Tensor Field Networks

Rotation- and Translation-Equivarient Neural Networks for 3D Point Clouds

Ziqiao Lin, Nick Zhang, Luka Rogic

Feb 11, 2026

Outline

- **Part 1: Motivation**
  - Power of Equivariance in 3D
- **Part 2: Mathematical Foundations**
  - $SO(3)$ Rotation & Wigner D-matrices
  - Feature Hierarchy ($l = 0, 1, 2$)
- **Part 3: The TFN Architecture**
  - Point Convolution
  - Self-Interaction
  - Nonlinearity
- **Part 4: Experiments and Conclusion**

EQUIVARIANCE

Translation **SO(3)**

TFN

Permutation **Irreps**

Clebsch-Gordan

Symmetry by Design

The Power of Equivariance in 3D

- **Efficiency vs. Data Augmentation**:
  - To achieve angular resolution δ in 3D, traditional CNNs require $O(\delta^{-3})$ more filters.
  - TFNs use equivariant filters to cover all orientations with a **single set of weights**.
- **Interpretation**: The network identifies the same local features (e.g., molecular functional groups) regardless of their orientation.
- **Physical Consistency**: Naturally encodes **Geometric Tensors** (Scalars, Vectors) that transform according to physical laws.

Dimension	Rotational Degrees of Freedom	Filter Scaling Cost
2D Plane	1 (θ)	$O(\delta^{-1})$
3D Space	3 (α, β, γ)	$O(\delta^{-3})$

Examples of Tensor Fields

$l = 0$: Scalars

Invariant to Rotation

- Mass density
- Charge at a point
- Temperature

$l = 1$: Vectors

Equivariant (3D Arrow)

- Force vectors
- Velocity of atoms
- Electric fields

$l = 2$: Tensors

Higher-order Geometry

- Stress/Strain
- Polarizability
- Inertia tensors

TFN maintains the "Geometric Identity" of these physical quantities across all layers.

Related Work & Mathematical Motivation

Foundational Pillars

- **Harmonic Networks [1]**: Achieved **2D rotation equivariance** using circular harmonics and discrete convolutions.
- **SchNet [2]**: A **rotation-invariant** network for molecules using continuous convolutions (but lacks equivariance).
- **TFN Contribution**: Emulates and extends both to achieve full **3D equivariance**.

The 2D Case (Simpler)

- Rotations **commute** ($AB = BA$).
- Symmetry is additive (like adding angles).

The 3D Challenge (Complex)

- Rotations **do not commute** ($AB \neq BA$).
- Order of operations matters.
- Requires the sophisticated group theory of $SO(3)$.

Equivariance and Representation

1. Group Representation

A **representation** D of a group G is a function from G to square matrices such that for all $g, h \in G$:

$$D(g)D(h) = D(gh) \tag{1}$$

Interpretation: Matrix multiplication preserves the algebraic structure of group operations.

2. Equivariance and Invariance

A function $\mathcal{L} : X \to Y$ is **equivariant** with respect to G if for all $g \in G$:

$$\mathcal{L} \circ D_X(g) = D_Y(g) \circ \mathcal{L} \tag{2}$$

- **Invariance**: A special case where $D_Y(g) = \mathbb{I}$ (the identity matrix) for all g.
- **Scope**: We focus on **3D Isometries** (rotations/translations) and **Point Permutations**.

Proof of Compositional Equivariance

The Theorem

If $\mathcal{L}_1 : X \to H$ and $\mathcal{L}_2 : H \to Y$ are both equivariant layers, then their composition $\mathcal{L}_{total} = \mathcal{L}_2 \circ \mathcal{L}_1$ is also equivariant.

Step-by-Step Derivation

Let $g \in G$. We apply the rotated input $D_X(g)x$ to the composed network:

$$
\begin{aligned}
(\mathcal{L}_2 \circ \mathcal{L}_1)(D_X(g)x) &= \mathcal{L}_2\left(\mathcal{L}_1(D_X(g)x)\right) && \text{(Definition of composition)} && (3)\\
&= \mathcal{L}_2\left(D_H(g)\mathcal{L}_1(x)\right) && \text{(By equivariance of } \mathcal{L}_1) && (4)\\
&= D_Y(g)\mathcal{L}_2(\mathcal{L}_1(x)) && \text{(By equivariance of } \mathcal{L}_2) && (5)\\
&= D_Y(g)(\mathcal{L}_2 \circ \mathcal{L}_1)(x) && \text{(Definition of composition)} && (6)
\end{aligned}
$$

- **Conclusion**: Equivariance for each layer is sufficient to prove whole network is equivariance.

