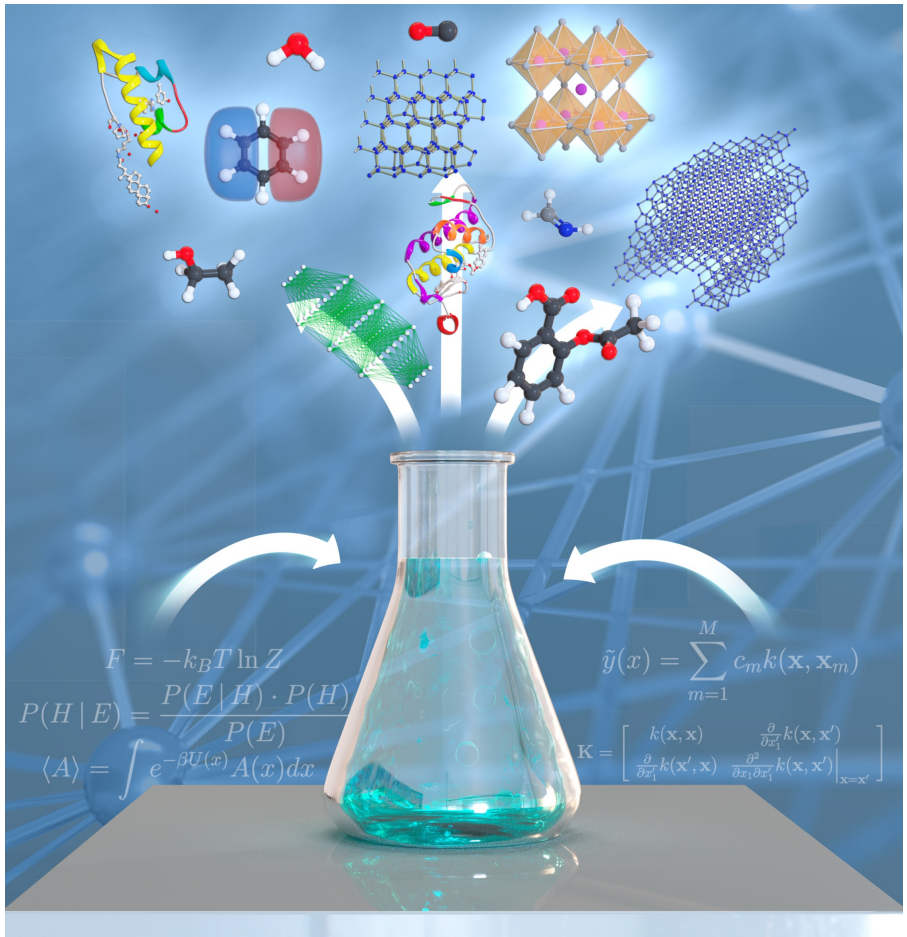
Machine Learning for Molecular Physics

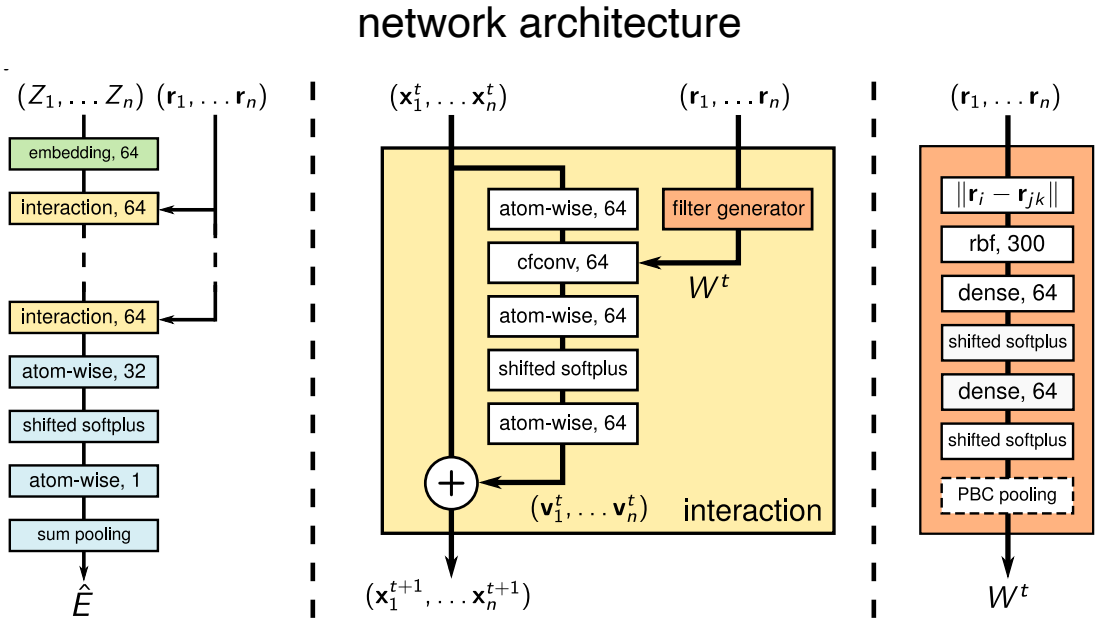
Cecilia Clementi

Freie Universität Berlin

cecilia.clementi@fu-berlin.de



SchNet



- Atoms are described by features $X^l = (\mathbf{x}_1^l, \ldots, \mathbf{x}_n^l), \quad \mathbf{x}_i^l \in \mathbb{R}^F$
- Atoms interact with surrounding through continuous filter convolutions $\mathbf{x}_i^{l+1} = (X^l * W^l)_i = \sum_{j=0}^{n_{\text{atoms}}} \mathbf{x}_j^l \circ W^l(\mathbf{r}_j - \mathbf{r}_i),$
- The interatomic distances are expanded on a Gaussian basis $e_k(\mathbf{r}_j - \mathbf{r}_i) = \exp(-\gamma(\|\mathbf{r}_j - \mathbf{r}_i\| - \mu_k)^2)$
- Trained on combined loss of matching energy and forces

$$\ell((\hat{E}, \hat{\mathbf{F}}_1, \ldots, \hat{\mathbf{F}}_n)), (E, \mathbf{F}_1, \ldots, \mathbf{F}_n))$$
$$= \rho \left\| E - \hat{E} \right\|^2 + \frac{1}{n_{\text{atoms}}} \sum_{i=0}^{n_{\text{atoms}}} \left\| \mathbf{F}_i - \left(-\frac{\partial \hat{E}}{\partial \mathbf{R}_i} \right) \right\|^2,$$

Schütt, et al. *J. Chem. Phys.,* 148(24):241722 (2018)

SchNet: Results

Trained on 131 000 small organic molecules with up to 9 atoms (QM9)
Validated on 10 000 molecules

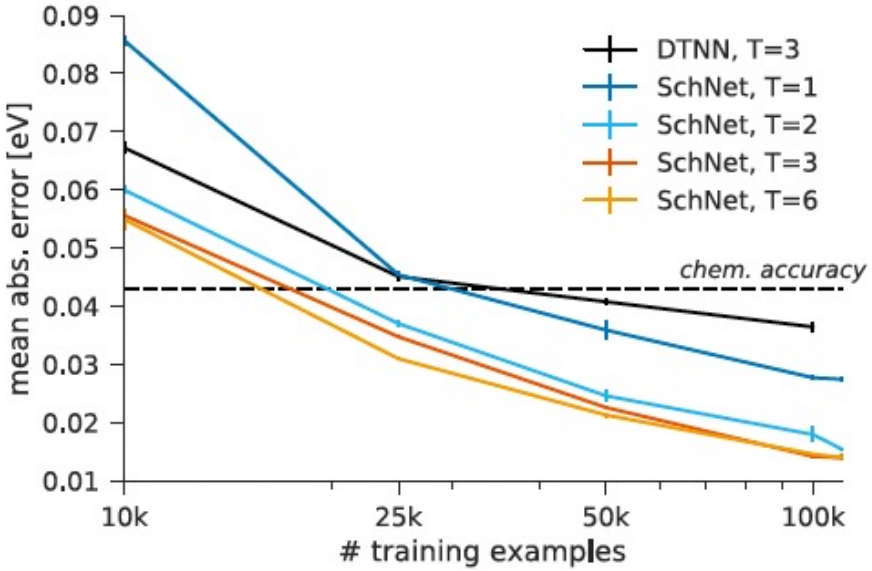


FIG. 3. Mean absolute error (in eV) of energy predictions (U_0) on the QM9 dataset[53–55] depending on the number of interaction blocks and reference calculations used for training. For reference, we give the best performing DTNN models (T = 3).[28]

TABLE I. Mean absolute errors for energy predictions on the QM9 dataset using 110k training examples. For SchNet, we give the average over three repetitions as well as standard errors of the mean of the repetitions. Best models in bold.

Property	Unit	SchNet ($T = 6$)	enn-s2s[29]
ϵ_{HOMO}	eV	**0.041 ± 0.001**	0.043
ϵ_{LUMO}	eV	**0.034 ± 0.000**	0.037
$\Delta\epsilon$	eV	**0.063 ± 0.000**	0.069
ZPVE	meV	1.7 ± 0.033	**1.5**
μ	D	0.033 ± 0.001	**0.030**
α	bohr^3	0.235 ± 0.061	**0.092**
$\langle R^2 \rangle$	bohr^2	**0.073 ± 0.002**	0.180
U_0	eV	**0.014 ± 0.001**	0.019
U	eV	**0.019 ± 0.006**	**0.019**
H	eV	**0.014 ± 0.001**	0.017
G	eV	**0.014 ± 0.000**	0.019
C_v	cal/mol K	**0.033 ± 0.000**	0.040

Schütt, et al. *J. Chem. Phys.,* 148(24):241722 (2018)

SchNet: Results

Learned embeddings reflect atomic properties

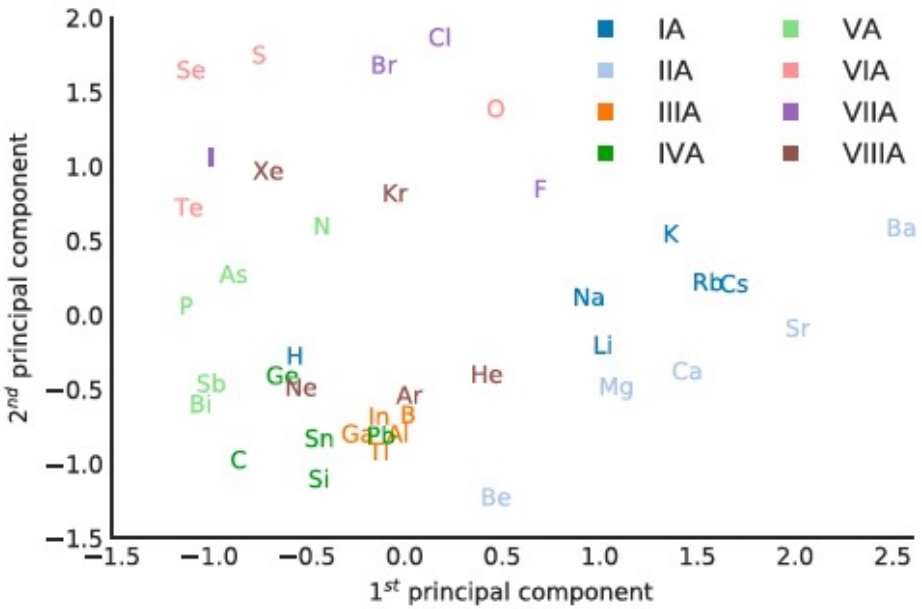


FIG. 4. The two leading principal components of the learned embeddings $\mathbf{x}^0$ of sp atoms learned by SchNet from the Materials Project dataset. We recognize a structure in the embedding space according to the groups of the periodic table (color-coded) as well as an ordering from lighter to heavier elements within the groups, e.g., in groups IA and IIA from light atoms (left) to heavier atoms (right).

