Deep learning symmetries and their Lie groups, algebras, and subalgebras from first principles

MACHINE LEARNING Science and Technology

OPEN ACCESS

RECEIVED
13 February 2023

REVISED
12 May 2023

ACCEPTED FOR PUBLICATION
26 May 2023

PUBLISHED
8 June 2023

PAPER

Deep learning symmetries and their Lie groups, algebras, and subalgebras from first principles

Roy T Forestano, Konstantin T Matchev[*], Katia Matcheva, Alexander Roman, Eyup B Unlu and Sarunas Verner

Institute for Fundamental Theory, Physics Department, University of Florida, Gainesville, FL 32611, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: matchev@ufl.edu

Keywords: symmetries, subalgebras, machine-learning, Lorentz group, Lie algebras, supervised learning, loss functions

Abstract

We design a deep-learning algorithm for the discovery and identification of the continuous group of symmetries present in a labeled dataset. We use fully connected neural networks to model the symmetry transformations and the corresponding generators. The constructed loss functions ensure that the applied transformations are symmetries and the corresponding set of generators forms a closed (sub)algebra. Our procedure is validated with several examples illustrating different types of conserved quantities preserved by symmetry. In the process of deriving the full set of symmetries, we analyze the complete subgroup structure of the rotation groups $SO(2)$, $SO(3)$, and $SO(4)$, and of the Lorentz group $SO(1,3)$. Other examples include squeeze mapping, piecewise discontinuous labels, and $SO(10)$, demonstrating that our method is completely general, with many possible applications in physics and data science. Our study also opens the door for using a machine learning approach in the mathematical study of Lie groups and their properties.

1. Introduction

Symmetries play a fundamental role in modern physics [1]. Physical systems with continuous symmetries exhibit conservation laws that are universally applicable and indispensable in understanding the system's behavior and evolution. In particle physics, symmetries provide an organizing principle behind the observed particle zoo and its interactions, and guide model-builders in the search for viable extensions of the standard model (SM) [2]. At the same time, the mathematical study of symmetries is interesting in its own right and has a rich history.

Over the last decade, there has been increased interest in applications of machine learning (ML) to high-dimensional physics data as a sensitive tool for event simulation, data analysis, and statistical inference [3–5]. More recently, ML is also being used to facilitate tasks that traditionally have fallen within the domain of theorists, e.g. performing symbolic computations [6, 7] or deriving analytical formulas by training a symbolic regression on synthetic data [8–20].

Applications of ML to the study of symmetries have been pursued by a number of groups in different contexts. One line of work investigates how a given symmetry is reflected in a learned representation of the data [21, 22] or in the ML architecture itself, e.g. in the embedding layer of a neural network (NN) [23]. Several proposals attempt to design special ML architectures (equivariant NNs) which have a desired symmetry property built in from the outset [24–30] and test their performance [31]. Incorporating the symmetry directly into the ML model makes it more economical (in terms of learned representations), interpretable and trainable. The approach can be extended to cover discrete (permutation) symmetries as well [32, 33]. Such efforts pave the way for data-driven blind searches for new physics which stress-test the data for violations of a well-established symmetry of the SM [34–36].

More recently, ML is also being applied to address more formal theoretical questions. For example, a good understanding of the symmetries present in the problem can reveal conserved quantities [37, 38] or hint at a more fundamental unified picture [39]. ML has been used to discover the symmetry of a potential [23, 40, 41], to decide whether a given pair of inputs is related by symmetry or not [42], to distinguish between scale-invariant and conformal symmetries [43], and to explore the string landscape [44–46]. Recent work made use of generative adversarial networks to learn transformations that preserve probability distributions [47]. ML applications have also found their way into group theory, which provides the abstract mathematical language of symmetries. For example, recent work used ML to compute tensor products and branching rules of irreducible representations of Lie groups [48] and to obtain Lie group generators of a symmetry present in the data [49, 50].

The main goal of this paper is to design a deep-learning method that mimics the traditional theorist's thinking and is capable of discovering and categorizing the full set of (continuous) symmetries in a given dataset from first principles, i.e. without any prior assumptions or prejudice. The only input to our procedure is a labeled dataset $\{\mathbf{x};y\}$ like the one in equation (1) below. An oracle $\varphi(\mathbf{x}) = y$ can then be learned from the dataset, or alternatively, can be provided externally. With those ingredients, we go through the following objectives:

- **Discovery of symmetries.** In the first step, described in section 3.1, we learn to generate a symmetry transformation, $\mathbf{x} \xrightarrow{\mathbf{f}} \mathbf{x}'$, which preserves the oracle values. The transformation $\mathbf{f}$ is encoded in a NN trained on the dataset. In general, there will be many possible symmetry transformations $\mathbf{f}$, and their study and categorization from a group theory point of view is the main goal of this paper.
- **Discovery of symmetry generators.** Having learned how to generate arbitrary (finite) symmetry transformations $\mathbf{f}$, we then focus on infinitesimal transformations $\delta\mathbf{f}$, which give us in turn the symmetry generators $\mathbf{J}$.
- **Discovery of Lie subalgebras.** By adding suitable terms to the loss function, our procedure requires that the learned set of generators $\{\mathbf{J}_\alpha\}$ forms a closed algebra. This allows us to discover subalgebras of the symmetry group, and identify them by their structure constants. Since the number of generators N_g is a free input, in cases when the training is unsuccessful (as quantified by the loss), we can rule out the existence of N_g-dimensional subalgebras.
- **Discovery of the full Lie algebra.** The maximum value of N_g which gives a vanishing loss, indicates the dimension of the full Lie algebra. The corresponding learned set of generators describes the full symmetry group of the dataset.
- **Identification of the symmetry group and its subgroups.** The learned sets of generators found in the previous two steps are then used to obtain the structure constants of the respective full algebra and its subalgebras, and thus to identify the corresponding symmetry group and its subgroups.

Our study complements and extends previous related work in [23, 40, 41, 49, 50]. We note that our procedure is completely general, and does not anticipate what symmetries might be present in the dataset—instead, the symmetries are learned from scratch. In addition, our method is basis-independent since we do not choose a specific convenient basis for the learned transformations and generators. Consequently, our results will not always match the nice canonical forms of the generators found in the group theory textbooks. Nevertheless, as the examples below explicitly demonstrate, all our learned transformations and generators will satisfy the defining properties of the respective symmetry groups and subgroups.

The paper is organized as follows. In section 2 we introduce the setup for our analysis and the corresponding notation and conventions. The main steps of our deep-learning procedure are outlined in section 3, which also explains and motivates the necessary ingredients for the loss function. For readers who are not yet fully comfortable with a deep-learning approach, section 4 outlines an analogous linear algebra approach that often (but not always—see section 8 for counterexamples) can accomplish similar objectives. Each of the remaining sections contains a separate completely worked-out example illustrating our method. The examples are distinguished by the choice of oracle and the dimensionality of the feature space. In section 5 we choose an oracle that returns the Euclidean distance in feature space, whose dimensionality, in turn, is chosen to be $n = 2$ in section 5.1, $n = 3$ in section 5.2, and $n = 4$ in section 5.3. In section 6 we focus on the Lorentz group, i.e. the oracle computes the pseudo-Euclidean distance in four-dimensional Minkowski space-time. In section 7 we consider the squeeze transformations in $n = 2$ dimensions whereby the oracle returns the product of the two features. To demonstrate the universal applicability of our technique, in section 8 we show two examples of discontinuous oracles. We summarize and conclude in section 9.

2. Setup and notations

Our starting point is a labeled dataset containing m samples of n features and a target label y:

$$m \text{ samples} \left\{ \begin{array}{ccccc} x_1^{(1)}, & x_1^{(2)}, & \dots, & x_1^{(n)}; & y_1 \\ x_2^{(1)}, & x_2^{(2)}, & \dots, & x_2^{(n)}; & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_m^{(1)}, & x_m^{(2)}, & \dots, & x_m^{(n)}; & y_m \end{array} \right. . \tag{1}$$

In ML parlance, the dataset (1) is an $m \times n$ dataframe with n features and m samples. In what follows, we use the sample index a lot more often than the feature index, thus we use an explicit subscript for the sample index and hide the feature index in the boldface vector notation $\mathbf{x}$:

$$\mathbf{x} \equiv \left\{ x^{(1)}, x^{(2)}, \dots, x^{(n)} \right\}. \tag{2}$$

This allows us to write the input features in a compact form as

$$\{\mathbf{x}_i\} \; \equiv \; \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}\,. \tag{3}$$

In order to study the symmetries of the data (1), we need to know the function $y(\mathbf{x})$, which can be given analytically or numerically in terms of an oracle $\varphi : \mathbb{R}^n \to \mathbb{R}$ capable of computing the corresponding output target labels $y_1, y_2, \dots, y_m$:

$$y_i \; = \; \varphi(\mathbf{x}_i)\,, \qquad i \; = \; 1, 2, \dots, m\,. \tag{4}$$

This leads to two basic scenarios:

- The function $y(\mathbf{x})$ is known analytically, and that same function has been used as in (4) to compute the labels in (1) exactly. This case is of interest to theorists, and this is the approach adopted in this paper as well—for each of our examples below, we specify the relevant analytic oracle $\varphi(\mathbf{x})$ and proceed to study the resulting symmetries. Note, however, that we never take advantage of the knowledge of the analytical form of the oracle, e.g. we do not differentiate or symbolically manipulate in any other way the function $\varphi(\mathbf{x})$. For our purposes, we only use the oracle numerically—our method treats it simply as a black box, which, given the values of $\mathbf{x}$, can produce the numerical value of the label y.
- The functional dependence $y(\mathbf{x})$ is *a priori* unknown and the dataset (1) is all that is available to us. This is the typical case encountered by data scientists. Now, one needs to go through a preliminary step of first creating the oracle φ (usually in the form of a NN trained on the dataset (1)), which is capable of computing and reporting the values of $y = \varphi(\mathbf{x})$ to us. This is a standard regression task which is of no interest here since it can be accomplished using one of the many established ML regression techniques. We can therefore safely assume that this preliminary step has already been completed and we have such a numerical oracle $y = \varphi(\mathbf{x})$ already available.

