Neural Networks 188 (2025) 107481

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

Symmetry discovery for different data types

Lexiang Hu [a], Yikang Li [a], Zhouchen Lin [a,b,c,*]

[a] State Key Lab of General AI, School of Intelligence Science and Technology, Peking University, China
[b] Institute for Artificial Intelligence, Peking University, China
[c] Pazhou Laboratory (Huangpu), Guangzhou, Guangdong, China

ARTICLE INFO

Keywords:
Equivariant networks
Symmetry discovery

ABSTRACT

Equivariant neural networks incorporate symmetries into their architecture, achieving higher generalization performance. However, constructing equivariant neural networks typically requires prior knowledge of data types and symmetries, which is difficult to achieve in most tasks. In this paper, we propose LieSD, a method for discovering symmetries via trained neural networks which approximate the input–output mappings of the tasks. It characterizes equivariance and invariance (a special case of equivariance) of continuous groups using Lie algebra and directly solves the Lie algebra space through the inputs, outputs, and gradients of the trained neural network. Then, we extend the method to make it applicable to multi-channel data and tensor data, respectively. We validate the performance of LieSD on tasks with symmetries such as the two-body problem, the moment of inertia matrix prediction, top quark tagging, and rotated MNIST. Compared with the baseline, LieSD can accurately determine the number of Lie algebra bases without the need for expensive group sampling. Furthermore, LieSD can perform well on non-uniform datasets, whereas methods based on GANs fail. Code and data are available at https://github.com/hulx2002/LieSD.

1. Introduction

Symmetries in different data types play important roles in deep learning. Many recent works embed symmetries into network structures (Cohen & Welling, 2016b; Weiler & Cesa, 2019; Weiler, Hamprecht et al., 2018), not only improving model generalization but also reducing the number of parameters, which significantly enhances training performance on specific tasks. For instance, convolutional neural networks (LeCun et al., 1998) introduce the translational symmetry in image data, outperforming multilayer perceptrons in computer vision. Group equivariant convolutional networks (Cohen & Welling, 2016a) incorporate the rotational symmetry of image data. Equivariant multilayer perceptrons (Finzi et al., 2021) propose a method for constructing networks with more general Lie group symmetries.

However, these methods for constructing equivariant networks all require prior knowledge of symmetries. For example, in the image classification problem, translational and rotational invariance stems from humans' prior knowledge of the task. When faced with complex tasks and messy data, such as physical dynamic systems, it is very difficult to know symmetries in advance, and embedding incorrect symmetries can lead to a significant decrease in network performance.

A series of works on symmetry discovery emerge subsequently (Dehmamy et al., 2021; Moskalev et al., 2022; Romero & Lohit, 2022; Tegnér & Kjellstrom, 2023). Augerino (Benton et al., 2020) parameterizes group transformations and discovers symmetries in tasks by optimizing these parameters during training. However, parameterization requires a rough understanding of the form of symmetries prior, which limits it to finding sub-group symmetries of given group transformations. Data augmentation can also incur significant computational overhead.

LieGAN (Yang, Walters et al., 2023) and LaLiGAN (Yang, Dehmamy et al., 2023) directly search for symmetries from the data by bringing the distributions of original and transformed data closer. However, this method demands a high level of symmetry in the distribution of the dataset, which is difficult to achieve in the real world. For example, in facial recognition datasets, most images are frontal faces, which leads to insufficient representation of the rotational symmetry. Additionally, in datasets of particle motion trajectories, particles may also be concentrated in specific regions rather than uniformly distributed throughout the entire space. LieGAN and LaLiGAN also need to specify the coefficient distribution of the Lie algebra bases, since they require group sampling. Furthermore, incorrect setting of the number of Lie algebra bases can lead to them learning incorrect Lie algebra bases.

In this paper, we propose a method based on trained neural networks to discover symmetries, not only invariance but also equivariance. It can identify single-connected Lie group symmetries without

* Corresponding author at: State Key Lab of General AI, School of Intelligence Science and Technology, Peking University, China.
E-mail addresses: hulx@stu.pku.edu.cn (L. Hu), liyk18@pku.edu.cn (Y. Li), zlin@pku.edu.cn (Z. Lin).

https://doi.org/10.1016/j.neunet.2025.107481
Received 11 October 2024; Received in revised form 26 February 2025; Accepted 10 April 2025
Available online 22 April 2025
0893-6080/© 2025 Published by Elsevier Ltd.

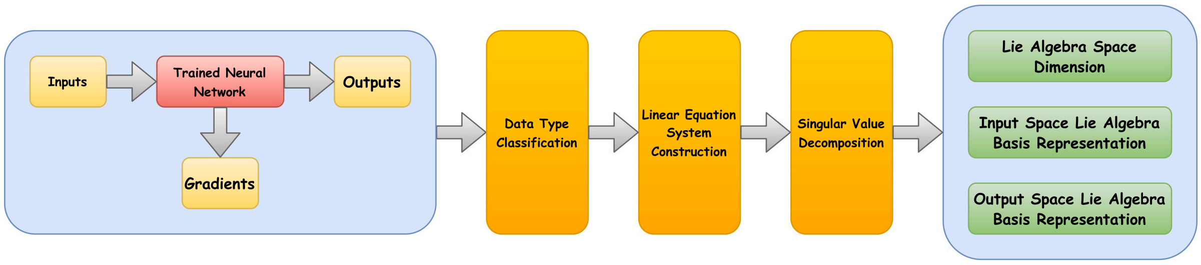

Inputs
Trained Neural Network
Outputs
Gradients
Data Type Classification
Linear Equation System Construction
Singular Value Decomposition
Lie Algebra Space Dimension
Input Space Lie Algebra Basis Representation
Output Space Lie Algebra Basis Representation

Fig. 1. The framework of LieSD. We first train a neural network on the task to approximate the input–output mapping, and then use its inputs, outputs, and gradients for symmetry discovery. Adapted to different data types, LieSD can solve the Lie algebra space, including the dimension of the Lie algebra space and the Lie algebra basis representations.

relying on uniform datasets. We first propose a theorem to demonstrate that equivariance of the network can be measured solely based on the network's inputs, outputs, gradients, and Lie algebra bases, without the need for data augmentation or group sampling. In practice, we need a trained neural network to approximate the true mapping of the task. Then, based on this theorem, we construct and solve a system of linear equations to obtain orthogonal Lie algebra bases, thereby discovering symmetries. We perform SVD on the coefficient matrix of the linear equations, where the number of zero singular values corresponds to the number of Lie algebra bases. Moreover, we extend our method to the multi-channel and tensor cases, whereas previous symmetry discovery methods only consider the single-channel vector case. The framework of our Lie algebra based symmetry discovery (LieSD) method is shown in Fig. 1.

In summary, our contributions are as follows:

- We provide a theorem that measures equivariance in the neural network using its inputs, outputs, and gradients, without the need for data augmentation or group sampling.
- We propose LieSD, a method for solving the Lie algebra space, thus discovering symmetries via trained neural networks. The dimension of the Lie algebra space can be determined by the number of zero singular values.
- We extend LieSD to discover symmetries in multi-channel and tensor data.
- We verify that LieSD can correctly discover symmetries in tasks such as the two-body problem, the moment of inertia matrix prediction, top quark tagging, rotated MNIST, and outperform the baseline, especially when the datasets are not uniform.

2. Related work

2.1. Equivariant networks

Many recent works have embedded equivariance into network structures layerwise, which improves their generalization on specific tasks. G-CNNs (Cohen & Welling, 2016a) and steerable CNNs (Cohen & Welling, 2016b) introduce discrete group equivariance into CNNs. SFC-NNs (Weiler, Hamprecht et al., 2018) and E(2)-Equivariant Steerable CNNs (Weiler & Cesa, 2019) extend discrete group equivariance to continuous cases, such as arbitrary angle rotations in group SO(2). 3D Steerable CNNs (Weiler, Geiger et al., 2018) extends the group action space to $\mathbb{R}^3$. EMLP (Finzi et al., 2021) and affConv/homConv (MacDonald et al., 2022) respectively embed arbitrary Lie group equivariance into MLP and CNNs. Furthermore, a series of works (He et al., 2022; Li et al., 2024; Shen et al., 2020, 2022, 2021) construct equivariant networks based on partial differential operators.

2.2. Symmetry discovery

Since the construction of equivariant networks requires prior knowledge of equivariance, many subsequent works attempt to discover symmetries. Some works are based on neural networks to detect symmetries. Augerino (Benton et al., 2020) parametrizes group actions. It performs augmentation on input data, then takes the average of the results as the final output of the network. However, it requires strong prior knowledge of group actions, namely the known parameterization of group actions. Krippendorf and Syvaeri (2020) uses the embedding layer of the neural network to identify orbits of the symmetries in the input. L-conv (Dehmamy et al., 2021) can automatically discover symmetries and serve as a building block to construct group equivariant feedforward architecture. Moskalev et al. (2022) propose a method for extracting invariance from neural networks, but it cannot handle situations of equivariance. van der Ouderaa et al. (2024) combine non-equivariant networks with equivariant networks. It assumes that the parameters of the two parts satisfy a given probability distribution, where the hyperparameters of the distribution can reflect the amount of equivariance. Similar to partial G-CNNs (Romero & Lohit, 2022), it can handle relaxed equivariant tasks well, but still requires specifying group transformations prior.

Some other works are based on dataset to discover symmetries. LieGAN (Yang, Walters et al., 2023) uses generative adversarial training to bring the data distributions before and after transformation closer, where the generator is used to generate group actions through group sampling. It eventually learns a set of orthogonal Lie algebra bases, but cannot accurately determine their number. LaLiGAN (Yang, Dehmamy et al., 2023) and Tegnér and Kjellstrom (2023) use an encoder–decoder architecture to map the dataset to a latent space with linear symmetries, which can indirectly discover nonlinear symmetries in the original space. Zhou et al. (2020) present a method for learning and embedding equivariance into networks by learning corresponding parameter sharing patterns from data. SGM (Allingham et al., 2024) captures symmetries under affine and color transformations based on generative models. These methods, based on dataset to discover symmetries, rely on uniform data distribution, while our work can perform well on non-uniform datasets.

3. Background

Before describing our method, we will first introduce some preliminary knowledge of group theory.

3.1. Lie group and Lie algebra

A Lie group G is a group with a smooth manifold structure. A Lie algebra $\mathfrak{g}$ is a vector space used to describe the local structure near the identity element of a Lie group. For a simply connected Lie group, any group element $g \in G$ can be obtained from a Lie algebra element $A \in \mathfrak{g}$ through the exponential map $\exp : \mathfrak{g} \to G$, i.e., $g = \exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$. Decomposing the space where the Lie algebra resides into several Lie algebra bases, we obtain:

$$\forall g \in G : \quad g = \exp\left(\sum_{i=1}^{D} \alpha_i A_i\right), \tag{1}$$

where D is the dimension of the Lie algebra space, $\{A_i\}_{i=1}^{D}$ are the Lie algebra bases, and $\{\alpha_i\}_{i=1}^{D}$ denotes their corresponding coefficients.

3.2. Group representation

Group representation describes how group elements act on a vector space through linear mappings. Formally, given a simply connected Lie group G and an n-dimensional vector space $\mathcal{X}$, a group representation $\rho_{\mathcal{X}} : G \to GL(n)$ maps group elements to n-dimensional invertible matrices. For $g_1, g_2 \in G$ it satisfies $\rho_{\mathcal{X}}(g_1 g_2) = \rho_{\mathcal{X}}(g_1)\rho_{\mathcal{X}}(g_2)$. Similarly, the Lie algebra representation can be defined as $\mathrm{d}\rho_{\mathcal{X}} : \mathfrak{g} \to \mathfrak{gl}(n)$. We can specialize Eq. (1) to the vector space:

$$\forall g \in G : \quad \rho_{\mathcal{X}}(g) = \exp\left(\sum_{i=1}^{D} \alpha_i \mathrm{d}\rho_{\mathcal{X}}(A_i)\right). \tag{2}$$

3.3. Equivariance and invariance

Symmetries can be divided into equivariance and invariance. Given a group G, with its group representations on the input space $\mathcal{X} \subseteq \mathbb{R}^n$ and output space $\mathcal{Y} \subseteq \mathbb{R}^m$ denoted as $\rho_{\mathcal{X}} : G \to GL(n)$ and $\rho_{\mathcal{Y}} : G \to GL(m)$ respectively. A function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant if $\forall g \in G, x \in \mathcal{X} : \rho_{\mathcal{Y}}(g)f(x) = f(\rho_{\mathcal{X}}(g)x)$. In particular, when $\rho_{\mathcal{Y}}(g) = I_m$, we call the function f invariant.

4. Measuring equivariance in neural networks

We need an efficient method to determine the equivariance of a trained neural network with respect to a given group. We consider the case of a simply connected Lie group, where group elements can be expressed as in Eq. (1). Naturally, we can perform data augmentation for testing, but this requires sampling of group elements. In Eq. (1), the Lie algebra bases $\{A_i\}_{i=1}^D$ can determine the information of the group, and different coefficients $\{\alpha_i\}_{i=1}^D$ correspond to different group elements, which means that we need to sample the coefficients. This method requires specifying the distribution of coefficients prior and entails significant computational cost. The following theorem tells us that equivariance in a neural network can be measured through its inputs, outputs, gradients, and Lie algebra bases, without the need for data augmentation.

Theorem 1. *Given a simply connected Lie group G, its group representations on the input space $\mathcal{X} \subseteq \mathbb{R}^n$ and output space $\mathcal{Y} \subseteq \mathbb{R}^m$ are denoted respectively as $\rho_{\mathcal{X}}(g) : G \to GL(n)$ and $\rho_{\mathcal{Y}}(g) : G \to GL(m)$. The function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant:*

$$\forall g \in G, x \in \mathcal{X} : \quad \rho_{\mathcal{Y}}(g)f(x) = f(\rho_{\mathcal{X}}(g)x), \tag{3}$$

if and only if the corresponding Lie algebra bases of $\rho_{\mathcal{X}}(g)$ and $\rho_{\mathcal{Y}}(g)$ in Eq. (2) satisfy:

$$\forall i \in \{1, 2, \ldots, D\}, x \in \mathcal{X} : \quad \mathrm{d}\rho_{\mathcal{Y}}(A_i)f(x) = \nabla f(x)\mathrm{d}\rho_{\mathcal{X}}(A_i)x. \tag{4}$$

The complete proof of Theorem 1 is provided in Appendix A.1. Note that Eq. (4) does not contain coefficients of the Lie algebra bases $\{\alpha_i\}_{i=1}^D$. Hence, to measure equivariance, we solely require knowledge of the group, namely the Lie algebra bases $\{A_i\}_{i=1}^D$. In practice, we directly use training data to obtain the inputs, outputs, and gradients of the neural network.

