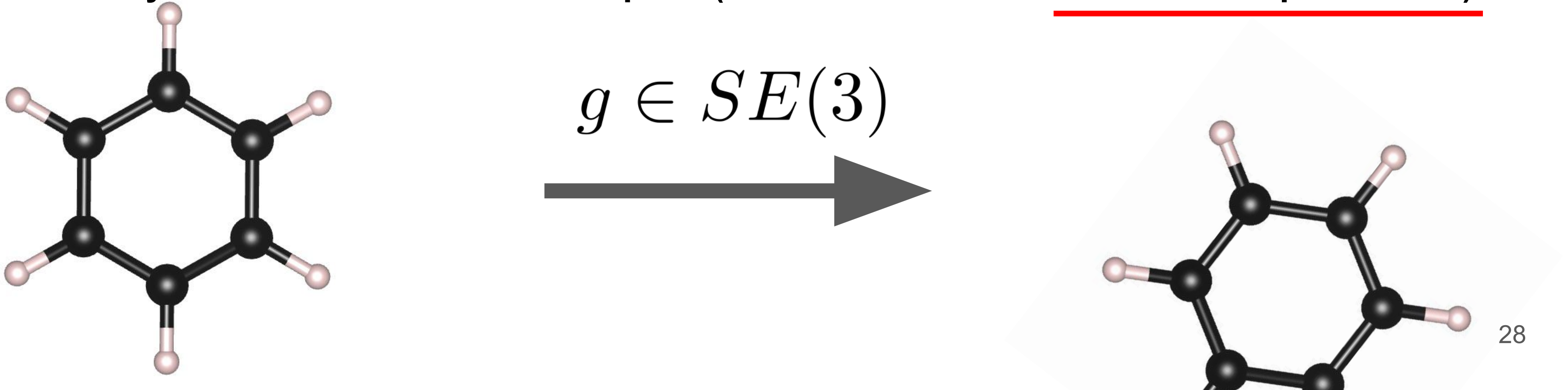


We use continuous convolutions with atoms as convolution centers.

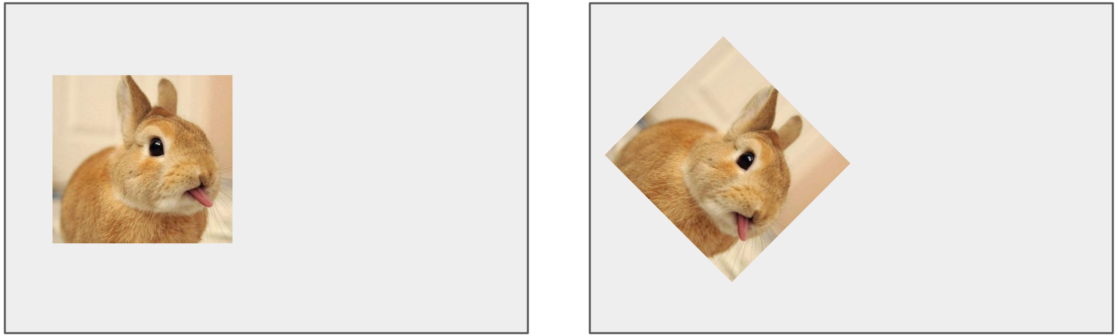


K. T. Schütt et al, NIPS 30 (2017). (arXiv: 1706.08566)

We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).



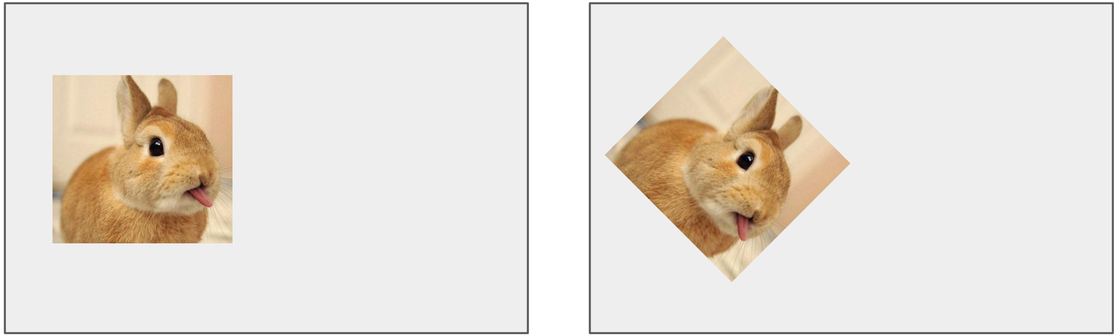
Translation equivariance



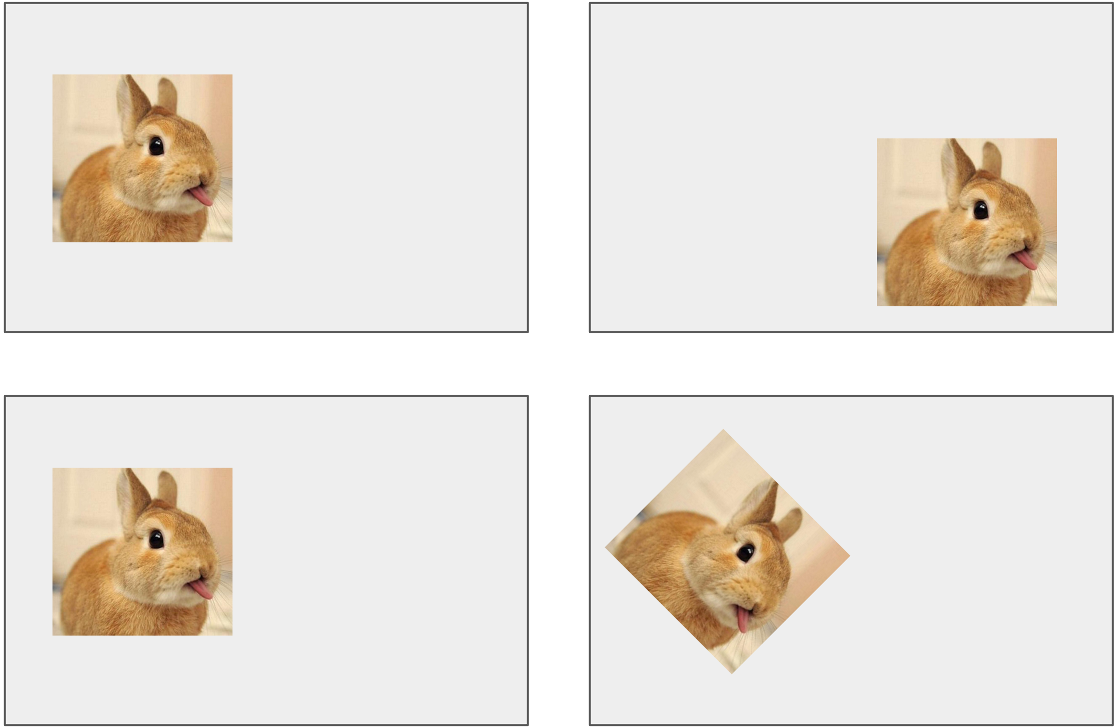
Rotation equivariance

Translation equivariance

Convolutional neural network ✓



Rotation equivariance?



Translation equivariance

Convolutional neural network ✓

Rotation equivariance

~~Data augmentation~~

~~Radial functions~~

Want a network that both preserves geometry and exploits symmetry.

Several groups converged on similar ideas around the same time.

Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds
(arXiv:1802.08219)
Tess Smidt*, Nathaniel Thomas*, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, Patrick Riley

Points, nonlinearity on norm of tensors

Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network
(arXiv:1806.09231)
Risi Kondor, Zhen Lin, Shubhendu Trivedi

Only use tensor product as nonlinearity, no radial function

3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data
(arXiv:1807.02547)
Mario Geiger*, Maurice Weiler*, Max Welling, Wouter Boomsma, Taco Cohen

Efficient framework for voxels, gated nonlinearity

**denotes equal contribution*

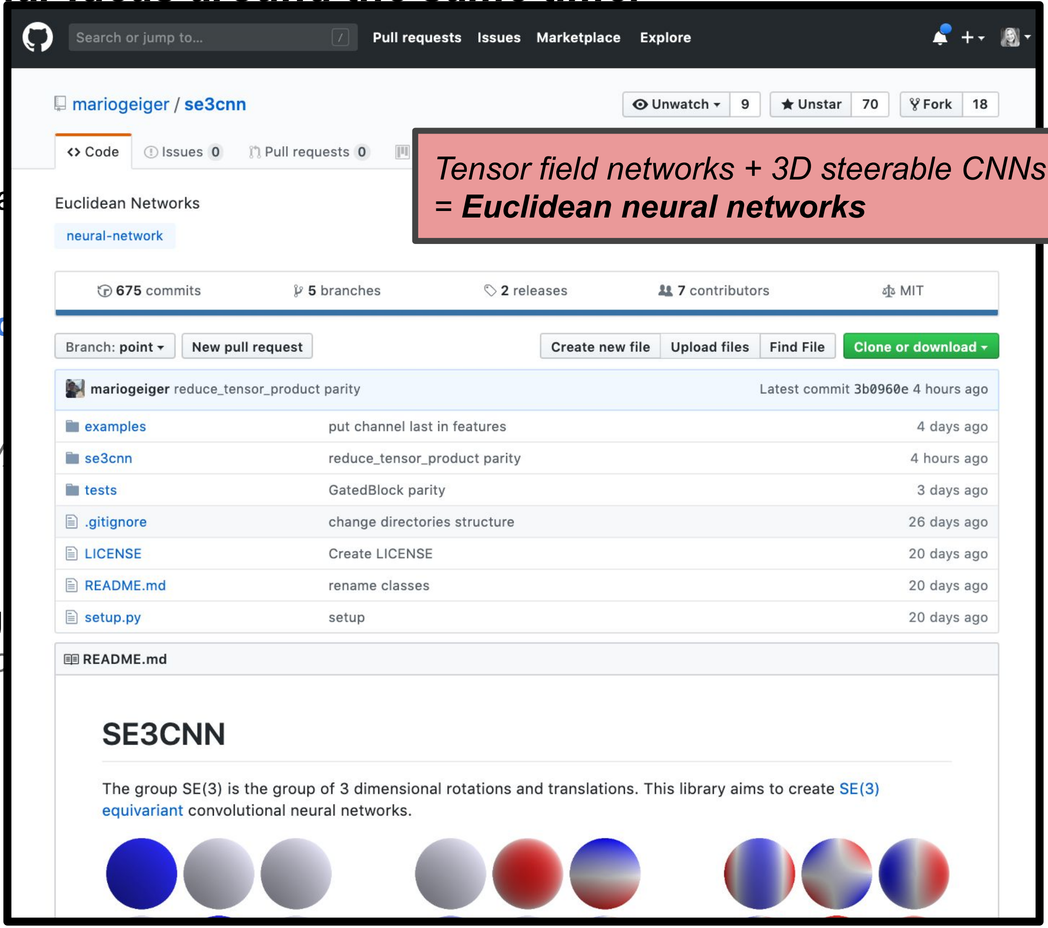
Several groups converged on similar ideas around the same time.