Since we are only using the oracle numerically, from our point of view there is no real difference between the above two cases. In what follows the oracle $\varphi(\mathbf{x})$ will be used in the exact same way, regardless of its origin.

Given this general setup, our main task is to derive the symmetry transformation $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$

$$\mathbf{x}' \; = \; \mathbf{f}(\mathbf{x})\,, \tag{5}$$

which preserves the φ-induced labels of our dataset (1). In other words, we want to find the function $\mathbf{f}(\mathbf{x})$ for which

$$\varphi(\mathbf{x}_i') \; \equiv \; \varphi(\mathbf{f}(\mathbf{x}_i)) = \varphi(\mathbf{x}_i), \quad \forall i \; = \; 1, 2, \dots, m\,. \tag{6}$$

The particular instantiation of the symmetry $\mathbf{f}(\mathbf{x})$ will depend on the initialization of our parameters, so by repeating the procedure with different initializations, we will in principle obtain a whole family of symmetry transformations.

Next, we focus on infinitesimal symmetry transformations and proceed to study the corresponding set of generators $\{\mathbf{J}_\alpha\}$, with $\alpha = 1, 2, \dots, N_g$, where we use lowercase Greek letters to label the generators of symmetry transformations. A given set of generators $\{\mathbf{J}_\alpha\}$ forms a Lie algebra if the closure condition is satisfied, i.e. if all Lie brackets $[\,.\,,\,.\,]$ can be represented as linear combinations of the generators in the set:

$$[\mathbf{J}_\alpha, \mathbf{J}_\beta] = \sum_{\gamma=1}^{N_g} a_{[\alpha\beta]\gamma} \mathbf{J}_\gamma. \tag{7}$$

The coefficients $a_{[\alpha\beta]\gamma}$ are the structure constants (anti-symmetric in their first two indices, as implied by the square brackets) whose values will reveal the symmetry group present in our dataset.

In principle, the number of generators N_g is a hyperparameter that must be specified ahead of time (similarly to the choice of the number of clusters in certain clustering algorithms like K-means). Therefore, when we find a closed algebra at a given N_g value, we are only guaranteed that it is a subalgebra, and we must proceed to try out higher values for N_g as well. The full algebra will then correspond to the maximum value of N_g for which a closed algebra of distinct generators is found to exist.

3. Deep learning approach

In our approach, we model the output function $\mathbf{f}$ with a NN $\mathcal{F}_\mathcal{W}$ with n neurons in the output layer, corresponding to the n transformed features of the data point $\mathbf{x}'$. The trainable network parameters (weights and biases) will be generically denoted with $\mathcal{W}$. During training, they will evolve and converge to the corresponding trained values $\widehat{\mathcal{W}}$ of the parameters of the trained network $\mathcal{F}_{\widehat{\mathcal{W}}}$, i.e. the hat symbol will denote the result of the training. Once the parameters $\widehat{\mathcal{W}}$ are found, we can find the structure constants using SMs. We choose a suitable loss function that ensures the desired properties of the trained network. The following subsections discuss the individual contributions to the loss function, which in our implementation are combined and minimized simultaneously.

The NN $\mathcal{F}_\mathcal{W}$ is implemented as a sequential feed-forward neural network in PyTorch [51]. The examples in sections 5 and 6 are simple enough to be done with no hidden layers, no bias, and no activation function, i.e. with linear transformations (see section 4). For the examples in the later sections, we do add hidden layers. Optimizations are performed with the Adam optimizer with a learning rate between 0.03 and 0.1. The loss functions were designed to achieve a fast and efficient training process without the need for extensive hyperparameter tuning. The training data (3) was 300 hundred points and was sampled from a standard normal distribution.

An alternative approach to predicting the generators from the model parameters, which utilizes an identical loss function, can be carried out where a NN, $\mathcal{F} : \{\mathcal{G}_\alpha, \mathbf{a}\} \to \{\widehat{\mathcal{G}}_\alpha, \hat{\mathbf{a}}\}$, takes a set of randomly initialized $n \times n$ generators $\{\mathcal{G}_\alpha\}$ and an $N_b \times N_g$ structure constant array $\{\mathbf{a}\}$, flattens each individual generator and the structure constant array, and proceeds to converge the elements to the desired set of generators $\{\widehat{\mathcal{G}_\alpha}\}$ and structure constant array $\{\hat{\mathbf{a}}\}$. Here $N_b = \binom{N_g}{2} = \frac{N_g(N_g-1)}{2}$ denotes the number of unique one-directional Lie brackets for a given number of generators N_g.

This alternative approach can be implemented as a module list of sequential NN layers consisting of two hidden layers for the generators and a single sequential layer consisting of two hidden layers for the structure constants. Each layer consists of a bias and the hidden layers use the rectified linear unit activation function. The optimizer, average learning rate, model hyperparameters, and training data generation were identical to the alternative implementation described above. Whereas the original method feeds the data directly into the network, the data in this approach is fed into the loss function to be transformed by the model's predicted generators.

3.1. Invariance

The invariance under the transformation (5) is enforced by requiring that the labels before and after the transformation remain the same. For this purpose, we include the following mean squared error term in the loss function L:

$$L_{\mathrm{inv}}(\mathcal{W}, \{\mathbf{x}_i\}) = \frac{1}{m} \sum_{i=1}^{m} \left[\varphi\left(\mathcal{F}_\mathcal{W}(\mathbf{x}_i)\right) - \varphi(\mathbf{x}_i)\right]^2 . \tag{8}$$

A NN trained with this loss function will find an arbitrarily general (finite) symmetry transformation $\mathcal{F}_{\widehat{\mathcal{W}}}$ parametrized by the values of the trained network parameters $\widehat{\mathcal{W}}$.

In order to find multiple symmetries, the process can be repeated. Alternatively, several networks can be trained concurrently, by modifying the loss function to ensure that the resulting transformations are sufficiently distinct (see section 3.3 below).

3.2. Infinitesimality

In order to focus on the generators of the possible set of symmetry transformations, we restrict ourselves to infinitesimal transformations $\delta\mathcal{F}$ in the vicinity of the identity transformation $\mathbf{I}$:

$$\delta\mathcal{F} \equiv \mathbf{I} + \varepsilon\mathcal{G}_{\mathcal{W}}, \tag{9}$$

where ε is an infinitesimal parameter and the parameters $\mathcal{W}$ of the new NN $\mathcal{G}$ will be forced to be finite, which ensures that (9) is an infinitesimal transformation. The loss function (8) can then be rewritten as

$$L_{\text{inf}}(\mathcal{W}, \{\mathbf{x}_i\}) = \frac{1}{m\varepsilon^2}\sum_{i=1}^{m}\left[\varphi(\mathbf{x}_i + \varepsilon\mathcal{G}_{\mathcal{W}}(\mathbf{x}_i)) - \varphi(\mathbf{x}_i)\right]^2, \tag{10}$$

where we have introduced an explicit factor of ε^2 in the denominator to account for the fact that generic transformations scale as ε [41]. Once we minimize the loss function, the trained NN $\mathcal{G}_{\widehat{\mathcal{W}}}$ will represent a corresponding generator

$$\mathbf{J} = \mathcal{G}_{\widehat{\mathcal{W}}}, \tag{11}$$

where

$$\widehat{\mathcal{W}} \equiv \underset{\mathcal{W}}{\arg\min}\left(L(\mathcal{W}, \{\mathbf{x}_i\})\right) \tag{12}$$

are the learned values of the NN parameters which minimize the loss function. The result for $\widehat{\mathcal{W}}$, and therefore, for $\mathbf{J}$, will in principle depend on the starting values $\mathcal{W}_0$ of the network parameters at initialization. If we now repeat the training N_g times under different initial conditions $\mathcal{W}_0$ (or with different values of the hyperparameters), we will obtain a set of N_g (in general distinct) generators $\{\mathbf{J}_\alpha\}$, $\alpha = 1, 2, \ldots, N_g$.

3.3. Orthogonality

In order to make the set of generators $\{\mathbf{J}_\alpha\}$, $\alpha = 1, 2, \ldots, N_g$, found in the previous step maximally different, we introduce the following additional orthogonality term to the loss function

$$L_{\text{ortho}}(\mathcal{W}, \{\mathbf{x}_i\}) = \sum_{\alpha<\beta}^{N_g}\left(\sum_p \mathcal{W}_\alpha^{(p)}\mathcal{W}_\beta^{(p)}\right)^2, \tag{13}$$

where the index p runs over the individual NN parameters $\mathcal{W}_\alpha^{(p)}$.

3.4. Closure

In order to test whether a certain set of distinct generators $\{\mathbf{J}_\alpha\}$, $\alpha = 1, 2, \ldots, N_g$, found in the previous steps, generates a group, we need to check the closure of the algebra (7). This can be done in several ways. The most principled would be to minimize

$$L_{\text{closure}}(a_{[\alpha\beta]\gamma}) = \sum_{\alpha<\beta}\text{Tr}\left(\mathbf{C}_{[\alpha\beta]}^T\mathbf{C}_{[\alpha\beta]}\right), \tag{14}$$

with respect to the candidate structure constant parameters $a_{[\alpha\beta]\gamma}$, where the closure mismatch is defined by

$$\mathbf{C}_{[\alpha\beta]}(a_{[\alpha\beta]\gamma}) \equiv \left[\mathbf{J}_\alpha, \mathbf{J}_\beta\right] - \sum_{\gamma=1}^{N_g} a_{[\alpha\beta]\gamma}\mathbf{J}_\gamma. \tag{15}$$

Since L_{closure} is positive semi-definite, $L_{\text{closure}} = 0$ would indicate that the algebra is closed and we are thus dealing with a genuine (sub)group. In practice[1], we perform the minimization of (14) simultaneously with the previously discussed contributions to the loss function, by replacing $\mathbf{J}_\alpha \to \mathcal{G}_{\mathcal{W}_\alpha}$ in (15). The advantage of this simultaneous construction is that every set of generators that we obtain at any given value of N_g is

[1] Another possible approach is to minimize the out-of-space components of the commutators with respect to the space of generators, after flattening and Gram–Schmidt orthonormalization.