To discover symmetries, we need to accurately express the function from input to output. However, since the training set is discrete, gradient information cannot be obtained. Therefore, we train a neural network, which can accurately map the input of the training set to the output and is capable of capturing the correct gradient information. Moreover, since we use information from the training set to discover symmetries, we only need it to generalize within the training space $\mathcal{X}$, without requiring it to generalize across the entire space $\mathbb{R}^n$. Compared with methods based on GANs that directly obtain symmetries from the dataset (Desai et al., 2022; Yang, Dehmamy et al., 2023; Yang, Walters et al., 2023), we use local information of each data point rather than the overall data distribution. Therefore, our method can perform better on non-uniform datasets.

Although using automatic differentiation of a trained neural network to estimate gradients is the most accurate, other models for estimating gradients can also be integrated with our approach. For example, we have also experimented with k-Nearest Neighbors algorithm (k-NN) (Cover & Hart, 1967) to obtain gradient information from the discrete dataset. Specifically, for a given point, we perform local linear regression on the k nearest points in the dataset to fit the gradient vector. We use a Gaussian kernel $w_i = \exp\left(-\frac{d_i^2}{2}\right)$ to weight the points, reducing the influence of more distant points, where d_i is the distance from the ith neighboring data point to the given point. We provide a detailed procedure in Algorithm 1.

Algorithm 1 Gradient estimation based on k-NN

```
Input: Dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$, a given point $(x^{(0)}, y^{(0)})$, and the number of nearest neighbors $k$.
Output: Gradient estimate $\nabla f(x^{(0)}) \in \mathbb{R}^{m\times n}$.
Execute:
Select the $k$ nearest points to $(x^{(0)}, y^{(0)})$ from $\mathcal{D}$, and denote their indices as $(i_1, \ldots, i_k)$ and their distances to $(x^{(0)}, y^{(0)})$ as $(d_1, \cdots, d_k)$.
Construct $A \in \mathbb{R}^{k\times n}$, where the $r$-th row is $A_{r\cdot} = \exp\left(-\frac{d_i^2}{2}\right)(x^{(i_r)} - x^{(0)})$.
Construct $B \in \mathbb{R}^{k\times m}$, where the $r$-th row is $B_{r\cdot} = \exp\left(-\frac{d_i^2}{2}\right)(y^{(i_r)} - y^{(0)})$.
Solve for $g^* = \arg\min_{g\in\mathbb{R}^{m\times n}} \|Ag^\top - B\|_F^2$ using least squares.
Return $g^*$.
```

5. Symmetry discovery for single-channel vector data

Our final goal is to discover symmetries via trained neural network, which amounts to solving the space where the Lie algebra resides. A general approach is to optimize the parameters of the Lie algebra bases using gradient descent, where the loss function can be defined as $\mathcal{L}_{equiv}(\{A_i\}_{i=1}^D) = \frac{1}{D}\sum_{i=1}^D \|\mathrm{d}\rho_{\mathcal{Y}}(A_i)f(x) - \nabla f(x)\mathrm{d}\rho_{\mathcal{X}}(A_i)x\|_2$ according to Theorem 1. Although we can ensure the orthogonality of Lie algebra bases by adding a regularization term $l_{chreg}(\{A_i\}_{i=1}^D) = \sum_{1\le i<j\le D} R_{ch}(A_i, A_j)$ (Yang, Walters et al., 2023), we cannot exactly determine the dimension D of the Lie algebra space, which may lead to unexpected results.

We propose a mathematical method for solving the Lie algebra bases. Note that Eq. (4) is linear with respect to Lie algebra representation. We can transform it into a system of linear equations for solution, and determine the dimension of the Lie algebra space by the number of zero singular values of the coefficient matrix. We summarize the method as the following theorem.

Theorem 2. *Suppose that the input space $\mathcal{X} \subseteq \mathbb{R}^n$ and the output space $\mathcal{Y} \subseteq \mathbb{R}^m$ are both single-channel vector spaces. The function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{x^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, and let:*

$$C(\mathcal{D}) = \begin{bmatrix} -\nabla f(x^{(1)}) \otimes x^{(1)\top} & I_m \otimes f^\top(x^{(1)}) \\ -\nabla f(x^{(2)}) \otimes x^{(2)\top} & I_m \otimes f^\top(x^{(2)}) \\ \vdots & \vdots \\ -\nabla f(x^{(N)}) \otimes x^{(N)\top} & I_m \otimes f^\top(x^{(N)}) \end{bmatrix}. \tag{5}$$

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ reach its maximum. Formally speaking, $\forall x \in \mathcal{X} : \mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\mathcal{X}$ and $\mathcal{Y}$.

To prove Theorem 2, we substitute the data samples into Eq. (4) and transform it into a system of linear equations:

$$C(\mathcal{D})v = \begin{bmatrix} -\nabla f(x^{(1)}) \otimes x^{(1)\top} & I_m \otimes f^\top(x^{(1)}) \\ -\nabla f(x^{(2)}) \otimes x^{(2)\top} & I_m \otimes f^\top(x^{(2)}) \\ \vdots & \vdots \\ -\nabla f(x^{(N)}) \otimes x^{(N)\top} & I_m \otimes f^\top(x^{(N)}) \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A)) \end{bmatrix} = 0. \quad (6)$$

Then, the problem transforms into finding the bases $\{v_i\}_{i=1}^D$ of the subspace spanned by v that satisfies Eq. (6). The complete proof of Theorem 2 can be found in Appendix A.2.

In practice, we directly use the training set as the data sample. Due to factors such as data noise and model error, singular values are not exactly zero. So we compare the relative magnitudes of singular values and select those that are much closer to zeros.

Efficiently computing. In Eq. (5), we notice that $C(\mathcal{D}) \in \mathbb{R}^{(N\times m)\times(n^2+m^2)}$. Therefore, when the size N of the training set is large, directly computing the coefficient matrix $C(\mathcal{D})$ will incur significant storage overhead. In practice, we only compute $C(\mathcal{D})^\top C(\mathcal{D}) \in \mathbb{R}^{(n^2+m^2)\times(n^2+m^2)}$:

$$C(\mathcal{D})^\top C(\mathcal{D}) = \begin{bmatrix} \sum_{i=1}^N \nabla f(x^{(i)})^\top \nabla f(x^{(i)}) \otimes x^{(i)} x^{(i)\top} & -\sum_{i=1}^N \nabla f(x^{(i)})^\top \otimes x^{(i)} f^\top(x^{(i)}) \\ -\sum_{i=1}^N \nabla f(x^{(i)}) \otimes f(x^{(i)}) x^{(i)\top} & \sum_{i=1}^N I_m \otimes f(x^{(i)}) f^\top(x^{(i)}) \end{bmatrix}.$$

By calculating the eigenvalues and eigenvectors of $C(\mathcal{D})^\top C(\mathcal{D})$, we can obtain the singular values and right singular vectors of $C(\mathcal{D})$. Formally, for $C(\mathcal{D}) = U\Sigma V^\top$, we have $C(\mathcal{D})^\top C(\mathcal{D}) = V\Sigma^2 V^\top$.

6. Extension to different data types

6.1. Why the extension is necessary

We face a crucial question: what kind of symmetries do we want? Recall the motivation behind discovering symmetries, that is to aid the design of equivariant networks, enhance understanding of problems, or contribute to the discovery of scientific laws. Therefore, we need meaningful and interpretable symmetries. Take the two-body problem in LieGAN (Yang, Walters et al., 2023) as an example. For the symmetry that enables interactions between the position or momentum of different bodies, we cannot explain its actual physical significance. It is due to the origin of the dataset being at the center of mass and $m_1 = m_2$, which results in $q_1 = -q_2$ and $p_1 = -p_2$. In other words, this symmetry arises from a specific property of the dataset rather than from inherent physical laws. Therefore, for different data types, we impose certain requirements on the form of the Lie algebra basis, which essentially uses our prior knowledge of the structure of the input and output spaces to ensure that symmetries are meaningful.

6.2. Extension to multi-channel data

We first discuss the case where both the input and output of the task are multi-channel vectors, which means that independent group transformations are performed within each channel. Formally, the vector space can be decomposed as $V = \bigoplus_{i=1}^c V_i = V_1 \oplus V_2 \oplus \cdots \oplus V_c$. Correspondingly, the group representation and Lie algebra representation are $\rho_V(g) = \bigoplus_{i=1}^c \rho_{V_i}(g)$ and $\mathrm{d}\rho_V(A) = \bigoplus_{i=1}^c \mathrm{d}\rho_{V_i}(A)$ respectively, where $\oplus$ is defined as $A \oplus B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$ (Finzi et al., 2021). Similarly, we can obtain the following theorem.

Theorem 3. *Suppose that the input space $\mathcal{X} = \bigoplus_{i=1}^{c_x} \mathcal{X}_i$ and the output space $\mathcal{Y} = \bigoplus_{i=1}^{c_y} \mathcal{Y}_i$ are both multi-channel vector spaces. The function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{x^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, and let:*

$$C(\mathcal{D}) = \begin{bmatrix} C_x^{(1)} & C_y^{(1)} \\ C_x^{(2)} & C_y^{(2)} \\ \vdots & \vdots \\ C_x^{(N)} & C_y^{(N)} \end{bmatrix}, \quad (7)$$

where

$$\begin{cases} C_x^{(i)} = \begin{bmatrix} -\nabla f_1(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_1(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_1(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ -\nabla f_2(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_2(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_2(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ \vdots & \vdots & \ddots & \vdots \\ -\nabla f_{c_y}(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_{c_y}(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_{c_y}(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \end{bmatrix}, \\ C_y^{(i)} = \mathrm{diag}[I_{m_1} \otimes f_1^\top(x^{(i)}), I_{m_2} \otimes f_2^\top(x^{(i)}), \ldots, I_{m_{c_y}} \otimes f_{c_y}^\top(x^{(i)})], \end{cases}$$

x_i and $f_i(x)$ are the ith channels of x and $f(x)$ respectively, $\nabla f_i(x_j)$ is the gradient of $f_i(x)$ with respect to x_j, and m_i represents the dimension of the ith output channel.

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ reach its maximum. Formally speaking, $\forall x \in \mathcal{X} : \mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\{\mathcal{X}_i\}_{i=1}^{c_x}$ and $\{\mathcal{Y}_i\}_{i=1}^{c_y}$.

The complete proof of Theorem 3 is provided in Appendix A.3. To save memory overhead, we compute $C(\mathcal{D})^\top C(\mathcal{D})$ instead of directly storing the coefficient matrix $C(\mathcal{D})$ in Eq. (7):

$$C(\mathcal{D})^\top C(\mathcal{D}) = \begin{bmatrix} \sum_{i=1}^N C_x^{(i)\top} C_x^{(i)} & \sum_{i=1}^N C_x^{(i)\top} C_y^{(i)} \\ \sum_{i=1}^N C_y^{(i)\top} C_x^{(i)} & \sum_{i=1}^N C_y^{(i)\top} C_y^{(i)} \end{bmatrix}. \quad (8)$$

6.3. Extension to tensor data

Sometimes neural networks take high-dimensional tensors as inputs and outputs rather than vectors. For example, graph neural networks are used to process adjacency matrices representing graph-structured data (Kipf & Welling, 2016), and some neural networks are employed for handling the neural weight space (Eilertsen et al., 2020; Schürholt et al., 2021; Unterthiner et al., 2020). Many works also attempt to encode symmetries in high-dimensional tensors (Navon et al., 2023; Satorras et al., 2021). We now explain how our method for symmetry discovery can be generalized to the tensor case.

For the tensor space $T = \bigotimes_{i=1}^d V_i = V_1 \otimes V_2 \otimes \cdots \otimes V_d$, its group representation and Lie algebra representation can be expressed as $\rho_V(g) = \bigotimes_{i=1}^d \rho_{V_i}(g)$ and $\mathrm{d}\rho_V(A) = \overline{\bigoplus}_{i=1}^d \mathrm{d}\rho_{V_i}(A)$, where $\overline{\oplus}$ is defined as $A \overline{\oplus} B = A \otimes I_b + I_a \otimes B$ (Finzi et al., 2021). For simplicity, we consider the matrix case, i.e. $\mathcal{X} = \mathcal{X}_1 \otimes \mathcal{X}_2 = \mathbb{R}^{n_1\times n_2}$ and $\mathcal{Y} = \mathcal{Y}_1 \otimes \mathcal{Y}_2 = \mathbb{R}^{m_1\times m_2}$. The discovery of symmetries in high-dimensional tensors is analogous to matrices. Then we present the following theorem.

