Tensor Field Networks:

NeurIPS Molecules and Materials Workshop
2018.12.08

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

Tess Smidt
Berkeley Lab

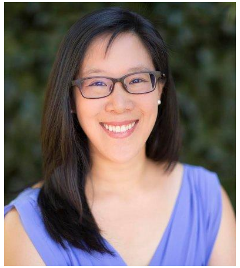
Stanford

Google Accelerated Science Team

Nate Thomas

Patrick Riley

Steve Kearnes



Lusann Yang

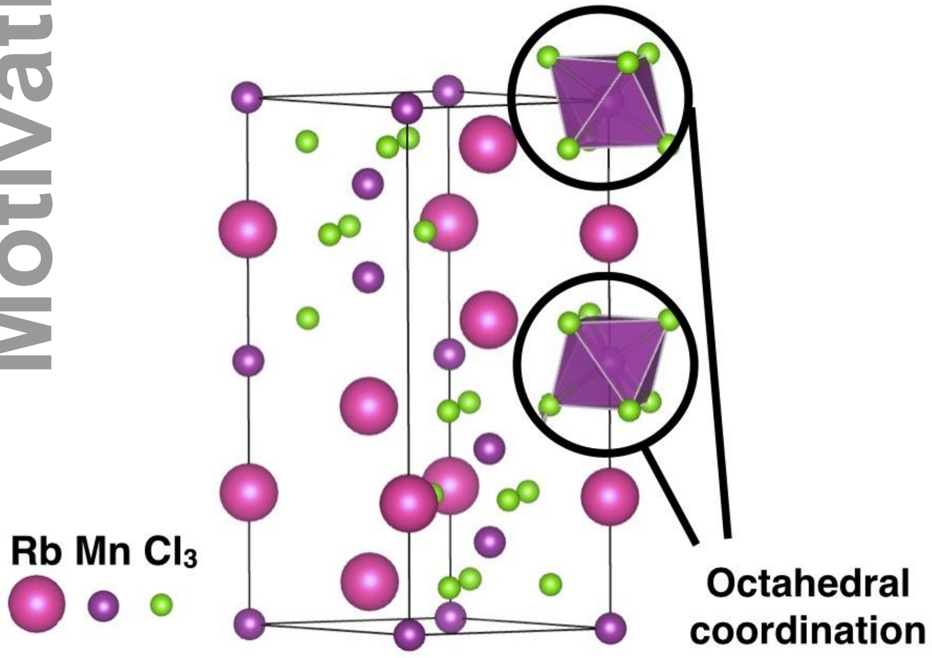
Li Li

Kai Kohlhoff

Motivation

Motivation

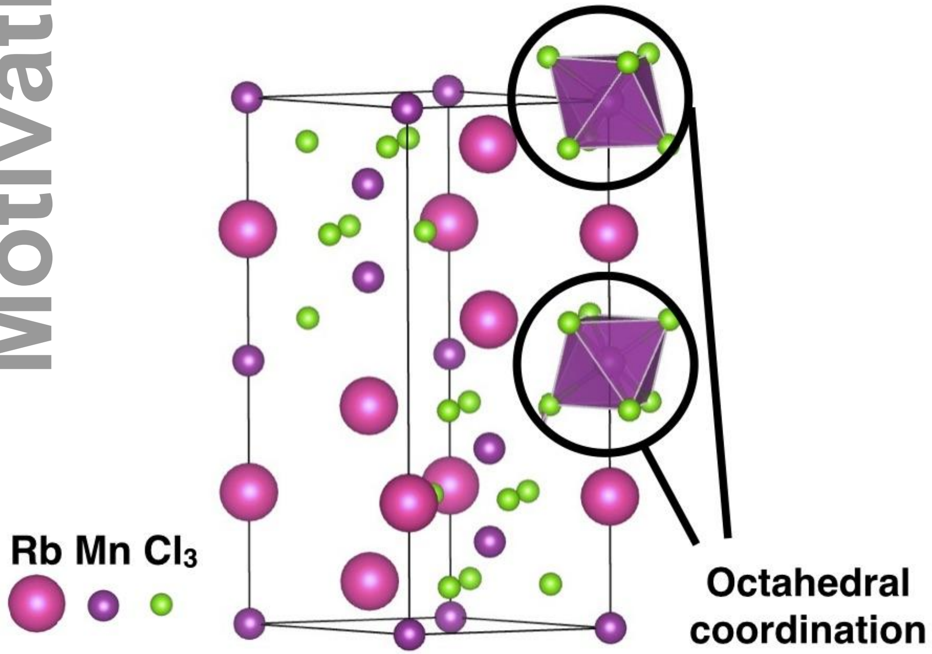
Atomic systems form geometric motifs that can appear at multiple locations and orientations.



How can we identify these rotated and translated motifs using the same filters?

Motivation

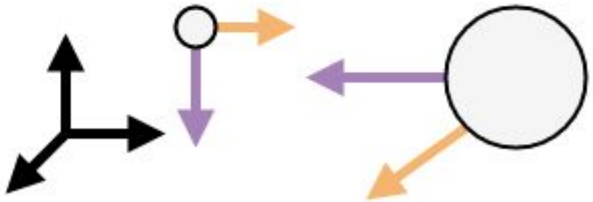
Atomic systems form geometric motifs that can appear at multiple locations and orientations.



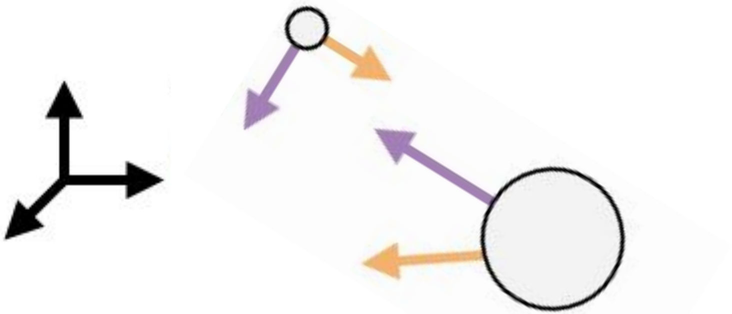
How can we identify these rotated and translated motifs using the same filters?

The properties of physical systems transform predictably under rotation.

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



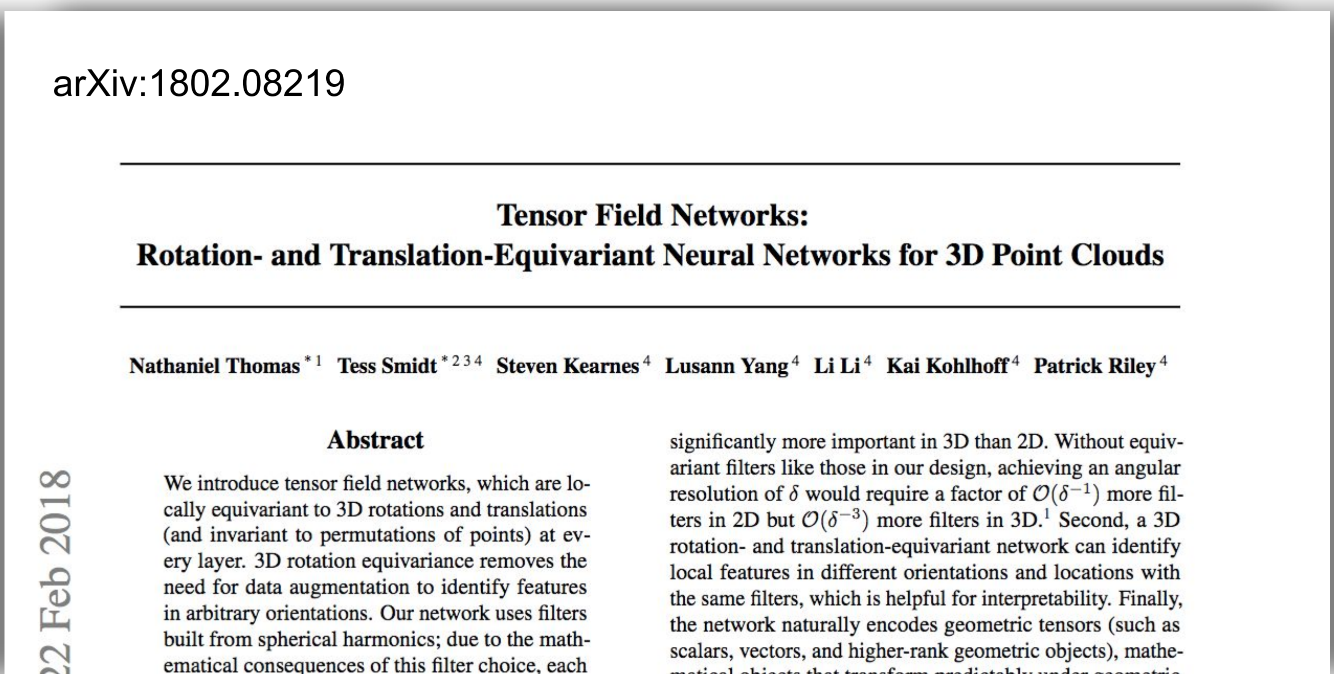
Same system, with rotated coordinates.



Can we construct a network that naturally handles these data types?

We created a network that can naturally handle 3D geometry and features of physical systems.

- It can be applied to any type of atomic system *(molecules, materials, proteins, hybrid systems, nanoclusters, etc.)*
- And preserves geometric information (lengths **and angles**).



arXiv:1802.08219

Tensor Field Networks:
Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds

Nathaniel Thomas [*1] Tess Smidt [*234] Steven Kearnes [4] Lusann Yang [4] Li Li [4] Kai Kohlhoff [4] Patrick Riley [4]

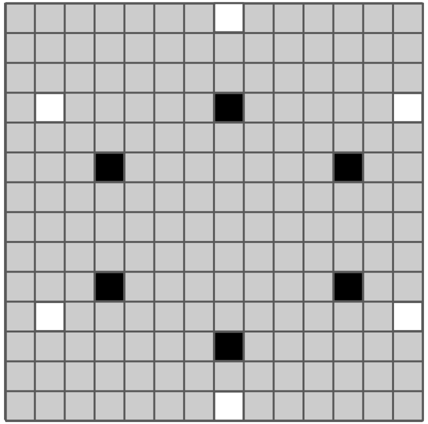
22 Feb 2018

Abstract

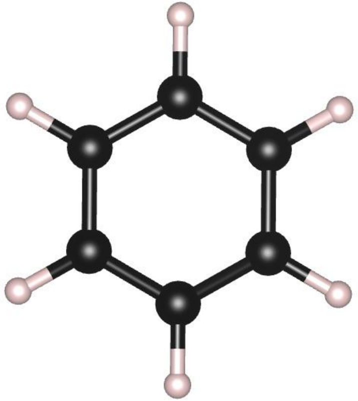
We introduce tensor field networks, which are locally equivariant to 3D rotations and translations (and invariant to permutations of points) at every layer. 3D rotation equivariance removes the need for data augmentation to identify features in arbitrary orientations. Our network uses filters built from spherical harmonics; due to the mathematical consequences of this filter choice, each

significantly more important in 3D than 2D. Without equivariant filters like those in our design, achieving an angular resolution of δ would require a factor of $\mathcal{O}(\delta^{-1})$ more filters in 2D but $\mathcal{O}(\delta^{-3})$ more filters in 3D.[1] Second, a 3D rotation- and translation-equivariant network can identify local features in different orientations and locations with the same filters, which is helpful for interpretability. Finally, the network naturally encodes geometric tensors (such as scalars, vectors, and higher-rank geometric objects), mathe-

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



vs.

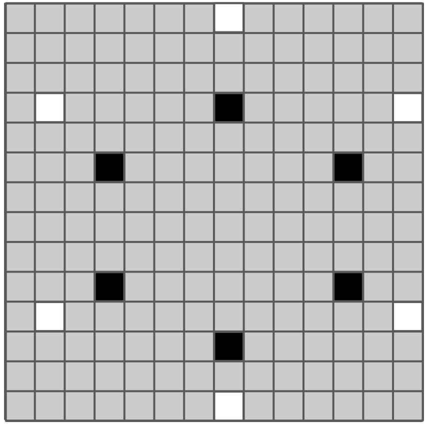


We use continuous convolutions with atoms as convolution centers.

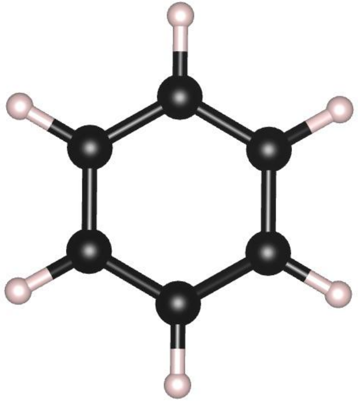
K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, Adv. in Neural Information Processing Systems 30 (2017). (arXiv: 1706.08566)

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

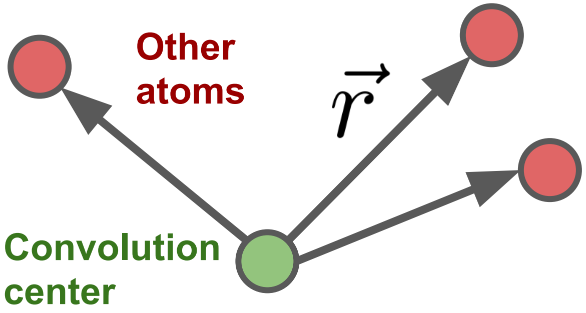
We use points. Images of atomic systems are sparse and imprecise.



vs.