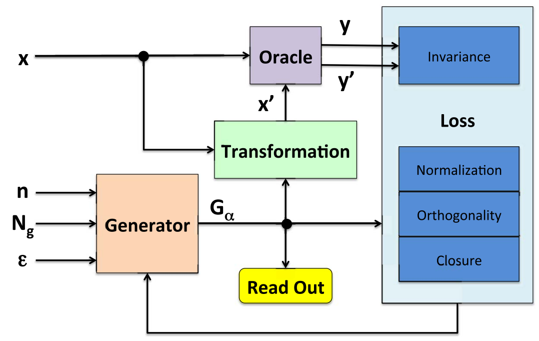

Figure 1. A flowchart of the symmetry finding procedure outlined in sections 3 and 4.

already forming an algebra that is 'as closed as possible'. Then, the size of the achieved total training loss is an indicator whether for that value of N_g a closed algebra exists or not.

Once a closed set of valid generators has been found, we can retrain the NN in a conveniently chosen canonical basis and obtain the canonical form of the set of generators, whose structure constants in turn reveal the nature of the group behind the found symmetry transformations $\mathcal{G}_{\widehat{\mathcal{W}}_\alpha}$, $\alpha = 1, 2, \ldots, N_g$.

Sections 5.1–8 illustrate the steps above with several examples of increasing complexity. A flowchart of our symmetry finding procedure is shown in figure 1. The chosen values for n (the dimensionality of the feature space), N_g (the requested number of independent symmetry generators) and ε (the parameter of the infinitesimal symmetry transformation (9), taken to be $\varepsilon = 0.001$ in the numerical experiments below) are passed to the generator module parametrizing the transformation in terms of either a NN $\mathcal{F}_{\mathcal{W}}$ (section 3) or a matrix $\mathbb{G}$ (section 4). The parameters of the transformation are directly accessible to the module evaluating the loss function. Next we create the sample of feature vectors (3), ideally aiming for a complete coverage of the domain of interest (for our numerical experiments below, we sample 300 points from a standard normal distribution). These feature vectors are then processed using the current state of the generator into the transformed feature set $\{\mathbf{x}_i'\}$. Both $\{\mathbf{x}_i\}$ and $\{\mathbf{x}_i'\}$ are subsequently passed to the oracle φ for the evaluation of the respective sets of target labels $\{y_i\}$ and $\{y_i'\}$. All of this information is fed into the loss module which in turns informs the generator through the backpropagation feedback loop about the required adjustments in the values of the generator parameters. When the process converges (by either reaching a target value for the loss, or after a sufficiently large number of epochs), the N_g symmetry generators can be read out directly from the current state of the generator module.

4. Linear algebra approach

The universal approximation theorems [52] guarantee that the deep-learning approach of the previous section can handle almost any symmetry transformation, including a highly non-linear one. At the same time, a large class of interesting symmetries arising in physics are linear transformations for which the usual formalism of linear algebra would suffice. Furthermore, the analysis of the symmetry generators involves infinitesimal transformations, which are represented with linear operators. For those reasons and to captivate the readers who are not yet at ease with the technical intricacies of ML, in this section, we reformulate our analysis from the previous section in the language of linear algebra. We follow the same notation and conventions, but replace the calligraphic font symbols representing NNs with corresponding blackboard-bold symbols representing $n \times n$ matrices acting on the n-dimensional feature space.

In this section we are interested in the linear subclass of the transformations (5), which are encoded in a generic matrix $\mathbb{F}$

$$\mathbf{x}' = \mathbb{F}\,\mathbf{x}, \tag{16}$$

whose n^2 components $\mathbb{F}_{ij}$ are determined by minimizing the loss function

$$L_{\text{inv}}(\mathbb{F}, \{\mathbf{x}_i\}) = \frac{1}{m}\sum_{i=1}^{m}\left[\varphi\left(\mathbb{F}\,\mathbf{x}_i\right) - \varphi(\mathbf{x}_i)\right]^2. \tag{17}$$

If the minimization is successful, then such a linear symmetry exists and is represented with the learned matrix $\widehat{\mathbb{F}}$.

In analogy to (9), we can write the corresponding infinitesimal linear transformation as

$$\delta\mathbb{F}(\varepsilon) = \mathbb{I} + \varepsilon\,\mathbb{G}\,, \tag{18}$$

where $\mathbb{I}$ is the unit $n\times n$ matrix and $\mathbb{G}$ is an $n\times n$ matrix whose components $\mathbb{G}_{ij}$ are yet to be determined through the optimization. In order to obtain a single generator matrix $\mathbb{J}$, we can use a loss function analogous to (10)

$$L_{\mathbb{J}}(\mathbb{G},\{\mathbf{x}_i\}) = \frac{h_{\text{inv}}}{m\varepsilon^2}\sum_i^m\left[\varphi\left(\mathbf{x}_i+\varepsilon\,\mathbb{G}\,\mathbf{x}_i\right)-\varphi(\mathbf{x}_i)\right]^2 + h_{\text{norm}}\left[\text{Tr}(\mathbb{G}^T\mathbb{G})-2\right]^2, \tag{19}$$

where the constants h_{inv} and h_{norm} in (19) are hyperparameter weights determining the relative contribution of the two terms in the loss function (19) enforcing the symmetry invariance and finite normalization[2] conditions, respectively. The actual generator $\mathbb{J}$ is then obtained by minimizing (19):

$$\mathbb{J} = \underset{\mathbb{G}}{\arg\min}\left(L_{\mathbb{J}}(\mathbb{G},\{\mathbf{x}_i\})\right). \tag{20}$$

By repeating this procedure several times with different initial conditions, we obtain a different generator $\mathbb{J}$ each time. Alternatively, we can produce all N_g generators in one go by adding together and minimizing simultaneously N_g copies of the loss function (19):

$$\sum_{\alpha=1}^{N_g} L_{\mathbb{J}}(\mathbb{G}_\alpha,\{\mathbf{x}_i\})\,, \tag{21}$$

which will lead to a set of N_g generators $\mathbb{J}_\alpha$, $\alpha = 1,2,\ldots,N_g$, and their respective infinitesimal transformations $\delta\mathbb{F}_\alpha \equiv \mathbb{I}+\varepsilon\,\mathbb{J}_\alpha$.

At this point the generators $\mathbb{J}_\alpha$ are completely decoupled and independent of each other. We can force them to be different by adding a loss term analogous to (13):

$$L_{\text{ortho}}(\mathbb{G},\{\mathbf{x}_i\}) = \sum_{\alpha<\beta}^{N_g}\left[\text{Tr}(\mathbb{G}_\alpha^T\mathbb{G}_\beta)\right]^2. \tag{22}$$

Finally, we can enforce the closure property by including a loss term analogous to (14):

$$L_{\text{closure}}(a_{[\alpha\beta]\gamma}) = \sum_{\alpha<\beta}\text{Tr}\left(\mathbb{C}_{[\alpha\beta]}^T\mathbb{C}_{[\alpha\beta]}\right), \tag{23}$$

where

$$\mathbb{C}_{[\alpha\beta]}(a_{[\alpha\beta]\gamma}) \equiv \left[\mathbb{G}_\alpha,\mathbb{G}_\beta\right] - \sum_{\gamma=1}^{N_g} a_{[\alpha\beta]\gamma}\mathbb{G}_\gamma\,, \tag{24}$$

and minimizing the total loss function with respect to the parameters $a_{[\alpha\beta]\gamma}$ as well.

5. Length-preserving transformations

In this section, we focus on transformations which preserve the Euclidean length of the feature vector, i.e. our oracle φ will return

$$\varphi(\mathbf{x}) \equiv |\mathbf{x}| = \left\{\sum_{i=1}^n [x^{(i)}]^2\right\}^{1/2}. \tag{25}$$

In this case we expect our method to discover the symmetry of the orthogonal group $O(n)$, whose generators can be written in terms of Kronecker deltas as

$$(\mathbb{O}_{mn})^{ij} = \delta^i_m\delta^j_n - \delta^j_m\delta^i_n\,, \tag{26}$$

[2] In order for (18) to be an infinitesimal transformation, the matrix $\mathbb{G}$ needs to be finite. We choose to normalize our generators as $\text{Tr}(\mathbb{G}_\alpha^T\mathbb{G}_\beta) = 2\delta_{\alpha\beta}$, hence the factor of 2 in the second line of (19).

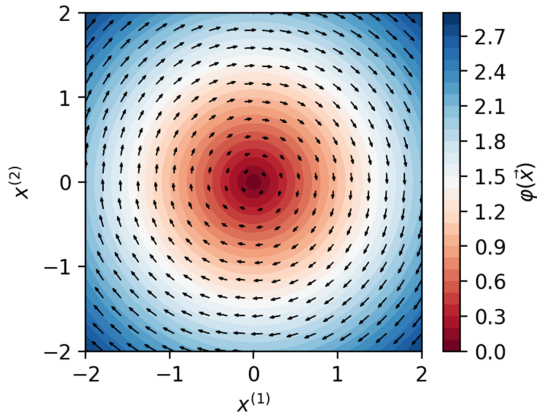

Figure 2. A representative symmetry transformation preserving the oracle function (25), found by our procedure in the two-dimensional exercise in section 5.1. In this and all subsequent such figures, the arrows represent the displacements $\mathbf{x}' - \mathbf{x}$ resulting from the symmetry transformation. The color map illustrates the oracle function $\varphi(\mathbf{x})$.

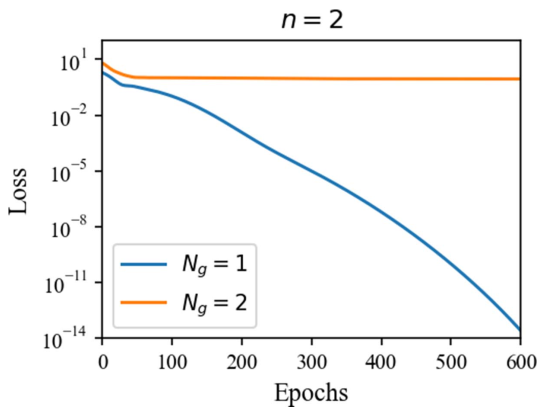

Figure 3. The evolution of the training loss with the number of epochs for the two-dimensional exercise in section 5.1. Our algorithm finds one symmetry generator (the training loss shown in blue steadily decreases) but not two different symmetry generators (the training loss shown in orange stays large).

in which case the algebra is given by

$$\left[\mathbb{O}_{mn}, \mathbb{O}_{pq}\right] = \delta_{np}\mathbb{O}_{mq} + \delta_{mq}\mathbb{O}_{np} - \delta_{mp}\mathbb{O}_{nq} - \delta_{nq}\mathbb{O}_{mp}. \tag{27}$$

Note that the generators $\mathbb{O}_{mn}$ are labeled by two indices, which indicate the plane of rotation.