Schütt, et al. *J. Chem. Phys.,* 148(24):241722 (2018)

SchNet: Results

Learned local chemical potentials (energy of a probe atom)

$$\mathbf{x}_{\text{probe}}^{l+1} = (X^l * W^l)_i = \sum_{i=0}^{n_{\text{atoms}}} \mathbf{x}_i^l \circ W^l(\mathbf{r}_{\text{probe}} - \mathbf{r}_i)$$

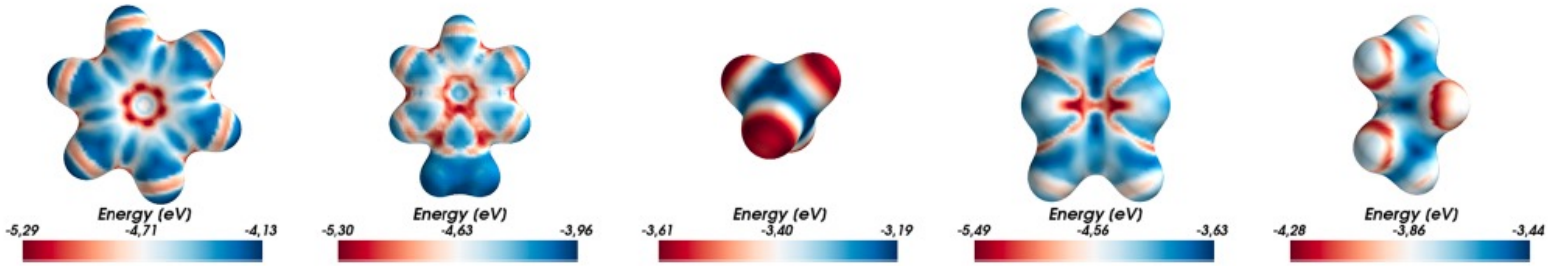


FIG. 5. Local chemical potentials $\Omega_C(\mathbf{r})$ of DTNN (top) and SchNet (bottom) using a carbon test charge on a $\sum_i \|\mathbf{r} - \mathbf{r}_i\| = 3.7$ Å isosurface are shown for benzene, toluene, methane, pyrazine, and propane.

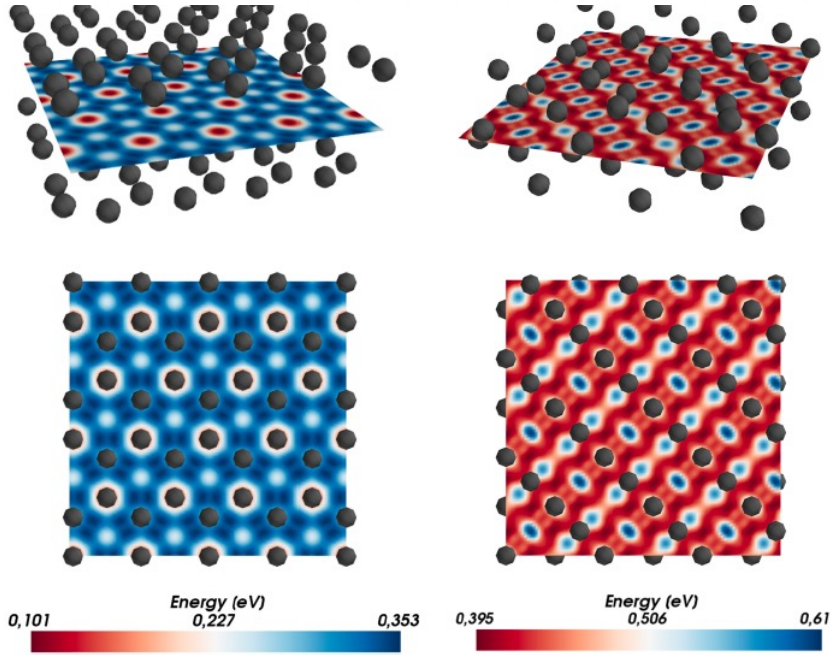


FIG. 6. Cuts through local chemical potentials $\Omega_C(\mathbf{r})$ of SchNet using a carbon test charge are shown for graphite (left) and diamond (right).

Schütt, et al. *J. Chem. Phys.,* 148(24):241722 (2018)

SchNet: Results

Molecular dynamics of Fullerene C_{20}

"Overall, with SchNet we could carry out 1.25 ns of PIMD, reducing the runtime compared to DFT by 3-4 orders of magnitude: from about 7 years to less than 7 h with much less computational resources."

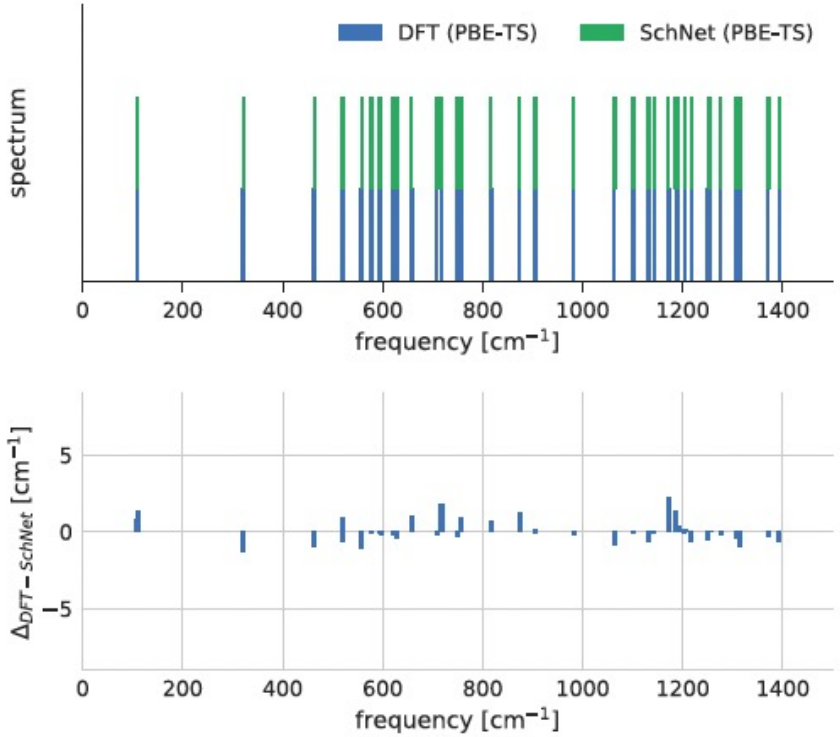


FIG. 7. Normal mode analysis of the fullerene C_{20} dynamics comparing SchNet and DFT results.

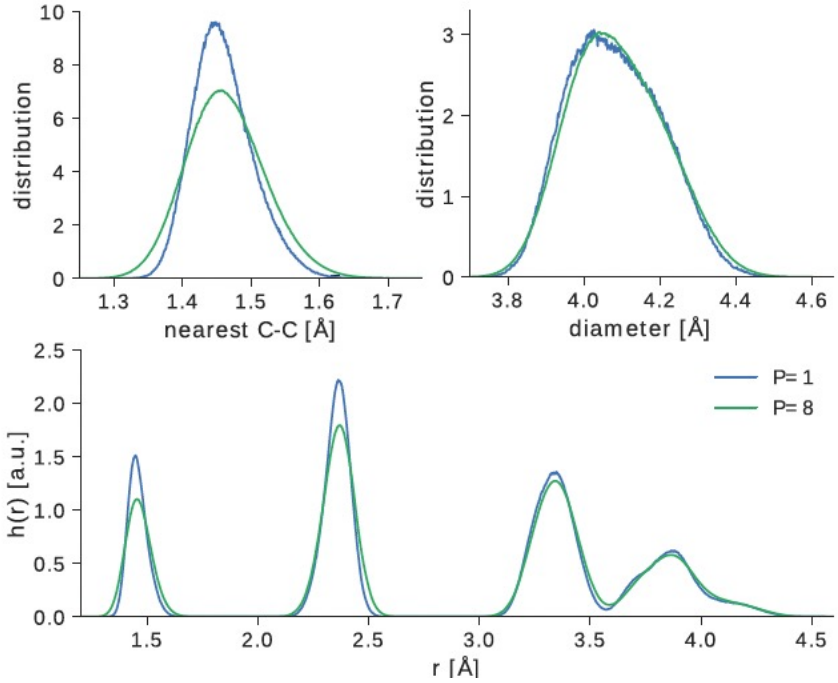


FIG. 8. Analysis of the fullerene C_{20} dynamics at 300 K using SchNet@DFT. Distribution functions for nearest neighbours, diameter of the fullerene, and the atomic-pair distribution function using classical MD (blue) and PIMD (green) with 8 beads.

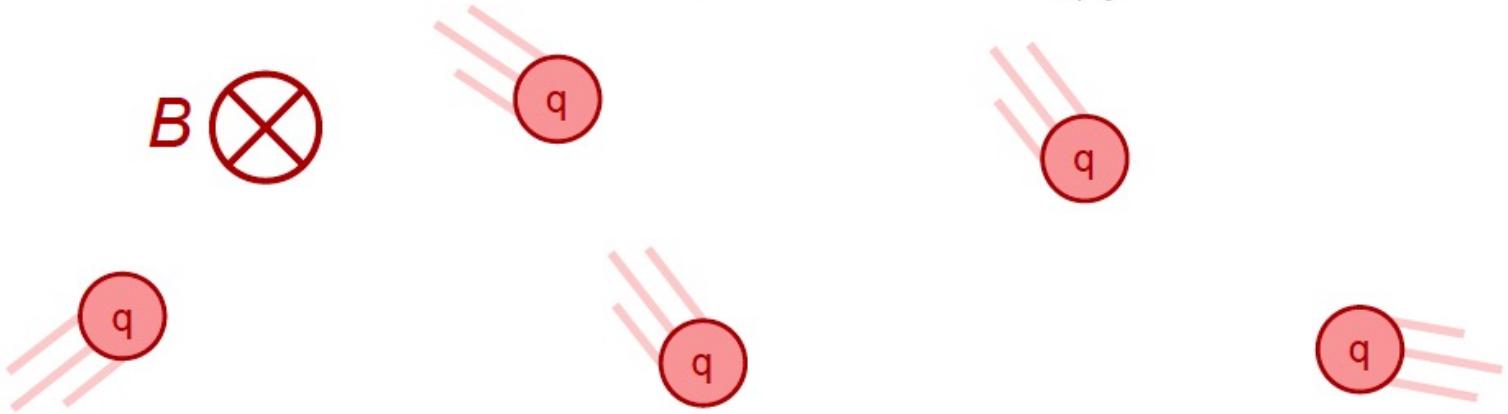
Schütt, et al. *J. Chem. Phys.,* 148(24):241722 (2018)

The laws of physics have rotational, translational, and (unless you're a particle physicist) **parity symmetry.**

We want machine learning models that also obey this symmetry.

e.g. a **network** is our model of **"physics"**. The **input** to the network is our **system**.