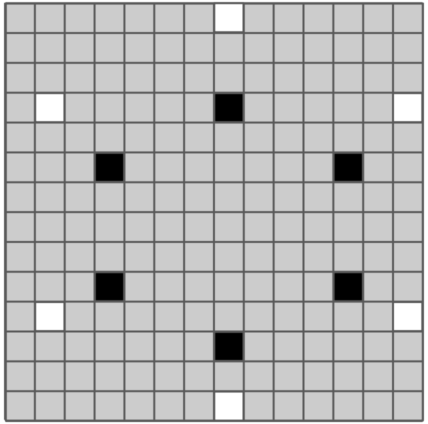
We use continuous convolutions with atoms as convolution centers.



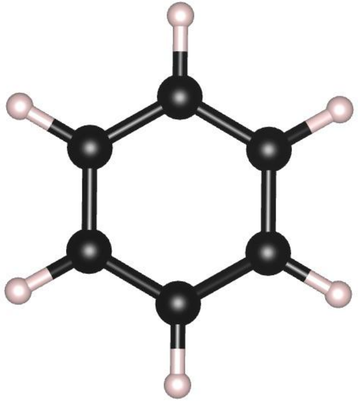
K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, Adv. in Neural Information Processing Systems 30 (2017). (arXiv: 1706.08566)

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

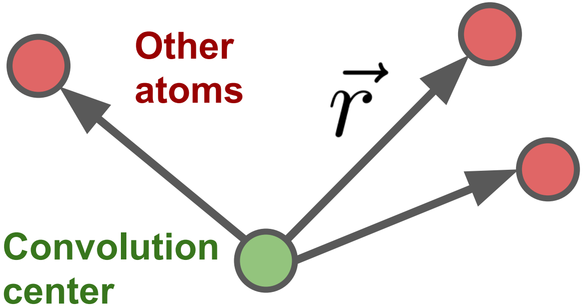
We use points. Images of atomic systems are sparse and imprecise.



vs.

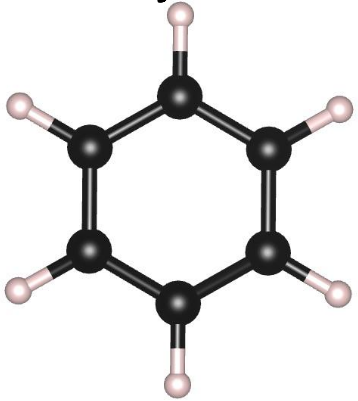


We use continuous convolutions with atoms as convolution centers.

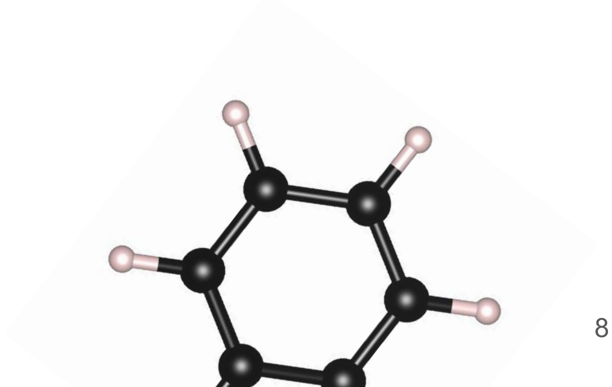


K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, Adv. in Neural Information Processing Systems 30 (2017). (arXiv: 1706.08566)

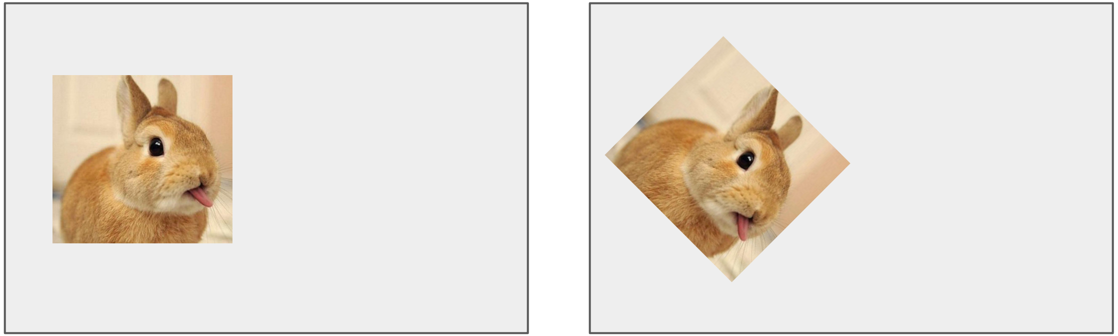
We encode the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).



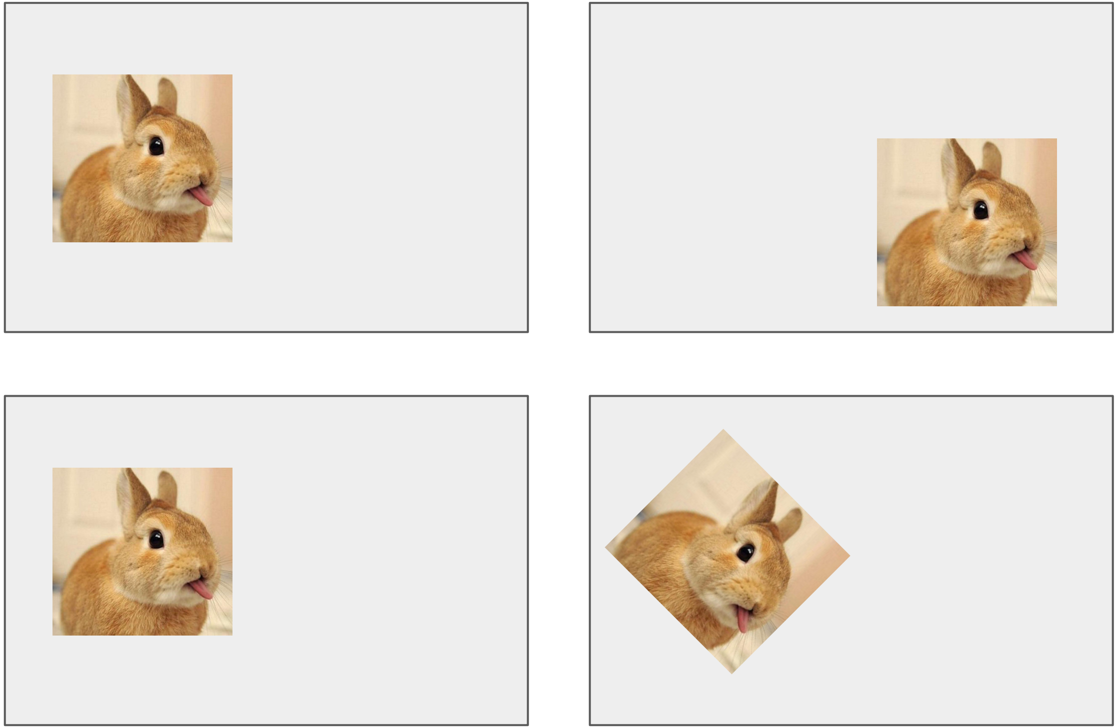
$$g \in SE(3)$$



Translation equivariance



Rotation equivariance



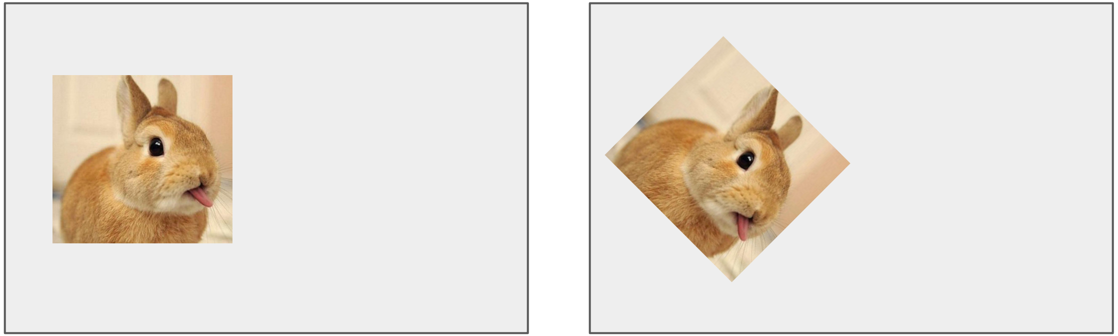
Translation equivariance

Convolutional neural network ✓

Rotation equivariance?

Translation equivariance

Convolutional neural network ✓



Rotation equivariance

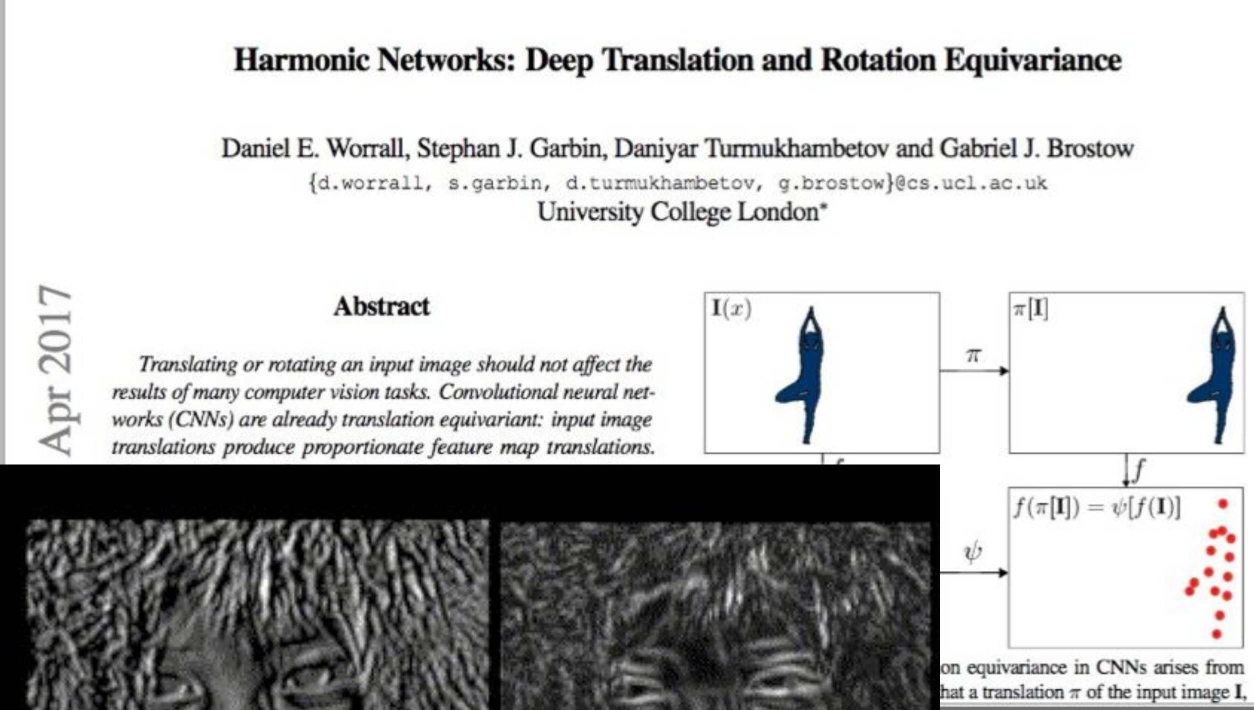
~~Data augmentation~~

~~Radial functions~~

Want a network that both preserves geometry and exploits symmetry.

Previous work on 2D rotation-equivariance uses filters based on circular harmonics.

To extend to 3D, we use spherical harmonics.



Harmonic Networks: Deep Translation and Rotation Equivariance

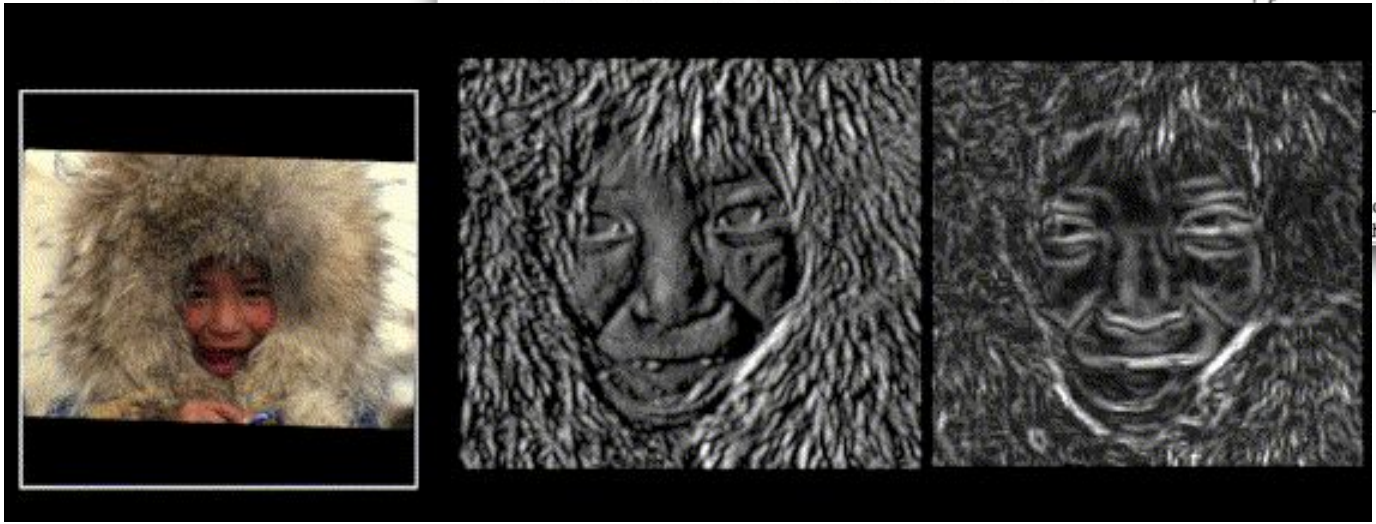
Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov and Gabriel J. Brostow
{d.worrall, s.garbin, d.turmukhambetov, g.brostow}@cs.ucl.ac.uk
University College London*

Apr 2017

Abstract

Translating or rotating an input image should not affect the results of many computer vision tasks. Convolutional neural networks (CNNs) are already translation equivariant: input image translations produce proportionate feature map translations.

on equivariance in CNNs arises from
hat a translation π of the input image $\mathbf{I}$,



Rotated image **CNN filter output** **Harmonic filter output**

http://visual.cs.ucl.ac.uk/pubs/harmonicNets/

Convolutional kernels...