5.1. Rotations in two dimensions

In this subsection we start with the simplest case of two dimensions, $n = 2$, which should correspond to the single-generator group $O(2)$. First we try to generate a single generic (i.e. not necessarily infinitesimal) symmetry transformation. For this purpose, we train our network $\mathcal{F}_{\mathcal{W}}$ with the invariance loss (8). This exercise was successful and, depending on the initialization, we found various symmetry transformations. They all involved a rotation around the origin in the $(x^{(1)}, x^{(2)})$ plane. One representative symmetry transformation is depicted in figure 2.

Next, we turn to the algebra of symmetry generators. We begin with a single generator, $N_g = 1$, in which case we do not need to include the orthogonality and closure terms in the loss function. The training is successful and the loss function is driven to zero, as seen by the blue solid line in figure 3. The resulting generator in matrix form is pictorially visualized in the top row of figure 4, and we immediately recognize the familiar matrix $\mathbb{O}_{12}$ from (26) generating rotations in the 12-plane

$$\mathbb{O}_{12} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{28}$$

Note that the generator has the expected antisymmetric property.

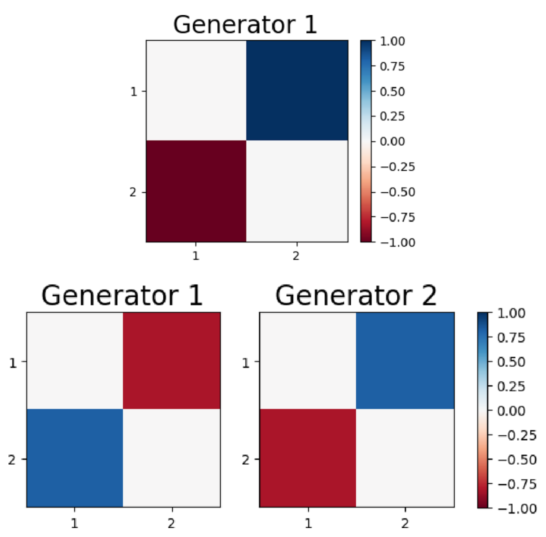

Figure 4. Top row: a successfully learned generator for $n = 2$ and $N_g = 1$. Bottom row: unsuccessfully learned 'generators' for $n = 2$ and $N_g = 2$. In this and all subsequent such figures, each panel represents a learned generator $\mathbb{J}_\alpha$ in matrix form, where the values of the individual elements of the matrix are indicated by the color bar.

Having found one symmetry generator, we next check if there is a second distinct generator. For this purpose, we add the orthogonality and closure terms in the loss function and repeat the training. This time the training is unsuccessful and the loss flattens after about 50 epochs, as seen by the orange solid line in figure 3. In the second row of figure 4 we show the result for the two candidate generators found in this case. We note that while they do have the expected form for a generator of rotations in two dimensions, they are essentially the same transformation, and differ only by an overall sign. This implies that they fail the orthogonality condition—indeed, we find that the dominant contribution to the large total loss in that case is from the orthogonality loss (13). Since the total loss is large and does not improve with further training (see the orange line for $N_g = 2$ in figure 3), these two are not valid generators and should be discarded. We thus conclude that there is no Lie algebra with $N_g = 2$ distinct generators in this scenario.

We note in passing that upon repeated training runs in the $N_g = 2$ case, we sometimes find that the algorithm chooses to create generators which are orthogonal to each other, but are not genuine symmetry transformations, i.e. the orthogonality loss is driven to zero, but only at the expense of a large invariance loss (10). In principle, it is difficult to predict what the machine will choose to do when presented with two mutually exclusive requirements. For example, in some runs the two candidate generators ended up being identical instead of differing by a minus sign. Also note that the normalization of the two candidate generators in the second row of figure 4 is off—this is because the program chose to violate the normalization condition as well, in order to help the orthogonality loss, which is the dominant penalty in that case.

Next, we check the cases with $N_g > 2$. Similarly, we find that the training ends up in a large loss and that there is no consistent solution for a closed algebra. We therefore conclude that in this $n = 2$ example there is only a one-parameter symmetry group with a generator given by (28).

Although the example of this subsection was rather trivial, it did outline and validate the main steps of our method. More complicated examples follow below. In each of those examples, we shall primarily be interested in the cases that lead to valid (sub)algebras, which are of interest to mathematicians. Hence we shall only show results from the successful training runs leading to small values for the loss. As already discussed above, the unsuccessful training runs leading to a large loss rule out the existence of (sub)algebras with that many generators. In those cases the exact value of the loss is determined by secondary factors like the choice of hyperparameters, the initial seed, etc

5.2. Rotations in three dimensions

In this subsection, we proceed to study symmetry transformations which preserve the oracle (25) in $n = 3$ dimensions. Once again, we find that training with the invariance loss (8) alone always leads to a valid symmetry transformation. We also observe that the matrix form $\mathbb{F}$ is antisymmetric, in agreement with the expectation for the orthogonal group $O(3)$. Now that the data is three-dimensional, however, it is difficult to visualize the symmetry transformation directly as in figure 2. Instead, we choose to plot the symmetry axis (in this case the axis of rotation) defined by the real eigenvector. Figure 5 illustrates the transformations found at different stages of the training for the case of $N_g = 3$ generators. At the top of each panel we list the

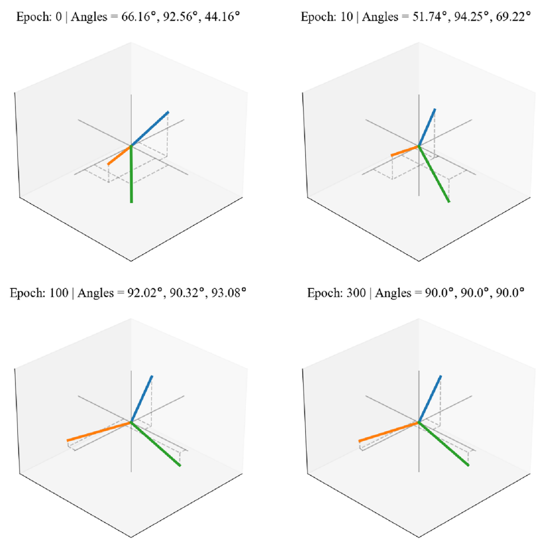
Epoch: 0 | Angles = 66.16°, 92.56°, 44.16°
Epoch: 10 | Angles = 51.74°, 94.25°, 69.22°
Epoch: 100 | Angles = 92.02°, 90.32°, 93.08°
Epoch: 300 | Angles = 90.0°, 90.0°, 90.0°

Figure 5. The axes of the eigenvectors of the three generators found at intermediate stages of the training: after 1 epoch (top left), after 10 epochs (top right), after 100 epochs (bottom left), and after 300 epochs (bottom right). For convenience, at the top of each panel we list the angle (in degrees) between each pair of axes.

relative angle in degrees for each pair of axes. Note how the symmetry axes start out oriented at random, but the orthogonality loss term (13) gradually drives them to a mutually orthogonal configuration.

In order to analyze the group structure of the found symmetry transformations, we proceed to study the generators of infinitesimal transformations. First we try to determine the dimensionality of the full algebra, i.e. the maximal number of linearly independent generators resulting in a closed algebra. Figure 6 shows loss curves for several different values of N_g: 1, 2, 3 and 4 (we do not show results for $N_g \geqslant 5$ since they had large losses). We observe that the training was successful only for the cases of $N_g = 1$ and $N_g = 3$. This implies that the full algebra has 3 generators, in agreement with the expectation of $n(n-1)/2$ generators for an orthogonal $O(n)$ group. At the same time, the successful training at $N_g = 1$ reveals a single generator subgroup of $O(3)$ which is nothing but the $O(2)$ discussed in the previous section.

The results from a typical training run at $N_g = 3$ are shown in figure 7, where the top row depicts the three successfully learned generators $\mathbb{J}_\alpha$, $\alpha = 1, 2, 3$, while the bottom panel is a pictorial representation of the structure constants in matrix form as follows. Each row (labeled $\alpha\beta = 12, 31, 23$) represents one of the three possible commutators, whereas the columns (labeled $\gamma = 1, 2, 3$) represent the found generators $\mathbb{J}_\gamma$ shown in the top panels of the figure. Then, the entry in each cell represents the structure constant $a_{[\alpha\beta]\gamma}$ from the defining equation (7).

A careful inspection of the top row in figure 7 reveals that the three generators found in our example are approximately

$$\mathbb{J}_1 \approx \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} = -\mathbb{O}_{31}, \tag{29a}$$

$$\mathbb{J}_2 \approx \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = -\frac{1}{\sqrt{2}}(\mathbb{O}_{12} + \mathbb{O}_{23}), \tag{29b}$$

$$\mathbb{J}_3 \approx \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = -\frac{1}{\sqrt{2}}(\mathbb{O}_{12} - \mathbb{O}_{23}). \tag{29c}$$

The bottom panel shows that the algebra of the found three generators $\mathbb{J}_\alpha$ is given by

$$[\mathbb{J}_\alpha, \mathbb{J}_\beta] = -\epsilon_{\alpha\beta\gamma}\mathbb{J}_\gamma, \tag{30}$$

in which we recognize the usual $SO(3)$ algebra involving the Levi-Civita permutation symbol $\epsilon_{\alpha\beta\gamma}$. From now on we will be using the Einstein summation convention for repeated generator-type indices $\alpha, \beta, \gamma, \ldots$.