Theorem 4. *Suppose that the input space $\mathcal{X} = \mathcal{X}_1 \otimes \mathcal{X}_2 = \mathbb{R}^{n_1\times n_2}$ and the output space $\mathcal{Y} = \mathcal{Y}_1 \otimes \mathcal{Y}_2 = \mathbb{R}^{m_1\times m_2}$ are both matrix spaces. The function $F : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{X^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, and let:*

$$C(\mathcal{D}) = \begin{bmatrix} C_{x_1}^{(1)} & C_{x_2}^{(1)} & C_{y_1}^{(1)} & C_{y_2}^{(1)} \\ C_{x_1}^{(2)} & C_{x_2}^{(2)} & C_{y_1}^{(2)} & C_{y_2}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ C_{x_1}^{(N)} & C_{x_2}^{(N)} & C_{y_1}^{(N)} & C_{y_2}^{(N)} \end{bmatrix}, \quad (9)$$

where

$$\begin{cases} C_{x_1}^{(i)} = -\sum_{k=1}^{n_2} \nabla f(X_{\cdot k}^{(i)}) \otimes X_{\cdot k}^{(i)\top}, \quad C_{x_2}^{(i)} = -\sum_{k=1}^{n_1} \nabla f(X_{k\cdot}^{(i)}) \otimes X_{k\cdot}^{(i)}, \\ C_{y_1}^{(i)} = I_{m_1} \otimes F^\top(X^{(i)}), \quad C_{y_2}^{(i)} = \begin{bmatrix} I_{m_2} \otimes F_{1\cdot}(X^{(i)}) \\ I_{m_2} \otimes F_{2\cdot}(X^{(i)}) \\ \vdots \\ I_{m_2} \otimes F_{m_1\cdot}(X^{(i)}) \end{bmatrix}, \end{cases}$$

$f(X) \in \mathbb{R}^{(m_1\times m_2)\times 1}$ is the vectorized form of the output matrix $F(X) \in \mathcal{Y}$, $X_{k\cdot} \in \mathbb{R}^{1\times n_2}$ and $F_{k\cdot}(X) \in \mathbb{R}^{1\times m_2}$ are the kth row vectors of X and $F(X)$ respectively, $X_{\cdot k} \in \mathbb{R}^{n_1\times 1}$ is the kth column vector of X, $\nabla f(X_{\cdot k}) \in$

$\mathbb{R}^{(m_1\times m_2)\times n_1}$ and $\nabla f(X_{k\cdot}) \in \mathbb{R}^{(m_1\times m_2)\times n_2}$ denote the gradient of $f(X)$ with respect to $X_{\cdot k}$ and $X_{k\cdot}$ respectively.

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ *reach its maximum. Formally speaking,* $\forall x \in \mathcal{X}$: $\mathrm{rank}(C(\mathcal{D}\cup\{x\})) = \mathrm{rank}(C(\mathcal{D}))$. *Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\{\mathcal{X}_i\}_{i=1}^2$ and $\{\mathcal{Y}_i\}_{i=1}^2$.*

We provide the complete proof of Theorem 4 in Appendix A.4. The expression $C(\mathcal{D})^\top C(\mathcal{D})$ can be computed in the same way as Eq. (8).

7. Experiments

We evaluate LieSD on tasks with symmetries. We will validate that LieSD can perform well on non-uniform datasets, handle multi-channel and tensor cases, and accurately determine the number of Lie algebra bases. As mentioned in Section 2, many previous works attempted to discover symmetries (Benton et al., 2020; Desai et al., 2022; Moskalev et al., 2022; Zhou et al., 2020), but LieGAN (Yang, Walters et al., 2023) has already proven to achieve state-of-the-art results. Therefore, we only compare with LieGAN.

7.1. Quantitative criteria

We hope to quantitatively analyze the accuracy of Lie algebra bases. The ideal Lie algebra bases span the same space as the real Lie algebra space and they are pairwise orthogonal. Therefore, we will define space error and orthogonality error to evaluate the symmetry discovery method. Suppose $\{v_i\}_{i=1}^D$ are the normalized vector expansion forms of the discovered Lie algebra bases, with their ground truth being $\{v_i^*\}_{i=1}^{D^*}$.

7.1.1. Space error

Two sets of bases $\{v_i\}_{i=1}^D$ and $\{v_i^*\}_{i=1}^{D^*}$ spanning the same space in $\mathbb{R}^n$ means that each v_i can be linearly represented by $\{v_i^*\}_{i=1}^{D^*}$, and conversely, each v_i^* can also be linearly represented by $\{v_i\}_{i=1}^D$. Formally:

$$\begin{cases} \exists X \in \mathbb{R}^{D\times D^*}: & V^* = VX, \\ \exists Y \in \mathbb{R}^{D^*\times D}: & V = V^*Y, \end{cases} \tag{10}$$

where $V \in \mathbb{R}^{n\times D}$ and $V^* \in \mathbb{R}^{n\times D^*}$ are matrices with $\{v_i\}_{i=1}^D$ and $\{v_i^*\}_{i=1}^{D^*}$ as column vectors, respectively.

Therefore, we can define the space error as the residual of the overdetermined system of linear Eqs. (10):

$$E_{space} = \min_X \|VX - V^*\|_F^2 + \min_Y \|V^*Y - V\|_F^2,$$

where the optimal solution can be obtained by solving the linear equation systems $V^\top V^* = V^\top VX$ and $V^{*T}V = V^{*T}V^*Y$.

7.1.2. Orthogonality error

Evaluating the accuracy of basis vectors solely using space error is insufficient. For instance, 9 basis vectors can also span the correct 7-dimensional subspace, while two of them are redundant. Therefore, we need to penalize such cases with orthogonal error. We define orthogonal error as the sum of the absolute values of the inner products of the basis vectors pairwise:

$$E_{orth} = \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} |\langle v_i, v_j\rangle|.$$

7.2. Two-body problem

We first consider the two-body problem (Greydanus et al., 2019). It studies the motion of two particles on a plane under the influence of gravity, with their center of mass as the origin. The task takes the motion states q_1, p_1, q_2, p_2 of the two particles at time t as inputs and predicts the motion states of the two particles at time $t+1$, where $q_i \in \mathbb{R}^2$ and $p_i \in \mathbb{R}^2$ represent the position and momentum coordinates of the ith particle, respectively.

The two-body problem has SO(2) equivariance, meaning that rotating the input at time t by a certain angle will correspondingly rotate the prediction at time $t+1$ by the same angle. The four channels q_1, p_1, q_2, p_2 undergo group transformations independently. Formally, the number of Lie algebra basis is $D = 1$, and $\mathrm{d}\rho_{\mathcal{X}}(A) = \mathrm{d}\rho_{\mathcal{Y}}(A) = \bigoplus_{i=1}^4 \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

We will validate LieSD on the two-body problem to demonstrate its ability to handle multi-channel cases. Additionally, we will shuffle the data distribution to evaluate its performance on the non-uniform dataset. For comparison, we specify the Lie algebra basis number of LieGAN as 1. We provide implementation details in Appendix B.1. We also present the space error and orthogonality error in Table 1, and the visualization results in Fig. 2.

As shown in Table 1 and Fig. 2, both LieSD and LieGAN can discover the correct Lie algebra basis on the uniform dataset. Note that when the Lie algebra basis differs by a coefficient, the space it spans remains the same. There is only one Lie algebra basis, so the orthogonality error is always zero. Furthermore, LieSD identifies one singular value that is almost zero, indicating that the number of Lie algebra basis is $D = 1$, whereas LieGAN requires manual specification. When we disrupt the data distribution, LieGAN fails, but LieSD still performs well, demonstrating stronger robustness on the non-uniform dataset.

To explore the potential of combining other gradient estimation methods with LieSD, we further evaluate the results of LieSD using k-NN instead of a trained neural network to estimate gradients. The number of neighboring points k is set to the number of points in the dataset, which we found to be the most accurate setting for gradient estimation. The experimental results are presented in Table 1 and Fig. 3, where we did not report error bars for the quantitative results because the k-NN gradient estimation is not stochastic. Although the accuracy of LieSD based on k-NN is not as high as that based on a trained neural network, the results of symmetry discovery are still largely correct. We anticipate that integrating better gradient estimation models in the future will make LieSD even more powerful.

7.3. The moment of inertia matrix prediction

We will now explore symmetries in predicting the moment of inertia matrix (Finzi et al., 2021). Given the masses $m_i \in \mathbb{R}$ and position coordinates $x_i \in \mathbb{R}^3$ of several particles on a rigid body, the moment of inertia matrix is defined as $M(\{m_i, x_i\}_i) = \sum_i m_i(x_i^\top x_i I - x_i x_i^\top)$. This task exhibits SO(3) and scaling equivariance. For $R \in \mathrm{SO}(3)$ and $S \in \{sI | s \in \mathbb{R}\}$, $M(\{m_i, Rx_i\}_i) = RM(\{m_i, x_i\}_i)R^\top$ and $M(\{m_i, Sx_i\}_i) = SM(\{m_i, x_i\}_i)S^\top$ hold. Formally, the input is a multi-channel vector $\mathcal{X} = \bigoplus_i \mathcal{X}_i$, and the output is a matrix $\mathcal{Y} = \mathcal{Y}_1 \otimes \mathcal{Y}_2$, where $\mathcal{X}_i = \mathbb{R}^3$ and $\mathcal{Y}_i = \mathbb{R}^3$.

We use the moment of inertia matrix prediction to verify that LieSD can handle tensor cases. In addition, we introduce uniformly distributed random noise of $\pm 10\%$ to the dataset to simulate real-world disturbances. We also perform an ablation study to demonstrate that processing the output as a matrix rather than an unfolded vector will be better. The implementation details can be found in Appendix B.2. We present the space error and orthogonality error in Table 2, and the visualization results in the dataset without noise in Figs. 4 and 5.

As shown in Fig. 4, LieSD finds 5 singular values that are almost zeros. Bases 24–26 correspond to SO(3) equivariance, and basis 23 corresponds to scaling equivariance. For basis 27, $\mathrm{d}\rho_{\mathcal{Y}}(A_1) =$

Table 1
The space error and orthogonality error of symmetry discovery in the two-body problem. We present the results in the format of averages (minimum ∼ maximum). LieSD exhibits stronger robustness on the non-uniform dataset.

Dataset	Method	Space error		Orthogonality error	
Uniform	LieSD	2.65×10^{-2}	$(4.35 \times 10^{-4} \sim 1.04 \times 10^{-1})$	0	$(0 \sim 0)$
	LieSD (k-NN)	9.74×10^{-2}		0	
	LieGAN	3.33×10^{-4}	$(1.97 \times 10^{-4} \sim 5.93 \times 10^{-4})$	0	$(0 \sim 0)$
Non-uniform	LieSD	7.17×10^{-2}	$(2.50 \times 10^{-2} \sim 1.85 \times 10^{-1})$	0	$(0 \sim 0)$
	LieGAN	9.35×10^{-1}	$(5.97 \times 10^{-1} \sim 1.35)$	0	$(0 \sim 0)$

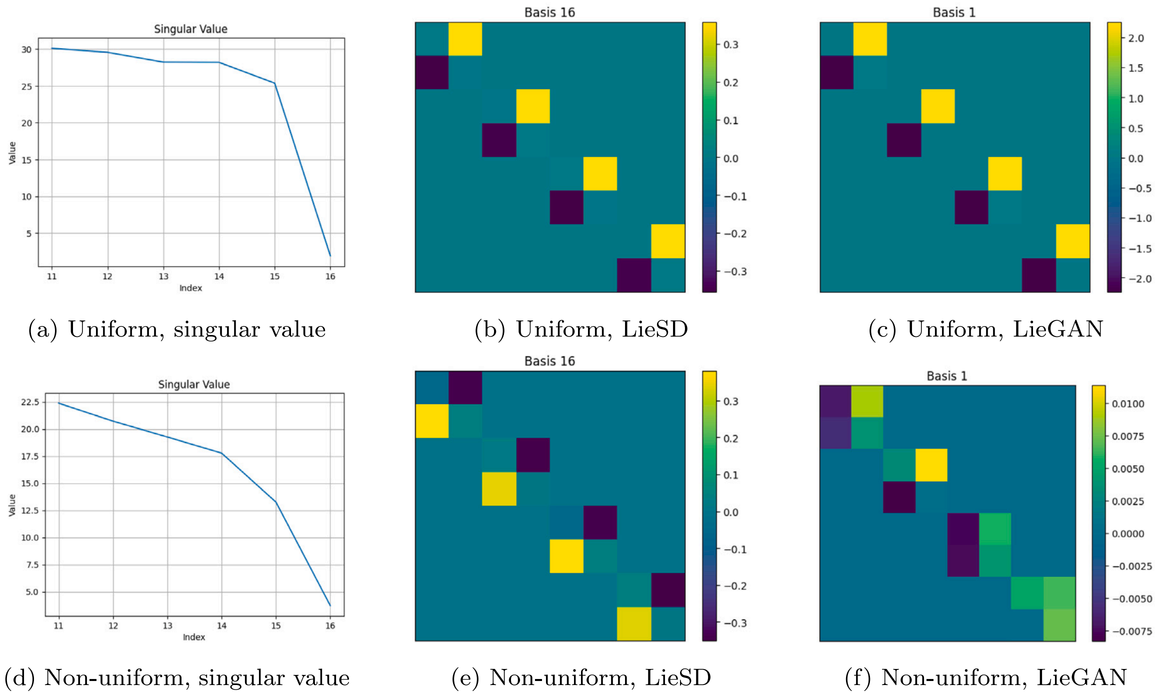


Fig. 2. The visualization results of symmetry discovery in the two-body problem. (a): The top 6 smallest singular values solved by LieSD on the uniform dataset, which are arranged in descending order. (b): The Lie algebra basis corresponding to the minimum singular value solved by LieSD on the uniform dataset. (c): The Lie algebra basis learned by LieGAN with 1 channel on the uniform dataset. (d-f): Results corresponding to (a-c) on the non-uniform dataset.