Learned Parameters

with no symmetry:

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

with 2D circular harmonics:

$$\boxed{R(r)}e^{im\phi}$$

with 3D spherical harmonics:

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

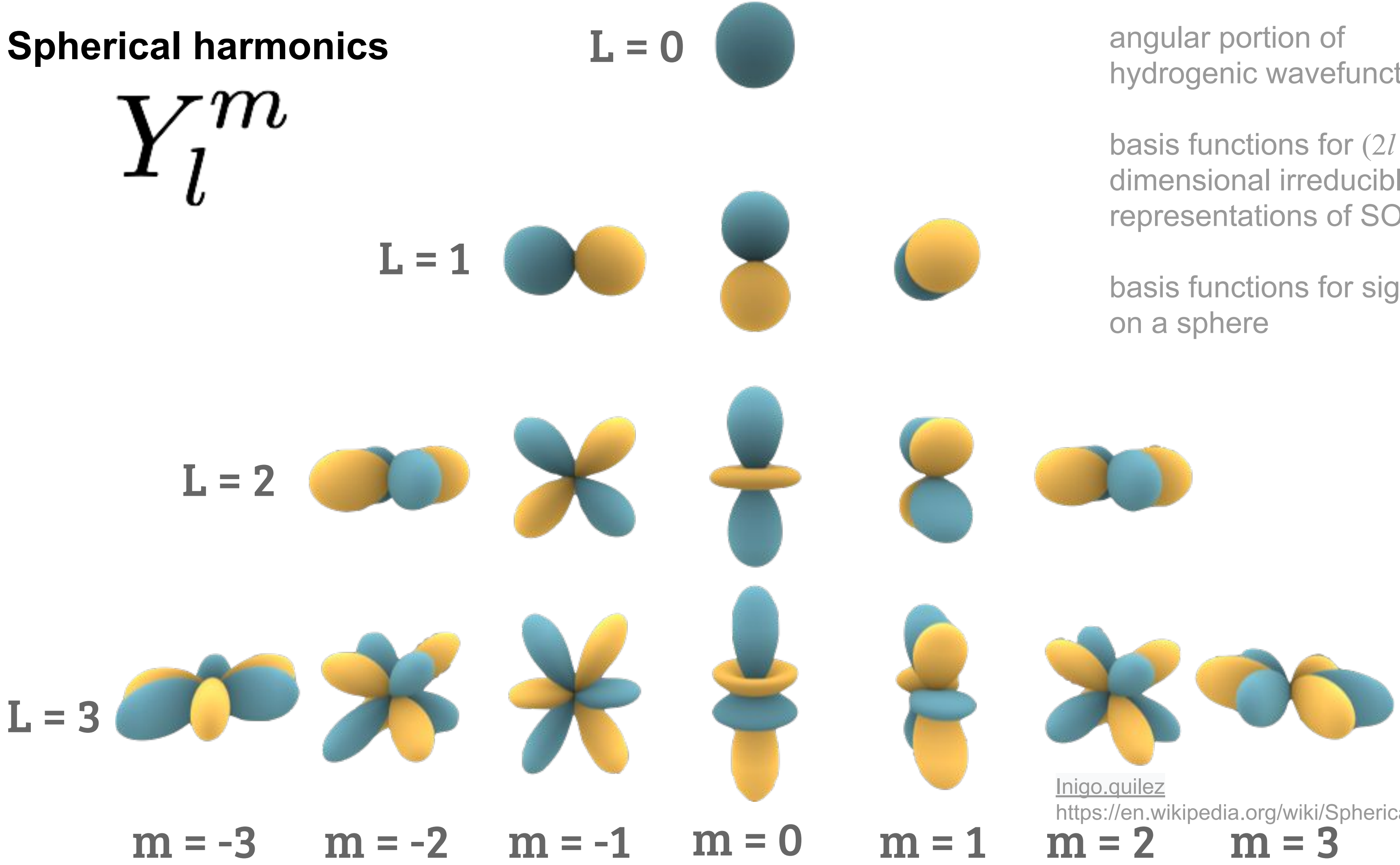
Spherical harmonics

$$Y_l^m$$

angular portion of hydrogenic wavefunctions

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

basis functions for signals on a sphere

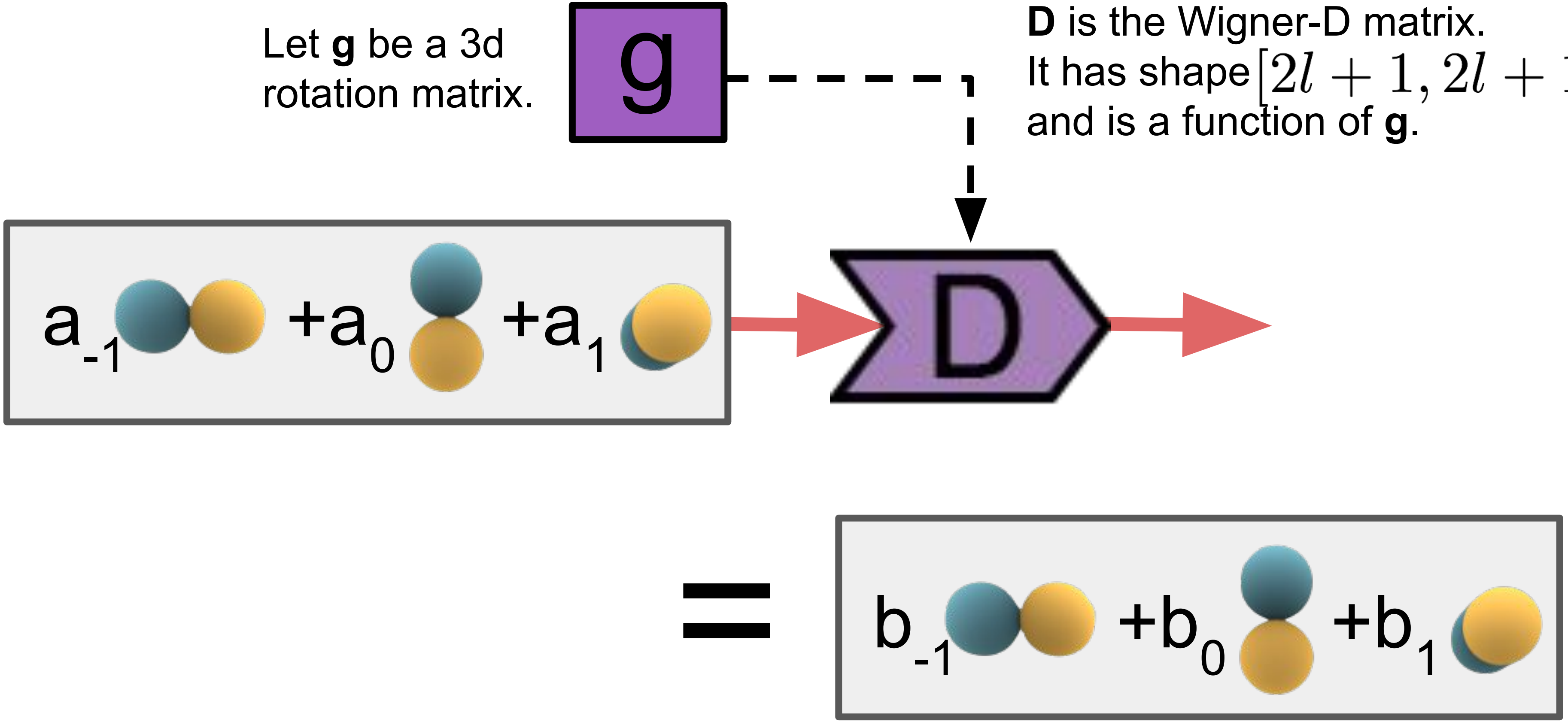


Inigo.quilez
https://en.wikipedia.org/wiki/Spherical_harmonics

Spherical harmonics of a given L transform together under rotation.

Let **g** be a 3d rotation matrix.

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



Going from 2D to 3D rotation-equivariance involves more than changing filters.

Going from 2D to 3D rotation-equivariance involves more than changing filters.

For rotation matrices **A** and **B**...

In 2D:

AB = **BA** (abelian)

In 3D:

AB ≠ **BA** (nonabelian)

Irreducible representations of SO(2) are all 1D.

Irreducible representations of SO(3) are $(2l+1)$ dimensional

Going from 2D to 3D rotation-equivariance involves more than changing filters.

For rotation matrices **A** and **B**...

In 2D:

AB = **BA** (abelian)

In 3D:

AB ≠ **BA** (nonabelian)

Irreducible representations of SO(2) are all 1D.

Irreducible representations of SO(3) are $(2l+1)$ dimensional

Our filter choice requires the input, filters, and output of our network to be **geometric tensors** and our network connectivity to be compatible with **tensor algebra**.

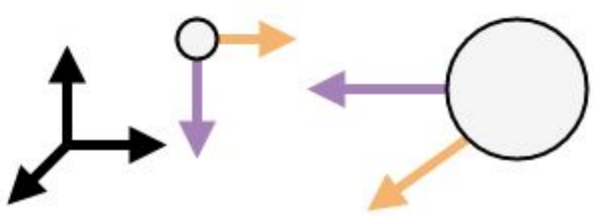
TensorFlow

MultidimensionalArray

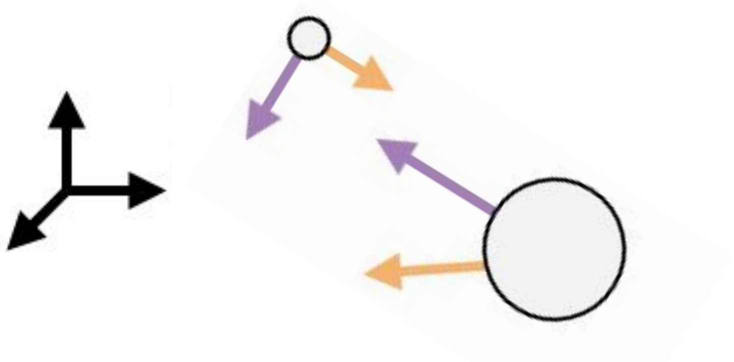
~~Tensor~~Flow

Geometric tensors transform predictably under 3D rotation.

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

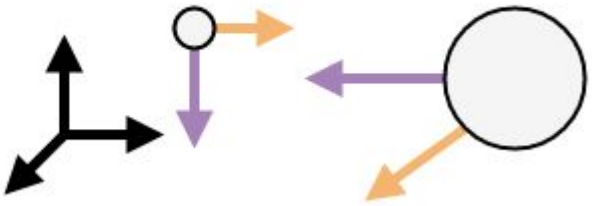


Same system, with rotated coordinates.

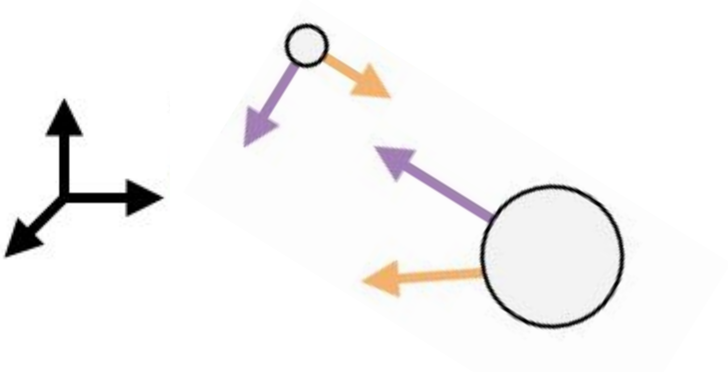


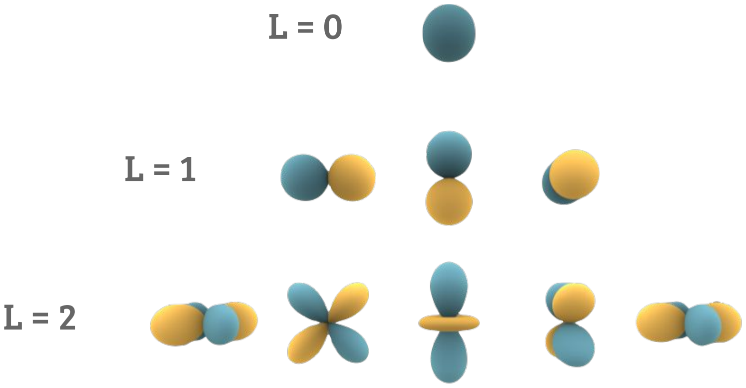
Geometric tensors transform predictably under 3D rotation.