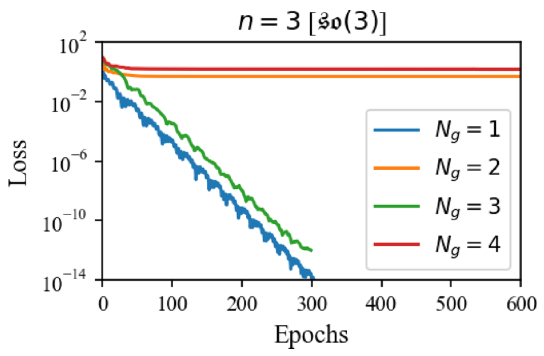

Figure 6. The same as figure 3, but for the $n = 3$ exercise considered in section 5.2. The training results in valid closed algebras with one or three generators, but not two or four.

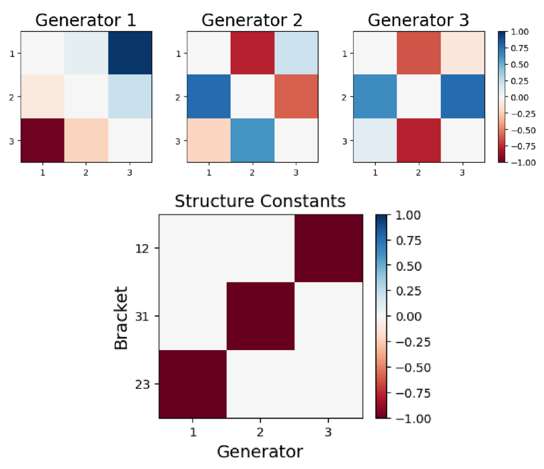

Figure 7. Top row: the successfully learned generators for the $n = 3$, $N_g = 3$ exercise considered in section 5.2. Bottom panel: the corresponding structure constants in matrix form (see text).

5.3. Rotations in four dimensions

In this subsection, we generalize the discussion from the previous two subsections to the case of $n = 4$ dimensions. The new twist here will be the existence of multiple nontrivial subalgebras of different dimensionality.

The discovery of a single finite symmetry transformation with the invariance loss (8) is straightforward and always succeeds in finding some finite rotation in four dimensions, represented with an orthogonal matrix. Therefore in this subsection, we focus only on the identification of the (sub)algebras. As before, we repeat the training for various number of multiple distinct generators ($N_g = 2, 3, 4, 5, 6, 7$) and with the orthogonality and closure losses turned on. Representative loss curves are shown in figure 8. We observe that a closed algebra is found in four of those cases, namely $N_g = 2, 3, 4, 6$, which we now discuss in turn (the trivial case of $N_g = 1$ is of course always possible and will not be specifically discussed from now on).

5.3.1. Two generator subalgebras

One of our found examples of a closed subalgebra with $N_g = 2$ generators is shown in figure 9. The top panels indicate that the two found generators can be roughly approximated as

$$\mathbb{J}_1 \approx \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \mathbb{O}_{12}, \tag{31a}$$

$$\mathbb{J}_2 \approx \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} = \mathbb{O}_{34}, \tag{31b}$$

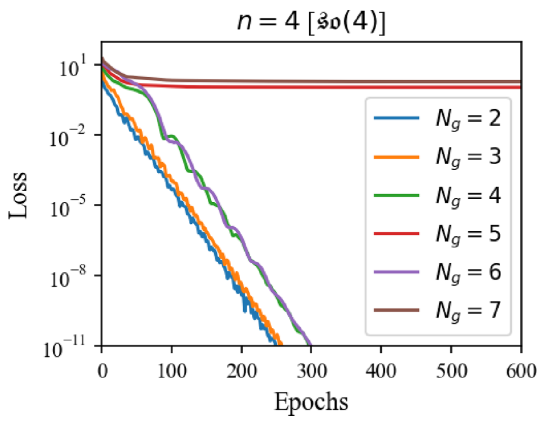

Figure 8. The same as figure 6, but for the $n = 4$ exercise considered in section 5.3. The training results in valid closed algebras with one (not shown), two, three, four and six generators, but not five or seven.

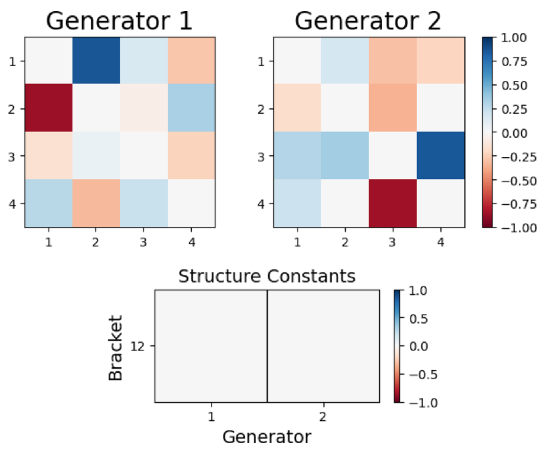

Figure 9. Top row: the successfully learned generators for the $n = 4$, $N_g = 2$ exercise considered in section 5.3.1. Bottom panel: the corresponding structure constants. The vanishing of the structure constants indicates that this is an abelian subgroup.

in which we recognize the rotation generators $\mathbb{O}_{12}$ and $\mathbb{O}_{34}$ from (26). As seen from the matrix forms in (31), these two generators commute, since the two rotations are done in completely different planes. Therefore, the algebra formed by $\mathbb{J}_1$ and $\mathbb{J}_2$ is abelian, which is independently verified by the bottom panel in figure 9.

One should keep in mind that we do not control the overall orientation of the generators found by our procedure. The example in figure 9 was judiciously chosen to be easily recognizable in terms of the canonical generators (26). A generic training run typically returns the generator set in some random orientation, which, however, still preserves the commutation properties. One such generic example is shown in figure 10.

This time, the two found generators $\mathbb{J}_1$ and $\mathbb{J}_2$ are more general linear combinations of the six canonical generators $\mathbb{O}_{12}$, $\mathbb{O}_{13}$, $\mathbb{O}_{14}$, $\mathbb{O}_{23}$, $\mathbb{O}_{24}$, and $\mathbb{O}_{34}$ of the $O(4)$ group. Nevertheless, the found generators $\mathbb{J}_1$ and $\mathbb{J}_2$ still commute and form an abelian two-dimensional subalgebra of the full symmetry group.

5.3.2. Three generator subalgebras

Next we discuss the discovered subalgebras with three generators ($N_g = 3$). Since $SO(4)$ contains $SO(3)$ as a subgroup[3], we know that such subalgebras should exist, and indeed, we find such solutions, as seen in figure 8. As mentioned above, a generic training run produces the three generators in a random orientation. For ease of interpretation, in figure 11 we show a judiciously chosen example, in which the generators depicted in the top panels can be recognized to be approximately

$$\mathbb{J}_1 \approx \mathbb{O}_{24}, \qquad \mathbb{J}_2 \approx \mathbb{O}_{23}, \qquad \mathbb{J}_3 \approx \mathbb{O}_{34}. \tag{32}$$

[3] Since $SO(4) = SO(3) \otimes SO(3)$, the $SO(4)$ algebra is a direct sum of two separate $SO(3)$ subalgebras.

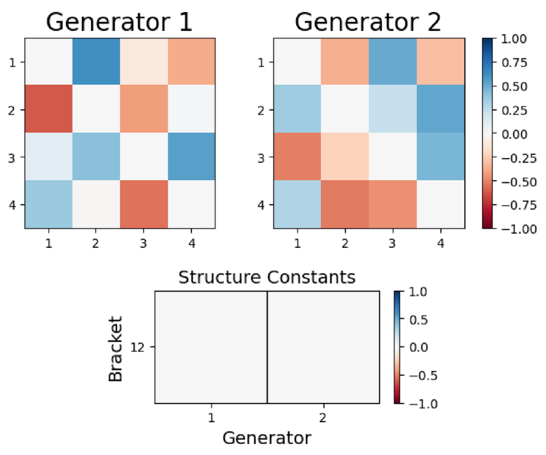

Figure 10. The same as figure 9, but for a different training run, still with $N_g = 2$.

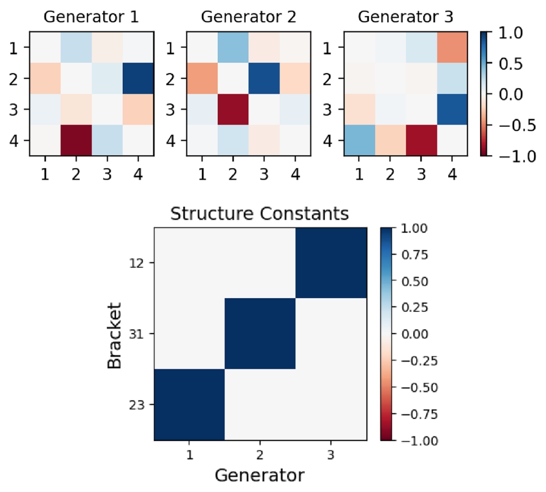

Figure 11. Top row: the successfully learned generators for the $n = 4$, $N_g = 3$ exercise considered in section 5.3.2. Bottom panel: the corresponding structure constants, which can be identified as those of the $SO(3)$ algebra (see text).

This algebra involves rotations primarily in the last three feature dimensions, while the first feature is largely unaffected by the symmetry. The bottom panel of figure 11 confirms that this is the $SO(3)$ algebra from equation (30). Of course, a more generic training run results in a set of three generators that involve all four feature dimensions, but still have the same commutation relations.