Tensor field networks: Rotation- and trans
(arXiv:1802.08219)
Tess Smidt*, Nathaniel Thomas*, Steven Kea
Points, nonlinearity on norm of tensors

Clebsch-Gordan Nets: a Fully Fourier Spac
(arXiv:1806.09231)
Risi Kondor, Zhen Lin, Shubhendu Trivedi
Only use tensor product as nonlinearity

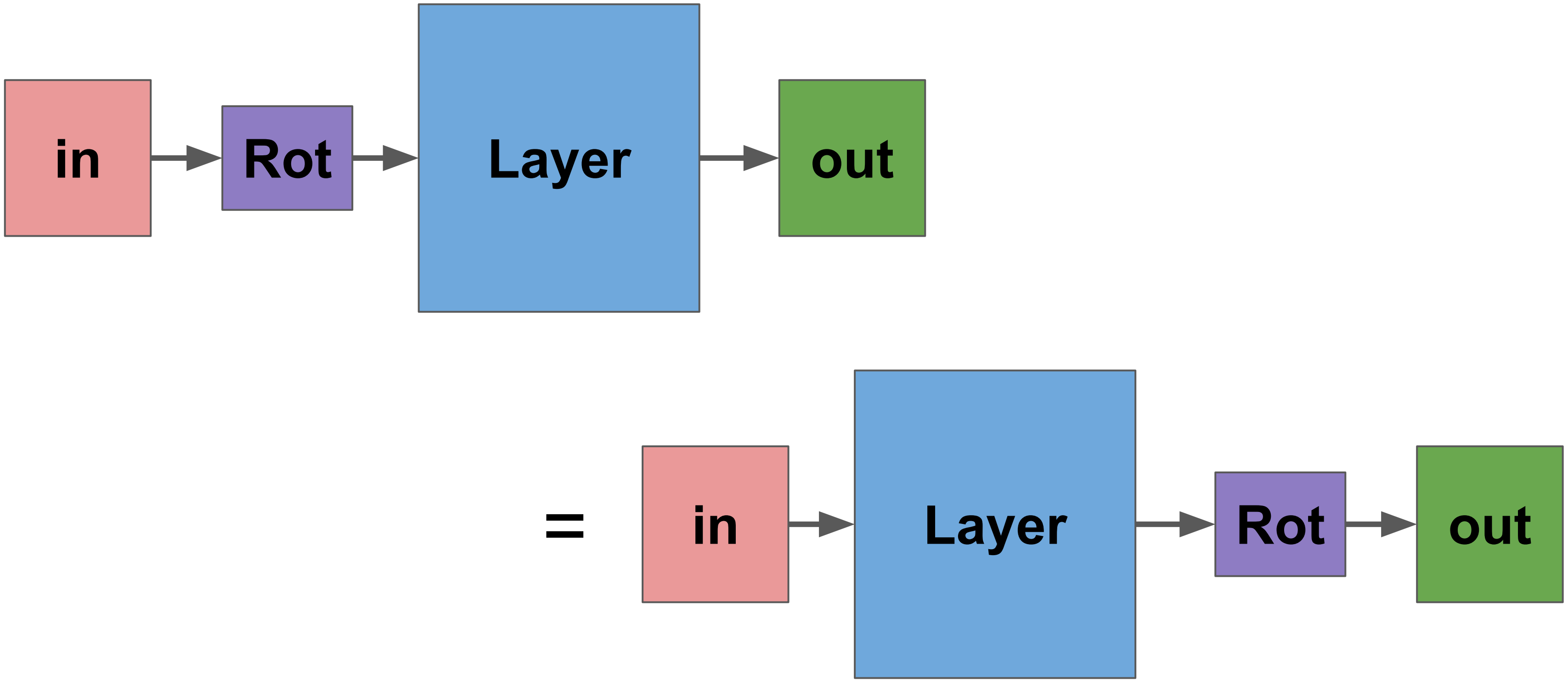
3D Steerable CNNs: Learning Rotationally
(arXiv:1807.02547)
Mario Geiger*, Maurice Weiler*, Max Welling
Efficient framework for voxels, gated no

**denotes equal contribution*

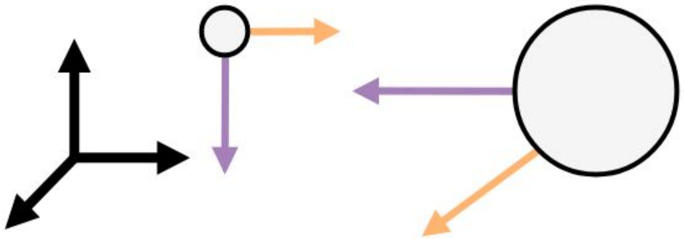


Tensor field networks + 3D steerable CNNs
*= **Euclidean neural networks***

To be rotation-equivariant means that we can rotate our inputs <u>OR</u> rotate our outputs and we get the same answer.



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

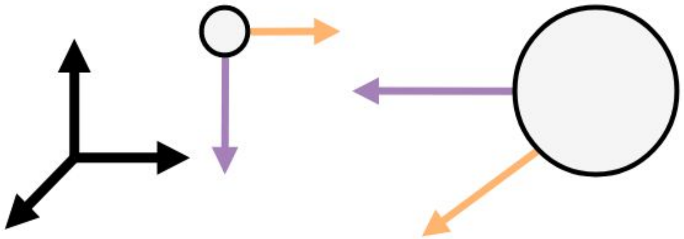


```
[[[m0]],[[m1]]],
[[[v0x, v0y, v0z],[a0x, a0y, a0z]],
 [[v1x, v1y, v1z],[a1x, a1y, a1z]]]
```

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

We categorize our features by how they transform under rotation.

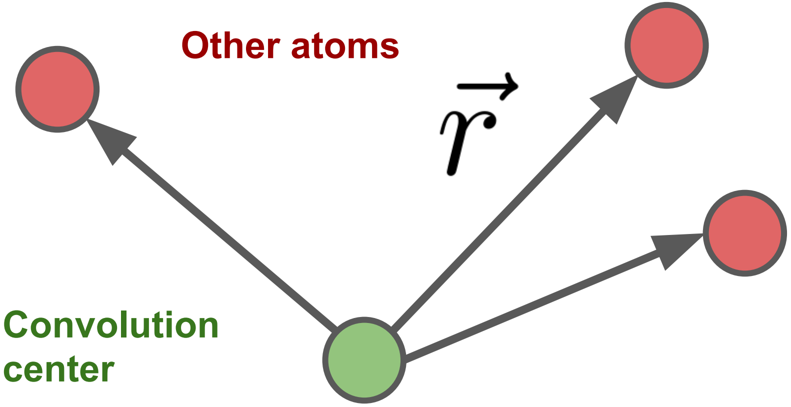
Features have "angular frequency" L where L is a positive integer.



```
{0: [[[m0]],[[m1]]],
 1: [[[v0x, v0y, v0z],[a0x, a0y, a0z]],
     [[v1x, v1y, v1z],[a1x, a1y, a1z]]]}
```

	Frequency	
Scalars	$l = 0$	Doesn't change with rotation
Vectors	$l = 1$	Changes with same frequency as rotation
3x3 Matrices	$l = 0 \oplus 1 \oplus 2$	

The convolutional kernels are built from functions with “angular frequency” L
⇨ *Spherical harmonics*.