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



Same system, with rotated coordinates.





Irreducible representations

Scalars fields $l = 0$

Vectors fields $l = 1$

3x3 Matrix fields $l = 0 \oplus 1 \oplus 2$

The input and output of our network is represented as tensors with **point** (or atom), **channel**, and **representation** indices organized by irreducible representation.

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

=

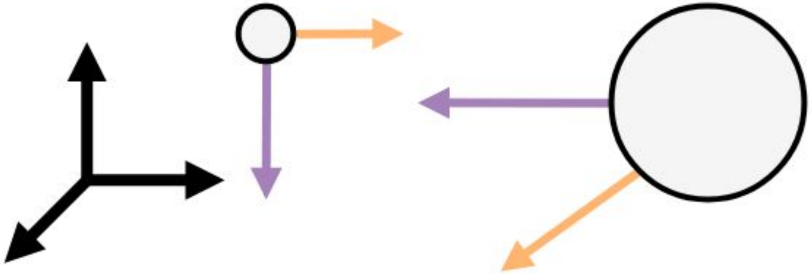
```
{0: [[[m0]],[[m1]]],
 1: [[[v0x, v0y, v0z],[a0x, a0y, a0z]],
     [[v1x, v1y, v1z],[a1x, a1y, a1z]]]}
```

`l`: dictionary key, l

[ ] point index, a

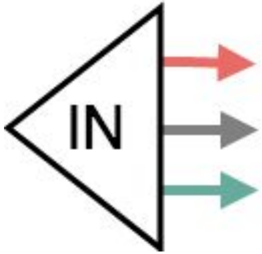
[ ] channel index, c

[ ] representation index, m



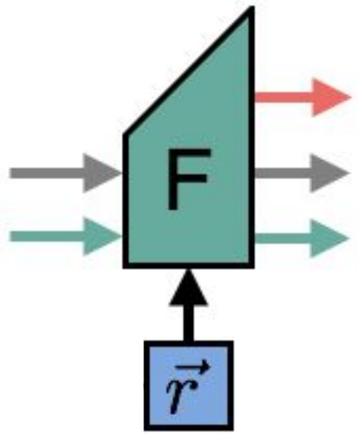
The input and output of our network is represented as tensors with **point** (or atom), **channel**, and **representation** indices organized by irreducible representation.

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



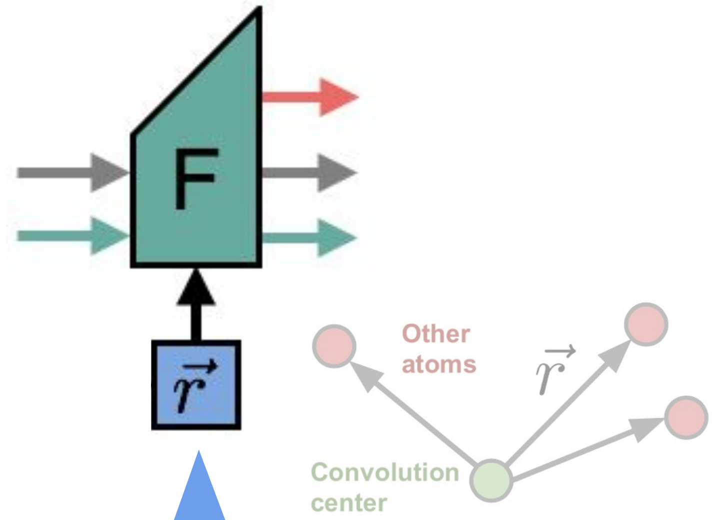
Filters contribute a **representation** index due to use of spherical harmonics.

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



Filters contribute a **representation** index due to use of spherical harmonics.

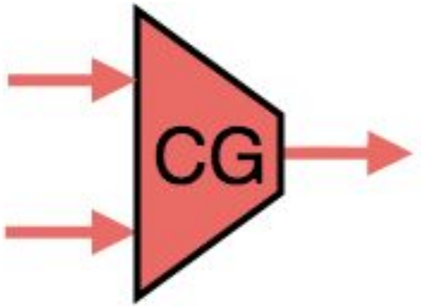
$$R(r)\boxed{Y_l^m(\hat{r})}$$



This is where the geometric information is used.

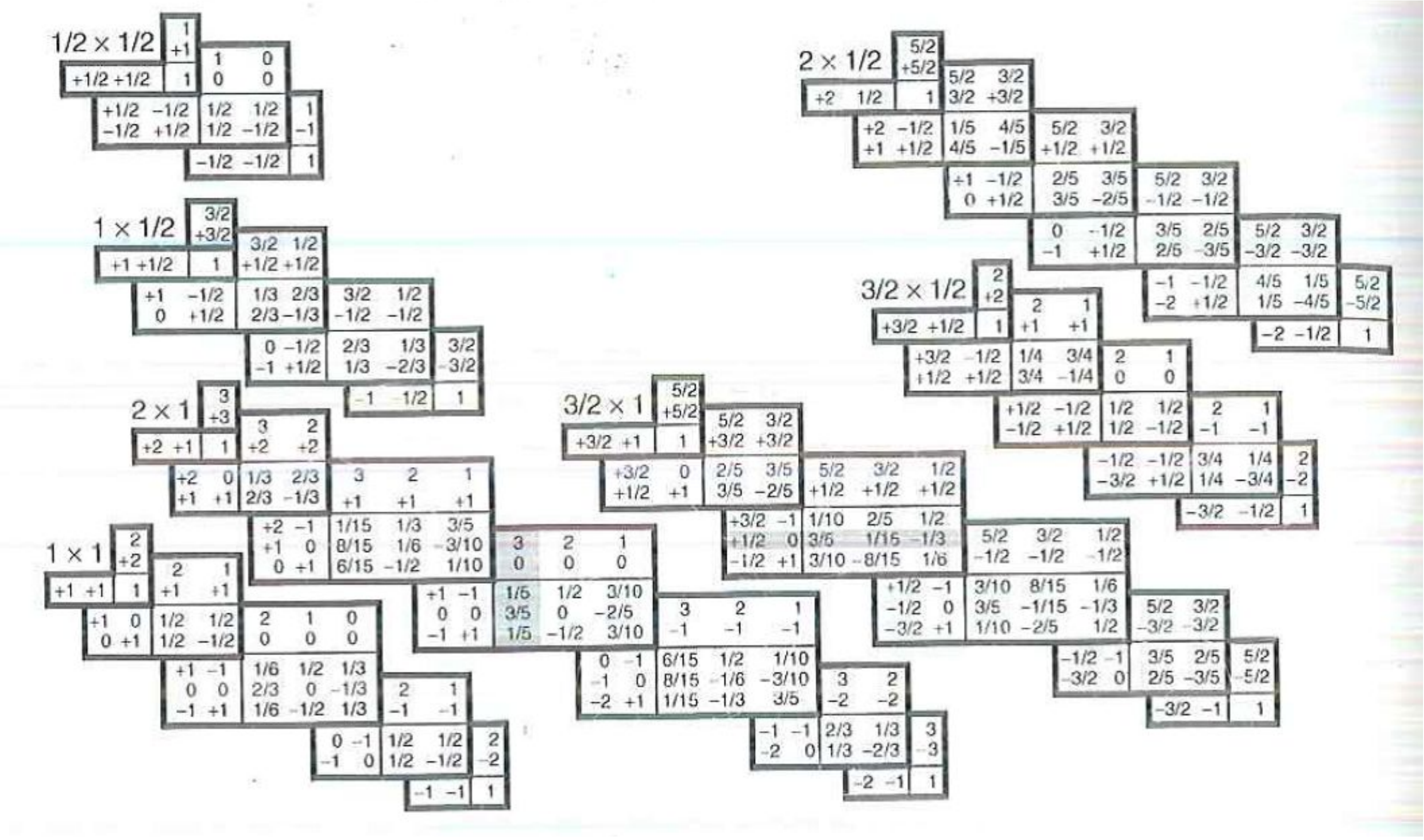
For images this is always the same (a grid). For points, this changes for every example.

To combine two tensors to create one tensor, we uses Clebsch-Gordan coefficients.



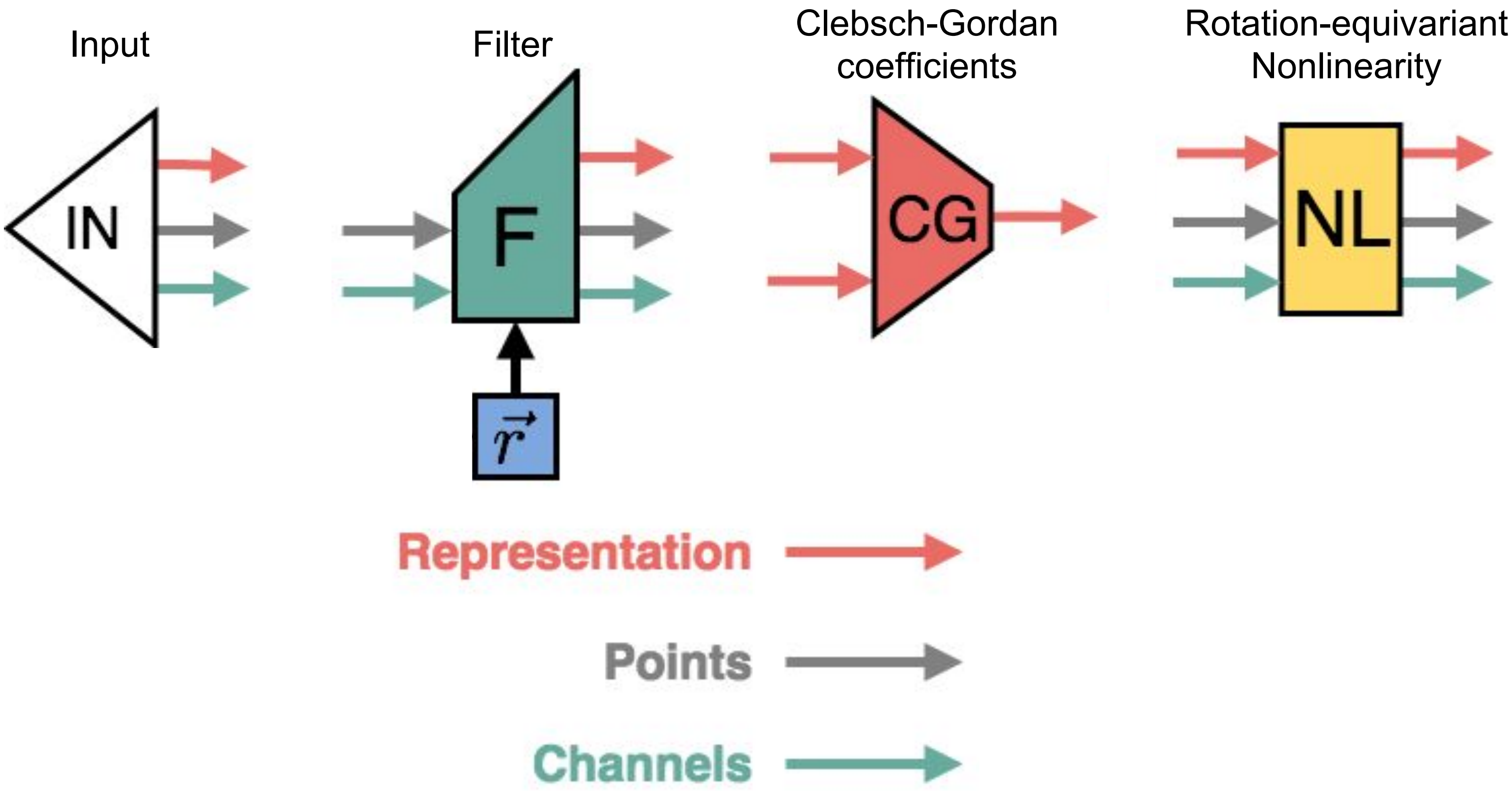
To combine two tensors to create one tensor, we uses Clebsch-Gordan coefficients.

Same math involved in the addition of angular momentum.