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



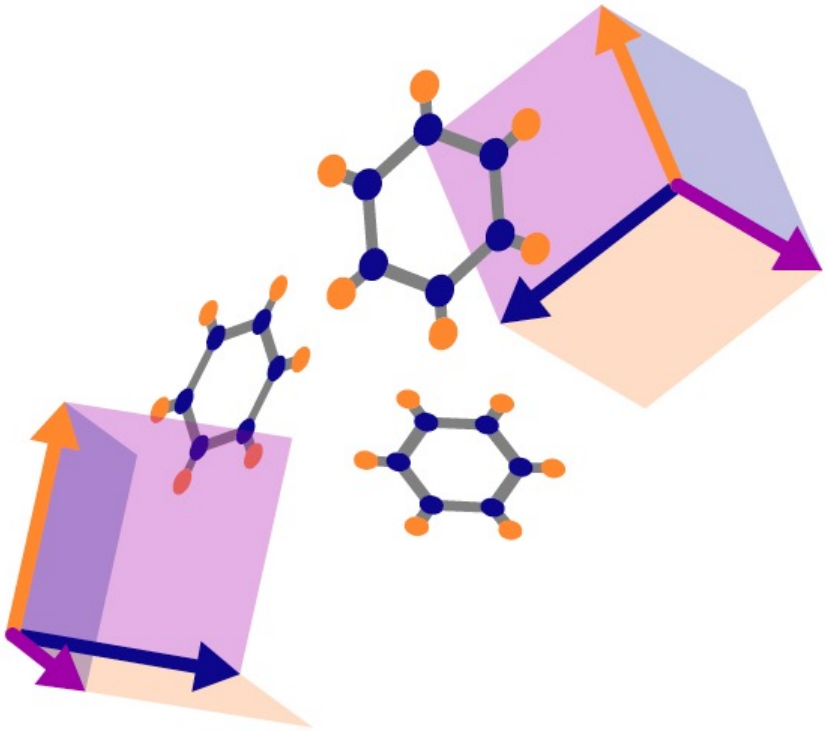
Symmetry emerges when different ways of representing something "mean" the same thing.

Symmetry of representation vs. objects

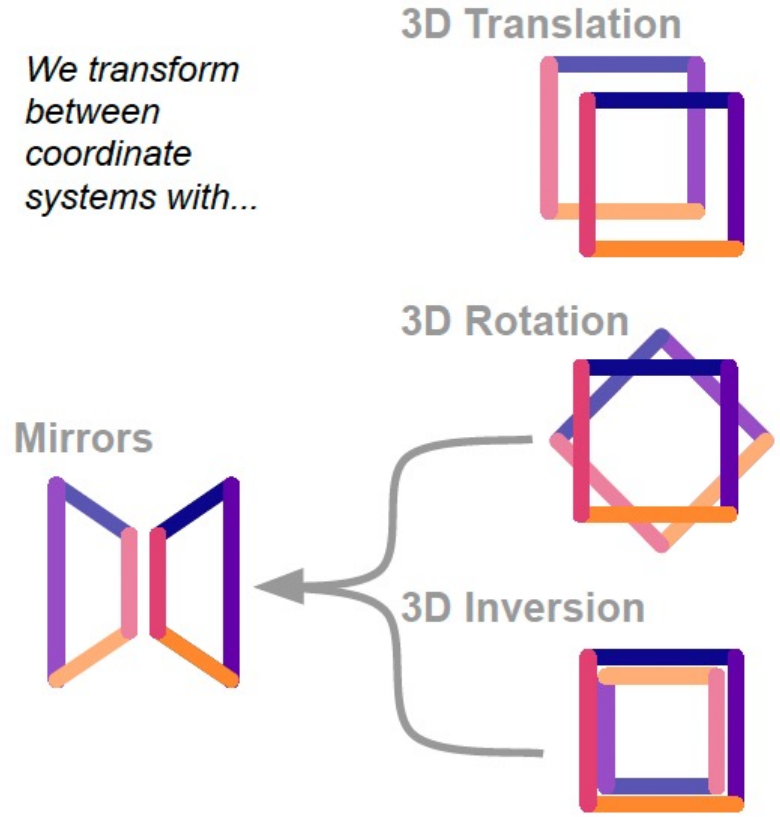
Euclidean symmetry, E(3):

Symmetry of 3D space

The freedom to choose your coordinate system



We transform between coordinate systems with...



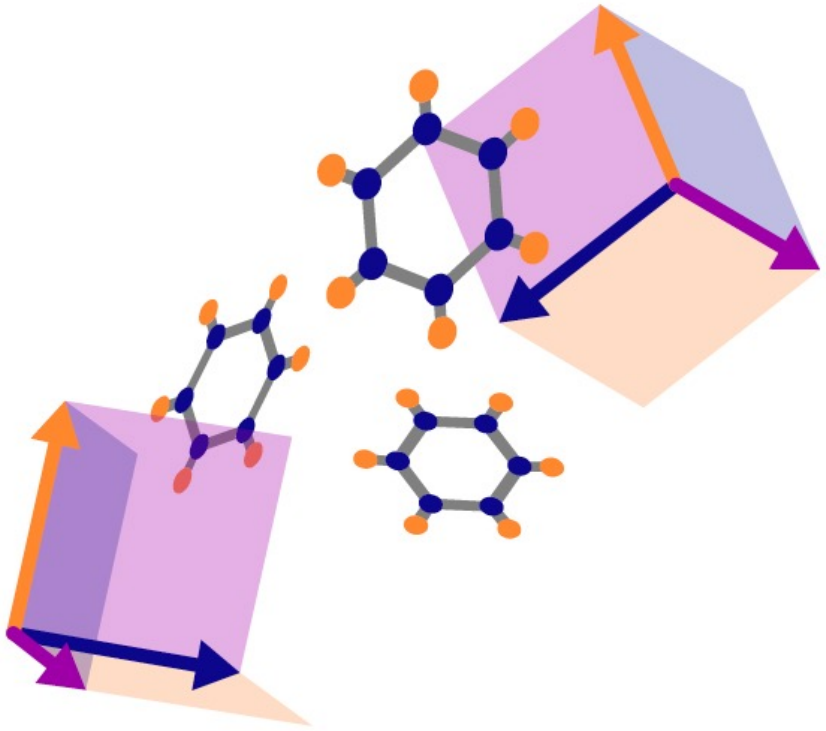
Symmetry emerges when different ways of representing something "mean" the same thing.

Symmetry of representation vs. objects

Euclidean symmetry, E(3):

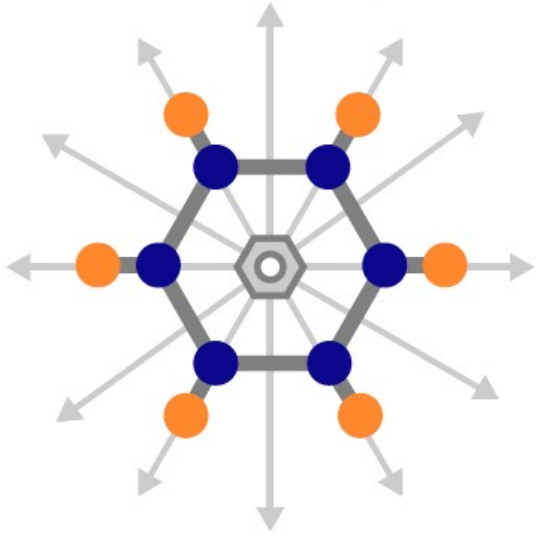
Symmetry of 3D space

The freedom to choose your coordinate system



Symmetry of geometric objects

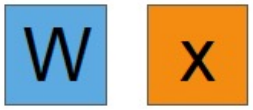
Looks the same under specific rotations, translations, and inversion (includes mirrors).



$$D_{6h} = 6/mmm$$

Equivariant Networks

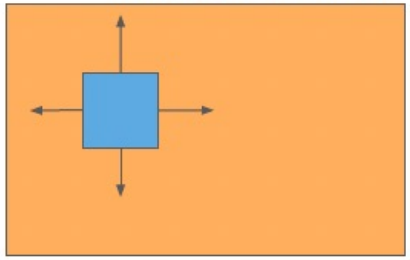
Neural networks are specially designed for different data types.
Assumptions about the data type are built into how the network operates.



Arrays ⇨ Dense NN

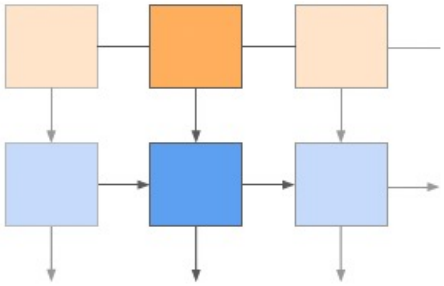
Components are independent.

2D images ⇨ Convolutional NN



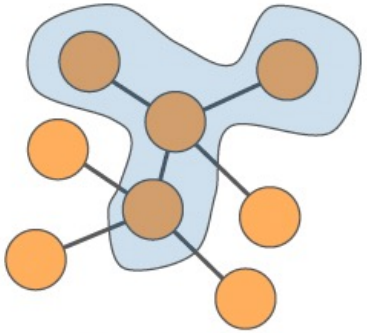
The same features can be found anywhere in an image. Locality.

Text ⇨ Recurrent NN



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

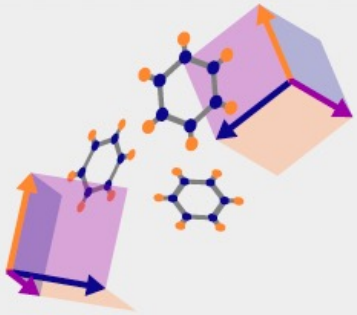
Graph ⇨ Graph (Conv.) NN



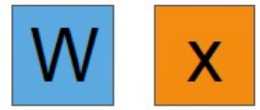
Topological data. Nodes have features and network passes messages between nodes connected via edges.

3D physical data ⇨ Euclidean NN

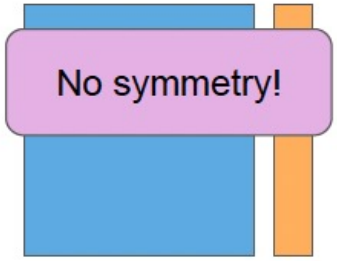
Data in 3D Euclidean space. Freedom to choose coordinate system.



Neural networks are specially designed for different data types.
Assumptions about the data type are built into how the network operates.
Thus, symmetries are encoded by tailoring network operations.