Input and Output of a TFN Layer

A TFN layer $\mathcal{L}$ transforms a finite set of points S in 3D space, mapping input features to output features while maintaining the point coordinates:

$$\mathcal{L} : (\vec{r}_a, x_a) \mapsto (\vec{r}_a, y_a) \tag{7}$$

- **Coordinates** ($\vec{r}_a \in \mathbb{R}^3$): Fixed position vectors used to compute relative geometry $\vec{r}_{ab} = \vec{r}_a - \vec{r}_b$.
- **Features** ($x_a \in X, y_a \in Y$): Geometric tensors (scalars, vectors, etc.) that transform under $SO(3)$.
- **Structure** ($\mathbb{R}^3 \oplus X$): A concatenation of spatial position vectors and their associated geometric properties.

---

*The network updates the **states** ($x \to y$) of the points based on their **spatial relationships**.*

Manifest Equivariance: Permutation & Translation

1. Permutation Equivariance

Condition: $\mathcal{L} \circ P_\sigma = P_\sigma \circ \mathcal{L}$

- **Definition**: $P_\sigma(\vec{r}_a, x_a) := (\vec{r}_{\sigma(a)}, x_{\sigma(a)})$ where σ permutes the indices.
- **Manifested by**: Treating point clouds as **sets** rather than ordered lists.
- **Mechanism**: Information is aggregated via symmetric operations (e.g., summation) that are independent of the storage order of points.

---

2. Translation Equivariance

Condition: $\mathcal{L} \circ T_t = T_t \circ \mathcal{L}$

- **Definition**: $T_t(\vec{r}_a, x_a) := (\vec{r}_a + \vec{t}, x_a)$ for any translation vector $\vec{t}$.
- **Manifested by**: Filters only depend on the **relative displacement**: $\vec{r}_i - \vec{r}_j$.
- **Mechanism**: Since $(\vec{r}_i + \vec{t}) - (\vec{r}_j + \vec{t}) = \vec{r}_i - \vec{r}_j$, the geometry remains invariant under global shifts.

The Rotation Group $SO(3)$

The **Special Orthogonal Group** $SO(3)$ represents all valid 3D rotations about the origin.

1. Mathematical Definition

$$SO(3) = \{R \in \mathbb{R}^{3\times 3} \mid R^T R = \mathbb{I}, \det(R) = 1\} \tag{8}$$

- **Orthogonality**: Preserves lengths and angles (No stretching/squeezing).
- **Special**: Preserves orientation ($\det = 1$ ensures no reflections).

2. Basic Rotation Matrices

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{9}$$

*Key Property: **Non-commutative** ($R_x R_y \neq R_y R_x$). Order matters in 3D!*

Rotation Equivariance in $SO(3)$

For a layer $\mathcal{L}$ to be rotation-equivariant, it must commute with the action of the rotation group $SO(3)$.

1. Equivariance Condition

$$\mathcal{L} \circ [R(g) \oplus D_X(g)] = [R(g) \oplus D_Y(g)] \circ \mathcal{L} \tag{10}$$

- $R(g)$: Standard 3×3 matrix acting on coordinates $\vec{r}_a$.
- $D(g)$: Wigner D-matrix acting on feature vectors x_a.

2. Wigner D-Matrices The "bridge" between $SO(3)$ and higher-order geometric features:

- Maps a rotation g to a $(2l+1) \times (2l+1)$ matrix.
- **Unitary & Irreducible**: They are the fundamental building blocks of 3D rotations.

Rotation Orders and Feature Types

TFN decomposes representations into **Irreducible Representations (irreps)** of order l.

Order l	Dim	Geometric Type	Example
$l = 0$	1	**Scalars**	Mass, Charge, Energy
$l = 1$	3	**Vectors**	Velocity, Force, Displacement
$l = 2$	5	**Rank-2 Tensors**	Stress, Quadrupole moment

Transformation Rules:

- **Order 0**: $D^{(0)}(g) = 1$ (Invariant under rotation).
- **Order 1**: $D^{(1)}(g) = R(g)$ (Standard 3D rotation).
- **Higher Orders**: Follow Wigner D-matrix $D^{(l)}(g)$ of size $(2l + 1)$.