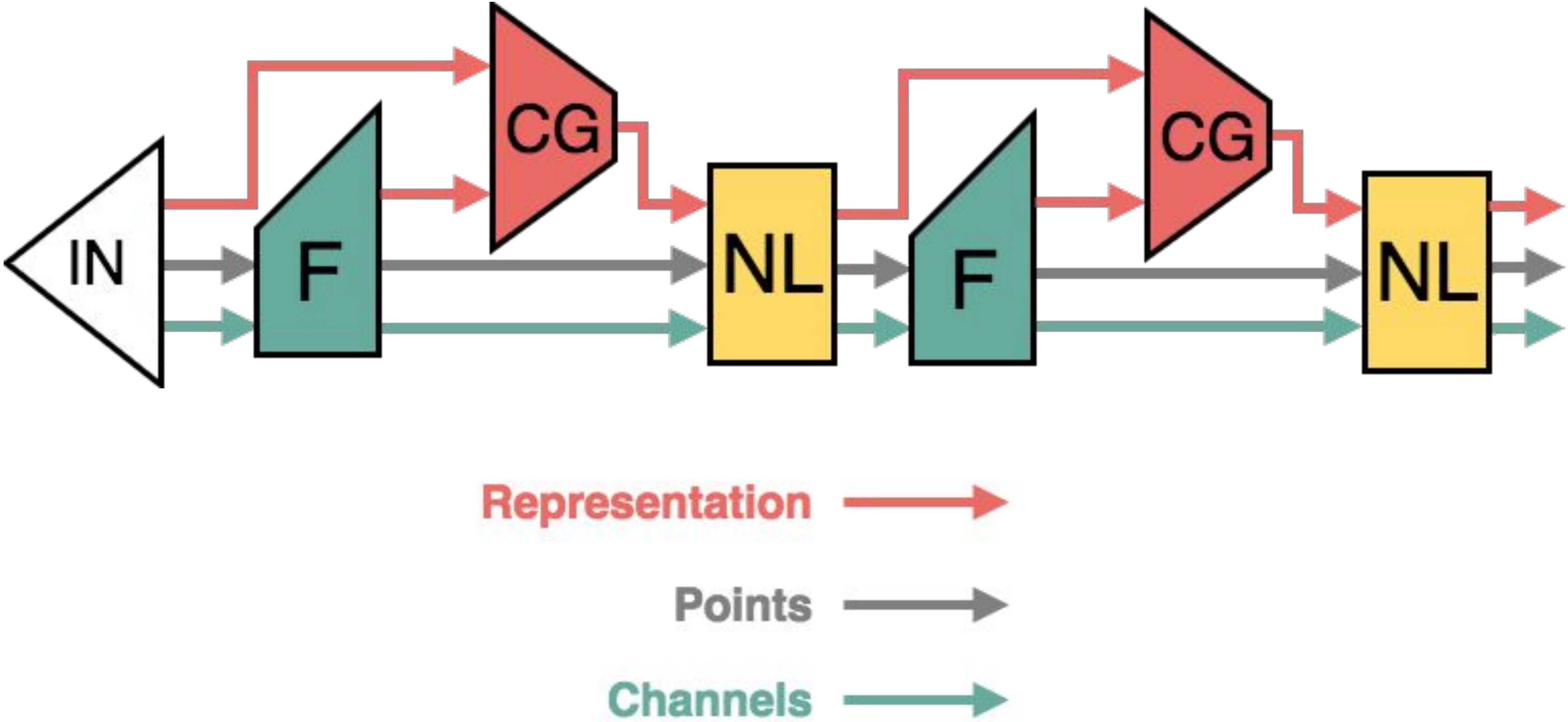


D. Griffiths, Introduction to quantum mechanics

These are components of **tensor field networks**



This is what a two-layer tensor field network looks like:



… and this is what one layer (without NL) looks like with all the indices written out:

$$
\begin{aligned}
&\mathcal{L}^{(l_O)}_{acm_O}\left(\vec{r}_a, V^{(l_I)}_{acm_I}\right) \\
&\quad := \sum_{m_F, m_I} \boxed{C^{(l_O, m_O)}_{(l_F, m_F)(l_I, m_I)}} \sum_{b \in S} \boxed{F^{(l_F, l_I)}_{cm_F}(\vec{r}_{ab})} \boxed{V^{(l_I)}_{bcm_I}}
\end{aligned}
$$

… and this is what one layer (without NL) looks like with all the indices written out:

$$\mathcal{L}^{(l_O)}_{acm_O}\left(\vec{r}_a, V^{(l_I)}_{acm_I}\right) := \sum_{m_F, m_I} \boxed{C^{(l_O, m_O)}_{(l_F, m_F)(l_I, m_I)}} \sum_{b \in S} \boxed{F^{(l_F, l_I)}_{cm_F}(\vec{r}_{ab})} \boxed{V^{(l_I)}_{bcm_I}}$$

$$\boxed{F^{(l_F, l_I)}_{cm_F}(\vec{r}_{ab}) = \boxed{R^{(l_F, l_I)}_{c}(r_{ab})} Y^{(l_F)}_{m_F}(\hat{r}_{ab})}$$

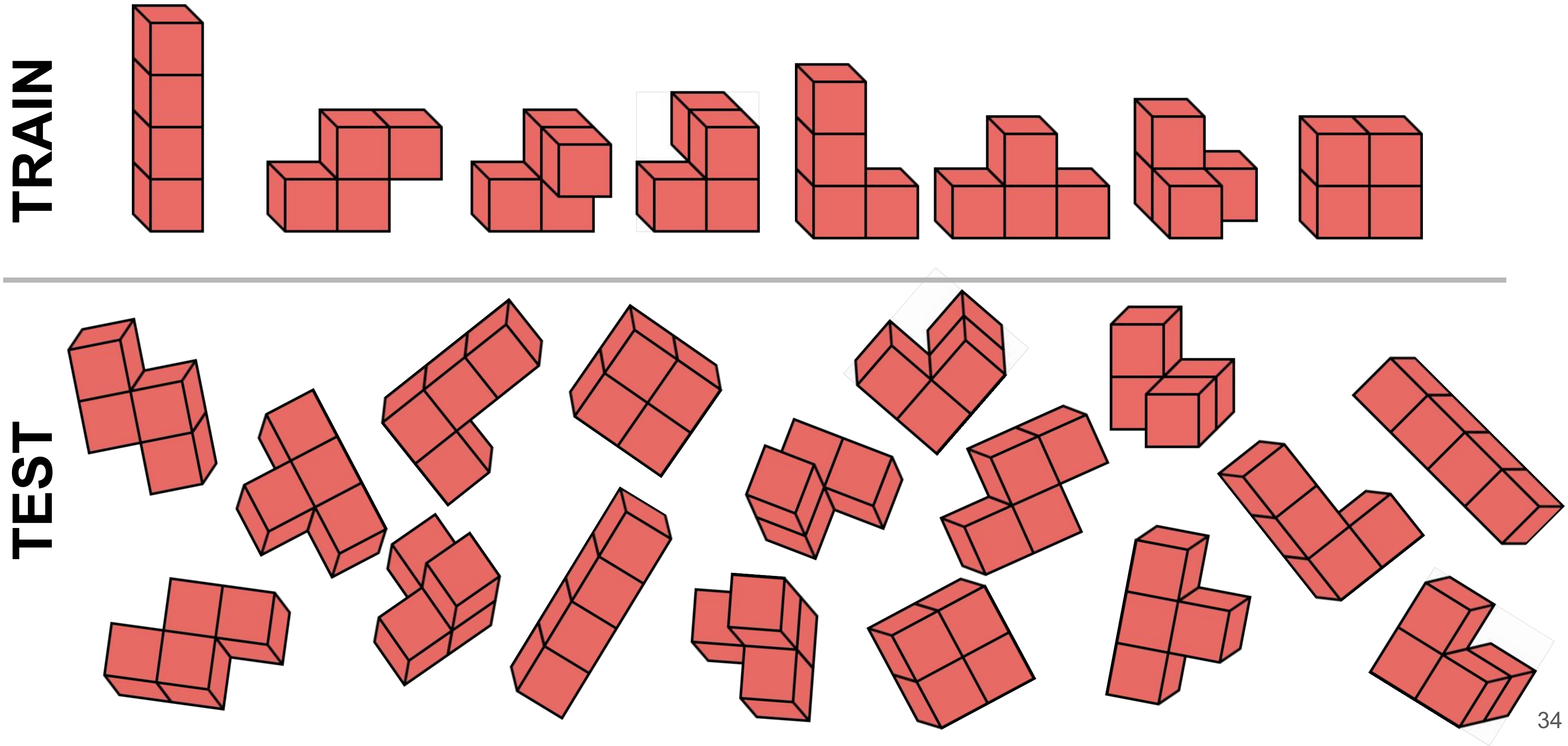
… and this is what one layer (without NL) looks like with all the indices written out:

$$\mathcal{L}^{(l_O)}_{acm_O}\left(\vec{r}_a, V^{(l_I)}_{acm_I}\right) := \sum_{m_F, m_I} \boxed{C^{(l_O, m_O)}_{(l_F, m_F)(l_I, m_I)}} \sum_{b \in S} \boxed{\boxed{F^{(l_F, l_I)}_{cm_F}(\vec{r}_{ab})} V^{(l_I)}_{bcm_I}}$$

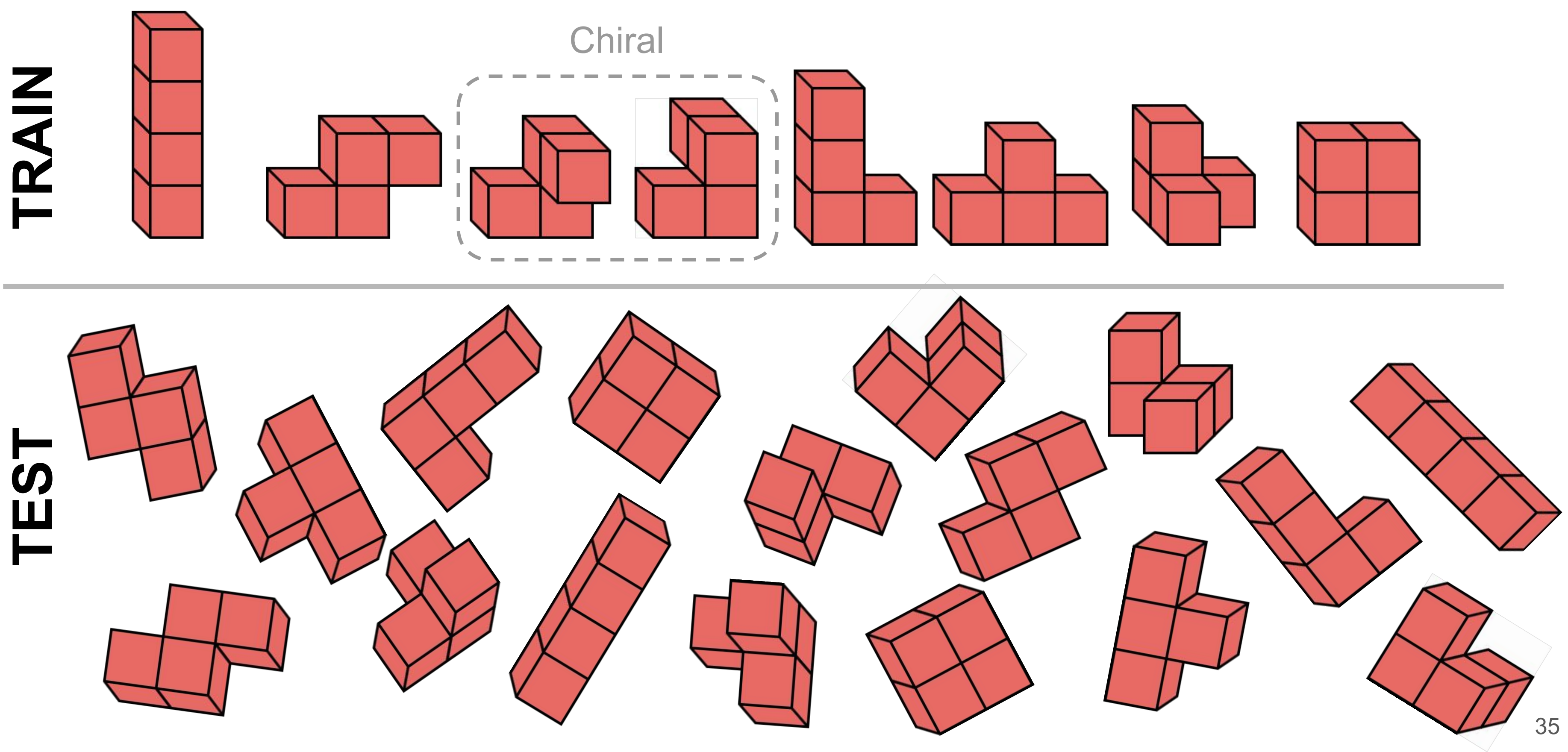
$$\boxed{F^{(l_F, l_I)}_{cm_F}(\vec{r}_{ab}) = \boxed{R^{(l_F, l_I)}_{c}(r_{ab})} Y^{(l_F)}_{m_F}(\hat{r}_{ab})}$$

$$\boxed{R^{(l_F, l_I)}_{c}(r_{ab}) = \sum_h \boxed{W_{ch}} \, \mathrm{NL}\Big(\sum_j \boxed{W_{hj}} B_j(r_{ab}) + \boxed{b_h}\Big) + \boxed{b_c}}$$

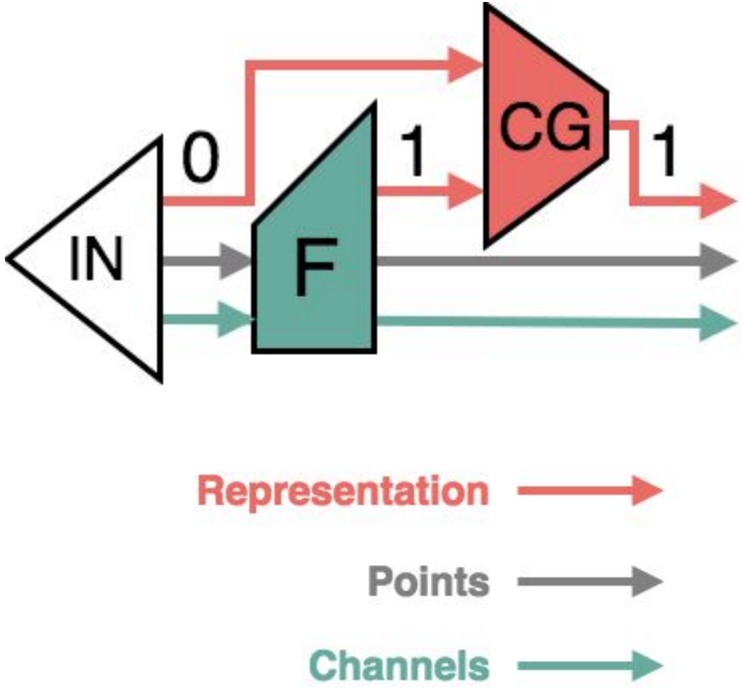
Test of 3D rotation equivariance: Trained on 3D Tetris shapes in one orientation, our network can perfectly identify these shapes in any orientation.