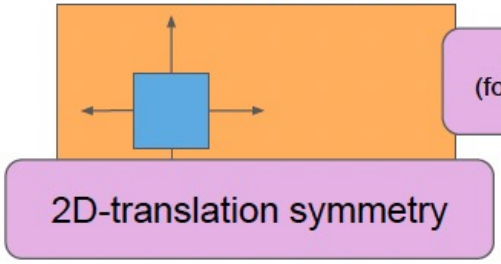


Arrays ⇨ Dense NN



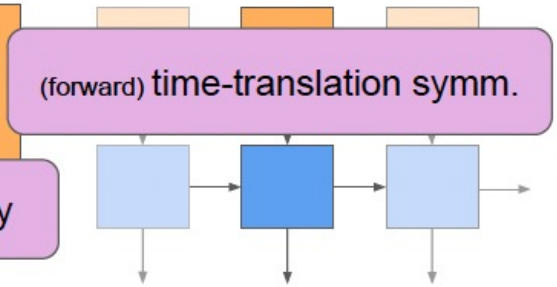
Components are independent.

2D images ⇨ Convolutional NN



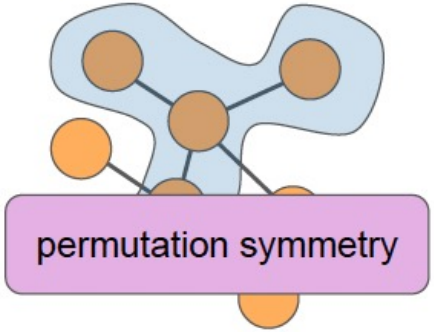
The same features can be found anywhere in an image. Locality.

Text ⇨ Recurrent NN



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

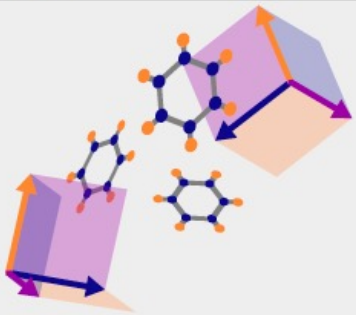
Graph ⇨ Graph (Conv.) NN



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

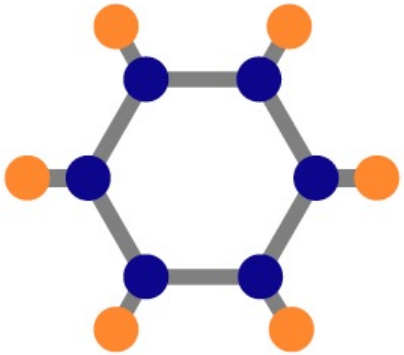
3D physical data ⇨ Euclidean NN

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



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

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



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

Coordinates are most general, but sensitive to translations, rotations, and inversion.

Approach 1:

Data Augmentation

Throw data at the problem and see what you get!

Approach 2:

Invariant Inputs

Convert your data to invariant representations so the neural network can't possibly mess it up.

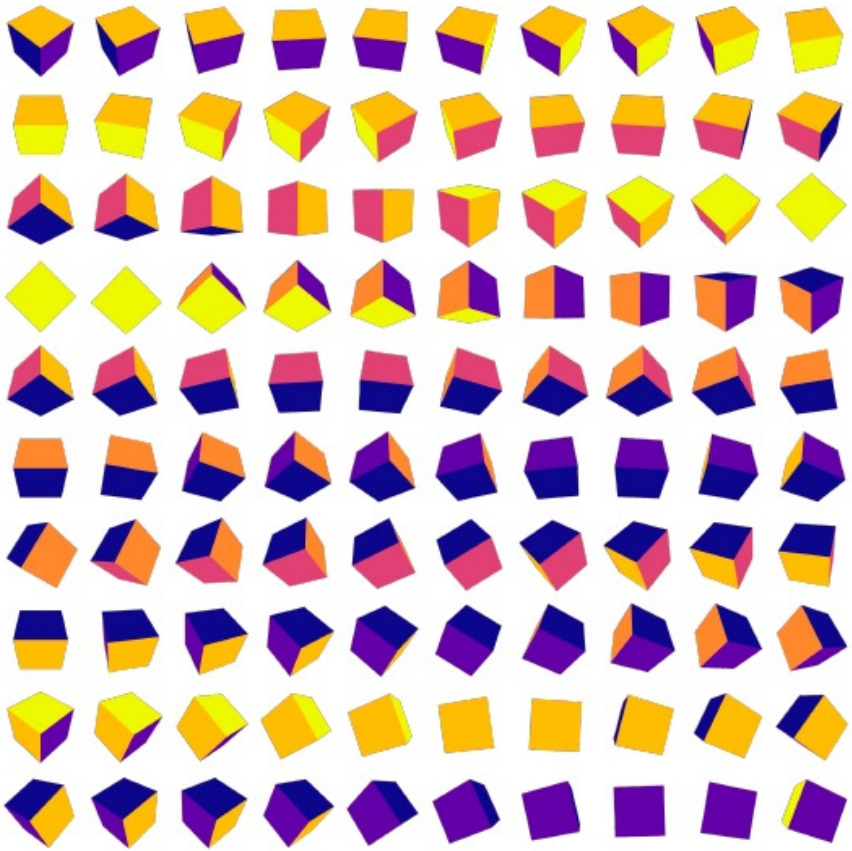
Approach 3:

Invariant models
Equivariant models

If there's no model that naturally handles coordinates, we will make one.

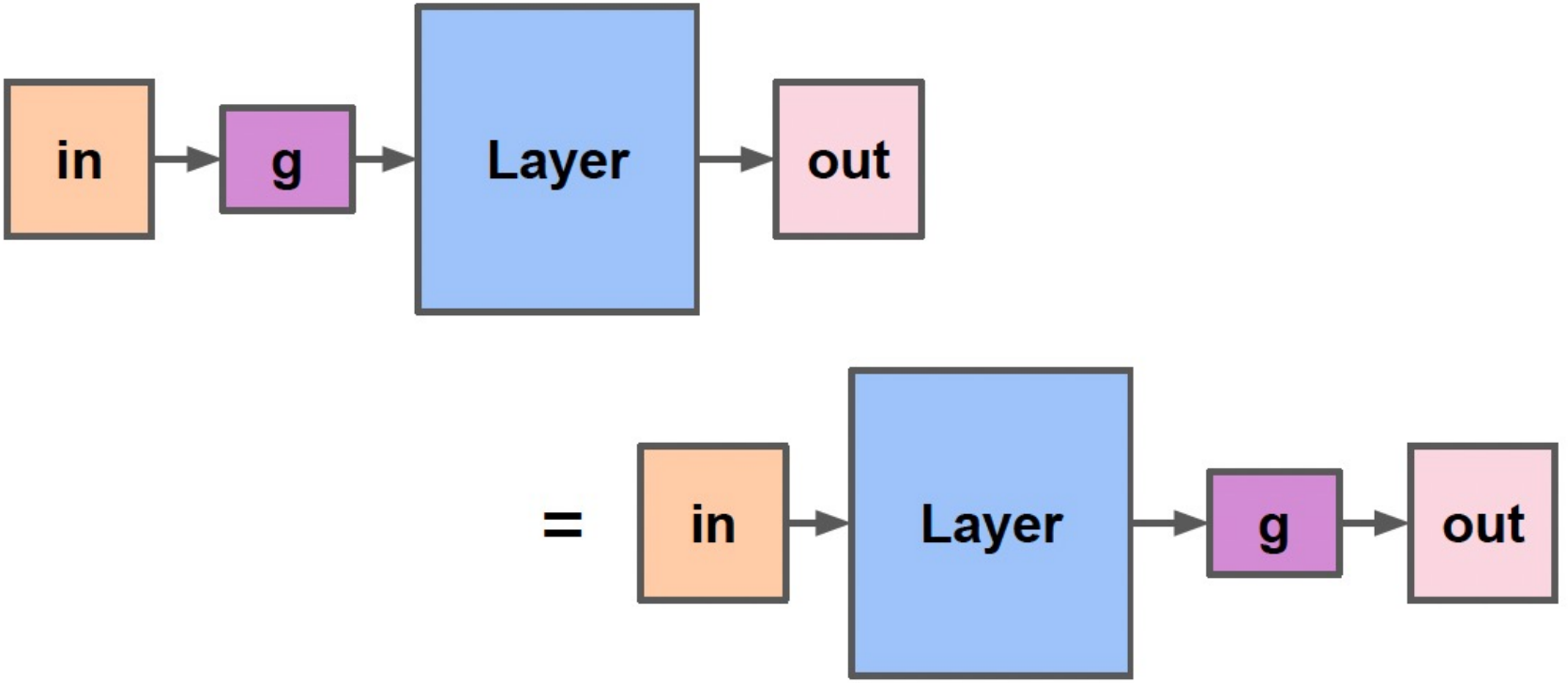
For 3D data, data augmentation is expensive, ~500 fold augmentation and you *still* don't get the guarantee of equivariance (it's only emulated).

training without rotational symmetry



training with symmetry

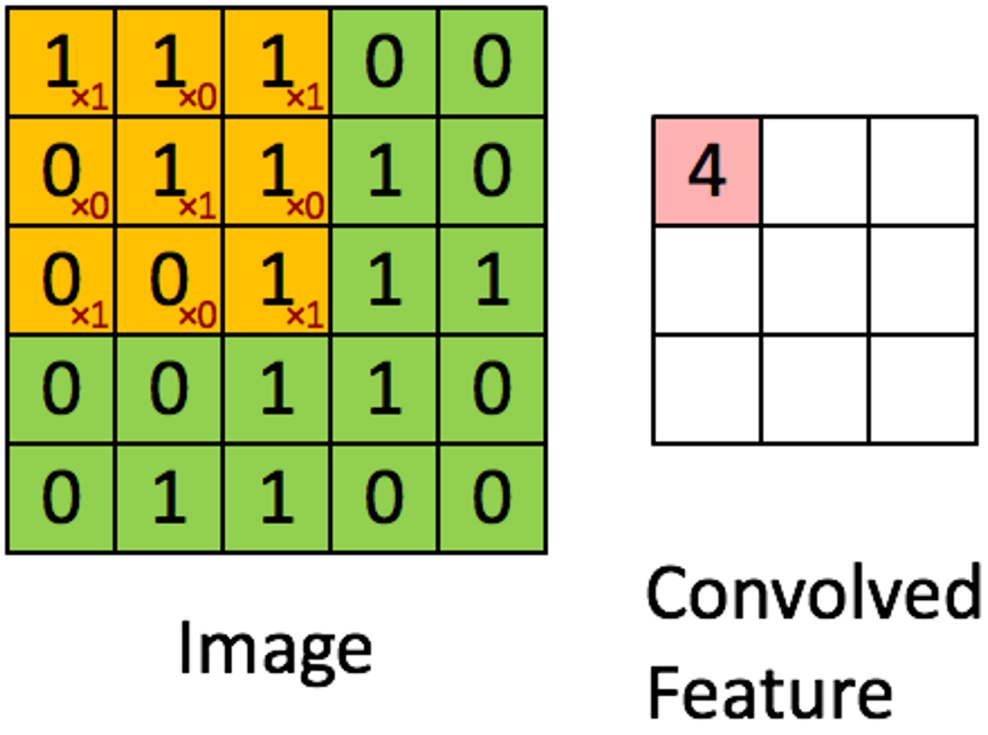
For a function to be *equivariant* means that we can act on our inputs with g
OR act our outputs with g and we get the same answer (for every operation).
For a function with *invariant* input (e.g. invariant models) means g is the identity (no change).