*TFN treats a point cloud as a collection of **Geometric Tensors** of mixed orders.*

Tensor Field Network Layers

What does the network need to do?

Inputs: 3D points, features associated with each point
Outputs: Processed features/representations for these points

We also want each layer to be $SO(3)$-equivariant.

What are the features?

- To have $SO(3)$-equivariance, we eventually want to apply rotations to the feature vectors.
- The irreducible representations of $SO(3)$ can only have dimensions $2\ell + 1$ for $\ell \in \mathbb{N}$.
- So we want the feature space to have dimensions $2\ell + 1$ for $\ell \in \mathbb{N}$, being the space the a representation $D^{(\ell)}(g)$ acts on.
- A point can have many feature spaces with different dimensions.

A feature vector in the feature space is called a **tensor**.

Examples:

- $\ell = 0$: mass, charge
- $\ell = 1$: velocity, acceleration

What are the features? – Implementation

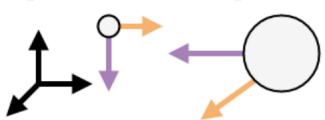
We have a finite $S \subseteq \mathbb{R}^3$. The features are implemented as a *dictionary* V.

- The *keys* are the ℓ's, representing feature spaces with different dims.
- The *values* are arrays of shape $[|S|, C_\ell, 2\ell + 1]$, where:
  - $2\ell + 1$: dimension
  - C_ℓ: number of feature types that has dim $2\ell + 1$, e.g. velocity and acceleration are two types. Also called the number of **channels**.