Learned Parameters

with no symmetry:

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

with SO(3) symmetry:

$$\boxed{R(r)}\,Y_l^m(\hat{r})$$

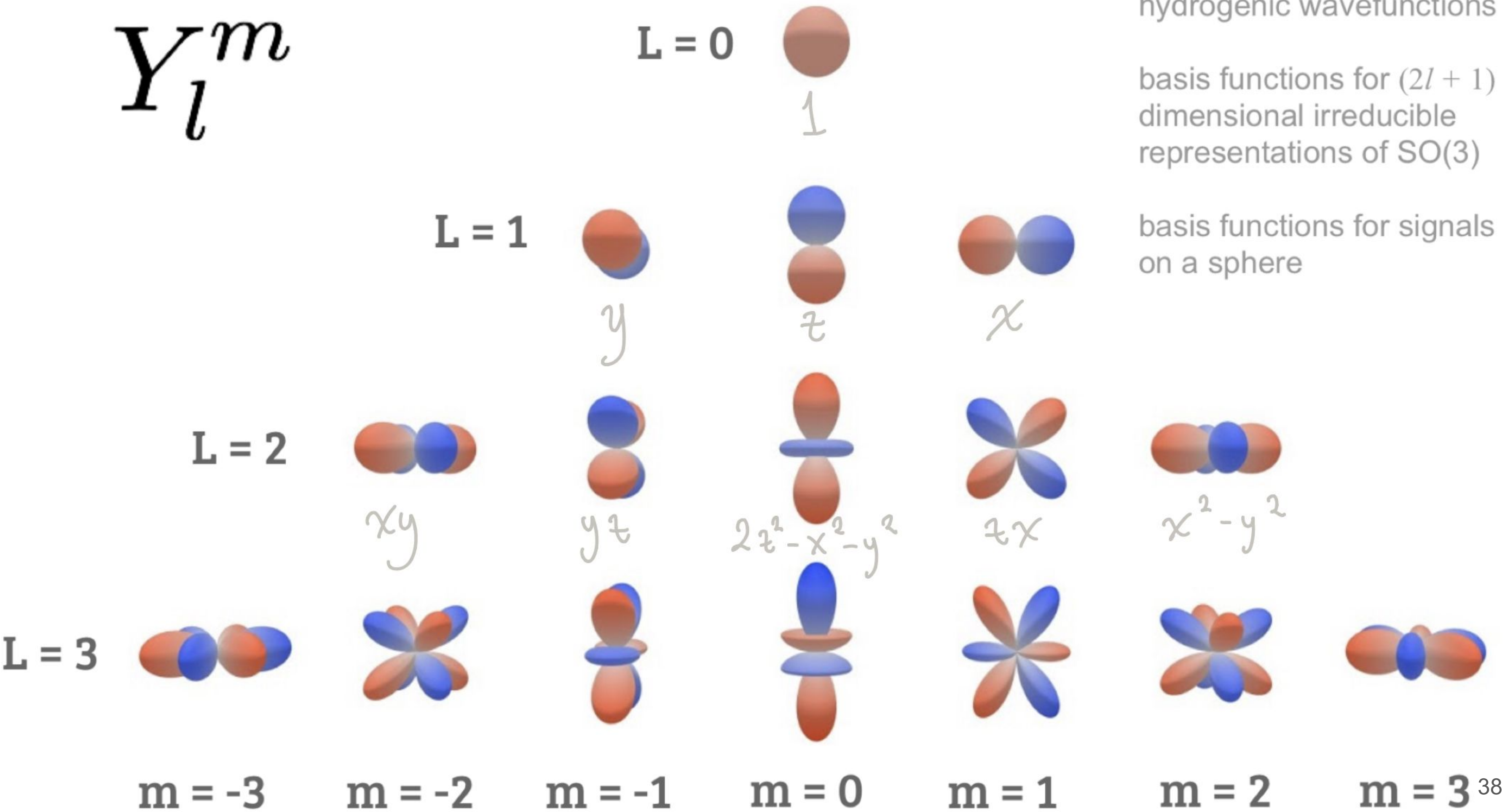
Spherical harmonics

$$Y_l^m$$

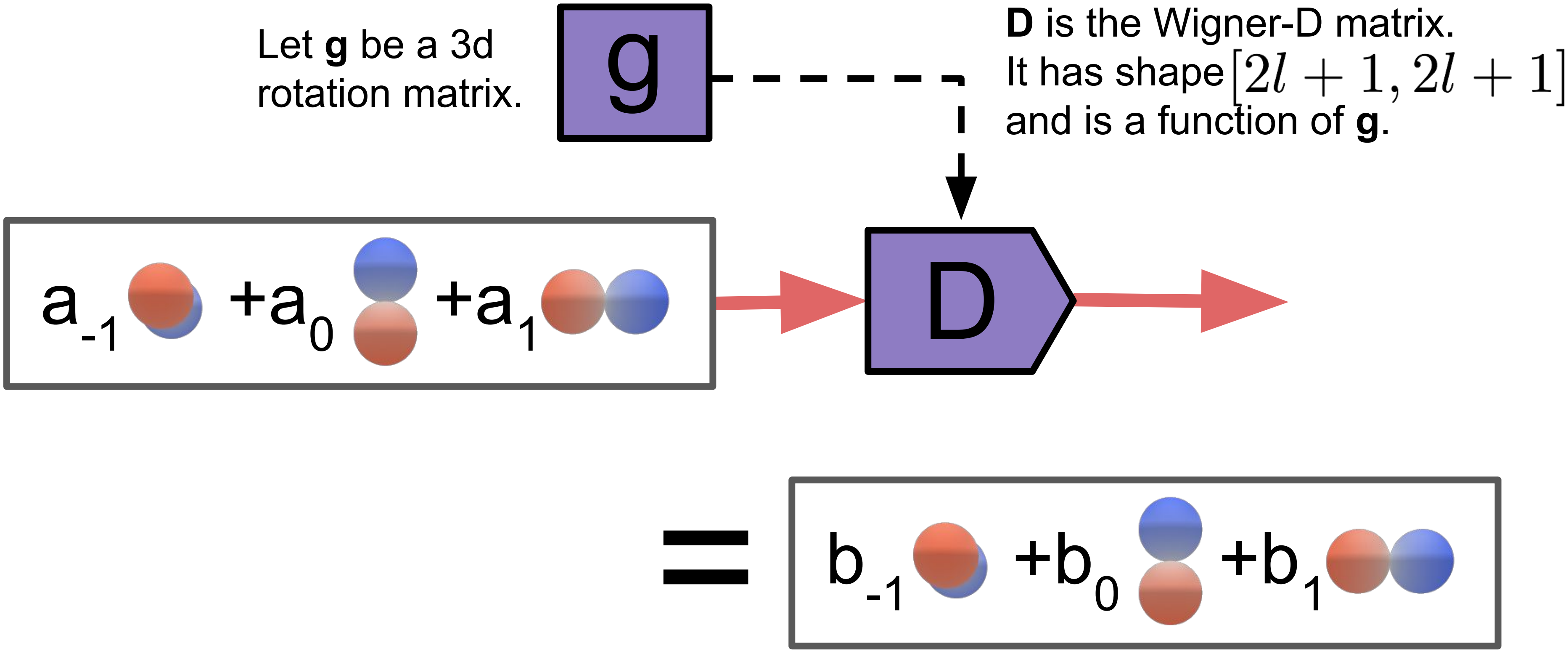
angular portion of hydrogenic wavefunctions

basis functions for $(2l + 1)$ dimensional irreducible representations of SO(3)

basis functions for signals on a sphere

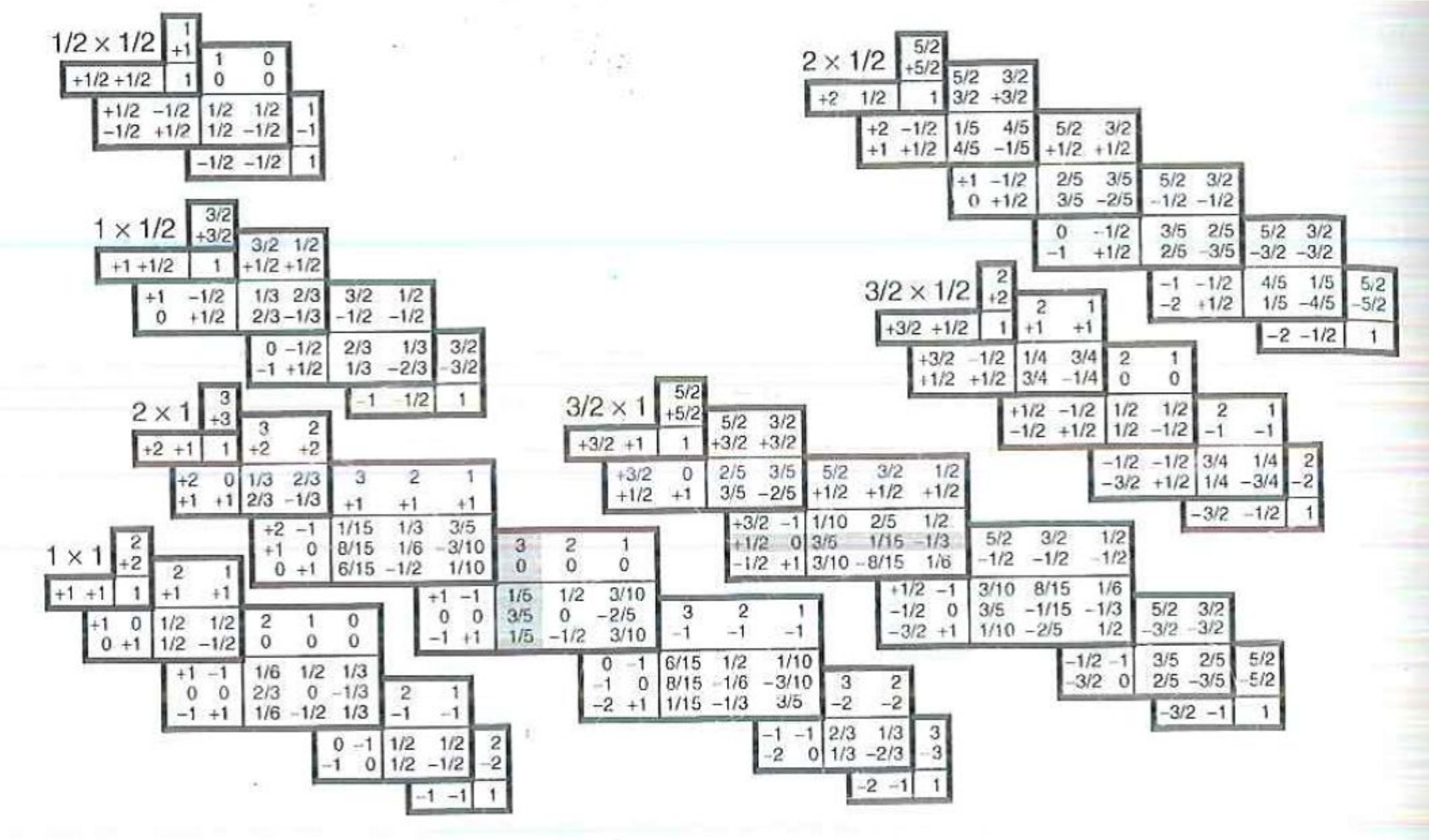


Spherical harmonics of a given L transform together under rotation.