Convolutional filters

convolutional neural networks:

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



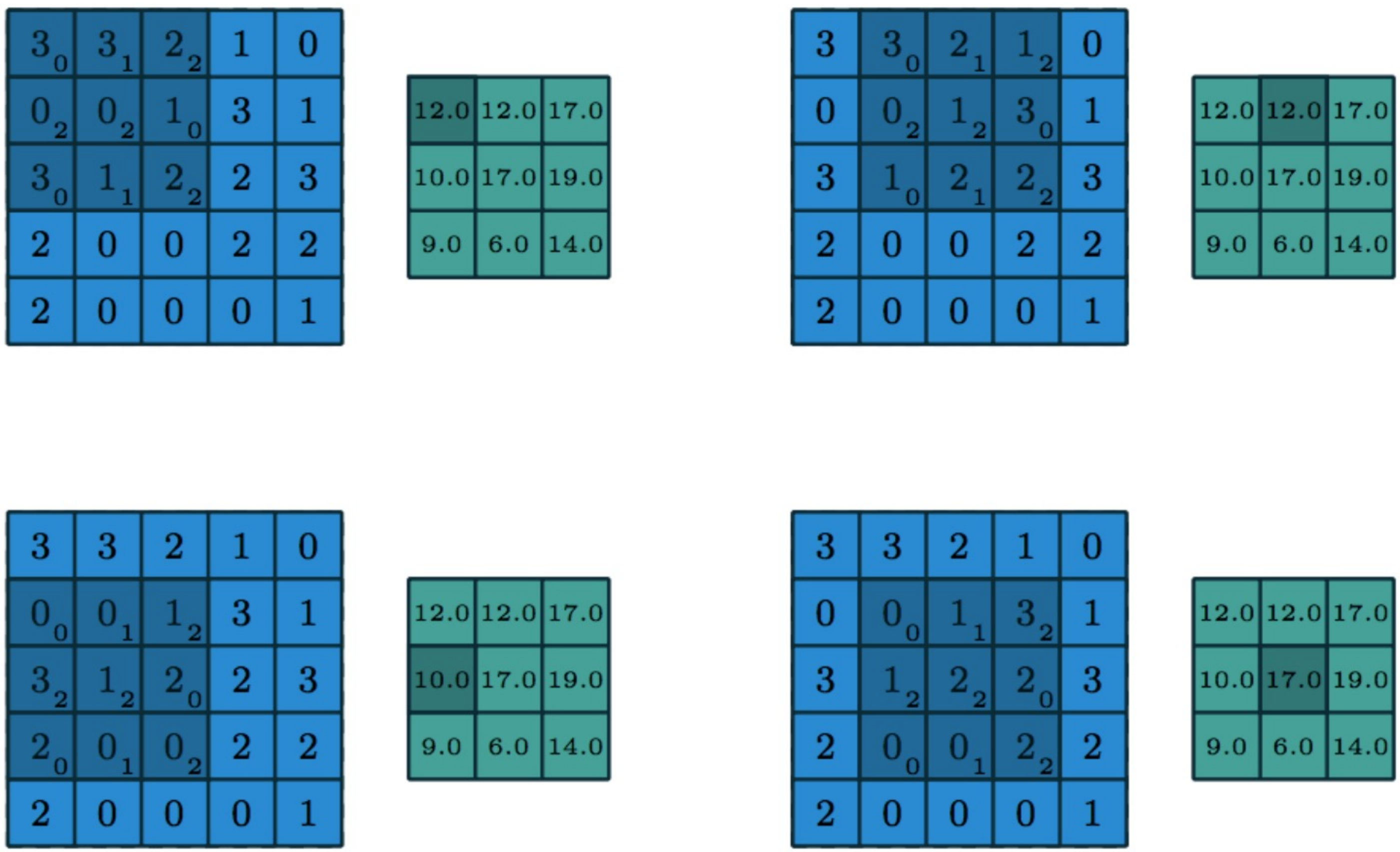
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolutional filters

Convolutional Kernel / Filter

0	1	2
2	2	0
0	1	2

Apply convolutions



Equivariant Networks

First paper:

Group Equivariant Convolutional Networks

Taco S. Cohen, Max Welling ICML 2016

generalization of convolutional networks to represent groups of symmetries such as rotations

Equivariance: network should transform as the input

$$\Phi(T_g\, x) = T'_g\, \Phi(x)$$

Main idea:

instead of convolutions as

$$[f * \psi^i](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K^l} f_k(y) \psi_k^i(x - y)$$

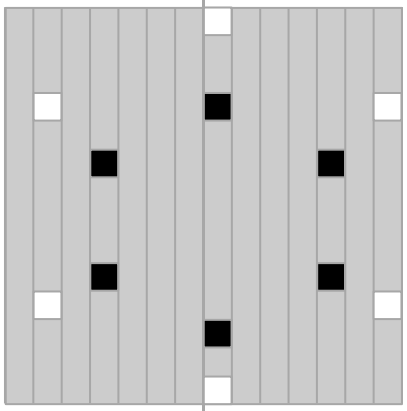
$$[f \star \psi^i](x) = \sum_{y \in \mathbb{Z}^2} \sum_{k=1}^{K^l} f_k(y) \psi_k^i(y - x)$$

perform convolutions as:

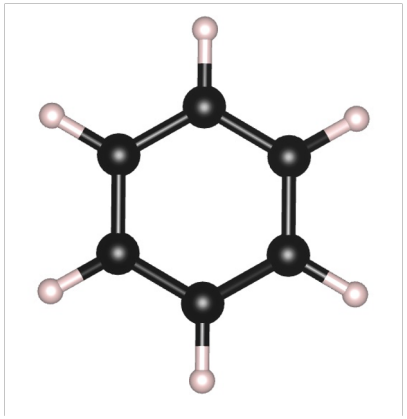
$$[f \star \psi](g) = \sum_{h \in G} \sum_{k} f_k(h) \psi_k(g^{-1} h)$$

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

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



VS.



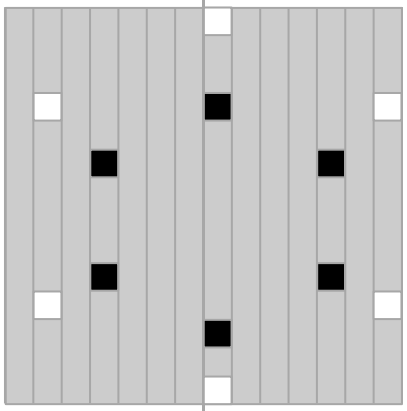
We use continuous convolutions with atoms as convolution centers.

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

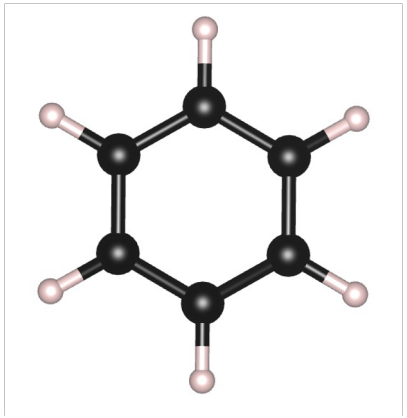
$$g \in SE(3)$$

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

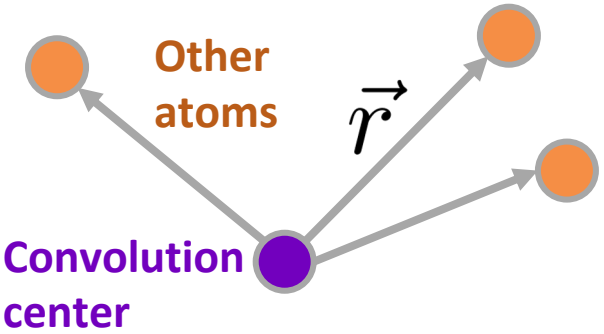
We use points. Images of atomic systems are sparse and imprecise.



vs.



We use continuous convolutions with atoms as convolution centers.

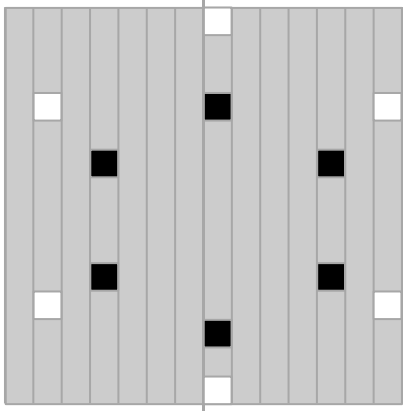


We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).

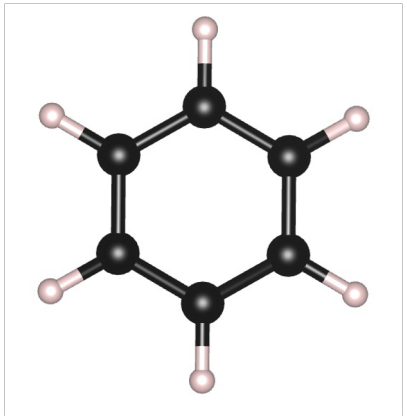
$$g \in SE(3)$$

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

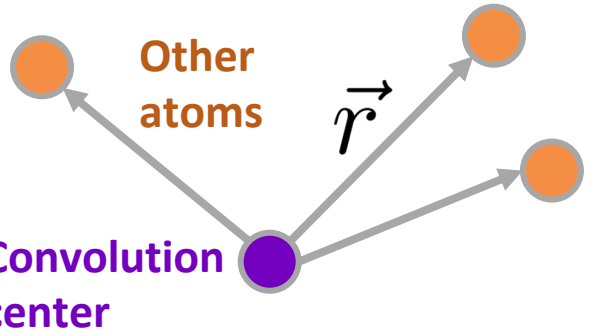
We use points. Images of atomic systems are sparse and imprecise.



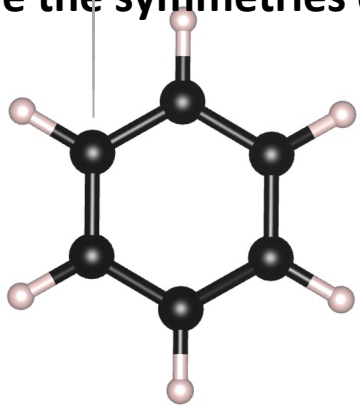
vs.