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

```
{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



Point Convolution Layers

Convolution Filters

Goal: We will build a *convolutional* layer, so we need a *filter/kernel*.

We use **spherical harmonics** $\left(Y_m^{(\ell)}\right)_{\ell\in\mathbb{N},\,-\ell\leq m\leq\ell}$:

- Each $Y_m^{(\ell)}$ is a function from the sphere S^2 to $\mathbb{R}$ or $\mathbb{C}$.
- They form a complete and orthonormal basis for the space of functions on S^2.
- They are $SO(3)$-equivariant!

$SO(3)$-Equivariance of Spherical Harmonics

Fix $\ell \in \mathbb{N}$, $g \in SO(3)$, and $\hat{r} \in S^2$. We can write

$$Y^{(\ell)}(\hat{r}) = \begin{bmatrix} Y^{(\ell)}_{-\ell}(\hat{r}) \\ Y^{(\ell)}_{-\ell+1}(\hat{r}) \\ \vdots \\ Y^{(\ell)}_{\ell}(\hat{r}) \end{bmatrix}$$

and view it as a $(2\ell+1)$-dim *tensor*. The equivariance property is

Proposition. [Equivariance of the Spherical Harmonics]

$$Y^{(\ell)}\big(\mathcal{R}(g)\hat{r}\big) = D^{(\ell)}(g) Y^{(\ell)}(\hat{r}).$$

Convolution Filters

For each input order ℓ_i, filter order ℓ_f, and channel c, define the filter to be

$$F_c^{(\ell_f,\ell_i)}(r;\boldsymbol{\theta}) = R_c^{(\ell_f,\ell_i)}\big(\|r\|;\boldsymbol{\theta}\big) \cdot Y^{(\ell_f)}\left(\frac{r}{\|r\|}\right),$$

where:

- $r \in \mathbb{R}^3$;
- $R_c^{(\ell_f,\ell_i)}\big(\cdot;\boldsymbol{\theta}\big) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ are learned functions with learnable parameters $\boldsymbol{\theta}$.

The filter is still $SO(3)$-equivariant!

The "Product" in Convolution

After constructing the filter, we can write out the convolution!

We want something like

$$\sum_{b \in S} F_c^{(\ell_f)}(r_a - r_b) \cdot V_{b,c}^{(\ell_i)}.$$

(Abuse of notation: When we write $a, b \in S$, a and b are the indices of the points and not the actual points. The actual points are denoted r_a and r_b.)

But we are "multiplying" two *tensors* here, which we have not defined yet!

Tensor Product

Definition. [Tensor Product]

Let u be an order-ℓ_1 tensor and let v be an order-ℓ_2 tensor. The **tensor product** of u and v can be defined for any output order ℓ between $|\ell_1 - \ell_2|$ and $(\ell_1 + \ell_2)$ inclusive, and is denoted as $(u \otimes v)^{(\ell)}$. In particular, we have

$$(u \otimes v)^{(\ell)}_m = \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} C^{(\ell,m)}_{(\ell_1,m_1),(\ell_2,m_2)} u^{(\ell_1)}_{m_1} v^{(\ell_2)}_{m_2}$$

for $m \in \{-\ell, \ldots, \ell\}$, where the C's are constants called the *Clebsch-Gordan* coefficients. We write $\ell_1 \otimes \ell_2 \to \ell$ for a tensor product with output order ℓ.

Tensor Product – Examples

We first consider an example of $1 \otimes 1 \to 0$.
The CG-coefficient satisfies $C^{(0,0)}_{(1,m_1),(1,m_2)} = \delta_{m_1 m_2}/\sqrt{3}$.

$$\begin{aligned}\mathbb{R} \ni (u \otimes v)^{(0)} &= \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} C^{(0,0)}_{(1,m_1),(1,m_2)} u^{(1)}_{m_1} v^{(1)}_{m_2} \\ &= \frac{1}{\sqrt{3}} \left(u_{-1}v_{-1} + u_0 v_0 + u_1 v_1\right).\end{aligned}$$

This gives the *dot product*!

One can show that $1 \otimes 1 \to 1$ is the usual *cross product*.

$SO(3)$-Equivariant of the Tensor Product

Tensor products are also $SO(3)$-equivariant!

Proposition. [Equivariance of the Tensor Product]

Let u be an order-ℓ_1 tensor and let v be an order-ℓ_2 tensor. Let $g \in SO(3)$. Let ℓ be the output order. Then

$$\Big(\big(D^{(\ell_1)}(g)u\big) \otimes \big(D^{(\ell_2)}(g)v\big)\Big)^{(\ell)} = D^{(\ell)}(g)(u \otimes v)^{(\ell)}.$$

$$\texttt{tensor\_product}\big(\texttt{rotate}(u), \texttt{rotate}(v)\big) = \texttt{rotate}\big(\texttt{tensor\_product}(u, v)\big)$$

Point Convolution

We are finally ready to define the **point convolution layer**.

Definition. [Point Convolution]