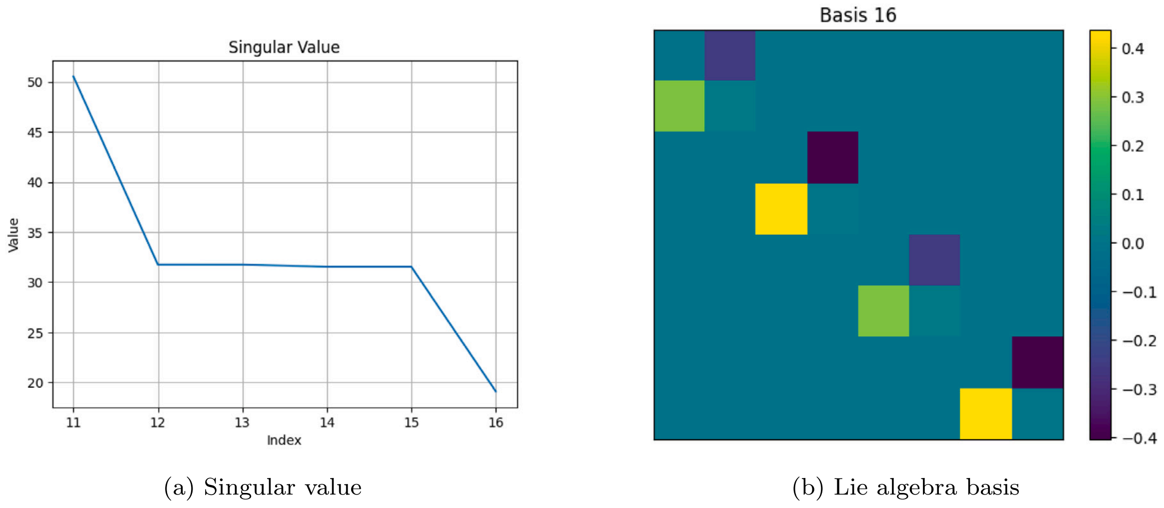


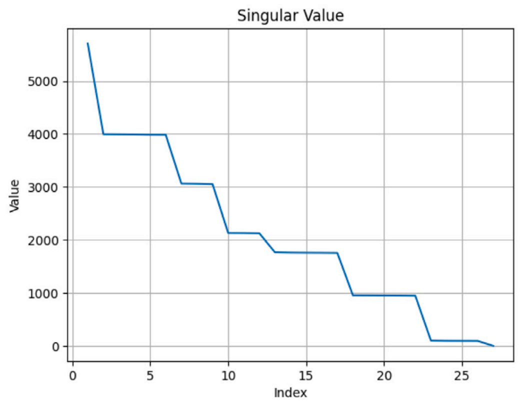
Fig. 3. The visualization results of LieSD (k-NN) in the two-body problem (uniform dataset). (a): The top 6 smallest singular values, which are arranged in descending order. (b): The Lie algebra basis corresponding to the minimum singular value.

$\mathrm{d}\rho_{\mathcal{Y}1}(A_1)\overline{\bigoplus}\mathrm{d}\rho_{\mathcal{Y}_2}(A_1) = \mathrm{diag}(a,a,a)\overline{\bigoplus}\mathrm{diag}(-a,-a,-a) = \mathbf{0}$ and $\mathrm{d}\rho_{\mathcal{X}}(A_1) = \mathbf{0}$. Therefore, basis 27 corresponds to the case where group representations of the input and output spaces are both trivial.

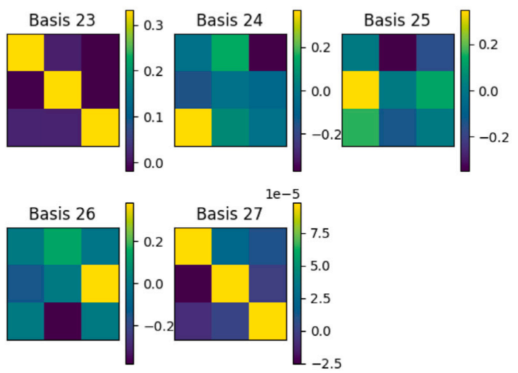
On the other hand, when we treat the moment of inertia matrix as a flattened vector, LieSD computes 26 nearly zero singular values and obtains a series of meaningless Lie algebra bases as shown in

Table 2
The space error and orthogonality error of symmetry discovery in predicting the moment of inertia matrix. We present the results in the format of averages (minimum ∼ maximum). Using the tensor form of LieSD allows for accurate computation of the Lie algebra space, while using the vector form of LieSD fails.

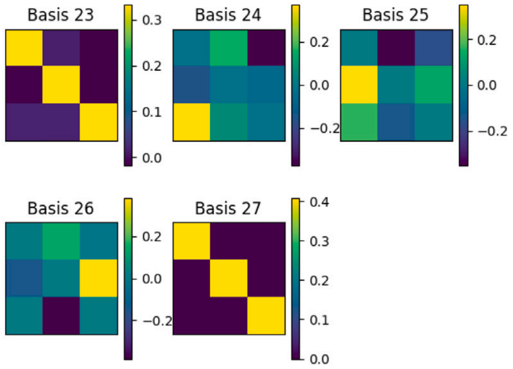
Dataset	Method	Space error		Orthogonality error	
w/o noise	LieSD (tensor)	7.08×10^{-5}	$(6.66 \times 10^{-5} \sim 7.55 \times 10^{-5})$	1.17×10^{-1}	$(5.91 \times 10^{-2} \sim 2.22 \times 10^{-1})$
	LieSD (k-NN, tensor)	6.06×10^{-1}		1.67×10^{-3}	
	LieSD (vector)	3.44×10^{1}	$(3.43 \times 10^{1} \sim 3.45 \times 10^{1})$	4.56×10^{-5}	$(1.78 \times 10^{-5} \sim 8.88 \times 10^{-5})$
w/ noise	LieSD (tensor)	1.99×10^{-3}	$(1.69 \times 10^{-3} \sim 2.32 \times 10^{-3})$	6.91×10^{-2}	$(4.09 \times 10^{-2} \sim 8.17 \times 10^{-2})$
	LieSD (k-NN, tensor)	6.06×10^{-1}		2.08×10^{-3}	
	LieSD (vector)	3.44×10^{1}	$(3.43 \times 10^{1} \sim 3.45 \times 10^{1})$	8.70×10^{-5}	$(6.27 \times 10^{-5} \sim 1.06 \times 10^{-4})$



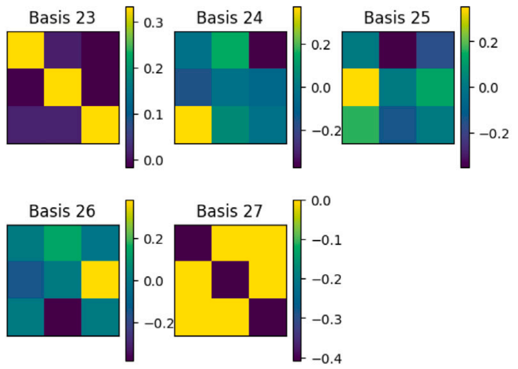
(a) Singular value



(b) Lie algebra basis representation in $\mathcal{X}_i$



(c) Lie algebra basis representation in $\mathcal{Y}_1$



(d) Lie algebra basis representation in $\mathcal{Y}_2$

Fig. 4. The visualization results of symmetry discovery in predicting the moment of inertia matrix (dataset without noise) of LieSD. (a): The computed singular values, which are arranged in descending order. (b-d): Lie algebra bases corresponding to five nearly zero singular values in spaces $\mathcal{X}_i, \mathcal{Y}_1, \mathcal{Y}_2$, where basis i corresponds to the singular value with index i.

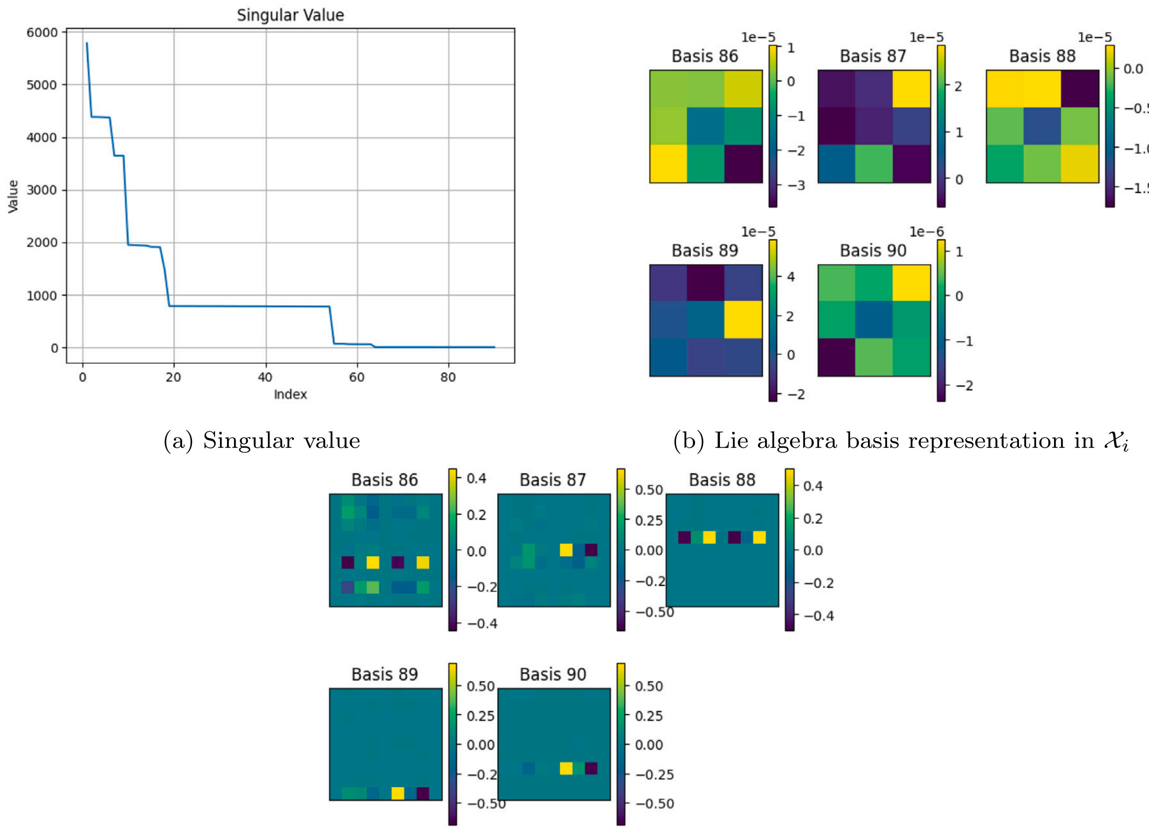
Fig. 5. In Table 2, we can confirm that using the tensor form of LieSD allows us to solve the correct Lie algebra space, while using the vector form of LieSD leads to completely incorrect Lie algebra space. Therefore, when the input or output is in tensor form, employing Theorem 4 instead of flattening into vectors and then using Theorem 2 to discover symmetries (essentially imposing formal constraints on Lie algebra representations) can help us more accurately identify the correct Lie algebra space. The reason why the orthogonal error in the tensor case is greater than in the vector case is that the orthogonality of $\mathrm{d}\rho_{\mathcal{Y}_1}(A)$ and $\mathrm{d}\rho_{\mathcal{Y}_2}(A)$ does not guarantee the orthogonality of $\mathrm{d}\rho_{\mathcal{Y}}(A) = \mathrm{d}\rho_{\mathcal{Y}_1}(A)\overline{\oplus}\mathrm{d}\rho_{\mathcal{Y}_2}(A)$.

Similar to the two-body problem, we further evaluate the results of LieSD based on k-NN. The number of neighboring points k is set to the number of points in the dataset, which we found to be the most accurate setting for gradient estimation. As shown in Table 2 and Fig. 6 (we do not report error bars for the quantitative results because the gradient estimation based on k-NN is not random), we once again confirm that the symmetry findings of LieSD based on k-NN are generally correct, but the accuracy is not as high as that of LieSD based on a trained neural network.

7.4. Top quark tagging

Top quark tagging (Yang, Walters et al., 2023) is a task of classifying hadronic tops from QCD background. It takes as input the four-momenta $p_i^\mu = (p_i^0, p_i^1, p_i^2, p_i^3) \in \mathbb{R}^4$ of the 20 jet constituents with the highest transverse momentum p_T produced in particle collisions and predicts labels for the particles, where $p_i^0 \in \mathbb{R}$ is the energy component of the ith jet constituent, and $(p_i^1, p_i^2, p_i^3) \in \mathbb{R}^3$ are the spatial momentum components. This task possesses $\mathrm{SO}(1,3)^+$ and scaling invariance. Rotations in three-dimensional space, boosts, and scaling transformations will not alter the classification result.

We use top quark tagging to illustrate the advantage of LieSD in accurately determining the number of Lie algebra bases, while LieGAN requires manual specification. Similar to the moment of inertia matrix prediction, we introduce uniformly distributed random noise of $\pm 10\%$



(a) Singular value (b) Lie algebra basis representation in $\mathcal{X}_i$

(c) Lie algebra basis representation in $\mathcal{Y}$

Fig. 5. The visualization results of the ablation study on symmetry discovery in predicting the moment of inertia matrix (dataset without noise). (a): The computed singular values, which are arranged in descending order. (b-c): Lie algebra bases corresponding to the five smallest singular values in spaces $\mathcal{X}_i$ and $\mathcal{Y}$, where basis i corresponds to the singular value with index i.

Table 3
The space error and orthogonality error of symmetry discovery in top quark tagging (dataset without noise). We present the results in the format of averages (minimum ∼ maximum). Compared with LieGAN, the Lie algebra space solved by LieSD is more accurate, and the Lie algebra bases are almost strictly orthogonal. What is more, incorrectly specifying the number of Lie algebra bases for LieGAN will make the results even worse.

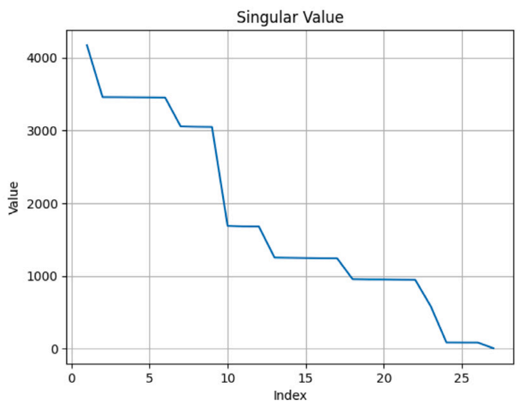
Dataset	Method	Space error		Orthogonality error	
w/o noise	LieSD	1.47×10^{-3}	$(1.19\times10^{-3}\sim2.11\times10^{-3})$	1.72×10^{-6}	$(1.46\times10^{-6}\sim2.07\times10^{-6})$
	LieGAN (7 channels)	2.49	$(1.28\sim4.07)$	5.82×10^{-1}	$(2.14\times10^{-1}\sim1.26)$
	LieGAN (9 channels)	2.22	$(1.65\sim2.75)$	1.13	$(5.54\times10^{-1}\sim1.43)$
w/ noise	LieSD	9.47×10^{-2}	$(6.65\times10^{-2}\sim1.74\times10^{-1})$	2.57×10^{-6}	$(2.11\times10^{-6}\sim3.03\times10^{-6})$
	LieGAN (7 channels)	6.07	$(5.31\sim6.49)$	1.50×10^{-1}	$(1.24\times10^{-1}\sim1.82\times10^{-1})$
	LieGAN (9 channels)	6.69	$(6.42\sim6.87)$	1.22	$(1.20\sim1.25)$

to the dataset to evaluate the robustness of LieSD against real-world disturbances. We will set different numbers of Lie algebra bases for LieGAN and compare the differences in their results. We provide implementation details in Appendix B.3. We also present the space error and orthogonality error in Table 3, and the visualization results in the dataset without noise in Fig. 7.