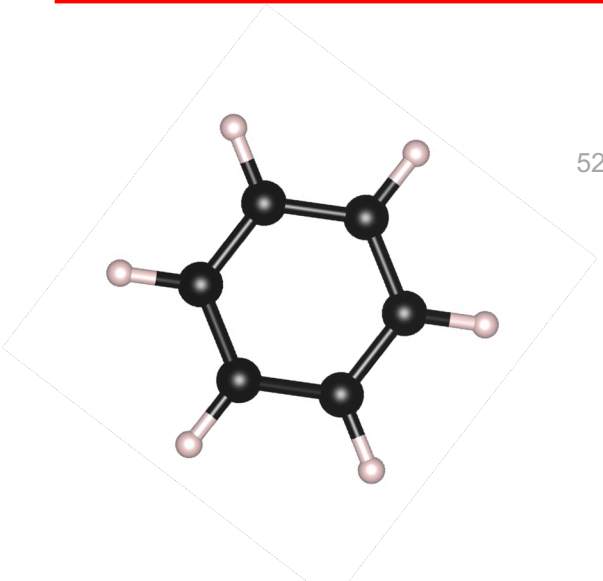
We use continuous convolutions with atoms as convolution centers.



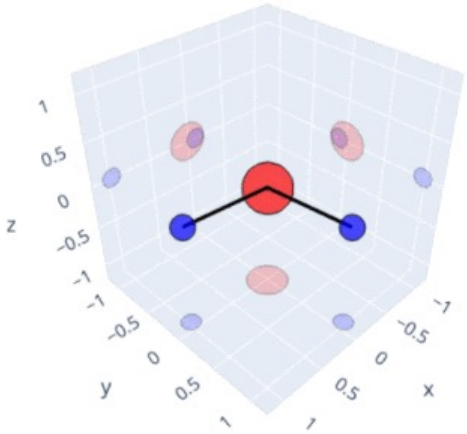
We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance)



$$g \in SE(3)$$

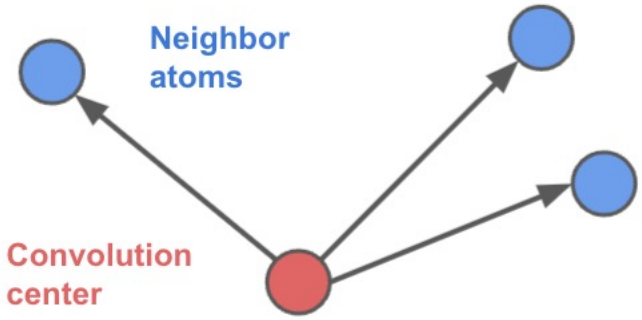


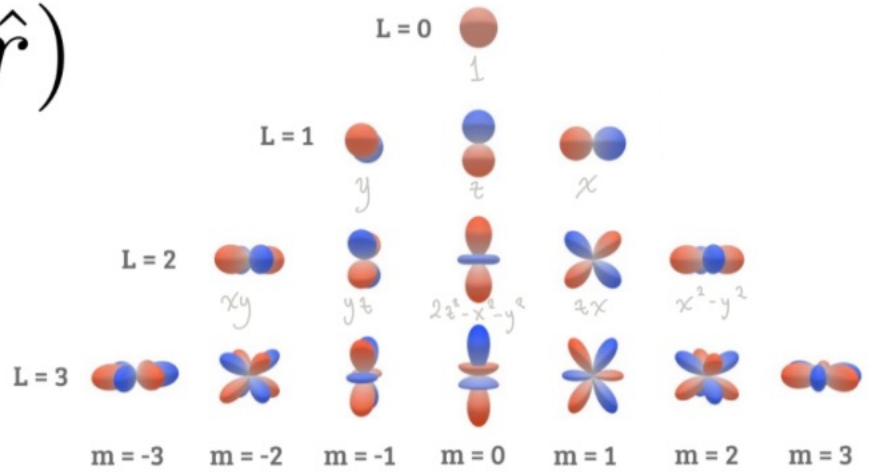
Euclidean Neural Networks



In Euclidean neural network, to achieve rotation equivariance, the convolution are constrained to be products of learnable radial functions and spherical harmonics, which are equivariant under SO(3). The radial function is implemented as a small neural network, operating on interatomic distances expressed in a basis of choice.

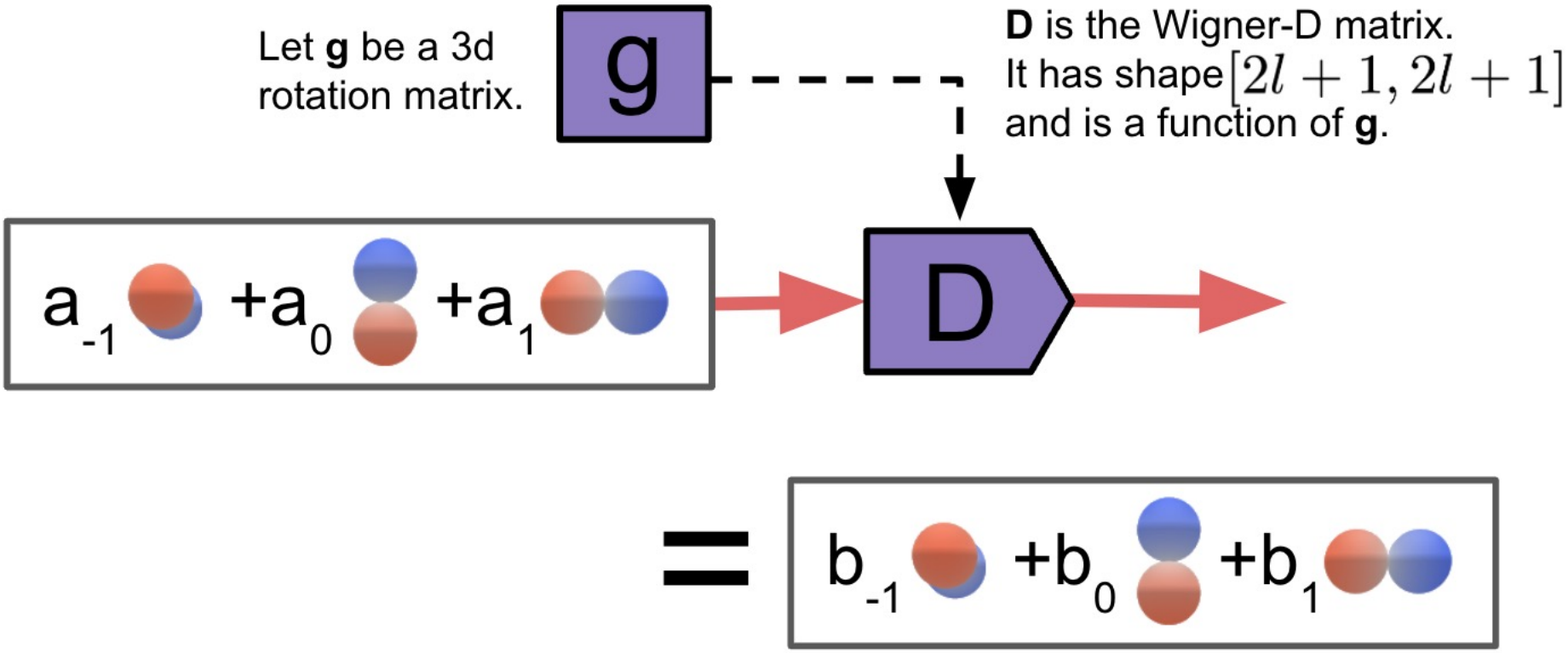
$$W(\vec{r}) = R(r) Y_l^m(\hat{r})$$





Euclidean Neural Networks

Spherical harmonics of a given L transform together under rotation.



$$Y_m^{(l)}(\mathcal{R}(g)\hat{r}) = \sum_{m'} D_{mm'}^{(l)}(g) Y_{m'}^{(l)}(\hat{r}).$$

$$D^{(0)}(g) = 1$$
$$D^{(1)}(g) = \mathcal{R}(g)$$

Euclidean Neural Networks

Euclidean Neural Networks are similar to convolutional neural networks,
EXCEPT with special filters and (geometric) tensor algebra!

Feature 1:
Everything in the network is a geometric tensor!

Scalar multiplication gets replaced with the more general geometric tensor product.

Contract two indices to one with Clebsch-Gordan Coefficients.

Example: How do you "multiply" two vectors?

Dot product

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

Scalar, Rank-0

Cross product

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

Vector, Rank-1

Outer product

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

Matrix, Rank-2

Euclidean Neural Networks

In the convolution, the input feature tensor and the filter have to be combined in an equivariant manner, which is achieved via a geometric tensor product, yielding an output feature that again is rotationally equivariant. A tensor product of two geometric tensors is computed via Clebsch-Gordan coefficients.

$$\mathcal{L}^{(l_o)}_{acm_o}(\vec{r}_a, V^{(l_i)}_{acm_i}) = \sum_{m_f, m_i} C^{(l_o,m_o)}_{(l_f,m_f)(l_i,m_i)} \sum_{b \in S} R^{(l_f,l_i)}_c(r_{ab}) Y^{(l_f)}_{m_f}(\hat{r}_{ab}) V^{(l_i)}_{bcm_i}$$

$V^{(l)}_{acm}$ features

$l = 0, 1, 2, \ldots$ "rotation order"
a atom index
c channel index (feature elements)
$m \in [-l, l]$ representation index

e.g. How to multiply two vectors?

$\vec{a} \cdot \vec{b} = c$ scalar

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

$\vec{a} \otimes \vec{b} = C$ 3x3 matrix

Euclidean Neural Networks

In the convolution, the input feature tensor and the filter have to be combined in an equivariant manner, which is achieved via a geometric tensor product, yielding an output feature that again is rotationally equivariant. A tensor product of two geometric tensors is computed via Clebsch-Gordan coefficients.

$$\mathcal{L}^{(l_o)}_{acm_o}(\vec{r}_a, V^{(l_i)}_{acm_i}) = \sum_{m_f, m_i} C^{(l_o,m_o)}_{(l_f,m_f)(l_i,m_i)} \sum_{b \in S} R^{(l_f,l_i)}_c(r_{ab}) Y^{(l_f)}_{m_f}(\hat{r}_{ab}) V^{(l_i)}_{bcm_i}$$

$V^{(l)}_{acm}$ features

$l = 0, 1, 2, \ldots$ "rotation order"
a atom index
c channel index (feature elements)
$m \in [-l, l]$ representation index

five distinct "interactions" between $l = 0$ and $l = 1$ filters and features that correspond to simple operations between scalars and vectors:

- $0 \otimes 0 \to 0$ (product of two scalars)
- $0 \otimes 1 \to 1$ (scalar multiplication of a vector)
- $1 \otimes 0 \to 1$ (scalar multiplication of a vector)
- $1 \otimes 1 \to 0$ (dot product of two vectors)
- $1 \otimes 1 \to 1$ (cross product of two vectors)

Euclidean Neural Networks

Formal definition of Clebsch–Gordan coefficients [edit]

The coupled states can be expanded via the completeness relation (resolution of identity) in the uncoupled basis

$$|J\,M\rangle = \sum_{m_1=-j_1}^{j_1} \sum_{m_2=-j_2}^{j_2} |j_1\,m_1\,j_2\,m_2\rangle\langle j_1\,m_1\,j_2\,m_2|J\,M\rangle$$

The expansion coefficients

$$\langle j_1\,m_1\,j_2\,m_2|J\,M\rangle$$

are the *Clebsch–Gordan coefficients*. Note that some authors write them in a different order such as $\langle j_1\,j_2;\,m_1\,m_2|J\,M\rangle$. Another common notation is $\langle j_1\,m_1\,j_2\,m_2\,|\,J\,M\rangle = C^{JM}_{j_1 m_1 j_2 m_2}$.