Let $a \in S$, and let c be a channel. Let ℓ_i be the input order, ℓ_f be the filter order, and ℓ_o be the output order. Then the **point convolution** centered at a is

$$\mathcal{L}_{a,c}^{(\ell_o)}\left(r_a, V_{a,c}^{(\ell_i)}; \boldsymbol{\theta}\right) := \sum_{b \in S} \left(F_c^{(\ell_f, \ell_i)}(r_a - r_b; \boldsymbol{\theta}) \otimes V_{b,c}^{(\ell_i)} \right)^{(\ell_o)},$$

which is a tensor of order ℓ_o.

$SO(3)$-Equivariance of Point Convolution

Theorem. [Equivariance of Point Convolution]

The point convolution layer is $SO(3)$-equivariant. That is, for $g \in SO(3)$, we have

$$\sum_{b \in S} \left(F_c^{(\ell_f, \ell_i)}(\mathcal{R}(g) r_{ab}) \otimes \left[D^{(\ell_i)}(g) V_{b,c}^{(\ell_i)} \right] \right)^{(\ell_o)} = D^{(\ell_f)}(g)\, \mathcal{L}_{a,c}^{(\ell_o)}\left(r_a, V_{a,c}^{(\ell_i)}\right),$$

where $r_{ab} := r_a - r_b$. Note the the LHS represents point convolution on the rotated point cloud.

$SO(3)$-Equivariance of Point Convolution Proof

Proof.

$$
\begin{aligned}
&\sum_{b\in S}\left(F_c^{(\ell_f,\ell_i)}(\mathcal{R}(g)r_{ab})\otimes\left[D^{(\ell_i)}(g)V_{b,c}^{(\ell_i)}\right]\right)^{(\ell_o)}\\
&=\sum_{b\in S}\left(\left[D^{(\ell_f)}(g)F_c^{(\ell_f,\ell_i)}(r_{ab})\right]\otimes\left[D^{(\ell_i)}(g)V_{b,c}^{(\ell_i)}\right]\right)^{(\ell_o)} \quad \text{by the eq. of filters}\\
&=\sum_{b\in S}D^{(\ell_o)}(g)\left(F_c^{(\ell_f,\ell_i)}(r_{ab})\otimes V_{b,c}^{(\ell_i)}\right)^{(\ell_o)} \quad \text{by the eq. of tensor products}\\
&=D^{(\ell_o)}(g)\sum_{b\in S}\left(F_c^{(\ell_f,\ell_i)}(r_{ab})\otimes V_{b,c}^{(\ell_i)}\right)^{(\ell_o)}\\
&=D^{(\ell_o)}(g)\,\mathcal{L}_{a,c}^{(\ell_o)}(r_a,V_{a,c}^{(\ell_i)}).
\end{aligned}
$$

□

Self-Interaction and Nonlinearity Layers

Self-Interaction

In point convolution, information is mixed *spatially between different points*.

In self-interaction, information is mixed *internally within each point*.

Definition. [Self-Interaction]

The **self-interaction** layer is defined by:

- **Learnable parameters:** A weight matrix $W^{(\ell)}$ with size $C_{\text{out}} \times C_{\text{in}}$ for each $\ell \in \mathbb{N}$, shared between all points.
- **Input:** Features of a point $V_a^{(\ell)}$ with size $C_{\text{in}} \times (2\ell + 1)$.
- **Output:**

$$W^{(\ell)} V_a^{(\ell)}.$$

$SO(3)$-Equivariance of Self-Interaction

Let $g \in SO(3)$. Note that rotating the input and then applying self-interaction corresponds to

$$W^{(\ell)} \cdot \left(V_a^{(\ell)} \left(D^{(\ell)}(g) \right)^\top \right).$$

Applying self-interaction first and then rotating the output corresponds to

$$\left(W^{(\ell)} V_a^{(\ell)} \right) \cdot \left(D^{(\ell)}(g) \right)^\top.$$

They are the same by associativity.

Nonlinearity

Standard nonlinearities like ReLU breaks $SO(3)$-equivariance.

We want nonlinearities that only act on scalars (e.g. magnitudes) and leave the orientation untouched.

Definition. [Nonlinearity]

For each $\ell \in \mathbb{N}$ and channel c, the nonlinearity layer is defined by:

- **Learnable parameters:** The bias term $b_c^{(\ell)}$.
- **Input:** Features of a point at the channel $V_{a,c}^{(\ell)}$.
- **Output:**

$$\eta^{(\ell)}\left(\|V_{a,c}^{(\ell)}\| + b_c^{(\ell)}\right) V_{a,c}^{(\ell)},$$

where $\eta^{(\ell)} : \mathbb{R} \to \mathbb{R}$ can be any standard nonlinearity.

$SO(3)$-Equivariance of the Nonlinearity Layer

Let $g \in SO(3)$. The Wigner D-matrices are *unitary*, i.e.

$$\left\| D^{(\ell)}(g) V_{a,c}^{(\ell)} \right\| = \left\| V_{a,c}^{(\ell)} \right\|.$$

Now note that rotating the input and then applying linearity corresponds to

$$\eta^{(\ell)} \left(\left\| D^{(\ell)}(g) V_{a,c}^{(\ell)} \right\| + b_c^{(\ell)} \right) \left[D^{(\ell)}(g) V_{a,c}^{(\ell)} \right] = D^{(\ell)}(g) \left[\eta^{(\ell)} \left(\| V_{a,c}^{(\ell)} \| + b_c^{(\ell)} \right) V_{a,c}^{(\ell)} \right],$$