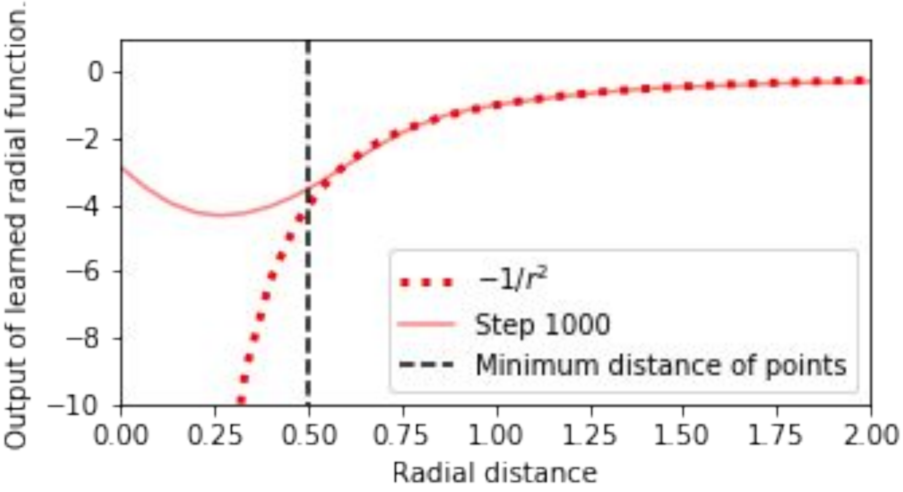


Test of 3D rotation equivariance: Trained on 3D Tetris shapes in one orientation, our network can perfectly identify these shapes in any orientation.

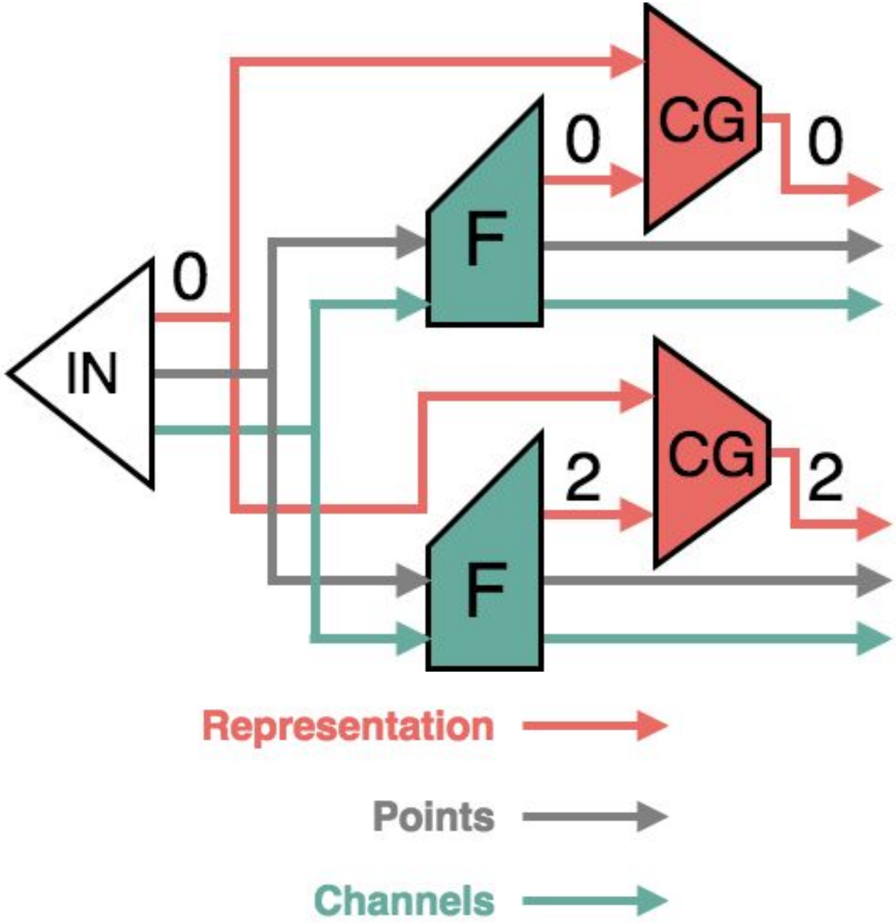


We can start with tensor input of any type and use filters to get tensor output of any type.
In this task, scalar masses are input and gravitational acceleration vectors are output.

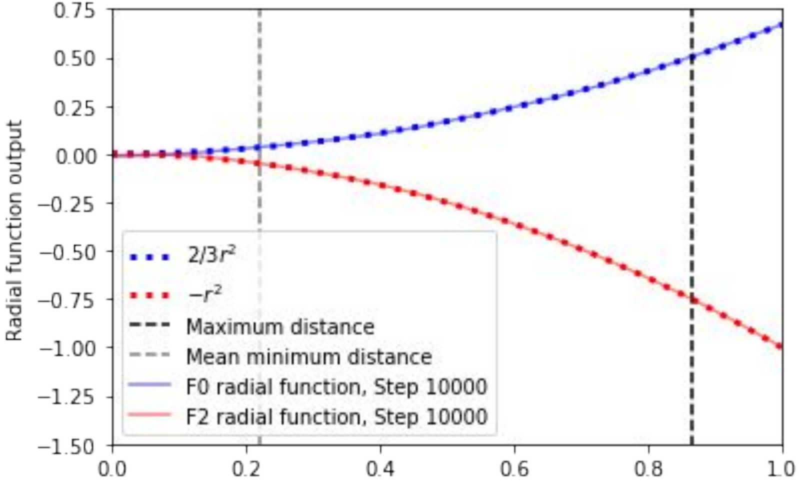




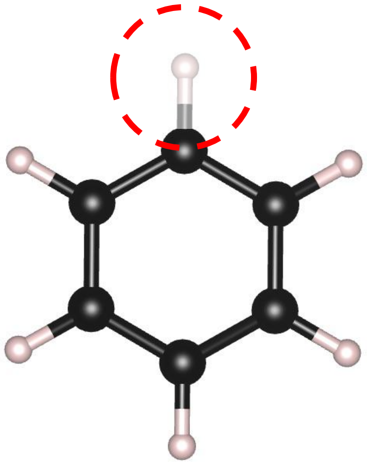
We can start with tensor input of any type and use filters to get tensor output of any type.
In this task, scalar masses are input and the moment of inertia tensor (a symmetric matrix) is output.



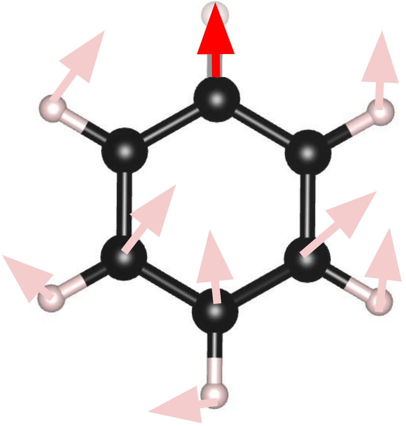
Moment of inertia:
0 (trace) + 2 (symmetric traceless)



Given a small organic molecule with an atom removed, replace the correct element at the correct location in space.



Input coordinates with missing atom.



Network outputs
(N-1) atom type features (scalars),
(N-1) displacement vectors, and
(N-1) scalars indicating confidence probability used for "voting".

DATASET
QM9: *http://www.quantum-machine.org/datasets/*
134k molecules with 9 or less heavy atoms (non-hydrogen) and elements H, C, N, O, F.

TRAIN
1,000 molecules with 5-18 atoms

TEST
1,000 molecules with 19 atoms
1,000 molecules with 23 atoms
1,000 molecules with 25-29 atoms

Atoms	Number of predictions	Accuracy (%) ($\leq$ 0.5 Å and atom type)	Distance MAE in Å
5-18 (train)	15 947	92.6	0.16
19	19 000	94.7	0.15
23	23 000	96.9	0.14
25-29	25 404	97.8	0.17

Learns to replace atoms with over 90% accuracy across train and test by seeing the same 1,000 molecules 200 times.

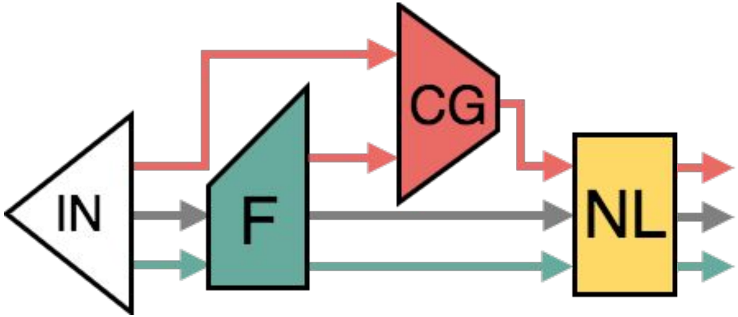
Conclusion

We've created a neural network architecture that operates on points and has the symmetries of 3D Euclidean space (3D translation- and 3D rotation-equivariance).

We use convolutional filters restricted to spherical harmonics with a learned radial function.

As a consequence of this choice of filter, the inputs, filters, and outputs of our network are geometric tensor fields.

We expect this network to be generally useful for tasks in geometry, physics and chemistry.



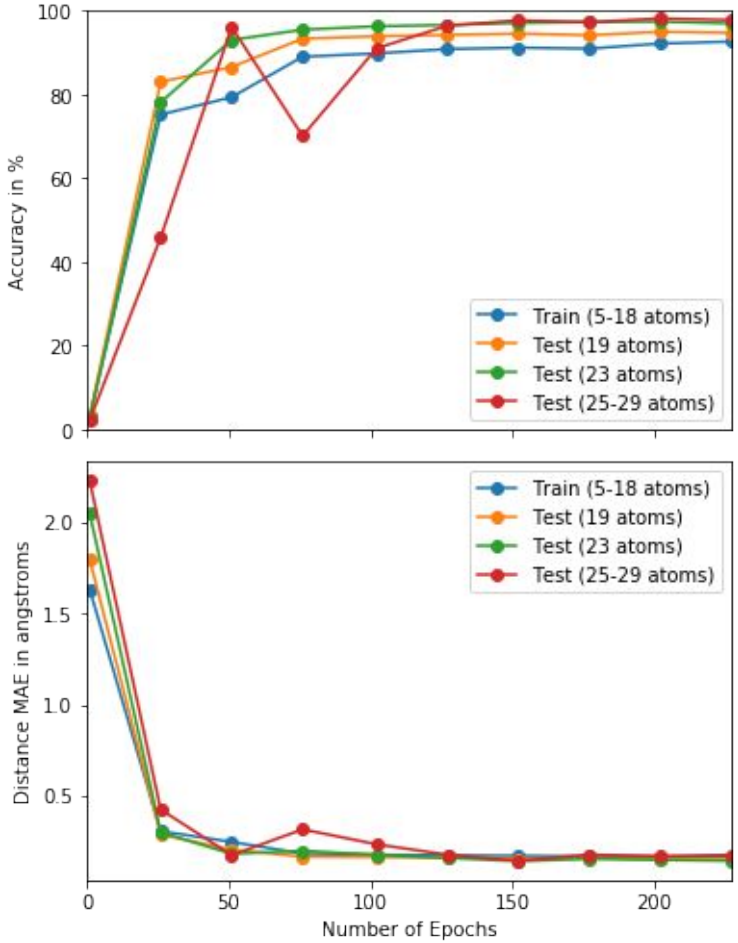
Paper:
arXiv:1802.08219

Code for paper:
https://github.com/tensorfieldnetworks/tensorfieldnetworks

Refactor coming soon:
https://github.com/mariogeiger/se3cnn

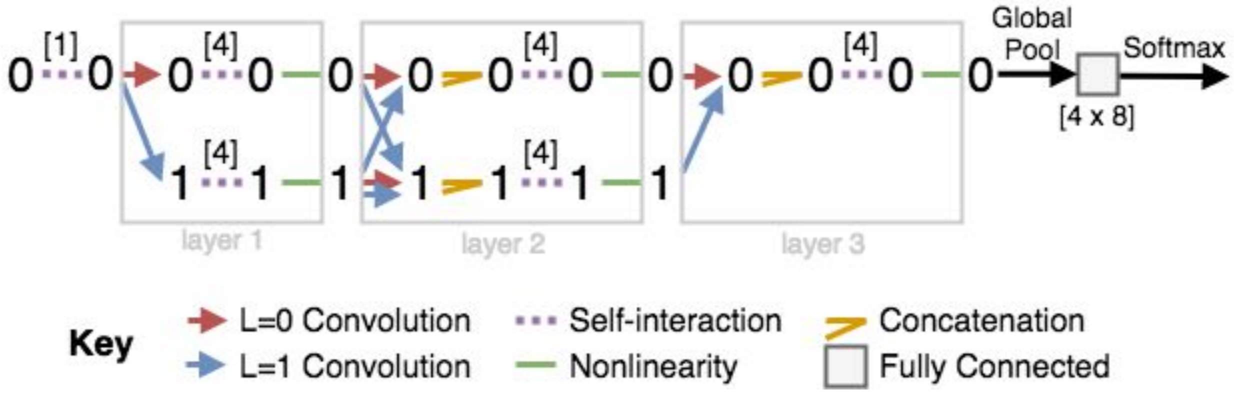
Calling in backup (slides)!