5.3.3. Four generator subalgebras

Figure 8 shows that our method finds an algebra with four generators as well. To see its origin theoretically, define a new generator basis in terms of sums and differences of pairs of commuting generators from the original basis (26) [53]

$$\mathbb{S}_1 \equiv \frac{1}{2}\left(\mathbb{O}_{34} + \mathbb{O}_{12}\right), \quad \mathbb{D}_1 \equiv \frac{1}{2}\left(\mathbb{O}_{34} - \mathbb{O}_{12}\right), \tag{33a}$$

$$\mathbb{S}_2 \equiv \frac{1}{2}\left(\mathbb{O}_{42} + \mathbb{O}_{13}\right), \quad \mathbb{D}_2 \equiv \frac{1}{2}\left(\mathbb{O}_{42} - \mathbb{O}_{13}\right), \tag{33b}$$

$$\mathbb{S}_3 \equiv \frac{1}{2}\left(\mathbb{O}_{23} + \mathbb{O}_{14}\right), \quad \mathbb{D}_3 \equiv \frac{1}{2}\left(\mathbb{O}_{23} - \mathbb{O}_{14}\right). \tag{33c}$$

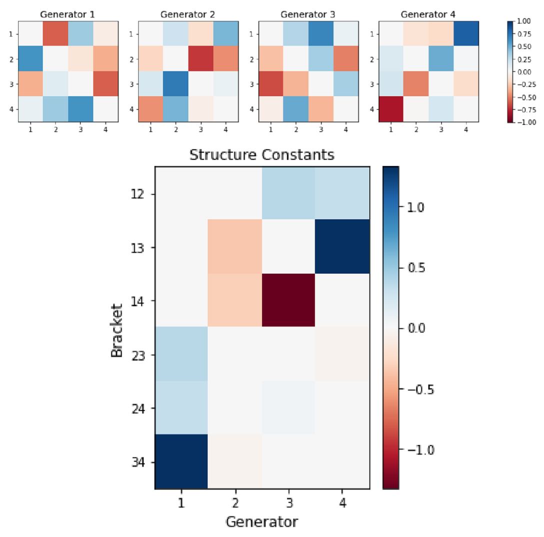

Figure 12. Top row: the successfully learned generators for the $n = 4$, $N_g = 4$ exercise considered in section 5.3.3. Bottom panel: the corresponding structure constants.

This change of basis decouples the algebra (27) of the original generators as follows

$$\left[\mathbb{S}_i, \mathbb{S}_j\right] = -\epsilon_{ijk}\mathbb{S}_k, \tag{34a}$$

$$\left[\mathbb{D}_i, \mathbb{D}_j\right] = -\epsilon_{ijk}\mathbb{D}_k, \tag{34b}$$

$$\left[\mathbb{S}_i, \mathbb{D}_j\right] = 0. \tag{34c}$$

Therefore, a closed algebra of four generators can be formed either from the set of three $\mathbb{S}$'s plus any one of the $\mathbb{D}$'s, or from the set of three $\mathbb{D}$'s plus any one of the $\mathbb{S}$'s. In either case, three of the generators will satisfy $SO(3)$-type commutation relations, while the fourth one will commute with everyone else. This expectation is confirmed in figure 12, which shows our usual representation of the found generators and their algebra for one representative result from a training run with $N_g = 4$. We observe that the found generators are approximately

$$\mathbb{J}_1 \equiv \frac{1}{2}\left(\mathbb{O}_{21} + \mathbb{O}_{43}\right) = -\mathbb{S}_1, \tag{35a}$$

$$\mathbb{J}_2 \equiv \frac{1}{2}\left(\mathbb{O}_{32} + \mathbb{O}_{14}\right) = -\mathbb{D}_3, \tag{35b}$$

$$\mathbb{J}_3 \equiv \frac{1}{2}\left(\mathbb{O}_{13} + \mathbb{O}_{42}\right) = \mathbb{S}_2, \tag{35c}$$

$$\mathbb{J}_4 \equiv \frac{1}{2}\left(\mathbb{O}_{14} + \mathbb{O}_{23}\right) = \mathbb{S}_3. \tag{35d}$$

Therefore, $\mathbb{J}_1$, $\mathbb{J}_3$ and $\mathbb{J}_4$ form an $SO(3)$ algebra as implied by equation (34a), and furthermore, all three of them commute with $\mathbb{J}_2$, as implied by equation (34c). This pattern is precisely what we observe in the lower panel of figure 12.

5.3.4. Six generator algebras

Finally, we get to the case of N_g which will reveal the full algebra of $SO(4)$. Figure 13 shows the result from a representative training run seeking a closed algebra with $N_g = 6$ generators. Among the set of learned generators we can approximately recognize $\mathbb{J}_1 \approx \mathbb{O}_{43}$, $\mathbb{J}_2 \approx -\mathbb{D}_3$, $\mathbb{J}_3 \approx \mathbb{S}_3$, $\mathbb{J}_4 \approx \mathbb{O}_{12}$, $\mathbb{J}_5 \approx \mathbb{O}_{42}$ and $\mathbb{J}_6 \approx \mathbb{O}_{13}$.

The analysis of the last three subsections can be readily extended to even higher dimensions ($n \geqslant 5$). We have checked a few more values of n and the method works each time—we obtain valid finite symmetry

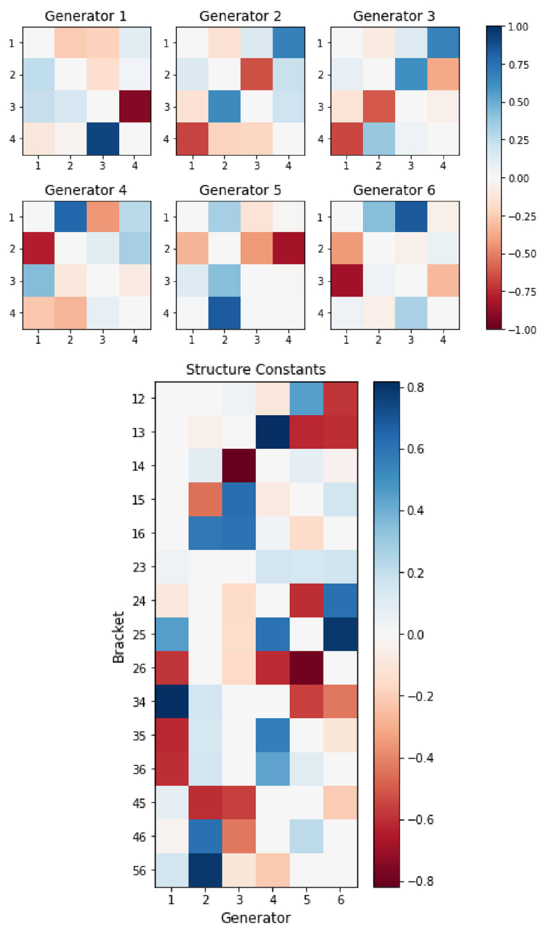

Figure 13. Top row: the successfully learned generators for the $n = 4$, $N_g = 6$ exercise considered in section 5.3.4. Bottom panel: the corresponding structure constants.

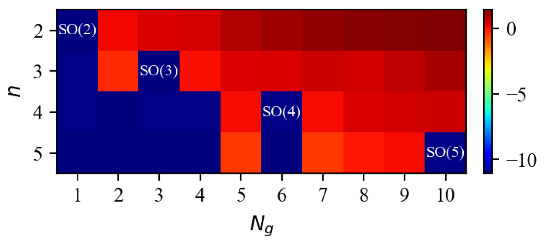

Figure 14. Results from our search for Lie algebras of transformations preserving the Euclidean oracle (25) in $n = 2, 3, 4, 5$ dimensions and for different number of generators N_g. The cells are color coded by the base-10 logarithm of the lowest value of the loss attained during training.

transformations which preserve the oracle (25), we find the closed algebra of the full set of $n(n-1)/2$ generators of the orthogonal $O(n)$ group, as well as valid subalgebras. For fun, in figure 28 we depict our derived 45 generators of the $SO(10)$ group.

In conclusion of our discussion of orthogonal groups, in figure 14 we summarize our results for the found closed algebras and subalgebras for $n \leqslant 5$ and different number of generators N_g. Each cell in the table represents a separate exercise at a fixed number of dimensions n and for a fixed number of generators N_g. The cells are color coded by the base-10 logarithm of the lowest value of the loss attained during training. The training is terminated once the loss reaches 10^{-12}, therefore blue cells correspond to successful training runs resulting in closed algebras. The right-most blue cell in each row corresponds to the full algebra, in this case $SO(n)$, as indicated. The preceding blue cells correspond to various subalgebras. In fact we did not anticipate the existence of the $N_g = 4$ subalgebras in the case of $n = 4$ and $n = 5$, but our model surprised us.

6. Lorentz transformations in four dimensional Minkowski space

In this section we consider the four-dimensional Minkowski spacetime (t,x,y,z) (in natural units with $c=1$). The usual Lorentz transformations preserve the quadratic form

$$\varphi(t,x,y,z) = t^2 - x^2 - y^2 - z^2, \quad (t,x,y,z) \in \mathbb{R}^4, \tag{36}$$

hence this will be our oracle in this section. The four input features are

$$x^{(0)} = t, \quad x^{(1)} = x, \quad x^{(2)} = y, \quad x^{(3)} = z, \tag{37}$$

where in keeping with the standard physics notation we start labelling the features from 0.

Our analysis proceeds as usual. First, we find symmetry transformations which are in general combinations of boosts and rotations. Upon inspection, we verify that they have the desired symmetry properties

$$\mathbb{F}_{0i} = \mathbb{F}_{i0}, \;\; \mathbb{F}_{ij} = -\mathbb{F}_{ji}, \;\; \mathbb{F}_{00} = \mathbb{F}_{ii} = 0, \;\; \forall i = 1,2,3.$$

Next we analyze the algebras of generators. Before presenting the numerical results, for the reader's convenience we summarize some relevant information about the mathematical structure of the Lorentz group $O(1,3)$. It has six generators: the three generators of boosts $\mathbb{K}_i$,

$$\mathbb{K}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{38a}$$

$$\mathbb{K}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{38b}$$

$$\mathbb{K}_3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \tag{38c}$$

and the three generators of rotations, $\mathbb{L}_i$, given by

$$\mathbb{L}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{39a}$$

$$\mathbb{L}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \tag{39b}$$

$$\mathbb{L}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{39c}$$

The full Lorentz algebra can be summarized as

$$[\mathbb{L}_i, \mathbb{L}_j] = \epsilon_{ijk} \mathbb{L}_k, \tag{40}$$

$$[\mathbb{L}_i, \mathbb{K}_j] = \epsilon_{ijk} \mathbb{K}_k, \tag{41}$$

$$[\mathbb{K}_i, \mathbb{K}_j] = -\epsilon_{ijk} \mathbb{L}_k. \tag{42}$$

Table 1. The Lorentz algebra in terms of the generators (43). The cells show the result from commuting one of the generators in the leftmost column with one of the generators in the top row.