which is applying linearity first and then rotating the output.

Experiments: Quick Summary

- **Core architecture:** A Tensor Field Network (TFN) is equivariant to:
  - 3D rotations ($SO(3)$),
  - translations in $\mathbb{R}^3$,
  - permutations of input points.
- **Feature mapping:** The network transforms input features $x_a \in X$ at points $r_a \in \mathbb{R}^3$ into output features $y_a \in Y$:

$$\mathcal{L}(r_a, x_a) = (r_a, y_a)$$

- **Equivariance constraint:**

$$\mathcal{L}(D_X(g) \cdot x) = D_Y(g) \cdot \mathcal{L}(x)$$

- **Experiments goal:** Show that TFNs give consistent predictions under rotations and strong generalization with limited data.

Experiment 1: Shape Classification

- **Inputs:** 8 "3D Tetris" shapes
  - Input geometry: 3D point coordinates of each shape. Each block corresponds to a point.
  - Input features: Dummy scalar features (all ones) at every point.
- **Outputs:**
  - For classification, output is scalar corresponding to class label.
- **Training procedure:**
  - Train only on shapes in a **single fixed orientation**.
  - Train until the model reaches 100% training accuracy.

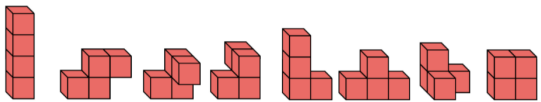


Figure: 3D Tetris shapes used for training/testing.

Experiment 1: Shape Classification — Test and Outcome

- **Test Set:**
  - Use the same 8 shapes.
  - Apply random 3D rotations and translations at test time.
- **Observed result:** 100% test accuracy.
- **Takeaway:** Final scalar class prediction for TFN is invariant to rotations and translations.
- **Additional evidence (from the paper):**
  - TFN was also trained on ModelNet40 (popular dataset of 3D objects) without rotational data augmentation.
  - It achieved decent (not state-of-the-art) accuracy.
  - Under the same no-augmentation setting, PointNet performs near random accuracy on randomly rotated/translated test data.

Experiment 2: Physics

- **Tasks:**
  - Predict the acceleration vector for each point mass in a collection of point masses.
  - Predict the moment of inertia tensor at a specific point in a collection of point masses.
- **Inputs:**
  - Random 3D point configurations with random scalar masses.
  - For the inertia task, an additional designated query point.
- **Outputs:**
  - Acceleration task: vector output $(0 \to 1)$.
  - Inertia task: $3x3$ symmetric matrix, which can be decomposed into a scalar $(\ell = 0)$ + symmetric traceless matrix $(\ell = 2)$, so we have $(0 \to 0 \oplus 2)$.
- **Training Procedure:**
  - For each task, ground-truth labels are computed analytically from their physics equations.

Aside: Why a symmetric matrix can be encoded as $0 \oplus 2$

- Decomposition of any symmetric 3×3 matrix:

$$S = \underbrace{\frac{\mathrm{tr}(S)}{3} I}_{\ell=0} + \underbrace{\left(S - \frac{\mathrm{tr}(S)}{3} I \right)}_{\ell=2}$$

- Example:

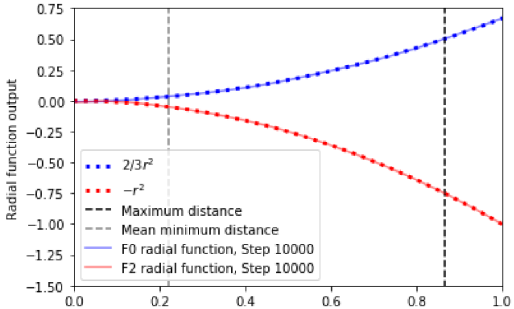
$$S = \begin{pmatrix} 4 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & 5 \end{pmatrix}, \quad \mathrm{tr}(S) = 12, \quad \frac{\mathrm{tr}(S)}{3} = 4$$

$$S^{(0)} = 4I, \qquad S^{(2)} = S - 4I = \begin{pmatrix} 0 & 1 & 2 \\ 1 & -1 & 0 \\ 2 & 0 & 1 \end{pmatrix}, \quad \mathrm{tr}(S^{(2)}) = 0$$

- symmetric matrix $\Rightarrow$ scalar $+$ symmetric traceless matrix $\Rightarrow 0 \oplus 2$.

Experiment 2: Physics — Test and Outcome

- **Observed result:**
  - Excellent agreement with the true physics.
  - Learned radial functions match.
- **Takeaway:** TFNs can learn to predict vector and tensor outputs that are equivariant under rotation and translation.



Experiment 3: Chemistry — Missing Atom Prediction

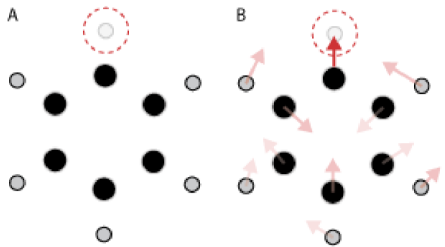
- **Task:**
  - Remove one atom from a molecular point cloud and predict the missing atom type and its location.
- **Inputs:**