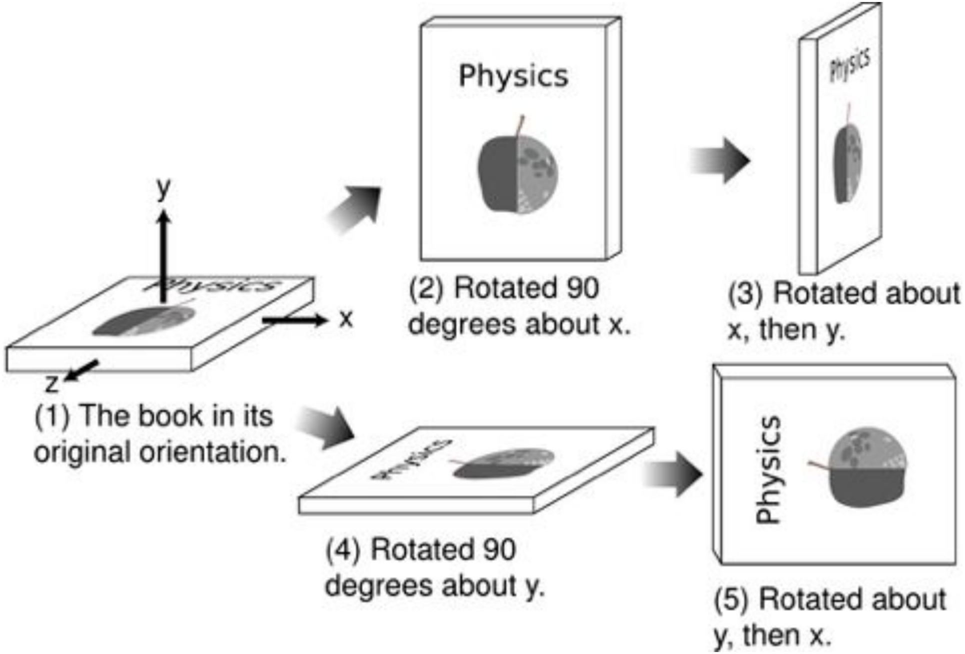
Missing point task results



Atoms	Number of atoms with given type in set	Accuracy (%) (≤ 0.5 Å and atom type)	Distance MAE in Å
Hydrogen			
5-18 (train)	7269	94.6	0.16
19	10 067	92.9	0.17
23	14 004	96.4	0.15
25-29	16 409	97.8	0.19
Carbon			
5-18 (train)	5713	94.6	0.16
19	6812	99.8	0.10
23	7851	99.9	0.11
25-29	8272	99.8	0.14
Nitrogen			
5-18 (train)	1426	84.2	0.16
19	599	86.3	0.19
23	48	91.7	0.19
25-29	17	58.8	0.20
Oxygen			
5-18 (train)	1498	85.7	0.16
19	1522	87.5	0.20
23	1097	82.9	0.21
25-29	706	73.1	0.21
Fluorine			
5-18 (train)	41	0.0	0.12
19	0		
23	0		
25-29	0		

3D Tetris classification network diagram

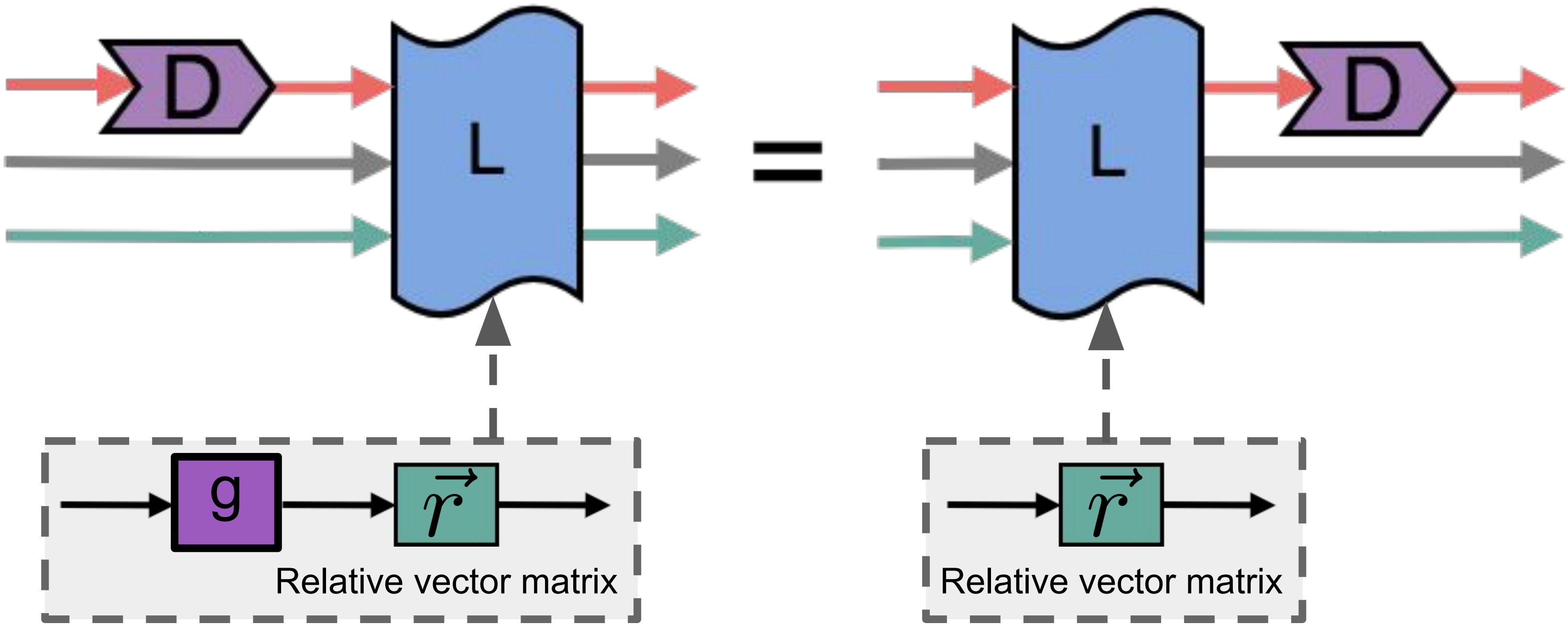




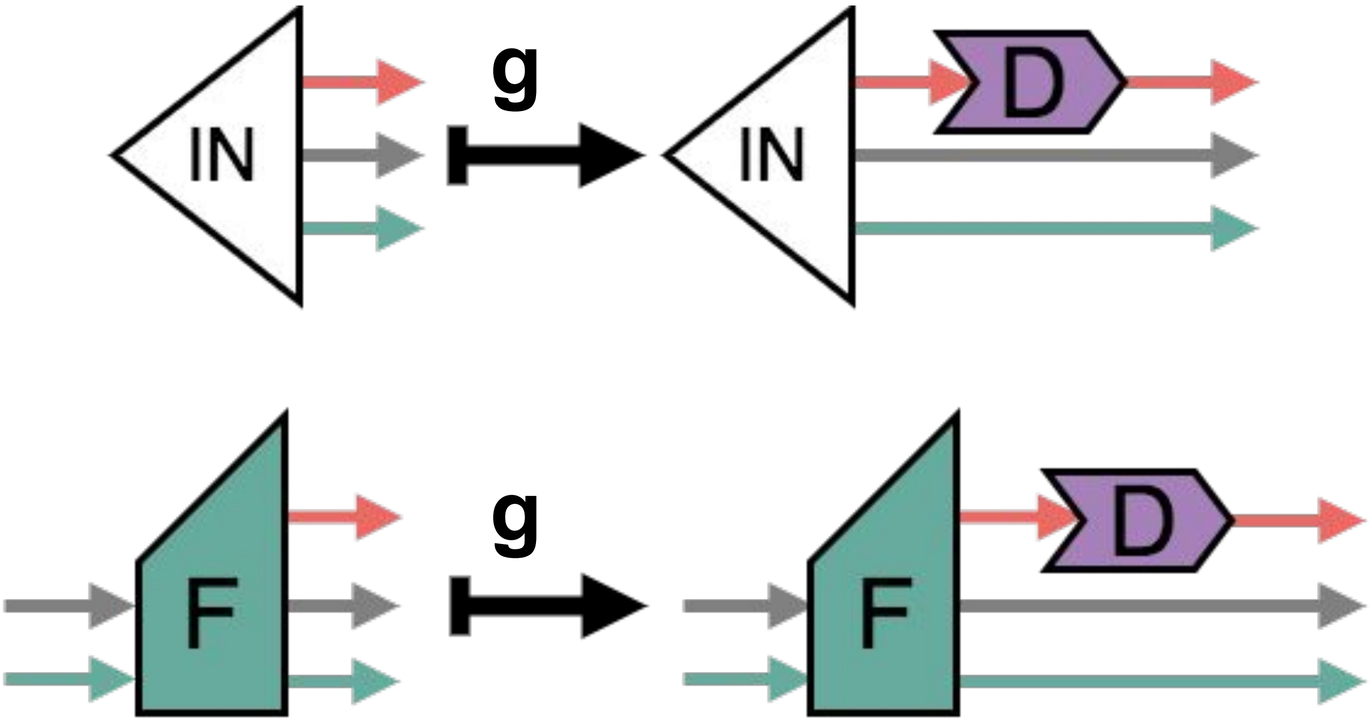
Benjamin Crowell, General Relativity, p. 256.

visual proof of 3d rotation equivariance

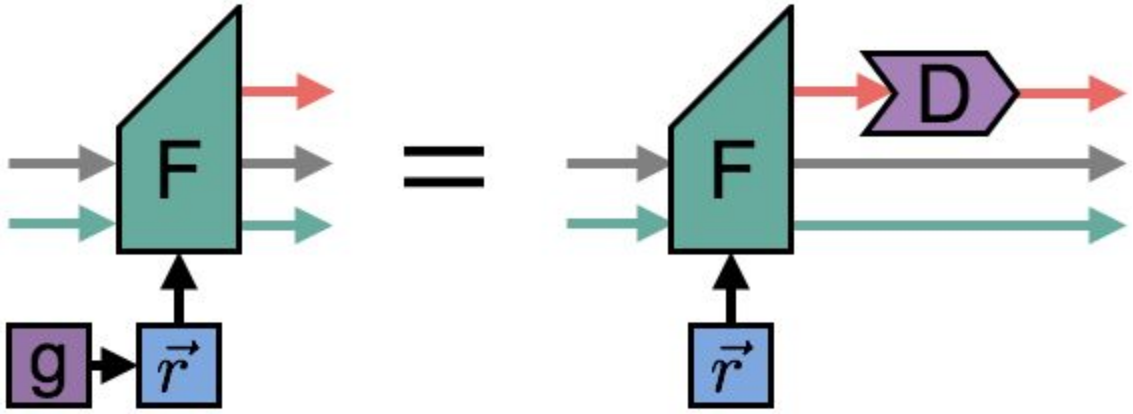
To be rotation-equivariant, the following relationship must hold for each layer:



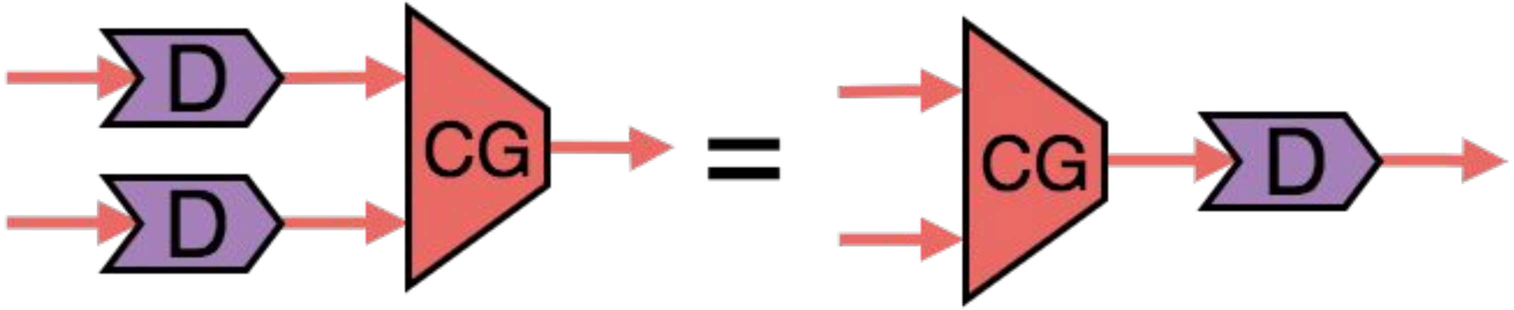
Each component of the network must be individually rotation-equivariant to guarantee the network rotation equivariance.