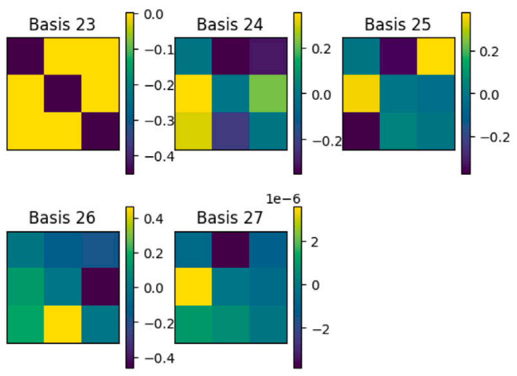
As shown in Fig. 7, LieSD solves for 7 singular values that are almost zeros, indicating that the number of Lie algebra bases is 7. Therefore, bases 10–16 in Fig. 7(b) are correct while bases 8–9 are not. Among them, basis 16 corresponds to scaling, bases 12, 13 and 15 correspond to rotations in three-dimensional space, and bases 10, 11 and 14 correspond to boosts. On the other hand, although we accurately set the number of Lie algebra bases for LieGAN to 7, LieSD can still outperform it in terms of accuracy and orthogonality as shown in Table 3. Furthermore, when we mistakenly set its number of Lie algebra bases to 9, LieGAN results in worse performance. It learns incorrect bases 5 and 7 as shown in Fig. 7(d), and we cannot distinguish them. In short, the Lie algebra bases obtained by LieSD are ordered, and the singular value can tell us which ones are valid, while LieGAN cannot achieve this because the Lie algebra bases it obtains are disordered.

7.5. Rotated MNIST

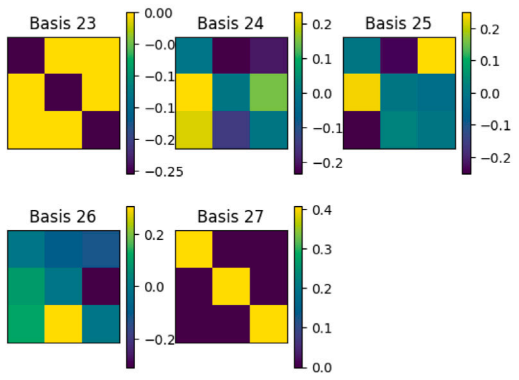
Finally, we evaluate the performance of LieSD in real high-dimensional scenarios on Rotated MNIST. For data generation, each image in MNIST is randomly rotated by an angle θ, where θ is sampled from a uniform distribution $\mathcal{U}(-\alpha,\alpha)$. We set the rotation range $\alpha=\{0,\frac{1}{4}\pi,\frac{1}{2}\pi,\frac{3}{4}\pi,\pi\}$ to simulate different levels of uniformity in the dataset. Note that the rotation transformation is applied to the image coordinates $x\in\mathbb{R}^2$, and all pixel coordinates share the same transformation, which means we can treat it as a multi-channel scenario. However, since the neural network takes the grayscale values $I(x)$ of the image as input, automatic differentiation is applied to obtain $\nabla_I f$ instead of $\nabla_x f$. Therefore, we further estimate the gradient of



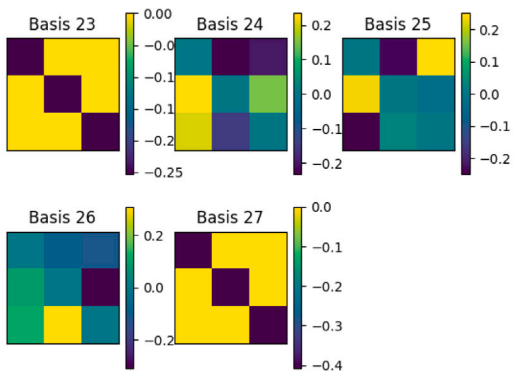
(a) Singular value



(b) Lie algebra basis representation in $\mathcal{X}_i$



(c) Lie algebra basis representation in $\mathcal{Y}_1$



(d) Lie algebra basis representation in $\mathcal{Y}_2$

Fig. 6. The visualization results of symmetry discovery in predicting the moment of inertia matrix (dataset without noise) of LieSD (k-NN). (a): The computed singular values, which are arranged in descending order. (b-d): Lie algebra bases corresponding to five nearly zero singular values in spaces $\mathcal{X}_i, \mathcal{Y}_1, \mathcal{Y}_2$, where basis i corresponds to the singular value with index i.

Table 4
The space error and orthogonality error of symmetry discovery in rotated MNIST. We present the results in the format of averages (minimum ∼ maximum). LieSD exhibits stronger robustness on real-world high-dimensional data.

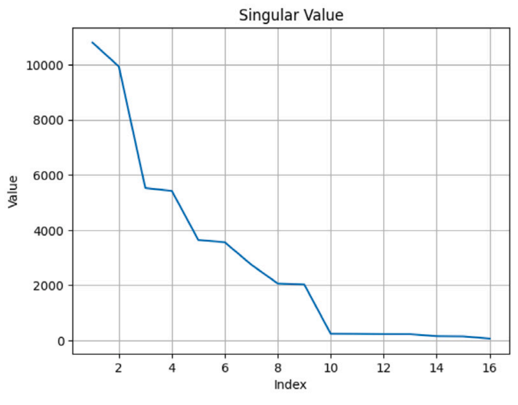
Rotation range	Method	Space error		Orthogonality error	
$\alpha = 0$	LieSD	6.33×10^{-1}	$(3.56 \times 10^{-1} \sim 9.31 \times 10^{-1})$	0	$(0 \sim 0)$
	LieGAN	2.00	$(1.99 \sim 2.00)$	0	$(0 \sim 0)$
$\alpha = \frac{1}{4}\pi$	LieSD	1.08×10^{-1}	$(6.78 \times 10^{-2} \sim 1.32 \times 10^{-1})$	0	$(0 \sim 0)$
	LieGAN	1.97	$(1.92 \sim 2.00)$	0	$(0 \sim 0)$
$\alpha = \frac{1}{2}\pi$	LieSD	4.45×10^{-2}	$(4.41 \times 10^{-3} \sim 7.99 \times 10^{-2})$	0	$(0 \sim 0)$
	LieGAN	1.96	$(1.94 \sim 1.98)$	0	$(0 \sim 0)$
$\alpha = \frac{3}{4}\pi$	LieSD	3.08×10^{-2}	$(8.23 \times 10^{-3} \sim 5.50 \times 10^{-2})$	0	$(0 \sim 0)$
	LieGAN	1.99	$(1.99 \sim 2.00)$	0	$(0 \sim 0)$
$\alpha = \pi$	LieSD	3.48×10^{-3}	$(5.28 \times 10^{-4} \sim 5.35 \times 10^{-3})$	0	$(0 \sim 0)$
	LieGAN	1.97	$(1.92 \sim 2.00)$	0	$(0 \sim 0)$

the grayscale values with respect to the coordinates $\nabla_x I$ using central differences, and obtain $\nabla_x f = \nabla_I f \cdot \nabla_x I$ using the chain rule. For comparison, we specify the Lie algebra basis number of LieGAN as 1. More implementation details are provided in Appendix B.4. We also present the space error and orthogonality error in Table 4, and the visualization results in Fig. 8.

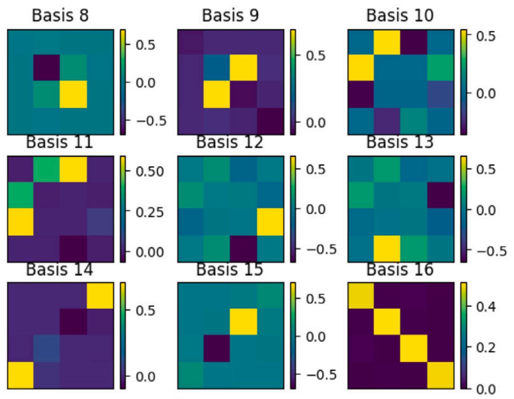
In Figs. 8(a)–8(e), we observe that although the accuracy of LieSD on the dataset with a smaller rotation range α is not as high as on that with a larger rotation range α, it still manages to capture the overall correct symmetry. This further confirms the capability of LieSD to handle cases with insufficient representation of rotational symmetry. On the other hand, LieGAN does not perform as expected in this task as shown in Fig. 8(f), while LieSD demonstrates stronger robustness when dealing with real-world high-dimensional data.

8. Conclusion

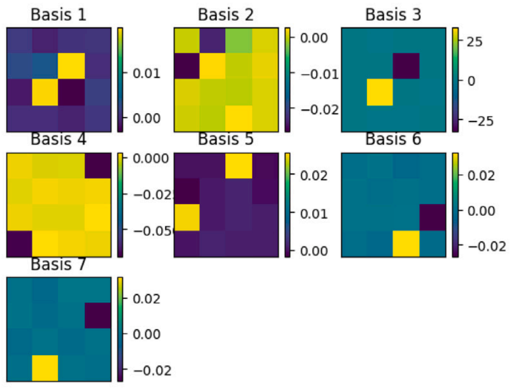
This work proposes a mathematical approach to discover symmetries via trained neural networks. The theorem we propose allows for measuring the equivariance in neural networks without the need for data augmentation or group sampling. Based on this theorem, we obtain the Lie algebra space by solving a system of linear equations, where the number of zero singular values corresponds to the dimension of the Lie algebra space. Furthermore, we extend the method to the multi-channel and tensor cases to adapt to different data types.



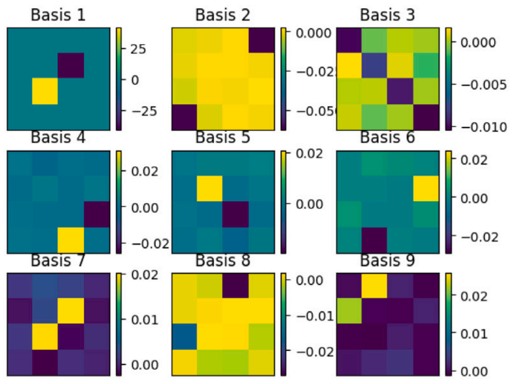
(a) LieSD, singular value



(b) LieSD, Lie algebra bases

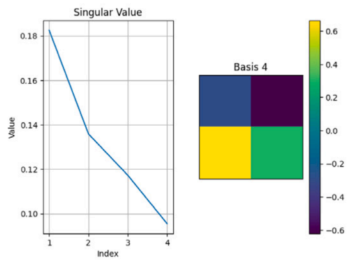


(c) LieGAN, 7 channels

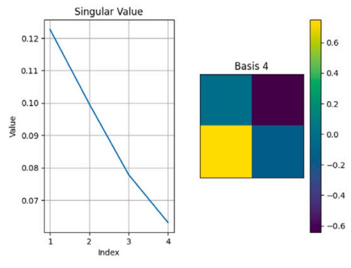


(d) LieGAN, 9 channels

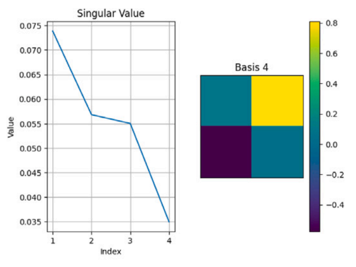
Fig. 7. The visualization results of symmetry discovery in top quark tagging (dataset without noise). (a): The singular values obtained by LieSD, which are sorted in descending order. (b): Lie algebra bases solved by LieSD, where basis i corresponds to the ith largest singular value. (c): The Lie algebra bases learned by LieGAN with 7 channels. (d): The Lie algebra bases learned by LieGAN with 9 channels.



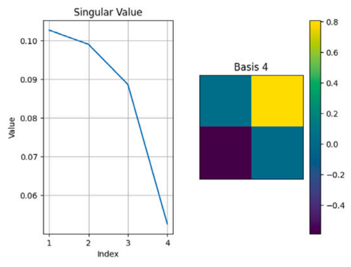
(a) LieSD, $\alpha = 0$



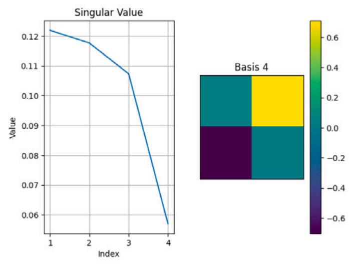
(b) LieSD, $\alpha = \frac{1}{4}\pi$



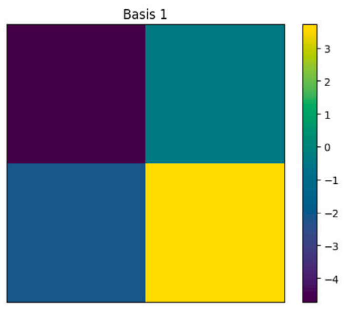
(c) LieSD, $\alpha = \frac{1}{2}\pi$



(d) LieSD, $\alpha = \frac{3}{4}\pi$



(e) LieSD, $\alpha = \pi$



(f) LieGAN, $\alpha = \pi$

Fig. 8. The visualization results of symmetry discovery in rotated MNIST. (a-e): Under different values of the rotation range α, the singular values obtained by LieSD and the Lie algebra basis corresponding to the smallest singular value. (f): The Lie algebra basis learned by LieGAN with 1 channel at the rotation range $\alpha = \pi$.

Compared with methods based on GANs that discover symmetries from the dataset, our approach does not rely on data distribution, thus performing well on non-uniform datasets.