Features and kernels are not simply scalars.
We use <u>tensor products</u> with <u>Clebsch-Gordan coefficients</u> to combine.

Same math involved in the addition of angular momentum.



D. Griffiths, Introduction to quantum mechanics

Examples of tensor product: How to combine a scalar and a vector? Easy!

$$a \times \vec{b} = \vec{c}$$

Angular Frequency

$$1$$

Examples of tensor product: How to combine two vectors? Many ways.

Angular Frequency

Dot product

$$\begin{pmatrix} a_i & a_j & a_k \end{pmatrix} \begin{pmatrix} b_i \\ b_j \\ b_k \end{pmatrix} = c$$

$$0$$

Cross product

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_i & a_j & a_k \\ b_i & b_j & b_k \end{vmatrix} = \vec{c}$$

$$1$$

Outer product

$$\begin{pmatrix} a_i \\ a_j \\ a_k \end{pmatrix} \begin{pmatrix} b_i & b_j & b_k \end{pmatrix} = \begin{pmatrix} a_i b_i & a_i b_j & a_i b_k \\ a_j b_i & a_j b_j & a_j b_k \\ a_k b_i & a_k b_j & a_k b_k \end{pmatrix}$$

$$0 \bigoplus 1 \bigoplus 2$$

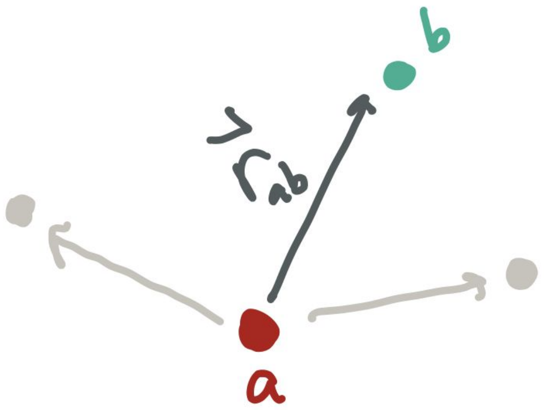
Features and kernels are not simply scalars.
We use tensor products and Clebsch-Gordan coefficients to combine.

Scalar convolution

Scalar multiply

$$\sum_b W_{ij}(\vec{r}_{ab}) \times I_{bi} = O_{aj}$$

"i" and "j" are scalar channels



Features and kernels are not simply scalars.
We use tensor products and Clebsch-Gordan coefficients to combine.

Scalar convolution

Scalar multiply

$$\sum_b W_{ij}(\vec{r}_{ab}) \times I_{bi} = O_{aj}$$

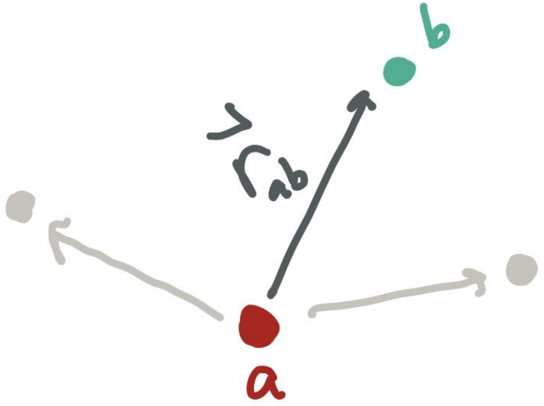
"i" and "j" are scalar channels

Tensor convolution

Tensor product + Clebsch-Gordan: Takes ij -> k

$$\sum_b W_{j}(\vec{r}_{ab}) \otimes I_{bi} = O_{ak}$$

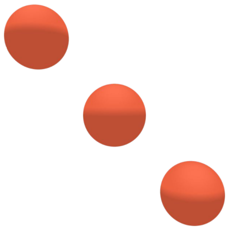
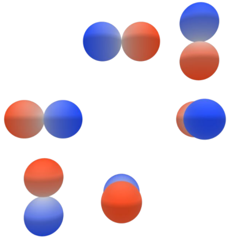
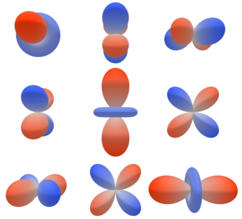
"channel" includes specific spherical harmonic L



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

$$\boxed{R(r)}\,Y_l^m(\hat{r})C_{lij} = K_{ij}$$

L=0 L=1 L=2

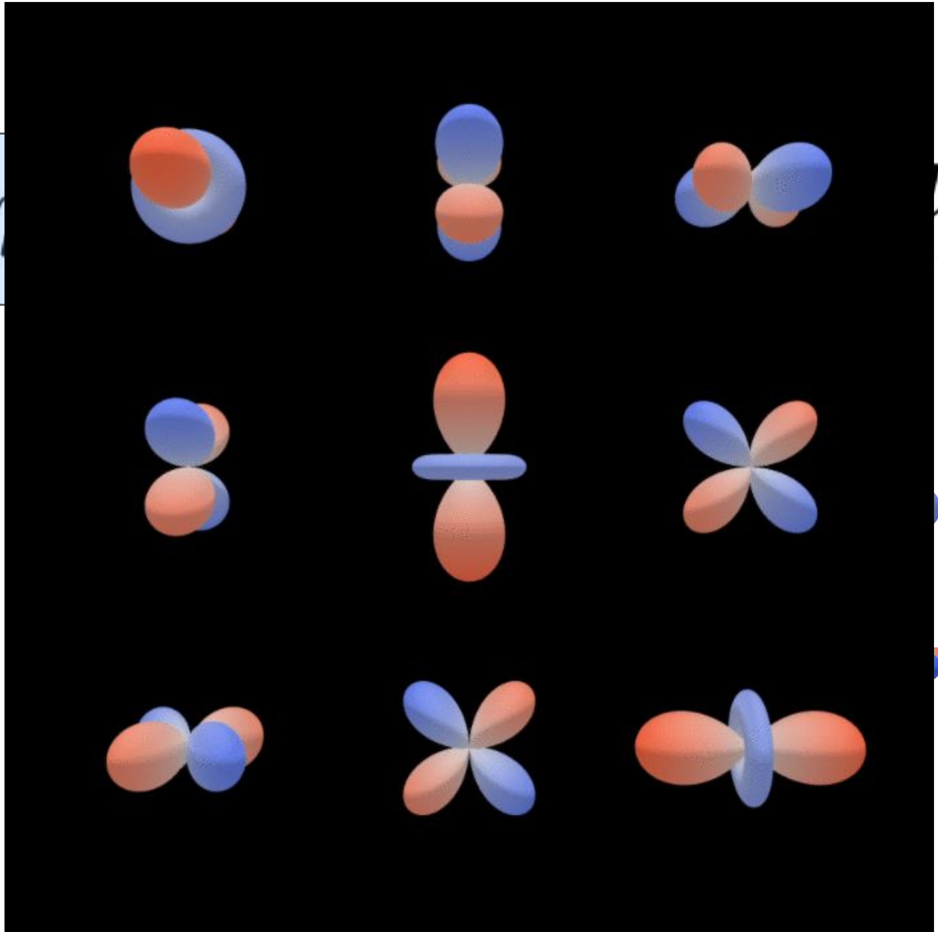
$R_0(r)$  $+R_1(r)$  $+R_2(r)$  $= K_{ij}$

(3 x 3) x 3 ➪ 3

(3 x 3) = 1 + 3 + 5

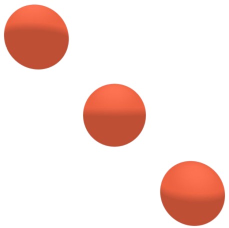
For L=1 ⇨ L=1, the filters will be a learned, radially-dependent linear combinations of the L = 0, 1, and 2 spherical harmonics.

$R($ K_{ij}

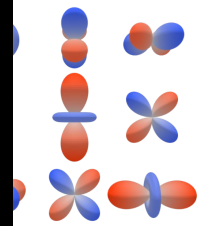


L=0

$R_0(r)$



L=2



$= K_{ij}$

(3 x 3) x 3 ⇨ 3
(3 x 3) = 1 + 3 + 5

We can interpret our outputs as numerical features...

Scalars
- Energy
- Mass
- Isotropic *

...

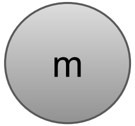
Vectors
- Force
- Velocity
- Acceleration
- Polarization

...