	$\mathbb{K}_1-\mathbb{L}_2$	$\mathbb{K}_2+\mathbb{L}_1$	$\mathbb{K}_3$	$\mathbb{L}_3$	$\mathbb{L}_2$	$\mathbb{L}_1$
	$\mathbb{X}_1$	$\mathbb{X}_2$	$\mathbb{X}_3$	$\mathbb{X}_4$	$\mathbb{X}_5$	$\mathbb{X}_6$
$\mathbb{X}_1$	0	0	$-\mathbb{X}_1$	$-\mathbb{X}_2$	$+\mathbb{X}_3$	$+\mathbb{X}_4$
$\mathbb{X}_2$	0	0	$-\mathbb{X}_2$	$+\mathbb{X}_1$	$+\mathbb{X}_4$	$-\mathbb{X}_3$
$\mathbb{X}_3$	$+\mathbb{X}_1$	$+\mathbb{X}_2$	0	0	$-\mathbb{X}_1-\mathbb{X}_5$	$\mathbb{X}_2-\mathbb{X}_6$
$\mathbb{X}_4$	$+\mathbb{X}_2$	$-\mathbb{X}_1$	0	0	$-\mathbb{X}_6$	$+\mathbb{X}_5$
$\mathbb{X}_5$	$-\mathbb{X}_3$	$-\mathbb{X}_4$	$+\mathbb{X}_1+\mathbb{X}_5$	$+\mathbb{X}_6$	0	$-\mathbb{X}_4$
$\mathbb{X}_6$	$-\mathbb{X}_4$	$+\mathbb{X}_3$	$-\mathbb{X}_2+\mathbb{X}_6$	$-\mathbb{X}_5$	$+\mathbb{X}_4$	0

The subgroup structure is most easily seen in a different basis,

$$\mathbb{X}_1=\mathbb{K}_1-\mathbb{L}_2, \tag{43a}$$

$$\mathbb{X}_2=\mathbb{K}_2+\mathbb{L}_1, \tag{43b}$$

$$\mathbb{X}_3=\mathbb{K}_3, \tag{43c}$$

$$\mathbb{X}_4=\mathbb{L}_3, \tag{43d}$$

$$\mathbb{X}_5=\mathbb{L}_2, \tag{43e}$$

$$\mathbb{X}_6=\mathbb{L}_1. \tag{43f}$$

The resulting algebra in the $\mathbb{X}_i$ basis is summarized in table 1. By inspection, we see that the Lorentz algebra has several non-trivial subalgebras.

6.1. Two generator subalgebras

There are several two-dimensional subalgebras:

- *Abelian subalgebras.* There are two abelian subalgebras: the first one is generated by the generator set $\{\mathbb{X}_1,\mathbb{X}_2\}$, while the second is generated by $\{\mathbb{X}_3,\mathbb{X}_4\}$. A representative example of the former case is shown in figure 15. We recognize the two found generators to be approximately

$$\mathbb{J}_1\approx-(\mathbb{K}_3-\mathbb{L}_1), \tag{44a}$$

$$\mathbb{J}_2\approx-(\mathbb{K}_1+\mathbb{L}_3). \tag{44b}$$

  This result is in agreement with equations (43a) and (43b)—the transformations are combinations of boosts along and rotations about two of the axes, in this case x and z.
- *Nonabelian subalgebras.* As seen in table 1, the Lie algebra of the Lorentz group has two nonabelian subalgebras, generated by $\{\mathbb{X}_1,\mathbb{X}_3\}$ and $\{\mathbb{X}_2,\mathbb{X}_3\}$, respectively. A representative training example for this nonabelian case is shown in figure 16. The top panels show that the two found generators are approximately

$$\mathbb{J}_1\approx-\mathbb{K}_3=-\mathbb{X}_3, \tag{45a}$$

$$\mathbb{J}_2\approx\mathbb{K}_1-\mathbb{L}_2=\mathbb{X}_1, \tag{45b}$$

while the bottom panel confirms that their Lie bracket is approximately given by $[\mathbb{J}_1,\mathbb{J}_2]\approx[-\mathbb{X}_3,\mathbb{X}_1]=-\mathbb{X}_1=-\mathbb{J}_2$, as expected from table 1.

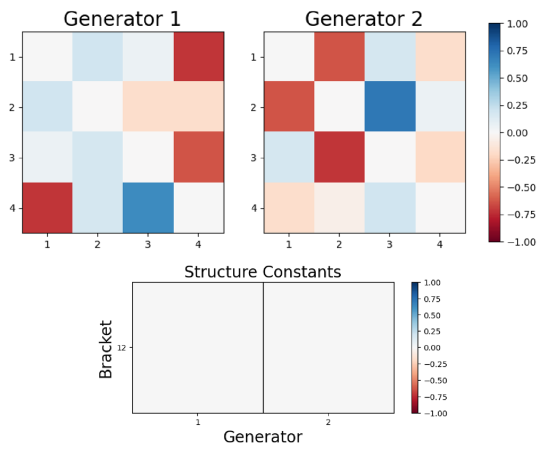

Figure 15. Top row: successfully learned generators for the $n = 4$, $N_g = 2$ exercise considered in section 6.1, using the pseudo-Euclidean oracle (36). Bottom panel: the corresponding structure constants. This example shows an abelian subalgebra.

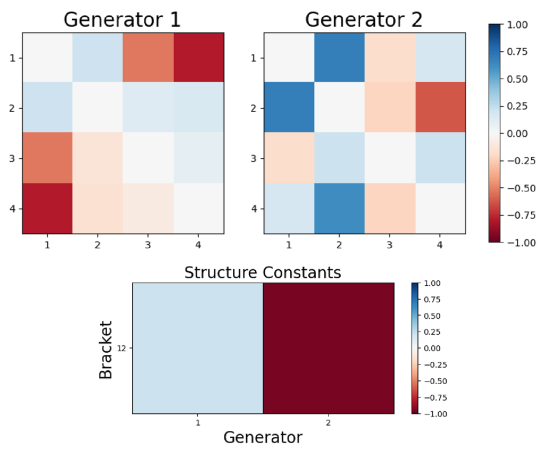

Figure 16. The same as figure 15, but showing a non-abelian subalgebra found by our method.

6.2. Three generator subalgebras

Table 1 reveals several three-dimensional subalgebras:

- *SO(3).* The set $\{\mathbb{X}_4, \mathbb{X}_5, \mathbb{X}_6\}$ generates a subalgebra isomorphic to the Lie algebra of the $SO(3)$ group of rotations in three dimensions, see section 5.2. A representative training example is shown in figure 17. The generators can be recognized as

$$\mathbb{J}_1 \approx \mathbb{L}_3, \tag{46a}$$

$$\mathbb{J}_2 \approx \mathbb{L}_1, \tag{46b}$$

$$\mathbb{J}_3 \approx \mathbb{L}_2. \tag{46c}$$

  In addition, we also found examples with subalgebras isomorphic to $SO(3)$ consisting of one rotation and two boosts, or one boost and two rotations.
- *Hom(2).* The set $\{\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3\}$ generates a subalgebra isomorphic to the Lie algebra of Hom(2), the group of Euclidean homotheities:

$$[\mathbb{X}_1, \mathbb{X}_2] = 0,\ [\mathbb{X}_3, \mathbb{X}_1] = \mathbb{X}_1,\ [\mathbb{X}_2, \mathbb{X}_3] = -\mathbb{X}_2. \tag{47}$$

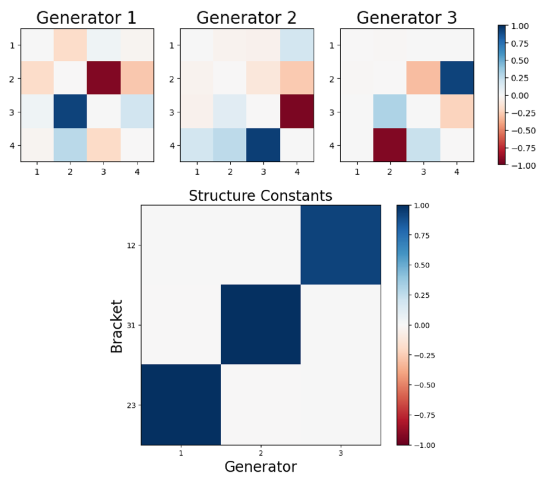

Figure 17. Top row: the successfully learned generators for the $n = 4$, $N_g = 3$ exercise considered in section 6.2, using the pseudo-Euclidean oracle (36). Bottom panel: the corresponding structure constants, which can be identified as those of an *SO(3)* type subalgebra.

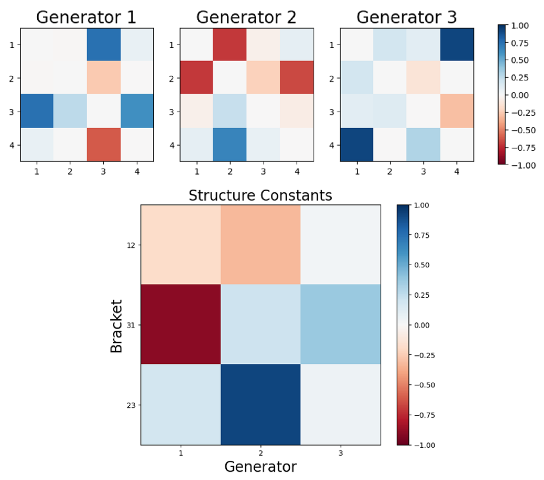

Figure 18. The same as figure 17, but showing an example of a *Hom(2)* subalgebra (47).

A representative training example is shown in figure 18. From the panels in the top row we recognize the found three generators as

$$\mathbb{J}_1 \approx +\mathbb{K}_2 - \mathbb{L}_1, \tag{48a}$$

$$\mathbb{J}_2 \approx -\mathbb{K}_1 - \mathbb{L}_2, \tag{48b}$$

$$\mathbb{J}_3 \approx +\mathbb{K}_3. \tag{48c}$$

The bottom panel in figure 18 confirms that their algebra is approximately that of equation (47).