9. Discussion

Since LieSD is based on the infinitesimal criterion of group symmetry as shown in Eq. (4), one of its limitations is the inability to discover discrete generators. A possible extension is to consider cases where some discrete generators $\{h_i\}_{i=1}^{M}$ are known in advance, i.e., the group elements in Eq. (1) can be rewritten as $g = \exp\left(\sum_{i=1}^{D} \alpha_i A_i\right) \prod_{i=1}^{N} h_{k_i}$. Then, we can add a constraint in Eq. (4) so that LieSD can obtain infinitesimal generators on several connected components:

$$\forall i \in \{1,2,\ldots,D\}, j \in \{1,2,\ldots,M\}, x \in \mathcal{X}:$$
$$\mathrm{d}\rho_{\mathcal{Y}}(A_i)\rho_{\mathcal{Y}}(h_j)f(x) = \nabla f(\rho_{\mathcal{X}}(h_j)x)\mathrm{d}\rho_{\mathcal{X}}(A_i)\rho_{\mathcal{X}}(h_j)x.$$

We anticipate that in the future, LieSD can be integrated with methods for discovering discrete symmetries to capture full symmetries.

Heavy reliance on gradient estimation is another limitation of LieSD. Although experiments on rotated MNIST have demonstrated the ability of LieSD to handle real-world high-dimensional data, our method may underperform when shattered gradients appear on more complex data manifolds (Dombrowski et al., 2023). As we attempt to combine LieSD with k-NN, we anticipate that more robust gradient estimation models in the future could further enhance the robustness of LieSD.

CRediT authorship contribution statement

Lexiang Hu: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Yikang Li:** Writing – review & editing, Methodology, Investigation, Formal analysis, Conceptualization. **Zhouchen Lin:** Writing – review & editing, Validation, Supervision, Software, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zhouchen Lin reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Z. Lin was supported by National Key R&D Program of China (2022ZD0160300) and the NSF China (No. 62276004).

Appendix A. Complete proofs

A.1. Proof of Theorem 1

Theorem 1. Given a simply connected Lie group G, its group representations on the input space $\mathcal{X} \subseteq \mathbb{R}^n$ and output space $\mathcal{Y} \subseteq \mathbb{R}^m$ are denoted respectively as $\rho_{\mathcal{X}}(g): G \to GL(n)$ and $\rho_{\mathcal{Y}}(g): G \to GL(m)$. The function $f: \mathcal{X} \to \mathcal{Y}$ is equivariant:

$$\forall g \in G, x \in \mathcal{X}: \quad \rho_{\mathcal{Y}}(g)f(x) = f(\rho_{\mathcal{X}}(g)x), \tag{3}$$

if and only if the corresponding Lie algebra bases of $\rho_{\mathcal{X}}(g)$ and $\rho_{\mathcal{Y}}(g)$ in equation Eq. (2) satisfy:

$$\forall i \in \{1,2,\ldots,D\}, x \in \mathcal{X}: \quad \mathrm{d}\rho_{\mathcal{Y}}(A_i)f(x) = \nabla f(x)\mathrm{d}\rho_{\mathcal{X}}(A_i)x. \tag{4}$$

Proof. First, we prove Eq. (3) ⇒ Eq. (4). Substitute Eq. (2) into Eq. (3), and note that different coefficients of the Lie algebra bases correspond to different group elements:

$$\forall \alpha \in \mathbb{R}^D, x \in \mathcal{X}:$$
$$h(\alpha) = \exp\left(\sum_{i=1}^{D} \alpha_i \mathrm{d}\rho_{\mathcal{Y}}(A_i)\right) f(x) - f\left(\exp\left(\sum_{i=1}^{D} \alpha_i \mathrm{d}\rho_{\mathcal{X}}(A_i)\right) x\right) = 0. \tag{A.1}$$

Taking the derivative element-wise with respect to α, and then setting $\alpha = 0$, we obtain:

$$\forall i \in \{1,2,\ldots,D\}, x \in \mathcal{X}: \quad \left.\frac{\partial h}{\partial \alpha_i}\right|_{\alpha=0} = \mathrm{d}\rho_{\mathcal{Y}}(A_i)f(x) - \nabla f(x)\mathrm{d}\rho_{\mathcal{X}}(A_i)x = 0.$$

Then, we prove Eq. (4) ⇒ Eq. (3). For convenience, we denote $B = \sum_{i=1}^{D} \alpha_i A_i$. By the linearity of the Lie algebra representation, we have $\mathrm{d}\rho_{\mathcal{X}}(B) = \sum_{i=1}^{D} \alpha_i \mathrm{d}\rho_{\mathcal{X}}(A_i)$ and $\mathrm{d}\rho_{\mathcal{Y}}(B) = \sum_{i=1}^{D} \alpha_i \mathrm{d}\rho_{\mathcal{Y}}(A_i)$. What is more:

$$\mathrm{d}\rho_{\mathcal{Y}}(B)f(x) = \nabla f(x)\mathrm{d}\rho_{\mathcal{X}}(B)x.$$

Replace x with $\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x$:

$$\mathrm{d}\rho_{\mathcal{Y}}(B)f(\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x) = \nabla f(\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x)\mathrm{d}\rho_{\mathcal{X}}(B)\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x. \tag{A.2}$$

Define $\tilde{h}: \mathbb{R} \to \mathbb{R}^m$ as:

$$\tilde{h}(\beta) = \exp(\beta \cdot \mathrm{d}\rho_{\mathcal{Y}}(B))f(x) - f(\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x).$$

Taking the derivative with respect to β:

$$\frac{\mathrm{d}\tilde{h}}{\mathrm{d}\beta} = \mathrm{d}\rho_{\mathcal{Y}}(B)\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{Y}}(B))f(x) - \nabla f(\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x)\mathrm{d}\rho_{\mathcal{X}}(B)\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x.$$

Substitute Eq. (A.2):

$$\frac{\mathrm{d}\tilde{h}}{\mathrm{d}\beta} = \mathrm{d}\rho_{\mathcal{Y}}(B)\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{Y}}(B))f(x) - \mathrm{d}\rho_{\mathcal{Y}}(B)f(\exp(\beta \cdot \mathrm{d}\rho_{\mathcal{X}}(B))x) = \mathrm{d}\rho_{\mathcal{Y}}(B)\tilde{h}(\beta).$$

Solving the differential equation, and note that $\tilde{h}(0) = 0$:

$$\tilde{h}(\beta) = \exp(\beta \cdot \mathrm{d}\rho_{\mathcal{Y}}(B))\tilde{h}(0) = 0.$$

Let $\beta = 1$, we obtain:

$$\exp(\mathrm{d}\rho_{\mathcal{Y}}(B))f(x) - f(\exp(\mathrm{d}\rho_{\mathcal{X}}(B))x) = 0,$$

which is equivalent to Eq. (A.1). Therefore, Eq. (3) is proven. □

A.2. Proof of Theorem 2

Theorem 2. Suppose that the input space $\mathcal{X} \subseteq \mathbb{R}^n$ and the output space $\mathcal{Y} \subseteq \mathbb{R}^m$ are both single-channel vector spaces. The function $f: \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$ from the input space $\mathcal{X}$, and let:

$$C(\mathcal{D}) = \begin{bmatrix} -\nabla f(x^{(1)}) \otimes x^{(1)\top} & I_m \otimes f^{\top}(x^{(1)}) \\ -\nabla f(x^{(2)}) \otimes x^{(2)\top} & I_m \otimes f^{\top}(x^{(2)}) \\ \vdots & \vdots \\ -\nabla f(x^{(N)}) \otimes x^{(N)\top} & I_m \otimes f^{\top}(x^{(N)}) \end{bmatrix}. \tag{5}$$

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ reach its maximum. Formally speaking, $\forall x \in \mathcal{X}: \mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\mathcal{X}$ and $\mathcal{Y}$.

Proof. Flatten the Lie algebra representation in Eq. (4) from matrices into vectors:

$$\forall x \in \mathcal{X}: \quad [I_m \otimes f^\top(x)]\mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A)) = [\nabla f(x) \otimes x^\top]\mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)).$$

Note that equivariance implies a one-to-one correspondence between the Lie algebra representations of the input space and the output space. We concatenate them together:

$$\forall x \in \mathcal{X}: \quad \begin{bmatrix} -\nabla f(x) \otimes x^\top & I_m \otimes f^\top(x) \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A)) \end{bmatrix} = 0. \tag{A.3}$$

Sample N data points $\{x^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, then Eq. (A.3) holds for any element $x^{(i)}$. We obtain a system of linear equations:

$$C(\mathcal{D})v = \begin{bmatrix} -\nabla f(x^{(1)}) \otimes x^{(1)\top} & I_m \otimes f^\top(x^{(1)}) \\ -\nabla f(x^{(2)}) \otimes x^{(2)\top} & I_m \otimes f^\top(x^{(2)}) \\ \vdots & \vdots \\ -\nabla f(x^{(N)}) \otimes x^{(N)\top} & I_m \otimes f^\top(x^{(N)}) \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A)) \end{bmatrix} = 0. \tag{A.4}$$

Obviously, $\langle \mathrm{d}\rho(A_i), \mathrm{d}\rho(A_j)\rangle = 0$ is equivalent to $\langle v_i, v_j\rangle = 0$. Therefore, the problem transforms into finding the bases $\{v_i\}_{i=1}^D$ of the subspace spanned by v that satisfies Eq. (A.4).

Perform singular value decomposition on the coefficient matrix $C(\mathcal{D})$:

$$C(\mathcal{D})v = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P^\top \\ Q^\top \end{bmatrix} v = 0.$$

Then we can obtain the solution space of Eq. (A.4):

$$v = Q\beta = \sum_{i=1}^{D} q_i \beta_i,$$

where $\{q_i\}_{i=1}^D$ are the column vectors of matrix Q, and $\{\beta_i\}_{i=1}^D$ are their corresponding coefficients. In other words, the column vectors of matrix Q form the bases of the solution space:

$$\begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A_i)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A_i)) \end{bmatrix} = v_i = q_i, \quad i \in \{1, 2, \ldots, D\}.$$

In general, the number of zero singular values of the coefficient matrix $C(\mathcal{D})$ in Eq. (A.4) matches the dimension of the Lie algebra space, and the corresponding right singular vectors are the vector expansion forms of the Lie algebra bases.

We are concerned that the discretization of Eq. (A.3) may lose information, which could lead to the solution of redundant and incorrect Lie algebra bases. In Eq. (A.4), note that $v \in \mathbb{R}^{n^2+m^2}$. If the row rank of $C(\mathcal{D})$ is r, then the dimension of the solution space v is $n^2 + m^2 - r$. Therefore, as long as we ensure that the row rank of $C(\mathcal{D})$ reaches its maximum, that is $n^2 + m^2 - D$, we can completely obtain the correct D Lie algebra bases. This assumption can be formalized as $\forall x \in \mathcal{X}$: $\mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. □

A.3. Proof of Theorem 3

Theorem 3. Suppose that the input space $\mathcal{X} = \bigoplus_{i=1}^{c_x} \mathcal{X}_i$ and the output space $\mathcal{Y} = \bigoplus_{i=1}^{c_y} \mathcal{Y}_i$ are both multi-channel vector spaces. The function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{x^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, and let:

$$C(\mathcal{D}) = \begin{bmatrix} C_x^{(1)} & C_y^{(1)} \\ C_x^{(2)} & C_y^{(2)} \\ \vdots & \vdots \\ C_x^{(N)} & C_y^{(N)} \end{bmatrix}, \tag{7}$$

where

$$\begin{cases} C_x^{(i)} \\ = \begin{bmatrix} -\nabla f_1(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_1(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_1(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ -\nabla f_2(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_2(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_2(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ \vdots & \vdots & \ddots & \vdots \\ -\nabla f_{c_y}(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_{c_y}(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_{c_y}(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \end{bmatrix}, \\ C_y^{(i)} = \mathrm{diag}[I_{m_1} \otimes f_1^\top(x^{(i)}), I_{m_2} \otimes f_2^\top(x^{(i)}), \ldots, I_{m_{c_y}} \otimes f_{c_y}^\top(x^{(i)})], \end{cases}$$

x_i and $f_i(x)$ are the ith channels of x and $f(x)$ respectively, $\nabla f_i(x_j)$ is the gradient of $f_i(x)$ with respect to x_j, and m_i represents the dimension of the ith output channel.

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ reach its maximum. Formally speaking, $\forall x \in \mathcal{X}$: $\mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\{\mathcal{X}_i\}_{i=1}^{c_x}$ and $\{\mathcal{Y}_i\}_{i=1}^{c_y}$.