  - Molecule point cloud where each point is an atom.
  - One atom is randomly deleted from each training example.
  - Each atom's input feature has its type encoded as a one hot array of scalars.
- **Outputs (network type $0 \to 0 \oplus 1$):**
  - Scalar outputs ($l = 0$): atom-type scores + confidence weight p at each point.
  - Vector output ($l = 1$): relative displacement δ from each point to the missing point.

Experiment 3: Chemistry — Missing Atom Prediction

- **Prediction rule:**
  - Each point casts a vote for missing location: $r_a + \delta_a$.
  - Final predicted position is a confidence-weighted sum:

$$\sum_a p_a(r_a + \delta_a).$$

  - Each point also casts atom-type scores which are weighted by the same confidence weights p_a to produce the final atom-type prediction.



Experiment 3: Chemistry — Training and Evaluation

- **Dataset:** QM9: a collection of 134,000 molecules.
  - Train on 1000 molecules with 5–18 atoms.
  - Each epoch train on all 1000 molecules, and randomly delete one atom from each molecule.
- **Evaluation protocol:**
  - Test on different molecules.
  - For each molecule, predict every possible removed atom.

Atoms	Number of predictions	Accuracy (%) (≤ 0.5 Å and atom type)	Distance MAE in Å
5-18	15863	91.3 (train)	0.16
19	19000	93.9 (test)	0.14
23	23000	96.5 (test)	0.13
25-29	25356	97.3 (test)	0.16

Strengths and Weaknesses

Strengths

- Strong theory: layer-wise equivariance implies whole-network equivariance.
- Experiments align with theory (geometry, physics, chemistry demos).
- Foundational early framework for continuous 3D rotation-equivariant learning on point clouds.

Weaknesses

- Experiments use only scalar input features.
- Data-efficiency is not directly benchmarked against non-equivariant baselines.
- Scaling can be unstable in practice: TFN often needed smaller settings (fewer points/channels) for stable training (as reported in the later paper SE(3)-Transformer).

Future Works

- **SE(3)-Transformer:**
  - Replaces convolution with self-attention.
  - Still equivariant under 3D rotations.
  - The authors wrote their own spherical harmonics library which uses the GPU to speed up computation significantly.
- **EGNN (E(n)-Equivariant GNN):**
  - EGNN avoids computing spherical harmonics, which are used in both TFN and SE(3)-Transformer so it is more computationally efficient.
  - TFN is built for 3D rotations, while EGNN can work in higher dimensions.

References

[1] **Thomas, N., et al. (2018).** Tensor Field Networks: Rotation and Translation Equivariant Neural Networks for 3D Point Clouds. *arXiv:1802.08219*. (The Core Paper)

[2] **Worrall, D. E., et al. (2017).** Harmonic Networks: Deep Translation and Rotation Equivariance. *CVPR*. (Foundation for 2D rotation equivariance)

[3] **Schütt, K. T., et al. (2017).** SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *NIPS*. (Reference for rotation-invariant continuous convolutions)

[4] **Reisert, M., & Burkhardt, H. (2007).** Learning object representations from 3D images and 3D point clouds with tensors. *International Journal of Computer Vision*. (Mathematical basis for tensors under 3D rotations)

[5] **Gilmore, R. (2008).** Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists. *Cambridge University Press*. (Reference for Wigner D-matrices and SO(3))

[6] **Goodman, R., & Wallach, N. R. (2009).** Symmetry, Representations, and Invariants. *Springer*. (Theory of group representations and irreducible representations)

[7] **Fuchs, F. B., et al. (2020).** SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. *arXiv:2006.10503*. (Equivariant self-attention extension of TFN)

[8] **Satorras, V. G., et al. (2022).** E(n) Equivariant Graph Neural Networks. *arXiv:2102.09844*. (Efficient equivariant GNN without spherical harmonics)