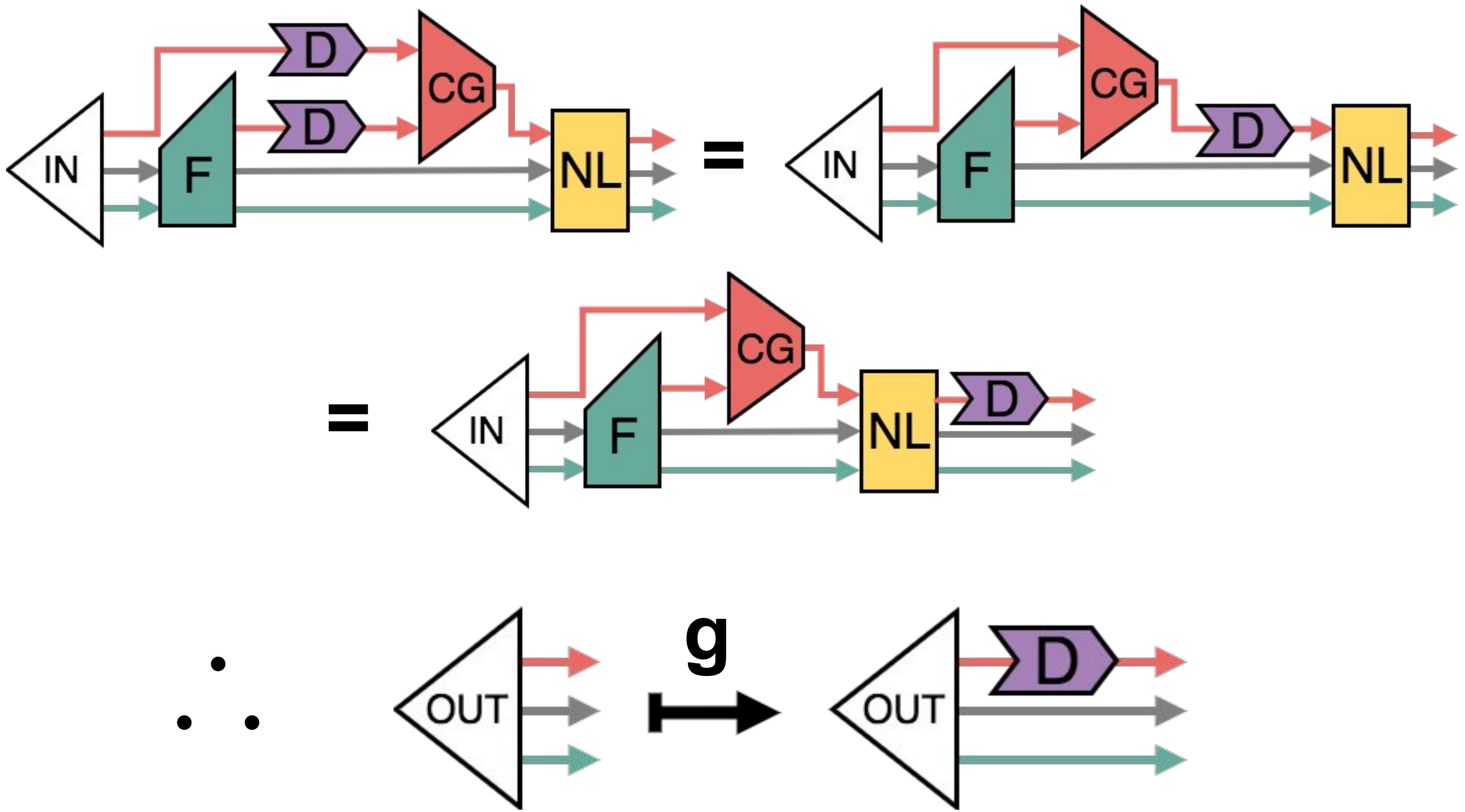
More explicitly...

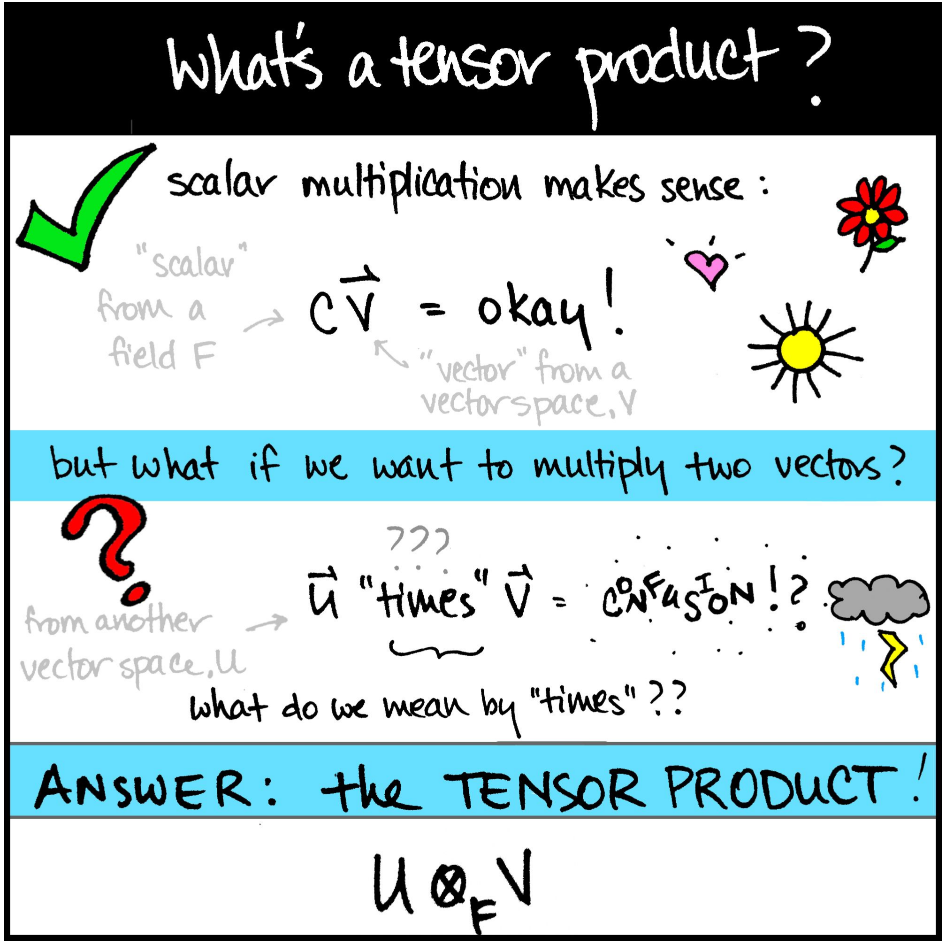


The key property of the Clebsch-Gordan coefficient tensor:



Putting it all together, we can show that the layers are equivariant.




What's a tensor product?
scalar multiplication makes sense:
"scalar" from a field F
cv⃗ = okay!
"vector" from a vectorspace, V
but what if we want to multiply two vectors?
???
u⃗ "times" v⃗ = CONFUSION!?
from another vector space, U
what do we mean by "times"??
ANSWER: the TENSOR PRODUCT!
U ⊗F V

Math3ma.com

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

Irreducible
Representations

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

$$1$$

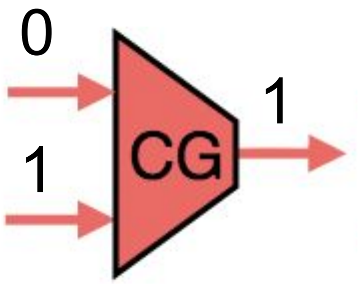
Examples of tensor algebra: How to combine two vectors? Many ways.

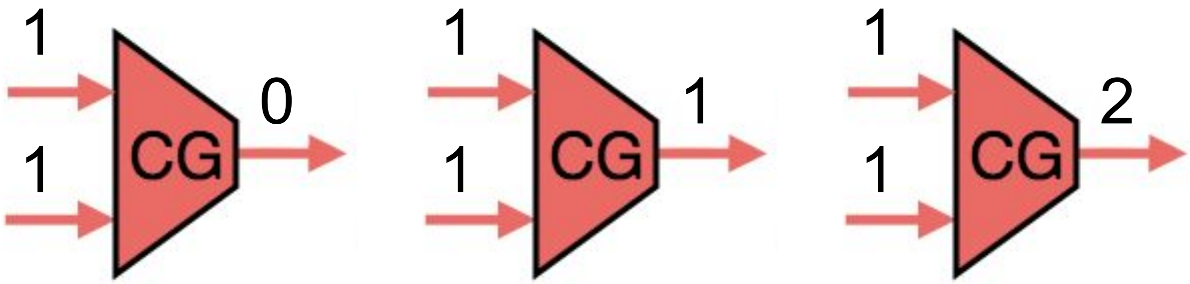
		Irreducible Representations
Dot product	$\begin{pmatrix} a_i & a_j & a_k \end{pmatrix} \begin{pmatrix} b_i \\ b_j \\ b_k \end{pmatrix} = c$	0
Cross product	$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_i & a_j & a_k \\ b_i & b_j & b_k \end{vmatrix} = \vec{c}$	1
Outer product	$\begin{pmatrix} a_i \\ a_j \\ a_k \end{pmatrix} \begin{pmatrix} b_i & b_j & b_k \end{pmatrix} = \begin{pmatrix} a_ib_i & a_ib_j & a_ib_k \\ a_jb_i & a_jb_j & a_jb_k \\ a_kb_i & a_kb_j & a_kb_k \end{pmatrix}$	$0 \oplus 1 \oplus 2$

To combine two tensors to create one tensor, we uses Clebsch-Gordan coefficients.

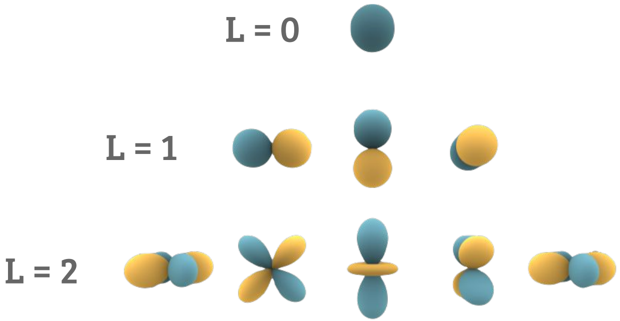
$$0 \otimes 1 = 1$$

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





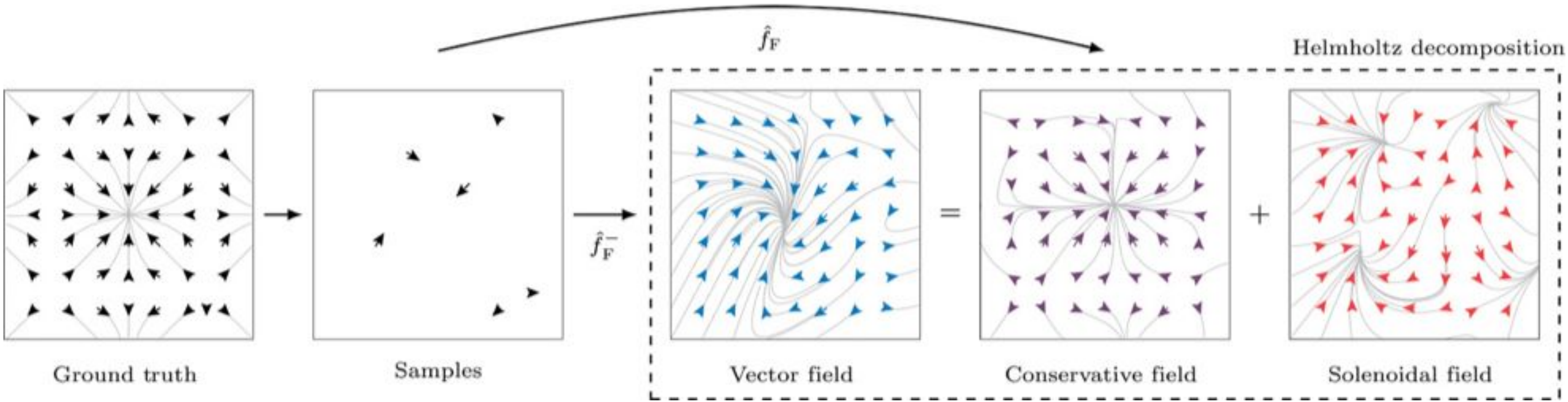
Our filter choice requires the structure of our network to be compatible with the algebra of **geometric tensors**.



$$\propto (x, y, z)$$

$$\propto (xy,\ yz,\ zx,\ 2z^2 - x^2 - y^2,\ x^2 - y^2)$$

(Smooth, decaying) Vector fields can be decomposed into a conservative field and a solenoid field. A conservative vector field can be expressed as the gradient of a scalar field and the solenoidal field as the curl of a vector field.



Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt and Klaus-Robert Müller,
Machine learning of accurate energy-conserving molecular force fields
Science Advances, Vol. 3, no. 5 e1603015 (2017)
DOI: 10.1126/sciadv.1603015