Proof. We can express Eq. (4) as:

$$\begin{bmatrix} \bigoplus_{i=1}^{c_y} \mathrm{d}\rho_{\mathcal{Y}_i}(A) \end{bmatrix} \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_{c_y}(x) \end{bmatrix} =$$

$$\begin{bmatrix} \nabla f_1(x_1) & \nabla f_1(x_2) & \cdots & \nabla f_1(x_{c_x}) \\ \nabla f_2(x_1) & \nabla f_2(x_2) & \cdots & \nabla f_2(x_{c_x}) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla f_{c_y}(x_1) & \nabla f_{c_y}(x_2) & \cdots & \nabla f_{c_y}(x_{c_x}) \end{bmatrix} \begin{bmatrix} \bigoplus_{i=1}^{c_x} \mathrm{d}\rho_{\mathcal{X}_i}(A) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{c_x} \end{bmatrix}.$$

In other words:

$$\forall i \in \{1, 2, \ldots, c_y\}, x \in \mathcal{X}: \quad \mathrm{d}\rho_{\mathcal{Y}_i}(A) f_i(x) = \sum_{j=1}^{c_x} \nabla f_i(x_j) \mathrm{d}\rho_{\mathcal{X}_j}(A) x_j.$$

Similar to the single-channel case, we flatten the Lie algebra representation from matrices to vectors:

$$\forall i \in \{1, 2, \ldots, c_y\}, x \in \mathcal{X}:$$

$$[I_{m_i} \otimes f_i^\top(x)]\mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_i}(A)) = \sum_{j=1}^{c_x} [\nabla f_i(x_j) \otimes x_j^\top]\mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}_j}(A)). \tag{A.5}$$

We discretize Eq. (A.5) using N data samples and obtain a system of linear equations:

$$C(\mathcal{D})v = \begin{bmatrix} C_x^{(1)} & C_y^{(1)} \\ C_x^{(2)} & C_y^{(2)} \\ \vdots & \vdots \\ C_x^{(N)} & C_y^{(N)} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = 0, \tag{A.6}$$

where

$$\begin{cases} C_x^{(i)} \\ = \begin{bmatrix} -\nabla f_1(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_1(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_1(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ -\nabla f_2(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_2(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_2(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \\ \vdots & \vdots & \ddots & \vdots \\ -\nabla f_{c_y}(x_1^{(i)}) \otimes x_1^{(i)\top} & -\nabla f_{c_y}(x_2^{(i)}) \otimes x_2^{(i)\top} & \cdots & -\nabla f_{c_y}(x_{c_x}^{(i)}) \otimes x_{c_x}^{(i)\top} \end{bmatrix}, \\ C_y^{(i)} = \mathrm{diag}[I_{m_1} \otimes f_1^\top(x^{(i)}), I_{m_2} \otimes f_2^\top(x^{(i)}), \ldots, I_{m_{c_y}} \otimes f_{c_y}^\top(x^{(i)})], \\ v_x = \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}_1}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}_2}(A)) \\ \vdots \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}_{c_x}}(A)) \end{bmatrix}, \quad v_y = \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_1}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_2}(A)) \\ \vdots \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_{c_y}}(A)) \end{bmatrix}. \end{cases}$$

Then we use the same method as in the single-channel case to solve Eq. (A.6) and obtain the Lie algebra representation for each channel. □

A.4. Proof of Theorem 4

Theorem 4. Suppose that the input space $\mathcal{X} = \mathcal{X}_1 \otimes \mathcal{X}_2 = \mathbb{R}^{n_1 \times n_2}$ and the output space $\mathcal{Y} = \mathcal{Y}_1 \otimes \mathcal{Y}_2 = \mathbb{R}^{m_1 \times m_2}$ are both matrix spaces. The function $F : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to a simply connected Lie group G. Sample N data points $\mathcal{D} = \{X^{(i)}\}_{i=1}^N$ from the input space $\mathcal{X}$, and let:

$$C(\mathcal{D}) = \begin{bmatrix} C_{x_1}^{(1)} & C_{x_2}^{(1)} & C_{y_1}^{(1)} & C_{y_2}^{(1)} \\ C_{x_1}^{(2)} & C_{x_2}^{(2)} & C_{y_1}^{(2)} & C_{y_2}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ C_{x_1}^{(N)} & C_{x_2}^{(N)} & C_{y_1}^{(N)} & C_{y_2}^{(N)} \end{bmatrix}, \tag{9}$$

where

$$\begin{cases} C_{x_1}^{(i)} = -\sum_{k=1}^{n_2} \nabla f(X_{\cdot k}^{(i)}) \otimes X_{\cdot k}^{(i)\top}, \quad C_{x_2}^{(i)} = -\sum_{k=1}^{n_1} \nabla f(X_{k\cdot}^{(i)}) \otimes X_{k\cdot}^{(i)}, \\ C_{y_1}^{(i)} = I_{m_1} \otimes F^\top(X^{(i)}), \quad C_{y_2}^{(i)} = \begin{bmatrix} I_{m_2} \otimes F_{1\cdot}(X^{(i)}) \\ I_{m_2} \otimes F_{2\cdot}(X^{(i)}) \\ \vdots \\ I_{m_2} \otimes F_{m_1\cdot}(X^{(i)}) \end{bmatrix}, \end{cases}$$

$f(X) \in \mathbb{R}^{(m_1 \times m_2)\times 1}$ is the vectorized form of the output matrix $F(X) \in \mathcal{Y}$, $X_{k\cdot} \in \mathbb{R}^{1\times n_2}$ and $F_{k\cdot}(X) \in \mathbb{R}^{1\times m_2}$ are the kth row vectors of X and $F(X)$ respectively, $X_{\cdot k} \in \mathbb{R}^{n_1 \times 1}$ is the kth column vector of X, $\nabla f(X_{\cdot k}) \in \mathbb{R}^{(m_1\times m_2)\times n_1}$ and $\nabla f(X_{k\cdot}) \in \mathbb{R}^{(m_1\times m_2)\times n_2}$ denote the gradient of $f(X)$ with respect to $X_{\cdot k}$ and $X_{k\cdot}$ respectively.

Assume that $\mathcal{D}$ makes $\mathrm{rank}(C(\mathcal{D}))$ reach its maximum. Formally speaking, $\forall x \in \mathcal{X} : \mathrm{rank}(C(\mathcal{D} \cup \{x\})) = \mathrm{rank}(C(\mathcal{D}))$. Then the number of zero singular values of $C(\mathcal{D})$ is the dimension of the Lie algebra space of group G, and the right singular vectors corresponding to zero singular values are the vector expansion forms of the orthogonal Lie algebra basis representations on $\{\mathcal{X}_i\}_{i=1}^2$ and $\{\mathcal{Y}_i\}_{i=1}^2$.

Proof. Eq. (4) can be written as:

$$[\mathrm{d}\rho_{\mathcal{Y}_1}(A) \otimes I_{m_2} + I_{m_1} \otimes \mathrm{d}\rho_{\mathcal{Y}_2}(A)] f(x) = \nabla f(x) [\mathrm{d}\rho_{\mathcal{X}_1} \otimes I_{n_2} + I_{n_1} \otimes \mathrm{d}\rho_{\mathcal{X}_2}] x, \tag{A.7}$$

where $x \in \mathbb{R}^{(n_1\times n_2)\times 1}$ is the vectorized form of the input matrix $X \in \mathcal{X}$. To rewrite Eq. (A.7) in a form similar to Eq. (4), we obtain:

$$\mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_1}(A) F(X)) + \begin{bmatrix} \mathrm{d}\rho_{\mathcal{Y}_2}(A) F_{1\cdot}^\top(X) \\ \mathrm{d}\rho_{\mathcal{Y}_2}(A) F_{2\cdot}^\top(X) \\ \vdots \\ \mathrm{d}\rho_{\mathcal{Y}_2}(A) F_{m_1\cdot}^\top(X) \end{bmatrix} =$$
$$\sum_{k=1}^{n_2} \nabla f(X_{\cdot k}) \mathrm{d}\rho_{\mathcal{X}_1}(A) X_{\cdot k} + \sum_{k=1}^{n_1} \nabla f(X_{k\cdot}) \mathrm{d}\rho_{\mathcal{X}_2}(A) X_{k\cdot}^\top,$$

Similar to Eq. (6), we construct a system of linear equations:

$$C(\mathcal{D}) v = \begin{bmatrix} C_{x_1}^{(1)} & C_{x_2}^{(1)} & C_{y_1}^{(1)} & C_{y_2}^{(1)} \\ C_{x_1}^{(2)} & C_{x_2}^{(2)} & C_{y_1}^{(2)} & C_{y_2}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ C_{x_1}^{(N)} & C_{x_2}^{(N)} & C_{y_1}^{(N)} & C_{y_2}^{(N)} \end{bmatrix} \begin{bmatrix} v_{x_1} \\ v_{x_2} \\ v_{y_1} \\ v_{y_2} \end{bmatrix} = 0, \tag{A.8}$$

where

$$\begin{cases} C_{x_1}^{(i)} = -\sum_{k=1}^{n_2} \nabla f(X_{\cdot k}^{(i)}) \otimes X_{\cdot k}^{(i)\top}, \quad C_{x_2}^{(i)} = -\sum_{k=1}^{n_1} \nabla f(X_{k\cdot}^{(i)}) \otimes X_{k\cdot}^{(i)}, \\ C_{y_1}^{(i)} = I_{m_1} \otimes F^\top(X^{(i)}), \quad C_{y_2}^{(i)} = \begin{bmatrix} I_{m_2} \otimes F_{1\cdot}(X^{(i)}) \\ I_{m_2} \otimes F_{2\cdot}(X^{(i)}) \\ \vdots \\ I_{m_2} \otimes F_{m_1\cdot}(X^{(i)}) \end{bmatrix}, \\ v_{x_i} = \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}_i}(A)), \quad v_{y_i} = \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_i}(A)). \end{cases}$$

Therefore, we can obtain the Lie algebra space by solving Eq. (A.8). □

Appendix B. Implementation details

B.1. Two-body problem

For data generation, we follow the setup of Hamiltonian Neural Networks (Greydanus et al., 2019), which we restate as follows. First, we generate 1000 initial conditions with the center of mass at the origin, total momentum of 0, and the distance between two particles $r = \|q_2 - q_1\|$ within the interval $[0.5, 1.5]$. Gaussian noise with a scaling factor of $\sigma^2 = 0.05$ is added to the initial velocities. Then, we perform numerical integration of the dynamics equations under gravitational force using the fourth-order Runge–Kutta method to obtain 20 observation points for each trajectory.

The two-body problem takes the positions and momentum coordinates q_1, p_1, q_2, p_2 of two particles on a plane as input and output. Therefore, we use a 3-layer MLP with input dimension of 8, hidden dimension of 384, output dimension of 8, and Tanh activation function to fit the mapping of the problem. We choose the Adan optimizer (Xie et al., 2024) for training and set the number of epochs to 10. To shuffle the data distribution, for data points where q_1 is in the first or third quadrant, we rotate both its input and output by 90 degrees. The size of the training set is 18,000, and we use the entire training set for symmetry discovery. We perform this experiment on a single-core NVIDIA GeForce RTX 3090 GPU with available memory of 24,576 MiB, and the execution time is approximately 3 min.

We consider the two-body problem as a multi-channel case, where position and momentum coordinates transform independently. For the equivariance discovery of such sequence prediction problems, we hope that the group representation of the input space is the same as that of the output space. Formally, $\rho_{\mathcal{X}}(g) = \rho_{\mathcal{Y}}(g) = \rho(g)$, and $\mathrm{d}\rho_{\mathcal{X}}(A) = \mathrm{d}\rho_{\mathcal{Y}}(A) = \mathrm{d}\rho(A)$. Setting $v = v_x = v_y$ in Eq. (A.6), we obtain:

$$C(\mathcal{D}) v = \begin{bmatrix} C_x^{(1)} + C_y^{(1)} \\ C_x^{(2)} + C_y^{(2)} \\ \vdots \\ C_x^{(N)} + C_y^{(N)} \end{bmatrix} v = 0.$$

Then, $C(\mathcal{D})^\top C(\mathcal{D})$ can be computed as follows:

$$C(\mathcal{D})^\top C(\mathcal{D}) = \sum_{i=1}^N (C_x^{(i)} + C_y^{(i)})^\top (C_x^{(i)} + C_y^{(i)}).$$

When conducting quantitative error analysis, we repeat experiments with five different random seeds. The random factor for LieSD is the parameter initialization of the neural network, while for LieGAN, it is the parameter initialization of the generator and discriminator. We report the average values of space error and orthogonality error in Table 1 (in Section 7.2), and indicate their minimum and maximum values in gray font.

B.2. The moment of inertia matrix prediction

For convenience, we assume that the mass of all particles on the rigid body is 1. Therefore, the problem takes the spatial coordinates of several particles $x_i \in \mathbb{R}^3$ as input and outputs the inertia tensor of the rigid body $M \in \mathbb{R}^{3\times 3}$. In practice, we set the number of particles to 3. We generate the dataset by sampling $x_i \in \mathcal{N}(0,1)^3$ and computing M according to the formula. To simulate real-world disturbances, we introduce uniformly distributed random noise of $\pm 10\%$ to the dataset. Specifically, for each value v_{origin} in the dataset, the perturbed value is $v_{noise} = v_{origin} \times (1 + \epsilon)$, where $\epsilon \sim \mathcal{U}(-0.1, 0.1)$.

The input dimension of the three-layer MLP used for training is 9, the hidden dimension is 384, and the output dimension is 9, with the ReLU function used as the activation function. We train for 100 epochs using the Adan optimizer (Xie et al., 2024). The size of the training set is 100,000, and we use $\frac{1}{10}$ of it for symmetry discovery. We perform this experiment on a single-core NVIDIA GeForce RTX 3090 GPU with available memory of 24,576 MiB, and the execution time is approximately 20 min.