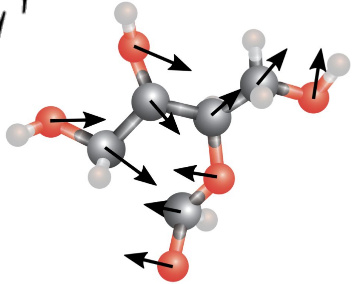
Matrices, Tensors, ...
- Moment of Inertia
- Polarizability
- Interaction of multipoles

...

$$E$$

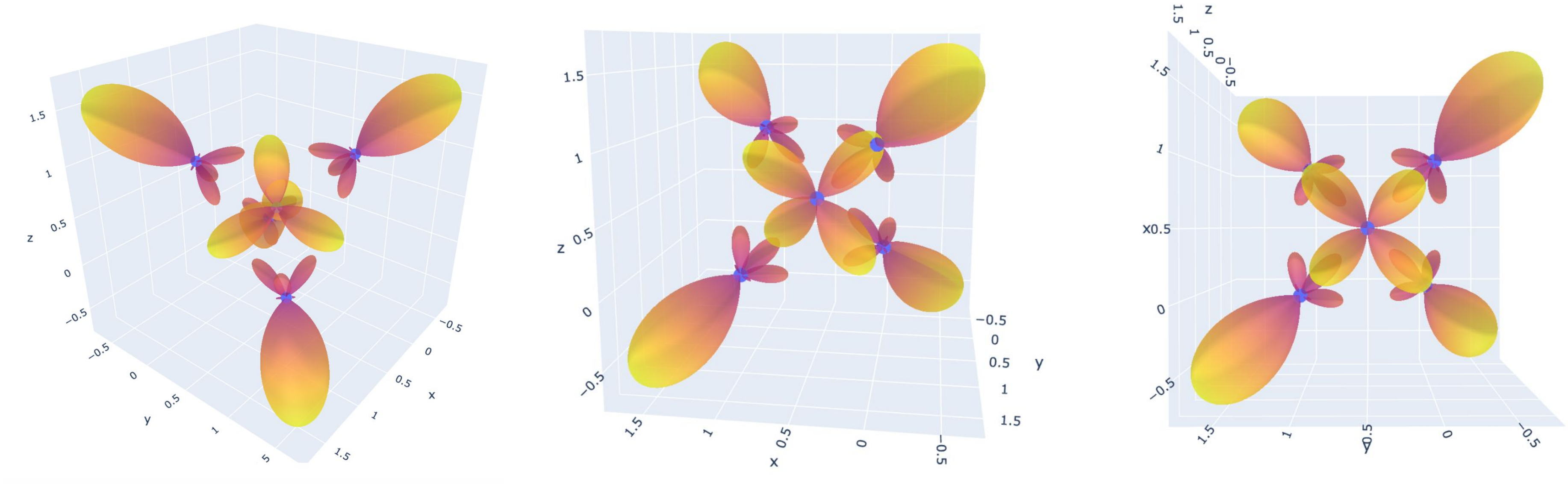


$$\vec{F}$$



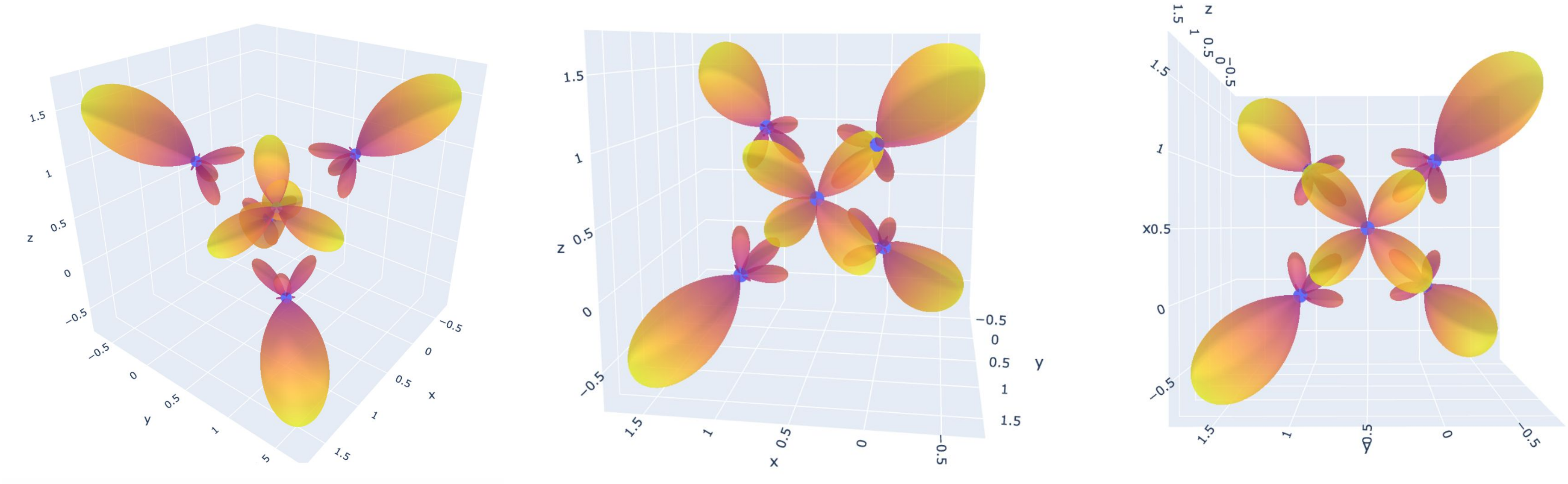
$$\alpha = \begin{bmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{bmatrix}$$

We can interpret our outputs as numerical features ***<u>or geometry.</u>***



Output (0 ≤ L< 6 coefficients) of randomly initialized network applied to a tetrahedron with a center.

We can interpret our outputs as numerical features ***or geometry.***



Output (0 ≤ L< 6 coefficients) of randomly initialized network applied to a tetrahedron with a center.

Can generate point sets from signal peaks as output!
Sets == permutation invariant

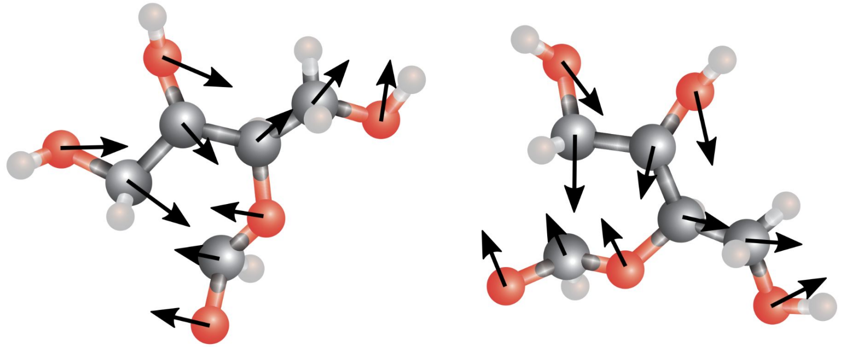
We build neural networks with Euclidean symmetry, E(3) and SE(3).

- What neural networks with Euclidean symmetry can do.
- How Euclidean Neural Networks work.
- **Applications of Euclidean Neural Networks.**

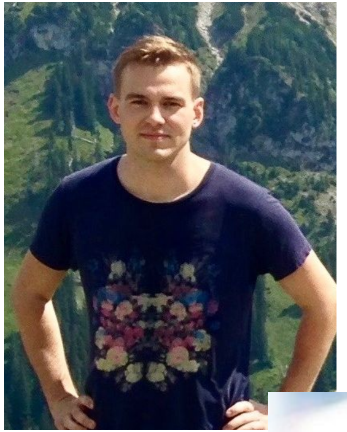
Applications: Predicting ab initio forces for molecular dynamics

Simon Batzner (MIT/Harvard) and Boris Kozinsky (Harvard)

Presented at APS March Meeting 2019

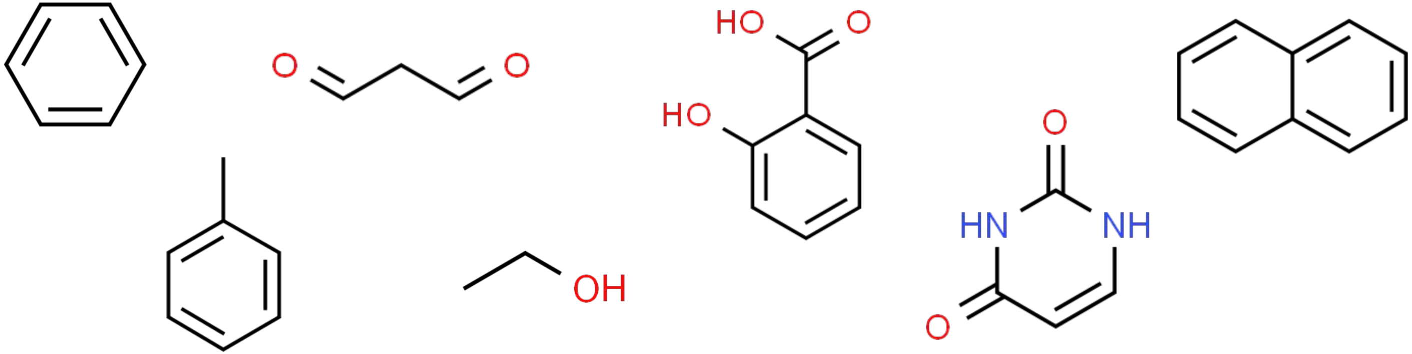


Direct prediction of forces rather than gradient of scalar energy.



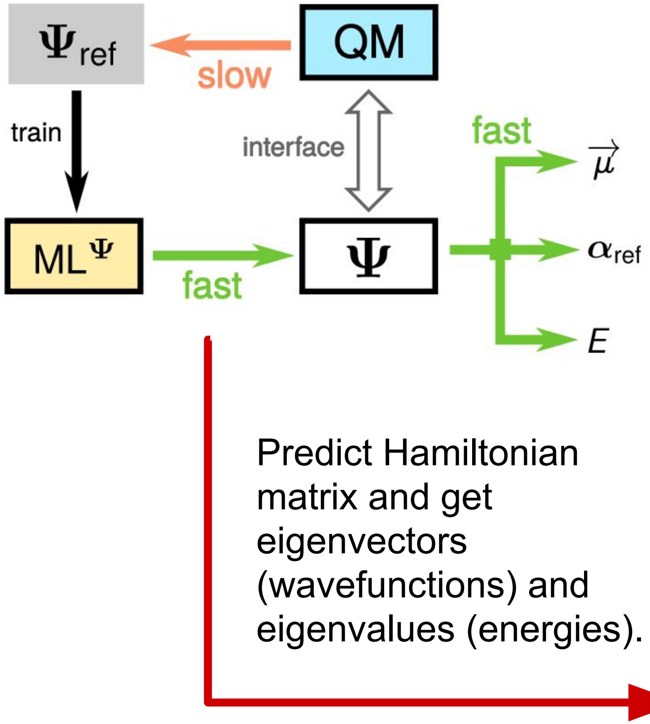
Dataset: **MD17**

ab initio molecular dynamics trajectories of...