Applying the operators

$$\begin{aligned} \mathrm{J} &= \mathrm{j}\otimes 1 + 1\otimes \mathrm{j} \\ \mathrm{J_z} &= \mathrm{j_z}\otimes 1 + 1\otimes \mathrm{j_z} \end{aligned}$$

to both sides of the defining equation shows that the Clebsch–Gordan coefficients can only be nonzero when

$$\begin{aligned} |j_1 - j_2| \le J \le j_1 + j_2 \\ M = m_1 + m_2 \end{aligned}.$$

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

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

Notation:

J	J	...
M	M	...

m_1	m_2	
m_1	m_2	Coefficients
.	.	
.	.	
.	.	

$1/2 \times 1/2$

	1	
$+1/2\ +1/2$	1	$1\quad 0$ $0\quad 0$

$+1/2\ -1/2$	$1/2\quad 1/2$	1
$-1/2\ +1/2$	$1/2\ -1/2$	-1

$-1/2\ -1/2$	1

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

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

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

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

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

$2 \times 1/2$

	$5/2$	
$+2\ +1/2$	$+5/2$	$5/2\quad 3/2$ $+3/2\ +3/2$

$+2\ -1/2$	$1/5\quad 4/5$	$5/2\quad 3/2$
$+1\ +1/2$	$4/5\ -1/5$	$+1/2\ +1/2$

$+1\ -1/2$	$2/5\quad 3/5$	$5/2\quad 3/2$
$0\ +1/2$	$3/5\ -2/5$	$-1/2\ -1/2$

$0\ -1/2$	$3/5\quad 2/5$	$5/2\quad 3/2$
$-1\ +1/2$	$2/5\ -3/5$	$-3/2\ -3/2$

$-1\ -1/2$	$4/5\quad 1/5$	$5/2$
$-2\ +1/2$	$1/5\ -4/5$	$-5/2$

$-2\ -1/2$	1

$1 \times 1/2$

	$3/2$	
$+1\ +1/2$	$+3/2$	$3/2\quad 1/2$ $+1/2\ +1/2$

$+1\ -1/2$	$1/3\quad 2/3$	$3/2\quad 1/2$
$0\ +1/2$	$2/3\ -1/3$	$-1/2\ -1/2$

$0\ -1/2$	$2/3\quad 1/3$	$3/2$
$-1\ +1/2$	$1/3\ -2/3$	$-3/2$

$-1\ -1/2$	1

$3/2 \times 1/2$

	2		
$+3/2\ +1/2$	$+2$	$2\quad 1$ $+1\quad +1$	

$+3/2\ -1/2$	$1/4\quad 3/4$	$2\quad 1$	
$+1/2\ +1/2$	$3/4\ -1/4$	$0\quad 0$	

$+1/2\ -1/2$	$1/2\quad 1/2$	$2\quad 1$	
$-1/2\ +1/2$	$1/2\ -1/2$	$-1\quad -1$	

$-1/2\ -1/2$	$3/4\quad 1/4$	2	
$-3/2\ +1/2$	$1/4\ -3/4$	-2	

$-3/2\ -1/2$	1

2×1

	3	
$+2\ +1$	$+3$	$3\quad 2$ $+2\quad +2$

$+2\quad 0$	$1/3\quad 2/3$	$3\quad 2\quad 1$
$+1\ +1$	$2/3\ -1/3$	$+1\quad +1\quad +1$

$+2\ -1$	$1/15$	$1/3$	$3/5$
$+1\quad 0$	$8/15$	$1/6$	$-3/10$
$0\ +1$	$2/5$	$-1/2$	$1/10$

$3/2 \times 1$

	$5/2$	
$+3/2\ +1$	$+5/2$	$5/2\quad 3/2$ $+3/2\ +3/2$

$+3/2\quad 0$	$2/5\quad 3/5$	$5/2\quad 3/2\quad 1/2$	
$+1/2\ +1$	$3/5\ -2/5$	$+1/2\quad +1/2\quad +1/2$	

$+3/2\ -1$	$1/10$	$2/5$	$1/2$
$+1/2\quad 0$	$3/5$	$1/15$	$-1/3$
$-1/2\ +1$	$3/10$	$-8/15$	$1/6$

$5/2$	$3/2$	$1/2$	
$-1/2$	$-1/2$	$-1/2$	

$+1/2\ -1$	$3/10$	$8/15$	$1/6$
$-1/2\quad 0$	$3/5$	$-1/15$	$-1/3$
$-3/2\ +1$	$1/10$	$-2/5$	$1/2$

$5/2$	$3/2$	
$-3/2$	$-3/2$	

$-1/2\ -1$	$3/5$	$2/5$
$-3/2\quad 0$	$2/5$	$-3/5$

$5/2$	
$-5/2$	

$-3/2\ -1$	1

1×1

	2		
$+1\ +1$	$+2$	$2\quad 1$ $+1\quad +1$	

$+1\quad 0$	$1/2\quad 1/2$	$2\quad 1\quad 0$	
$0\ +1$	$1/2\ -1/2$	$0\quad 0\quad 0$	

$+1\ -1$	$1/6$	$1/2$	$1/3$
$0\quad 0$	$2/3$	0	$-1/3$
$-1\ +1$	$1/6$	$-1/2$	$1/3$

3	2	1	
0	0	0	

$+1\ -1$	$1/5$	$1/2$	$3/10$
$0\quad 0$	$3/5$	0	$-2/5$
$-1\ +1$	$1/5$	$-1/2$	$3/10$

3	2	1	
-1	-1	-1	

$0\ -1$	$2/5$	$1/2$	$1/10$
$-1\quad 0$	$8/15$	$-1/6$	$-3/10$
$-2\ +1$	$1/15$	$-1/3$	$3/5$

2	1	0	
0	0	0	

$+1\ -1$	$1/2$	$1/2$	2
$0\quad 0$	$1/2$	$-1/2$	-2
$-1\ +1$			

2	1		
-1	-1		

$0\ -1$	$1/2$	$1/2$	2
$-1\quad 0$	$1/2$	$-1/2$	-2
$-1\ -1$			

3	2		
-2	-2		

$-1\ -1$	$2/3$	$1/3$	3
$-2\quad 0$	$1/3$	$-2/3$	-3
$-2\ -1$			

Euclidean Neural Networks

Recent application to learn atomic force fields:

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials
S. Batzner, et al. arXiv:2101.03164 (2021)

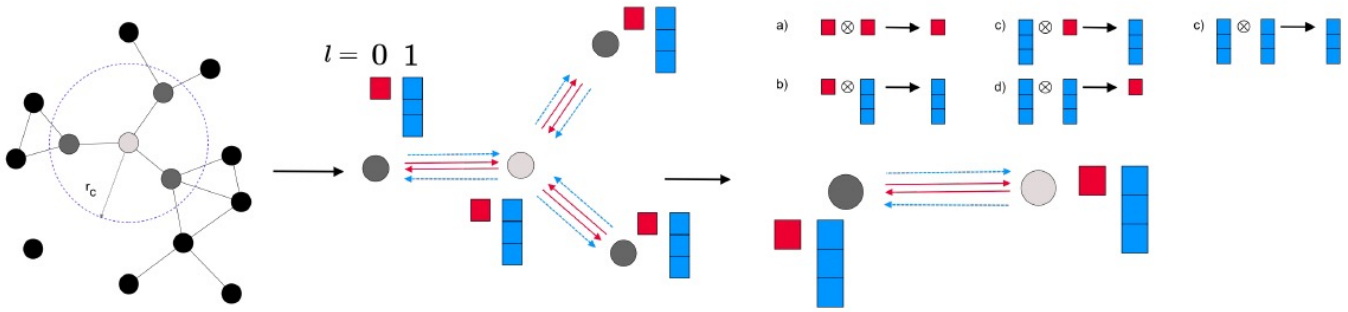
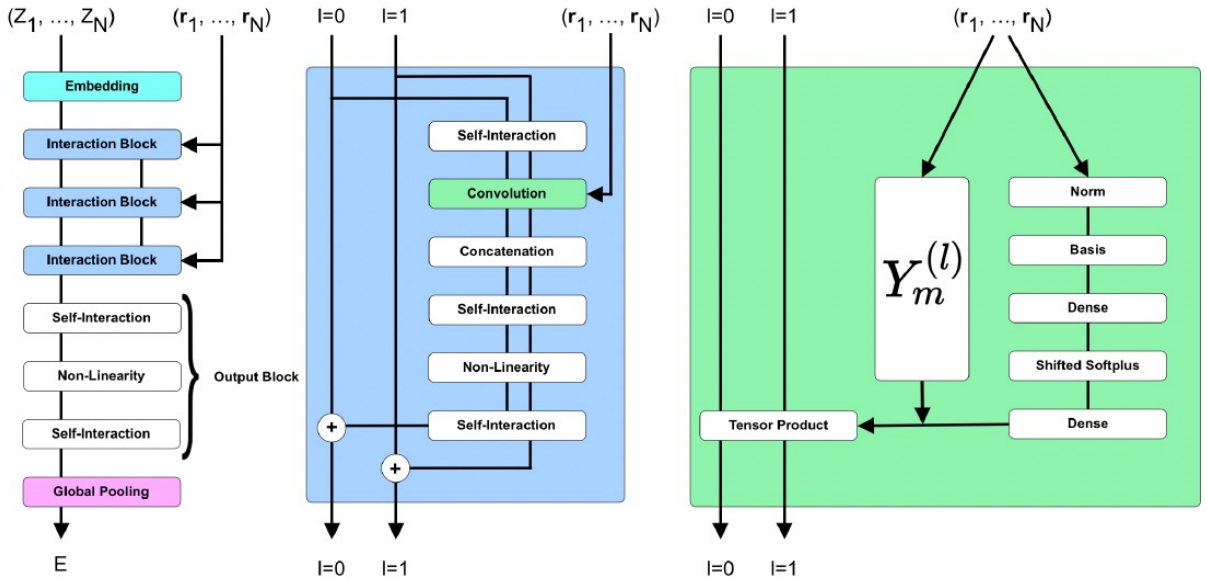


FIG. 1: Left: a set of atoms is interpreted as an atomic graph with local neighborhoods. Middle: every atom carries a set of scalar and vector features with it. Right: atoms exchange information via filters, that are again scalars and vectors. The interactions of features and filters define five interactions.



Networks are trained using a loss function based on atomic forces

Euclidean Neural Networks

Recent application to learn atomic force fields:

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials
S. Batzner, et al. arXiv:2101.03164 (2021)

Molecule	NequIP ($l = 1$)	SchNet	sGDML	DimeNet	FCHL19/GPR
Aspirin	15.1	58.5	29.5	21.6	20.7
Benzene [17]	8.1	13.4	n/a	8.1	n/a
Benzene [31]	2.3	n/a	2.6	n/a	n/a
Ethanol	9.0	16.9	14.3	10.0	5.9
Malonaldehyde	14.6	28.6	17.8	16.6	10.6
Naphthalene	4.2	25.2	4.8	9.3	6.5
Salicylic Acid	10.3	36.9	12.1	16.2	9.6
Toluene	4.4	24.7	6.1	9.4	8.8
Uracil	7.5	24.3	10.4	13.1	4.6

TABLE I: MAE of force components on the MD-17 data set, trained on 1,000 configurations, forces in units of [meV/Å]. For the benzene molecule, two different data set exists from [17], [31] with different levels of accuracy in the DFT reference data.