For this problem, the input is a multi-channel vector $\mathcal{X} = \bigoplus_i \mathcal{X}_i$, and the output is a matrix $\mathcal{Y} = \mathcal{Y}_1 \otimes \mathcal{Y}_2$. Furthermore, different spatial coordinates of particles should share the same group transformation. Formally, $\rho_{\mathcal{X}_1}(g) = \cdots = \rho_{\mathcal{X}_{c_x}}(g) = \rho_{\mathcal{X}}(g)$, and $\mathrm{d}\rho_{\mathcal{X}_1}(A) = \cdots = \mathrm{d}\rho_{\mathcal{X}_{c_x}}(A) = \mathrm{d}\rho_{\mathcal{X}}(A)$. Combining Eqs. (A.6) and (A.8), we obtain:

$$C(\mathcal{D})v = \begin{bmatrix} C_x^{(1)} & C_{y_1}^{(1)} & C_{y_2}^{(1)} \\ C_x^{(2)} & C_{y_1}^{(2)} & C_{y_2}^{(2)} \\ \vdots & \vdots & \vdots \\ C_x^{(N)} & C_{y_1}^{(N)} & C_{y_2}^{(N)} \end{bmatrix} \begin{bmatrix} v_x \\ v_{y_1} \\ v_{y_2} \end{bmatrix} = 0,$$

where

$$\begin{cases} C_x^{(i)} = -\sum_{j=1}^{c_x} \nabla f(x_j^{(i)}) \otimes x_j^{(i)\top}, \\ C_{y_1}^{(i)} = I_{m_1} \otimes F^\top(x^{(i)}), \quad C_{y_2}^{(i)} = \begin{bmatrix} I_{m_2} \otimes F_{1\cdot}(x^{(i)}) \\ I_{m_2} \otimes F_{2\cdot}(x^{(i)}) \\ \vdots \\ I_{m_2} \otimes F_{m_1\cdot}(x^{(i)}) \end{bmatrix}, \\ v_x = \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)), \quad v_{y_i} = \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}_i}(A)). \end{cases}$$

Similarly, we compute $C^\top C$ to save memory overhead:

$$C(\mathcal{D})^\top C(\mathcal{D}) = \begin{bmatrix} \sum_{i=1}^N C_x^{(i)\top} C_x^{(i)} & \sum_{i=1}^N C_x^{(i)\top} C_{y_1}^{(i)} & \sum_{i=1}^N C_x^{(i)\top} C_{y_2}^{(i)} \\ \sum_{i=1}^N C_{y_1}^{(i)\top} C_x^{(i)} & \sum_{i=1}^N C_{y_1}^{(i)\top} C_{y_1}^{(i)} & \sum_{i=1}^N C_{y_1}^{(i)\top} C_{y_2}^{(i)} \\ \sum_{i=1}^N C_{y_2}^{(i)\top} C_x^{(i)} & \sum_{i=1}^N C_{y_2}^{(i)\top} C_{y_1}^{(i)} & \sum_{i=1}^N C_{y_2}^{(i)\top} C_{y_2}^{(i)} \end{bmatrix}.$$

Then, the real Lie algebra bases are $\{\mathrm{d}\rho_{\mathcal{X}_i}(A_j)\}_j = \{\mathrm{d}\rho_{y_1}(A_j)\}_j = \{\mathrm{d}\rho_{y_2}(A_j)\}_j = \left\{ \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right\}$.
In the ablation study, we process the output as a single-channel vector, which means the group representation on $\mathcal{Y}$ is a 9×9 matrix $\rho_{\mathcal{Y}}(g) \in \mathbb{R}^{9\times 9}$ instead of decomposing it into two 3×3 matrices $\rho_{\mathcal{Y}_1}(g) \in \mathbb{R}^{3\times 3}$ and $\rho_{\mathcal{Y}_2}(g) \in \mathbb{R}^{3\times 3}$. Then we transform Eq. (A.6) into:

$$C(\mathcal{D})v = \begin{bmatrix} -\sum_{j=1}^{c_x} \nabla f(x_j^{(1)}) \otimes x_j^{(1)\top} & I_m \otimes f^\top(x^{(1)}) \\ -\sum_{j=1}^{c_x} \nabla f(x_j^{(2)}) \otimes x_j^{(2)\top} & I_m \otimes f^\top(x^{(2)}) \\ \vdots & \vdots \\ -\sum_{j=1}^{c_x} \nabla f(x_j^{(N)}) \otimes x_j^{(N)\top} & I_m \otimes f^\top(x^{(N)}) \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) \\ \mathrm{vec}(\mathrm{d}\rho_{\mathcal{Y}}(A)) \end{bmatrix} = 0.$$

For quantitative error analysis, similar to Table 1 (in Section 7.2), we conduct five repeated experiments with different random seeds and report the average, minimum, and maximum values in Table 2 (in Section 7.3).

B.3. Top quark tagging

For this task, we directly utilize a publicly available reference dataset (Kasieczka et al., 2019). This dataset is generated using the PYTHIA8 Monte Carlo event generator to simulate particle collision events, and the ATLAS detector response is modeled using the DELPHES software package. Similar to the moment of inertia matrix prediction, we introduce uniformly distributed random noise of ±10% to the four-momenta of the jet constituents to verify the robustness of LieSD against real-world perturbations. Formally, for the original value of the four-momentum $(p_i^\mu)_{origin}$, the perturbed value is $(p_i^\mu)_{noise} = (p_i^\mu)_{origin} \times (1+\epsilon)$, where $\epsilon \sim \mathcal{U}(-0.1, 0, 1)$.

Top quark tagging takes the four-momenta $p_i^\mu \in \mathbb{R}^4$ of 20 jet constituents as input and predicts labels for the particles as output. We set the input dimension of a 3-layer MLP to 80, the hidden dimension to 200, and the output dimension to 1, choosing the ReLU function as the activation function. We use the Adan optimizer (Xie et al., 2024) for training, with the number of epochs set to 100. The size of the training set is 1,211,000, and we use $\frac{1}{10}$ of the training set for symmetry discovery. We perform this experiment on a single-core NVIDIA GeForce RTX 3090 GPU with available memory of 24,576 MiB, and the execution time is approximately 2 h.

For such problems with invariance, we strictly require $\rho_{\mathcal{Y}}(g) = I_m$, i.e., $\mathrm{d}\rho_{\mathcal{Y}}(A) = 0$. Additionally, the four-momentum of different jet constituents should share the same group representation. Setting $v_y = 0$ and $\mathrm{d}\rho_{\mathcal{X}_1}(A) = \cdots = \mathrm{d}\rho_{\mathcal{X}_{c_x}}(A) = \mathrm{d}\rho_{\mathcal{X}}(A)$ in Eq. (A.6), we derive the system of linear equations:

$$C(\mathcal{D})v = \begin{bmatrix} -\sum_{j=1}^{c_x} \nabla f(x_j^{(1)}) \otimes x_j^{(1)\top} \\ -\sum_{j=1}^{c_x} \nabla f(x_j^{(2)}) \otimes x_j^{(2)\top} \\ \vdots \\ -\sum_{j=1}^{c_x} \nabla f(x_j^{(N)}) \otimes x_j^{(N)\top} \end{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) = 0.$$

The computation of $C(\mathcal{D})^\top C(\mathcal{D})$ can be done as follows:

$$C(\mathcal{D})^\top C(\mathcal{D}) = \sum_{i=1}^N \sum_{j=1}^{c_x} \nabla f(x_j^{(i)})^\top \nabla f(x_j^{(i)}) \otimes x_j^{(i)} x_j^{(i)\top}.$$

Then we conduct quantitative error analysis. Five different random seeds are used for LieSD, and three for LieGAN. We report the average, minimum, and maximum values of space error and orthogonality error in Table 3 (in Section 7.4).

B.4. Rotated MNIST

Rotated MNIST takes 28×28 grayscale images of handwritten digits as input, with the goal of predicting the digit labels. We use LeNet-5 (LeCun et al., 1998) and the Adan optimizer (Xie et al., 2024) to train this classic image classification task. The size of the training set is 60,000, and we use the entire training set for symmetry discovery. We perform this experiment on a single-core NVIDIA GeForce RTX 3090 GPU with available memory of 24,576 MiB, and the execution time is approximately 30 min.

For image data, group transformations act on the planar coordinates $x \in \mathbb{R}^2$. All pixel coordinates $\{x_i\}_{i=1}^{n_{pixel}}$ share the same group transformation, which implies that we can treat it as a multi-channel case. A grayscale image can be formalized as a mapping from planar coordinates to grayscale values $I : \mathbb{R}^2 \to \mathbb{R}$, and the classifier is a mapping from the grayscale values of n_{pixel} pixels to the probabilities of n_{class} categories $f : \mathbb{R}^{n_{pixel}} \to \mathbb{R}^{n_{class}}$. By the chain rule, we can derive the gradient of $f(I_1, I_2, \ldots, I_{n_{pixel}}) = f(I(x_1), I(x_2), \ldots, I(x_{n_{pixel}}))$ with respect to x_i as $\nabla_{x_i} f(x_1, x_2, \ldots, x_{n_{pixel}}) = \nabla_{I_i} f(I_1, I_2, \ldots, I_{n_{pixel}}) \cdot \nabla_x I(x_i)$, where $\nabla_{I_i} f$ is obtained through automatic differentiation of the classifier, and $\nabla_x I$ is estimated using central differences. Substituting into Eq. (A.6) and setting $v_y = 0, \mathrm{d}\rho_{\mathcal{X}_1}(A) = \cdots = \mathrm{d}\rho_{\mathcal{X}_{c_x}}(A) = \mathrm{d}\rho_{\mathcal{X}}(A)$ similarly to top quark tagging, we have:

$$C(\mathcal{D})v = \begin{bmatrix} -\sum_{j=1}^{n_{pixel}} [\nabla_{I_j} f(I^{(1)}) \cdot \nabla_x I(x_j^{(1)})] \otimes x_j^{(1)\top} \\ -\sum_{j=1}^{n_{pixel}} [\nabla_{I_j} f(I^{(2)}) \cdot \nabla_x I(x_j^{(2)})] \otimes x_j^{(2)\top} \\ \vdots \\ -\sum_{j=1}^{n_{pixel}} [\nabla_{I_j} f(I^{(N)}) \cdot \nabla_x I(x_j^{(N)})] \otimes x_j^{(N)\top} \end{bmatrix} \mathrm{vec}(\mathrm{d}\rho_{\mathcal{X}}(A)) = 0.$$

When conducting quantitative error analysis, we repeat experiments with three different random seeds. We report the average, minimum, and maximum values in Table 4 (in Section 7.5).

Data availability

Data will be made available on request.

References

Allingham, J. U., Mlodozeniec, B. K., Padhy, S., Antorán, J., Krueger, D., Turner, R. E., Nalisnick, E., & Hernández-Lobato, J. M. (2024). A generative model of symmetry transformations. arXiv preprint arXiv:2403.01946.

Benton, G., Finzi, M., Izmailov, P., & Wilson, A. G. (2020). Learning invariances in neural networks from training data. *Advances in Neural Information Processing Systems*, *33*, 17605–17616.

Cohen, T., & Welling, M. (2016a). Group equivariant convolutional networks. In *International conference on machine learning* (pp. 2990–2999). PMLR.

Cohen, T. S., & Welling, M. (2016b). Steerable CNNs. arXiv preprint arXiv:1612.08498.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers. Transactions on Information Theory*, *13*(1), 21–27.

Dehmamy, N., Walters, R., Liu, Y., Wang, D., & Yu, R. (2021). Automatic symmetry discovery with Lie algebra convolutional network. *Advances in Neural Information Processing Systems*, *34*, 2503–2515.

Desai, K., Nachman, B., & Thaler, J. (2022). Symmetry discovery with deep learning. *Physical Review D*, *105*(9), Article 096031.

Dombrowski, A.-K., Gerken, J. E., Müller, K.-R., & Kessel, P. (2023). Diffeomorphic counterfactuals with generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(5), 3257–3274.

Eilertsen, G., Jönsson, D., Ropinski, T., Unger, J., & Ynnerman, A. (2020). Classifying the classifier: dissecting the weight space of neural networks. In *ECAI 2020* (pp. 1119–1126). IOS Press.

Finzi, M., Welling, M., & Wilson, A. G. (2021). A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning* (pp. 3318–3328). PMLR.

Greydanus, S., Dzamba, M., & Yosinski, J. (2019). Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, *32*.

He, L., Chen, Y., Shen, Z., Yang, Y., & Lin, Z. (2022). Neural ePDOs: Spatially adaptive equivariant partial differential operator based networks. In *The eleventh international conference on learning representations*.

Kasieczka, G., Plehn, T., Thompson, J., & Russel, M. (2019). Top quark tagging reference dataset. http://dx.doi.org/10.5281/zenodo.2603256.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Krippendorf, S., & Syvaeri, M. (2020). Detecting symmetries with neural networks. *Machine Learning: Science and Technology*, *2*(1), Article 015010.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, Y., Qiu, Y., Chen, Y., He, L., & Lin, Z. (2024). Affine equivariant networks based on differential invariants. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5546–5556).

MacDonald, L. E., Ramasinghe, S., & Lucey, S. (2022). Enabling equivariance for arbitrary Lie groups. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8183–8192).

Moskalev, A., Sepliarskaia, A., Sosnovik, I., & Smeulders, A. (2022). LieGG: Studying learned Lie group generators. *Advances in Neural Information Processing Systems*, *35*, 25212–25223.

Navon, A., Shamsian, A., Achituve, I., Fetaya, E., Chechik, G., & Maron, H. (2023). Equivariant architectures for learning in deep weight spaces. In *International conference on machine learning* (pp. 25790–25816). PMLR.

van der Ouderaa, T., Immer, A., & van der Wilk, M. (2024). Learning layer-wise equivariances automatically using gradients. *Advances in Neural Information Processing Systems*, *36*.

Romero, D. W., & Lohit, S. (2022). Learning partial equivariances from data. *Advances in Neural Information Processing Systems*, *35*, 36466–36478.

Satorras, V. G., Hoogeboom, E., & Welling, M. (2021). E (n) equivariant graph neural networks. In *International conference on machine learning* (pp. 9323–9332). PMLR.

Schürholt, K., Kostadinov, D., & Borth, D. (2021). Self-supervised representation learning on neural network weights for model characteristic prediction. *Advances in Neural Information Processing Systems*, *34*, 16481–16493.

Shen, Z., He, L., Lin, Z., & Ma, J. (2020). PDO-econvs: Partial differential operator based equivariant convolutions. In *International conference on machine learning* (pp. 8697–8706). PMLR.

Shen, Z., Hong, T., She, Q., Ma, J., & Lin, Z. (2022). PDO-s3DCNNs: Partial differential operator based steerable 3D CNNs. In *International conference on machine learning* (pp. 19827–19846). PMLR.

Shen, Z., Shen, T., Lin, Z., & Ma, J. (2021). PDO-eS2CNNs: Partial differential operator based equivariant spherical CNNs. *vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 9585–9593).

Tegnér, G., & Kjellstrom, H. (2023). Self-supervised latent symmetry discovery via class-pose decomposition. In *NeurIPS 2023 workshop on symmetry and geometry in neural representations*.

Unterthiner, T., Keysers, D., Gelly, S., Bousquet, O., & Tolstikhin, I. (2020). Predicting neural network accuracy from weights. arXiv preprint arXiv:2002.11448.

Weiler, M., & Cesa, G. (2019). General E (2)-equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, *32*.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., & Cohen, T. S. (2018). 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, *31*.

Weiler, M., Hamprecht, F. A., & Storath, M. (2018). Learning steerable filters for rotation equivariant CNNs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 849–858).

Xie, X., Zhou, P., Li, H., Lin, Z., & Yan, S. (2024). Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yang, J., Dehmamy, N., Walters, R., & Yu, R. (2023). Latent space symmetry discovery. arXiv preprint arXiv:2310.00105.

Yang, J., Walters, R., Dehmamy, N., & Yu, R. (2023). Generative adversarial symmetry discovery. In *International conference on machine learning* (pp. 39488–39508). PMLR.

Zhou, A., Knowles, T., & Finn, C. (2020). Meta-learning symmetries by reparameterization. arXiv preprint arXiv:2007.02933.