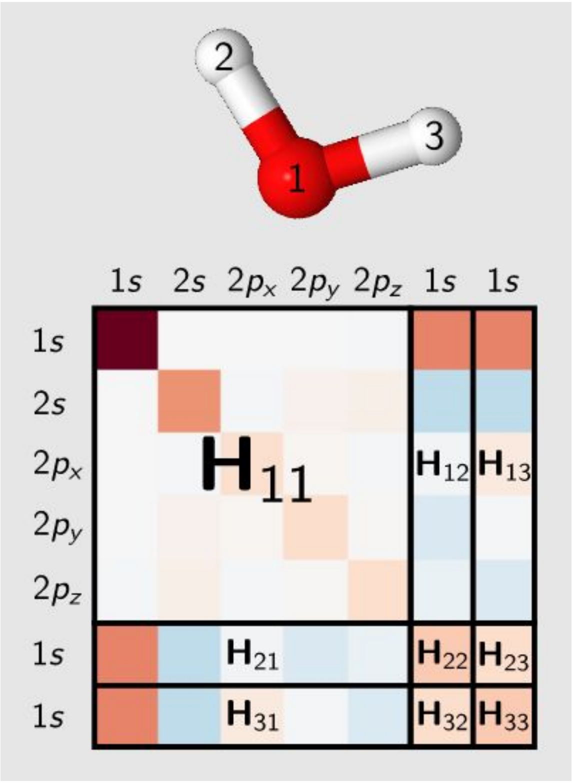


Applications: Predicting molecular Hamiltonians with atom-centered basis sets

K. T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller, R. J. Maurer. arXiv:1906.10033 (2019)

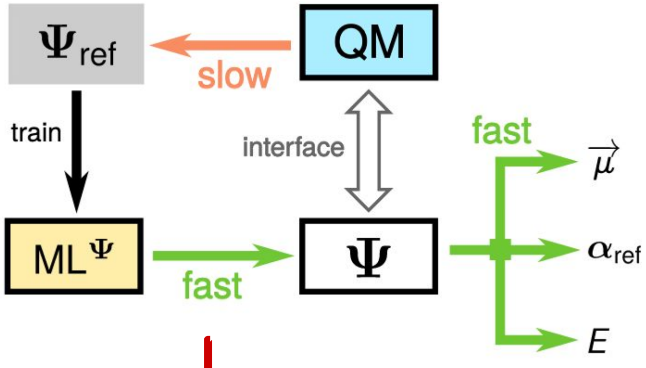


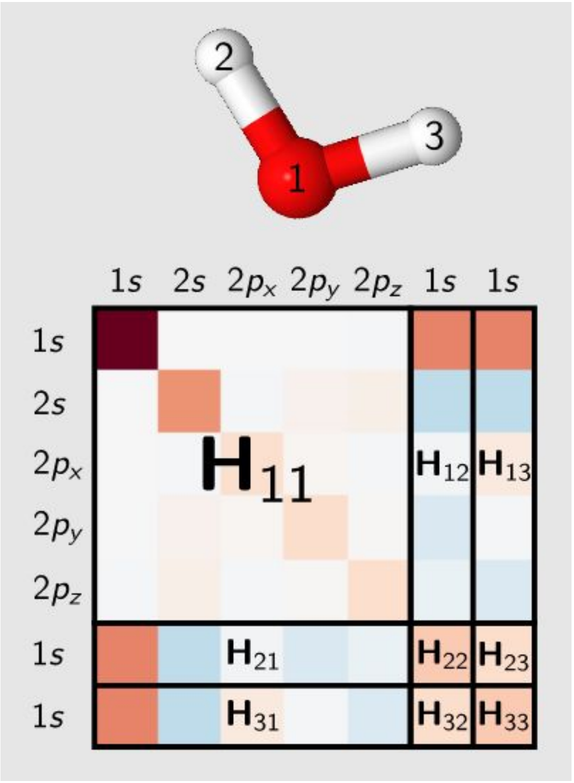
Predict Hamiltonian matrix and get eigenvectors (wavefunctions) and eigenvalues (energies).



Applications: Predicting molecular Hamiltonians with atom-centered basis sets

K. T. Schütt, M. Gastegger, A. Tkatchenko, K.-R. Müller, R. J. Maurer. arXiv:1906.10033 (2019)





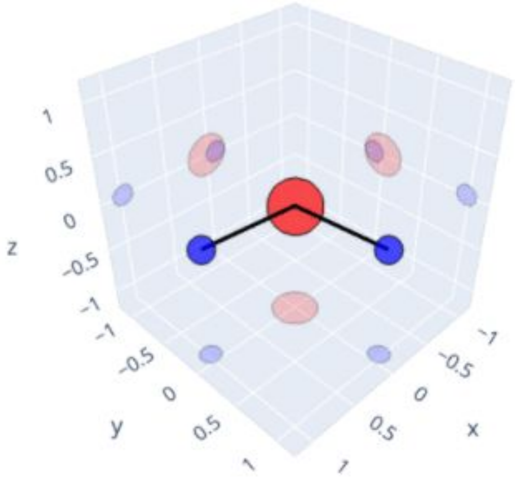
Predict Hamiltonian matrix and get eigenvectors (wavefunctions) and eigenvalues (energies).

Problem! Hamiltonian depends on coordinate system -- traditionally requires augmenting data.

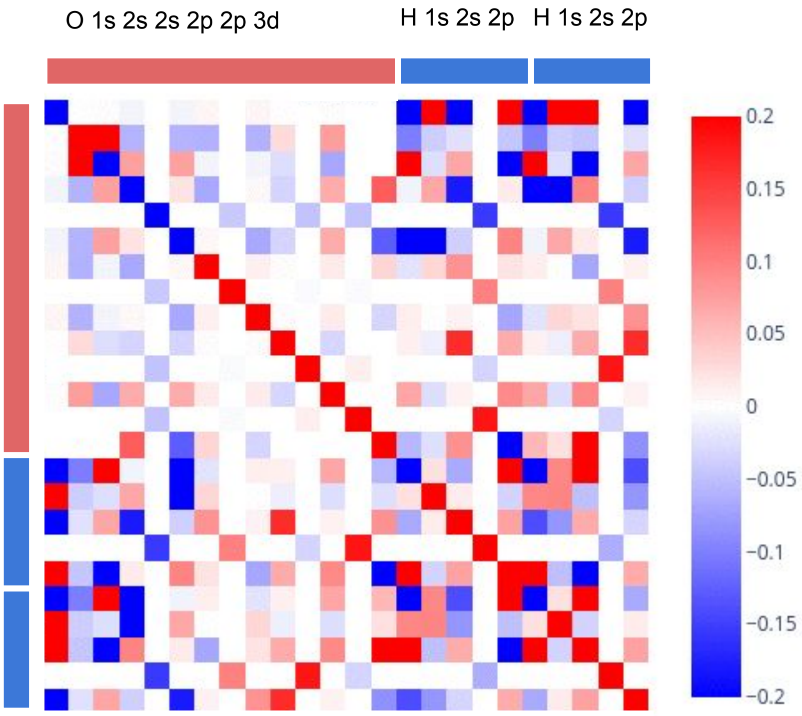
Applications: Predicting molecular Hamiltonians with atom-centered basis sets

With Euclidean neural networks -- only need one example.
Output is guaranteed to be equivariant!

$$R\,\overrightarrow{v}$$

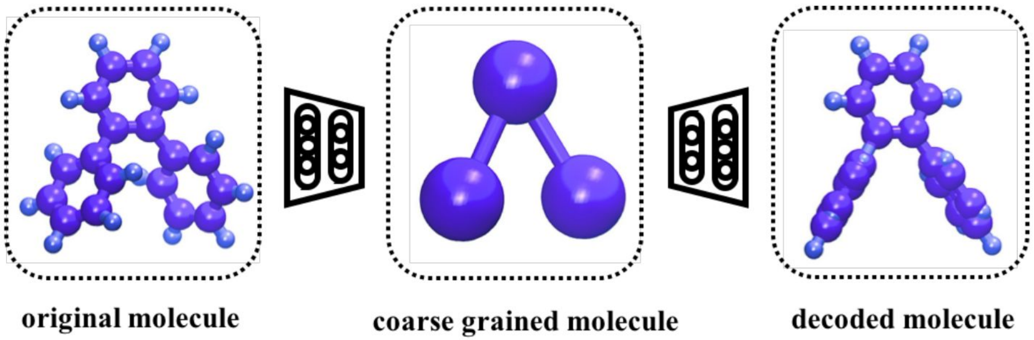


$$R\,H\,R^T$$



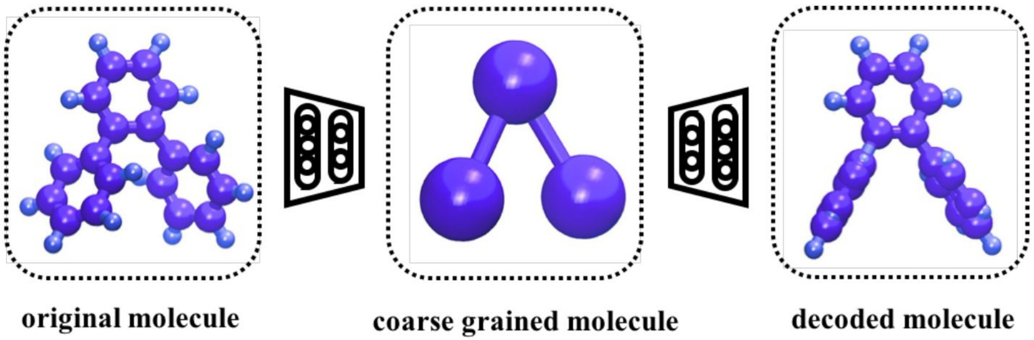
Applications:
Coarse-grained geometries and recover all atoms picture

W. Wang, R. Gómez-Bombarelli. arXiv:1812.02706 (revised 2019)

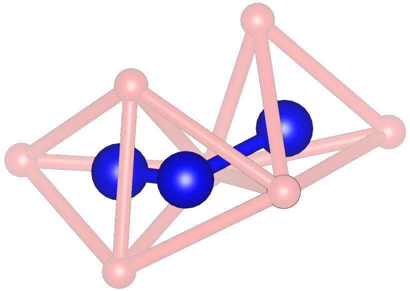


Applications:
Coarse-grained geometries and recover all atoms picture

W. Wang, R. Gómez-Bombarelli. arXiv:1812.02706 (revised 2019)

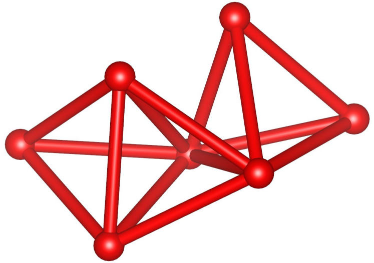


Test problem



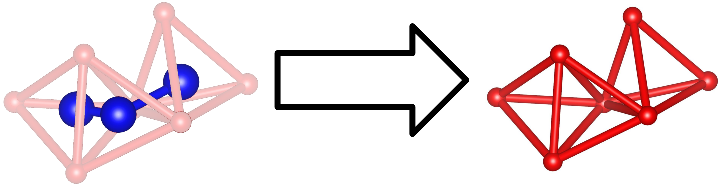
Centers of a tetrahedral chain.