Molecule	NequIP ($l = 1$)	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).

Euclidean Neural Networks

Recent application to learn atomic force fields:

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials
S. Batzner, et al. arXiv:2101.03164 (2021)

Lithium Phosphate Amorphous Glass Formation

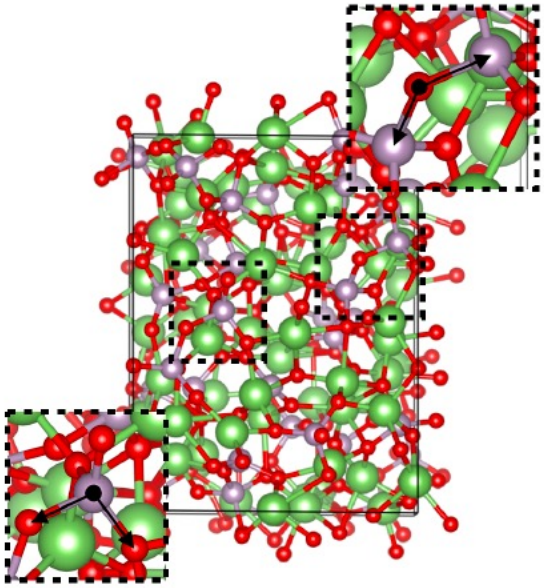
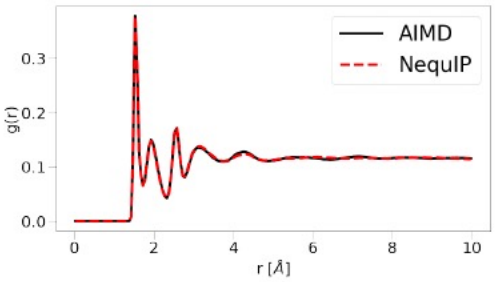
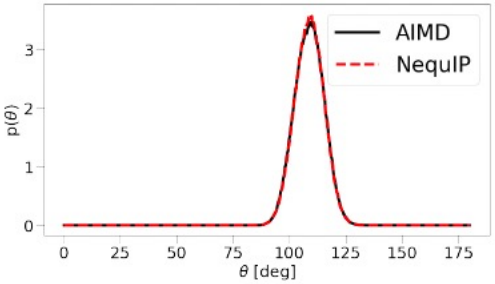
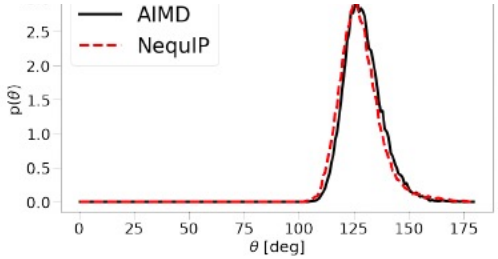


FIG. 4: Quenched glass structure of $Li_4P_2O_7$. The insets show the P-O-O tetrahedral bond angle (bottom left) as well as the O-P-P bridging angle between corner-sharing phosphate tetrahedra (top right).







System	Data Set Size	MAE
LiPS	10	157.1
LiPS	100	50.0
LiPS	1,000	25.1
LiPS	2,500	24.1
$Li_4P_2O_7$, melt	1,000	63.2
$Li_4P_2O_7$, quench	1,000	36.9

TABLE V: NequIP ($l = 1$) Force MAE for LiPS and $Li_4P_2O_7$ for different data set sizes in units of [meV/A]. The model for $Li_4P_2O_7$ was trained exclusively on structures from the melted trajectory, the reported test errors show the MAE on both the test set of the melted trajectory as well as the full quench trajectory.

Euclidean Neural Networks

Recent application to learn atomic force fields:

SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials
S. Batzner, et al. arXiv:2101.03164 (2021)

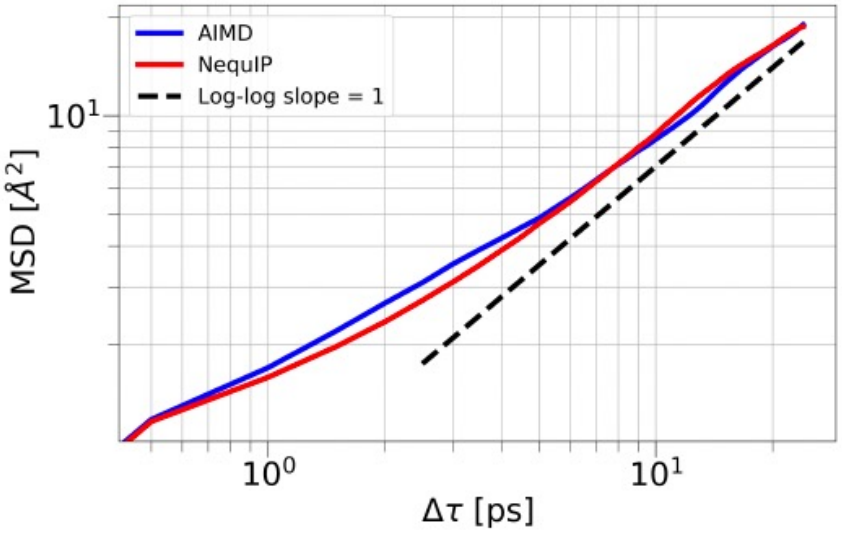


FIG. 6: Comparison of Lithium mean square displacement of AIMD and NequIP trajectories.

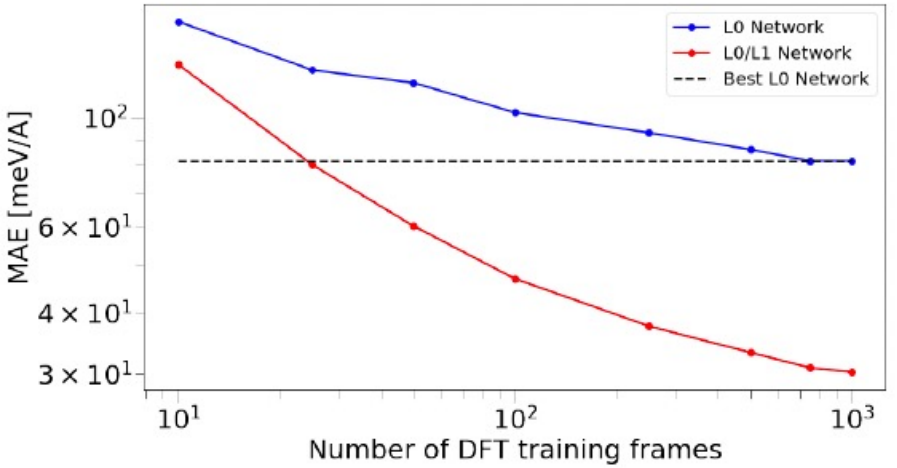


FIG. 7: Log-log plot of the predictive error in forces of NequIP with $l = 0$ vs. $l = 0/l = 1$ interactions as a function of data set size, measured via the force MAE.

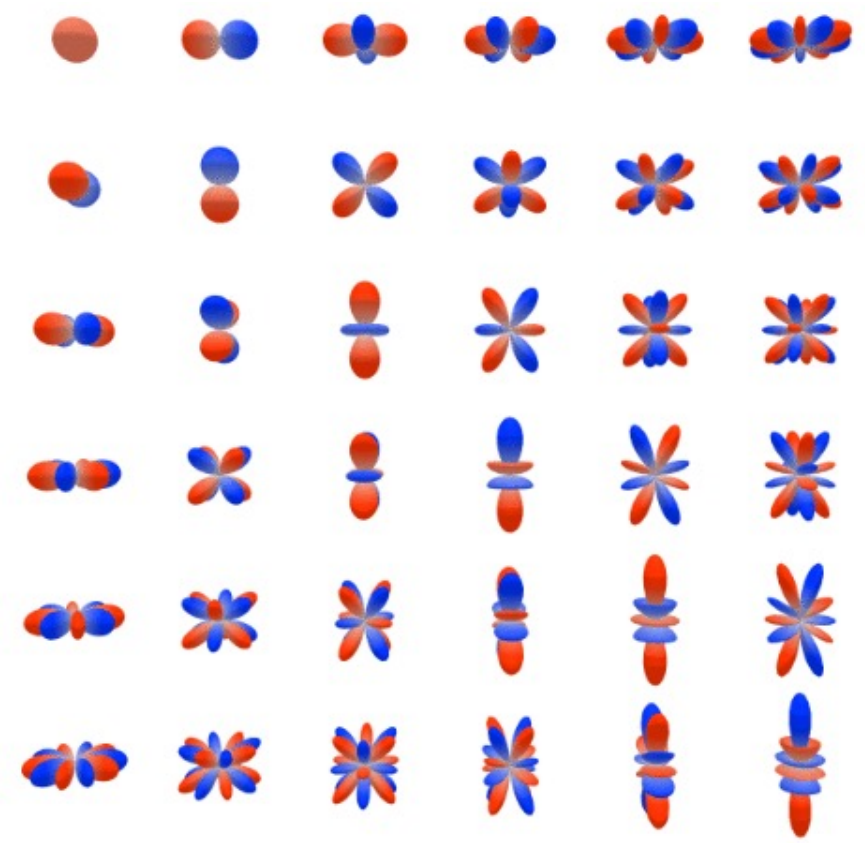
System	Number of atoms	NequIP	Ab-initio	Speed-up
Toluene	15	16 ms	4 hours*	900,000
Formate on Cu	52	58 ms	1045.6 s	18,028

TABLE VI: Time required for a single force call for NequIP ($l = 1$)in comparison to CCSD(T) for Toluene and DFT for Formate on Cu; * personal communication with Stefan Chmiela and Alexandre Tkatchenko.

Euclidean Neural Networks

Many incarnations, check this webpage for all the papers and more info:
https://e3nn.org/

E(3) is the Euclidean group in dimension 3. That is the group of rotations, translations and mirror. **e3nn** is a pytorch library that aims to create **E(3)** equivariant **n**eural **n**etworks.



A bit of group theory! Don't worry just a bit!
Formally, what are invariant vs. equivariant functions

vector in vector space

$x \in X$ inputs

$y \in Y$ outputs

$w \in W$ weights

element of group

$g \in G$

function (neural network)...

$f(x, w) = y$

...which is equivalent to writing.

$f : X, W \to Y$

representation of g acting on vector space

$D_x(g)$

$D_y(g)$

$D_x(g) : X \to X$

$D_y(g) : Y \to Y$

$D_w(g) = \mathbb{1}$

If we want to be equivariant to x, this has to be the case… weights must be "scalars"

equivariant to x if

$$f(D_x(g)\,x, D_w(g)w) = f(D_x(g)x, w)$$

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

(special case) invariant to x if

$$D_x(g) = D_y(g) = \mathbb{1}$$