Predict using spherical harmonic signal



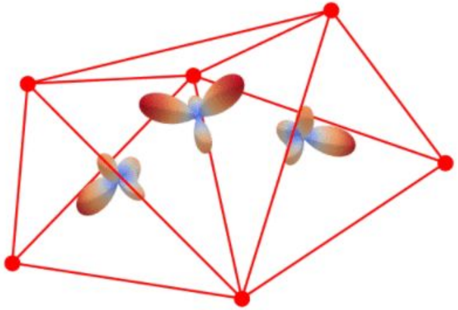
Tetrahedral chain.

Applications:
Coarse-grained geometries and recover all atoms picture



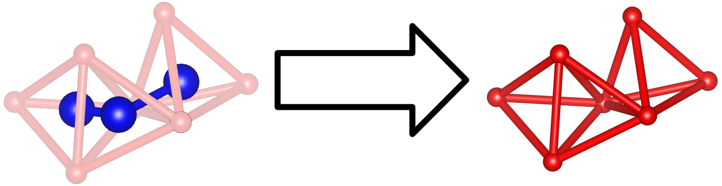
Peaks == point locations

Symmetric configurations == degeneracy



Without symmetry breaking

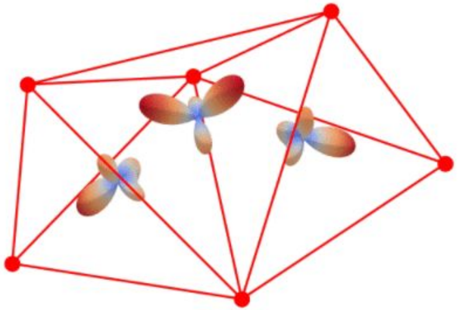
Applications:
Coarse-grained geometries and recover all atoms picture



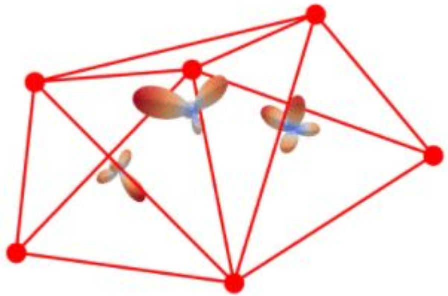
Peaks == point locations

Symmetric configurations == degeneracy

Use second input to break symmetry

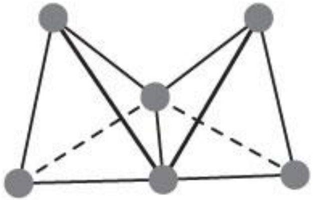


Without symmetry breaking



With symmetry breaking

Applications: Creating an autoencoder for discrete geometry



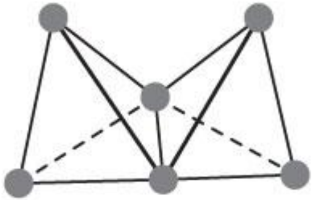
Discrete geometry

Reduce geometry to single point.

$\mathbb{R}^N$

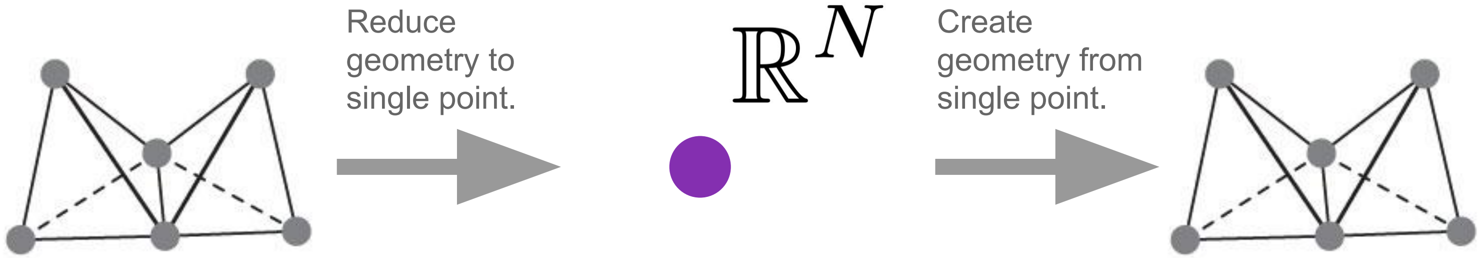
Continuous
Latent Representation
(N dimensional vector)

Create geometry from single point.



Discrete geometry

Applications: Creating an autoencoder for discrete geometry



Discrete geometry

Continuous
Latent Representation
(N dimensional vector)

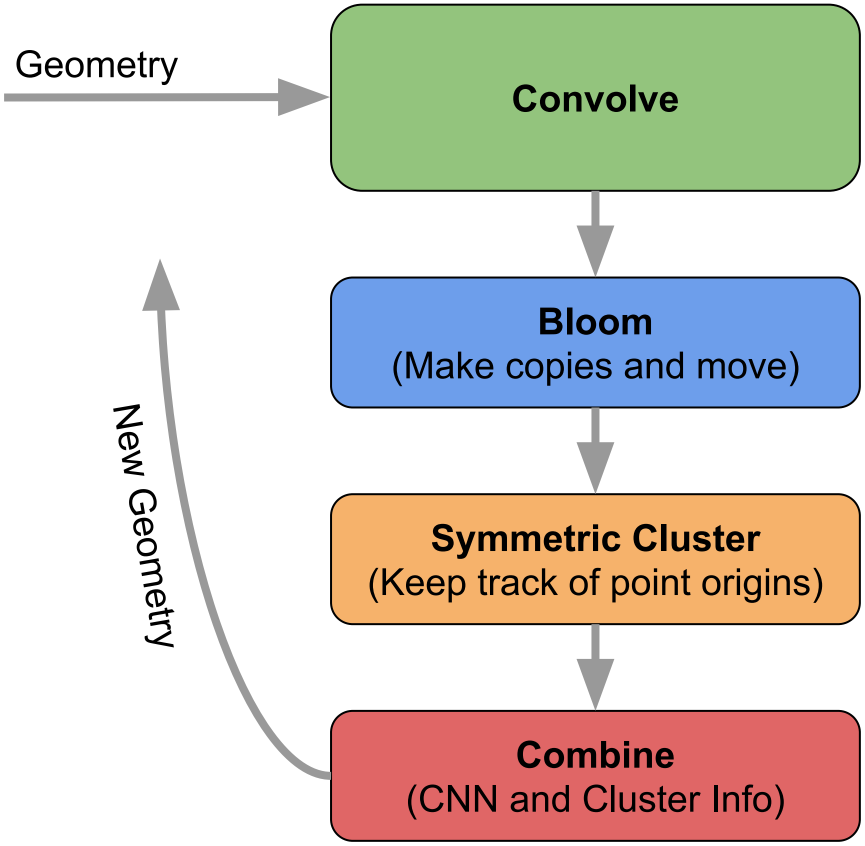
Discrete geometry

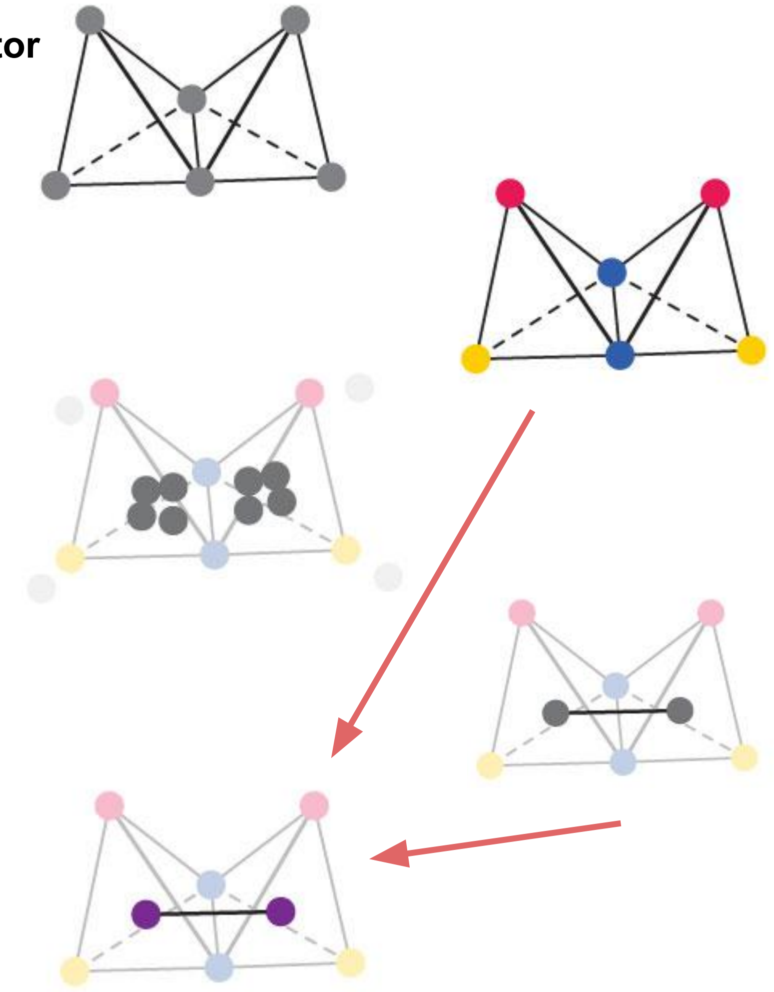
Atomic structures are hierarchical and can be constructed from geometric motifs.

+ Encode geometry ✓
+ Encode hierarchy ?
+ Decode geometry ?
+ Decode hierarchy ?

(Need to do this in a recursive manner)

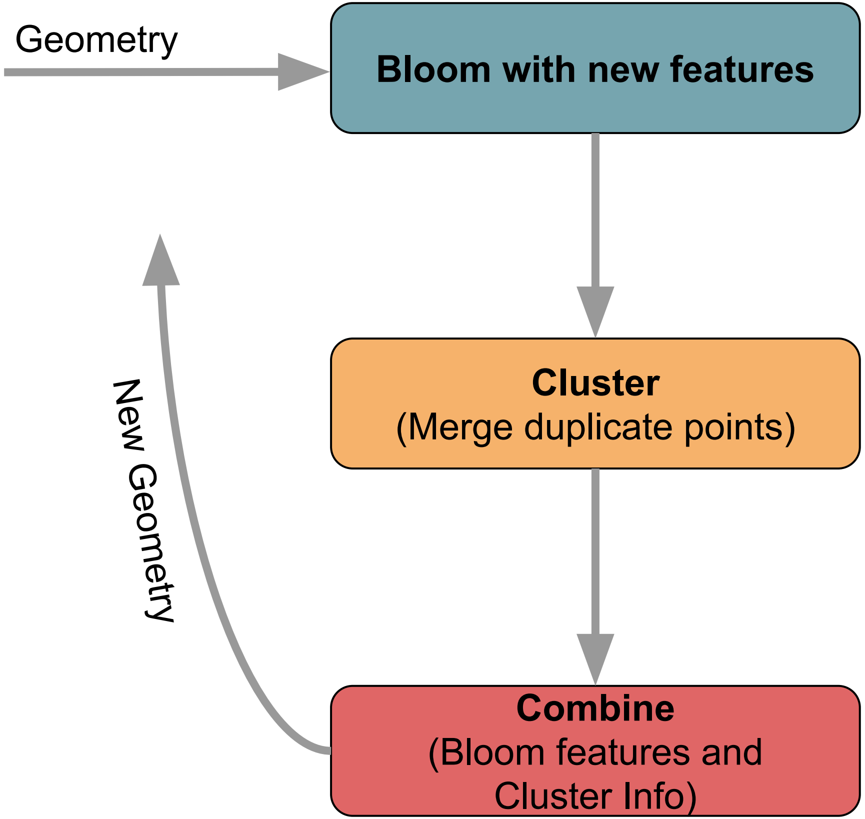
How to encode: Recursively convert geometry to a vector

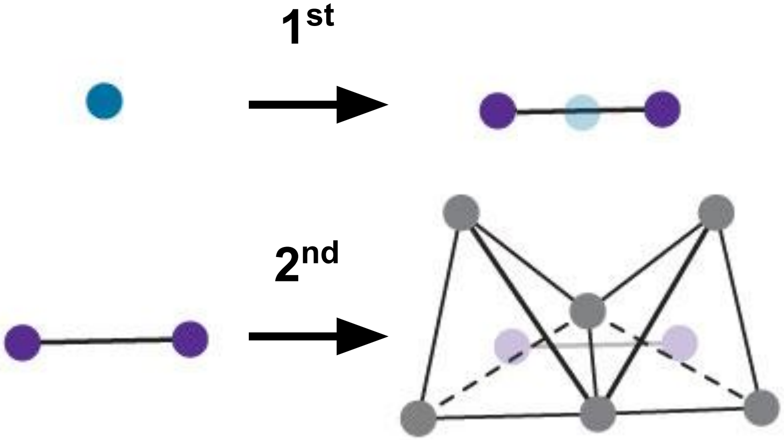




**Edges are shown for visualization. May not be included.*

How to decode: Recursively convert a vector to geometry





Edges are shown for visualization. May not be included.

atomic architects
summer 2019

Tess Smidt

Hashim Piracha

Mario Geiger

Ben Miller

tensor field networks

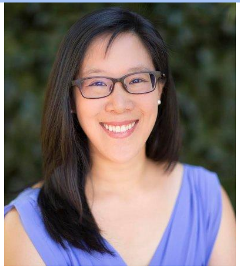
Stanford

Nate Thomas

Google Accelerated Science Team

Patrick Riley

Steve Kearnes



Lusann Yang

Li Li

Kai Kohlhoff

- ***Euclidean neural networks*** operate on points/voxels and have symmetries of E(3).
- Inputs to the network lower this symmetry to a subgroup of E(3).
- Symmetry of outputs are constrained to the symmetry of the inputs.
- The inputs and outputs of our network are geometry and geometric tensors.
- Convolutional filters are built from spherical harmonics with a learned radial function.

Applications: Molecular dynamics, predicting Hamiltonians, coarse-graining, autoencoders...

We expect these networks to be generally useful for physics, chemistry, and geometry.
Reach out to me if you are interested and/or have any questions!

`se3cnn` Code (PyTorch):
https://github.com/mariogeiger/se3cnn

Tensor Field Networks (arXiv:1802.08219)
3D Steerable CNNs (arXiv:1807.02547)

Tess Smidt
tsmidt@lbl.gov

Calling in backup (slides)!

Features and kernels are not simply scalars.
We use tensor products and Clebsch-Gordan coefficients to combine.

Clebsch-Gordan coeffs.

$$S_i \otimes S_j = C_{ijk} S_{ij} = S_k$$

$$|i, i_m\rangle |j, j_m\rangle \langle k, k_m|$$

non-zero when $-|i-j| \leq k \leq i+j$

Features and kernels are not simply scalars.
We use <u>tensor products</u> and <u>Clebsch-Gordan coefficients</u> to combine.

$\frac{1}{2}$-Spin eigenstates

$|\uparrow\rangle$ and $|\downarrow\rangle$

$|\frac{1}{2},\frac{1}{2}\rangle$ $\quad$ $|\frac{1}{2},-\frac{1}{2}\rangle$

2 $\frac{1}{2}$-Spin eigenstates

$|0,0\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle\right)$ singlet

$|1,1\rangle = |\uparrow\uparrow\rangle$

$|1,0\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle\right)$ triplet

$|1,-1\rangle = |\downarrow\downarrow\rangle$

$$\frac{1}{2}\otimes\frac{1}{2} \rightarrow S_i \otimes S_j = C_{ijk} S_k \rightarrow 0 \oplus 1$$

$$\ldots,\ \underbrace{|\tfrac{1}{2},-\tfrac{1}{2}\rangle}_{i}\ \underbrace{|\tfrac{1}{2},\tfrac{1}{2}\rangle}_{j}\ \underbrace{\langle 0,0|}_{k},\ \ldots$$

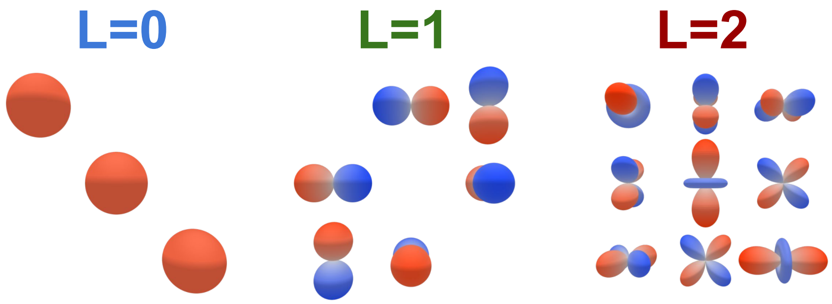
Features and kernels are not simply scalars.
We use <u>tensor products</u> and <u>Clebsch-Gordan coefficients</u> to combine.

$$M_{ij} = \begin{bmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{bmatrix}$$

$$M_{ij} = C_{ijk} S_k$$

$$\ldots, \underset{z}{|1,0\rangle} \underset{z}{|1,0\rangle} \underset{2z^2-x^2-y^2}{\langle 2,0|}, \ldots$$

$$1 \otimes 1 = 0 \oplus 1 \oplus 2$$



$$a_{x^2} = \frac{1}{\sqrt{3}} Y_{0,0} - \frac{1}{\sqrt{30}} Y_{2,0} + \frac{1}{\sqrt{10}} Y_{2,2}$$

$$a_{y^2} = \frac{1}{\sqrt{3}} Y_{0,0} - \frac{1}{\sqrt{30}} Y_{2,0} - \frac{1}{\sqrt{10}} Y_{2,2}$$

$$a_{z^2} = \frac{1}{\sqrt{3}} Y_{0,0} + \frac{2}{\sqrt{30}} Y_{2,0}$$

$$a_{xy/yx} = \frac{1}{\sqrt{10}} Y_{2,-2} \pm \frac{1}{\sqrt{6}} Y_{1,0}$$

$$a_{xz/zx} = \frac{1}{\sqrt{10}} Y_{2,1} \pm \frac{1}{\sqrt{6}} Y_{1,-1}$$

$$a_{yz/zy} = \frac{1}{\sqrt{10}} Y_{2,-1} \pm \frac{1}{\sqrt{6}} Y_{1,1}$$