- *E(2).* The set $\{\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_4\}$ generates a subalgebra isomorphic to the Lie algebra of $E(2)$, the Euclidean group:

$$[\mathbb{X}_1, \mathbb{X}_2] = 0,\ [\mathbb{X}_4, \mathbb{X}_1] = \mathbb{X}_2,\ [\mathbb{X}_2, \mathbb{X}_4] = \mathbb{X}_1. \tag{49}$$

A representative example is shown in figure 19, from which we recognize the found three generators as

$$\mathbb{J}_1 \approx -\mathbb{K}_3 - \mathbb{L}_2, \tag{50a}$$

$$\mathbb{J}_2 \approx -\mathbb{L}_1, \tag{50b}$$

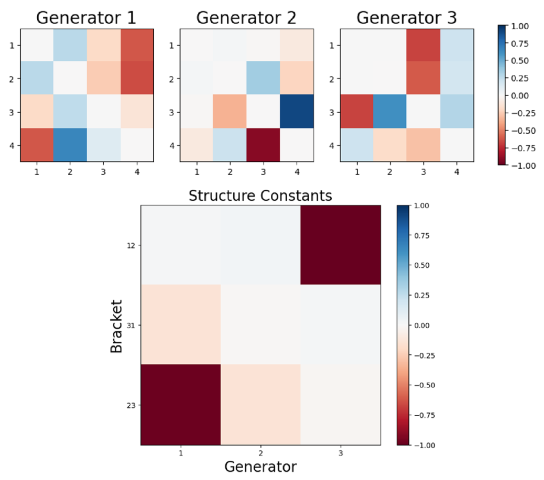

Figure 19. The same as figure 17, but showing an example of a $E(2)$ subalgebra (49).

$$\mathbb{J}_3 \approx -\mathbb{K}_2 + \mathbb{L}_3. \tag{50c}$$

As expected, two of the generators involve combinations of boosts along and rotations about two of the axes, in this case y and z, while the third generator is a rotation about the remaining axis, namely x. The bottom panel in figure 19 shows that the resulting algebra is approximately that of equation (49).

- *Bianchi VII$_a$*. The set $\{\mathbb{X}_1, \mathbb{X}_2, \mathbb{X}_3 + a\mathbb{X}_4\}$ with $a \neq 0$ generates a Bianchi VII$_a$ subalgebra. A representative training example is shown in figure 20, from which we recognize the found three generators as

$$\mathbb{J}_1 \approx -\mathbb{K}_3 - \mathbb{L}_3 = -(\mathbb{X}_3 + \mathbb{X}_4), \tag{51a}$$

$$\mathbb{J}_2 \approx +\mathbb{K}_1 - \mathbb{L}_2 = \mathbb{X}_1, \tag{51b}$$

$$\mathbb{J}_3 \approx -\mathbb{K}_2 - \mathbb{L}_1 = -\mathbb{X}_2. \tag{51c}$$

This set corresponds to a Bianchi VII$_a$ subalgebra with $a = 1$, whose structure constants are indeed consistent with the lower panel in figure 20.

- *SL(2,R)*. The set $\{\mathbb{X}_1, \mathbb{X}_3, \mathbb{X}_5\}$ generates a subalgebra isomorphic to the Lie algebra of $SL(2,R)$, the group of isometries of the hyperbolic plane:

$$[\mathbb{X}_1, \mathbb{X}_3] = -\mathbb{X}_1, \tag{52a}$$

$$[\mathbb{X}_5, \mathbb{X}_1] = -\mathbb{X}_3, \tag{52b}$$

$$[\mathbb{X}_3, \mathbb{X}_5] = -\mathbb{X}_1 - \mathbb{X}_5. \tag{52c}$$

A representative training example is shown in figure 21, from which we recognize the found three generators as

$$\mathbb{J}_1 \approx \mathbb{K}_1 - \mathbb{L}_3, \tag{53a}$$

$$\mathbb{J}_2 \approx -\mathbb{L}_3, \tag{53b}$$

$$\mathbb{J}_3 \approx \mathbb{K}_2. \tag{53c}$$

It is easy to verify that their structure constants given in the lower panel of figure 21 are consistent with equation (52).

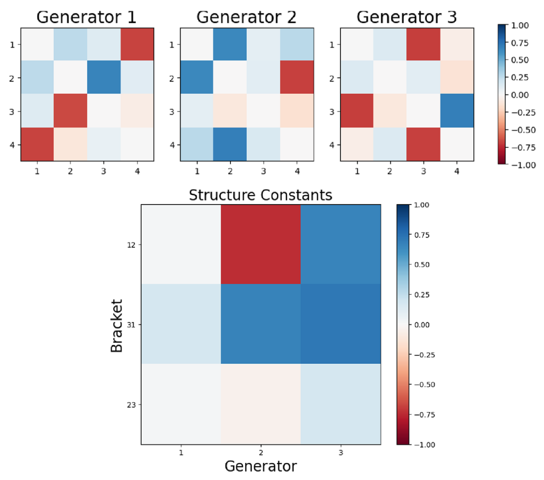

Figure 20. The same as figure 17, but showing an example of a Bianchi VII$_a$ subalgebra.

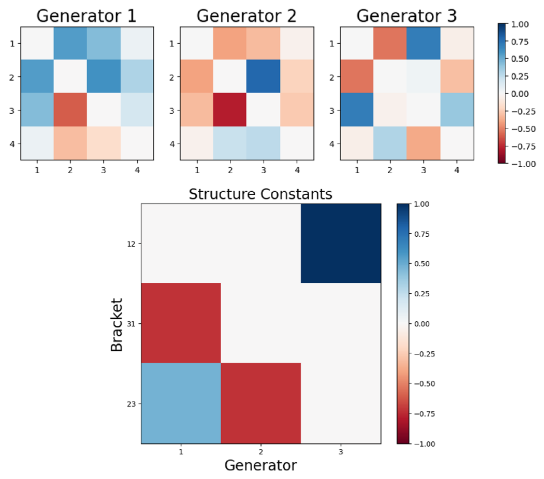

Figure 21. The same as figure 17, but showing an example of an $SL(2,R)$ subalgebra (52).

6.3. Four generator subalgebras

The only four-dimensional subalgebra is generated by $\{\mathbb{X}_1,\mathbb{X}_2,\mathbb{X}_3,\mathbb{X}_4\}$. A representative example is shown in figure 22. The four found generators are given by

$$\mathbb{J}_1 \approx \mathbb{L}_3 = \mathbb{X}_4, \tag{54a}$$

$$\mathbb{J}_2 \approx \mathbb{K}_2 + \mathbb{L}_1 = \mathbb{X}_2, \tag{54b}$$

$$\mathbb{J}_3 \approx -\mathbb{K}_1 + \mathbb{L}_2 = -\mathbb{X}_1, \tag{54c}$$

$$\mathbb{J}_4 \approx -\mathbb{K}_3 = -\mathbb{X}_3. \tag{54d}$$

The lower panel of figure 22 illustrates the corresponding structure constants. The results are consistent with table 1. For example, $\mathbb{J}_1$ and $\mathbb{J}_4$ commute because $[\mathbb{X}_3,\mathbb{X}_4] = [\mathbb{K}_3,\mathbb{L}_3] = 0$. Similarly, $\mathbb{J}_2$ and $\mathbb{J}_3$ commute due to $[\mathbb{X}_1,\mathbb{X}_2] = 0$. The results shown in the remaining rows in the lower panel of figure 22 can be analogously verified with the help of table 1.

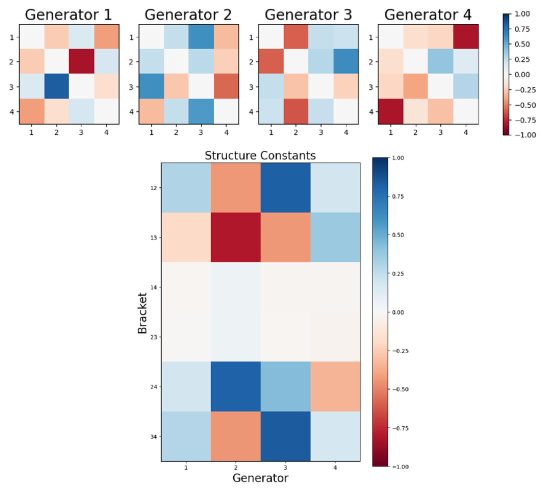

Figure 22. Top row: the successfully learned generators for the $n = 4$, $N_g = 4$ exercise considered in section 6.3, using the pseudo-Euclidean oracle (36). Bottom panel: the corresponding structure constants.

6.4. Six generator algebras

Our method is capable of finding the full six-dimensional algebra as well. The result is shown in figure 23, and can be roughly identified as

$$\mathbb{J}_1 \approx -\mathbb{L}_1, \quad \mathbb{J}_2 \approx -\mathbb{K}_2, \quad \mathbb{J}_3 \approx -\mathbb{L}_3, \tag{55a}$$

$$\mathbb{J}_4 \approx +\mathbb{K}_1, \quad \mathbb{J}_5 \approx -\mathbb{K}_3, \quad \mathbb{J}_6 \approx -\mathbb{L}_2. \tag{55b}$$

The corresponding structure constants are shown in the lower panel of figure 23.

Figure 24 summarizes the results for the value of the overall loss function obtained in typical training runs, as a function of the requested number of generators N_g. The green circles represent the cases when a closed subalgebra was successfully found, and the loss is essentially machine zero. The remaining cases in figure 24 correspond to unsuccessful trainings, i.e. no closed algebra with that many generators was found. In those cases, the total loss remained relatively large, due to tension with either the orthogonality condition (red crosses) or the closure condition (magenta diamonds).

7. Squeeze mapping in two dimensions

Consider again two dimensions, but now let the oracle return the product of the two input features:

$$\varphi(\mathbf{x}) = x^{(1)} x^{(2)}. \tag{56}$$

This oracle function is illustrated in figure 25 as a color heatmap representing the oracle values in the $(x^{(1)}, x^{(2)})$ Cartesian plane. We see that the set of points with the same oracle values form hyperbolas.

Proceeding as before, our method finds the symmetry transformation illustrated with the vector field in figure 25. The corresponding generator is

$$\mathbb{J} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{57}$$

We verified that in this example the method is unable to find more than one generator.

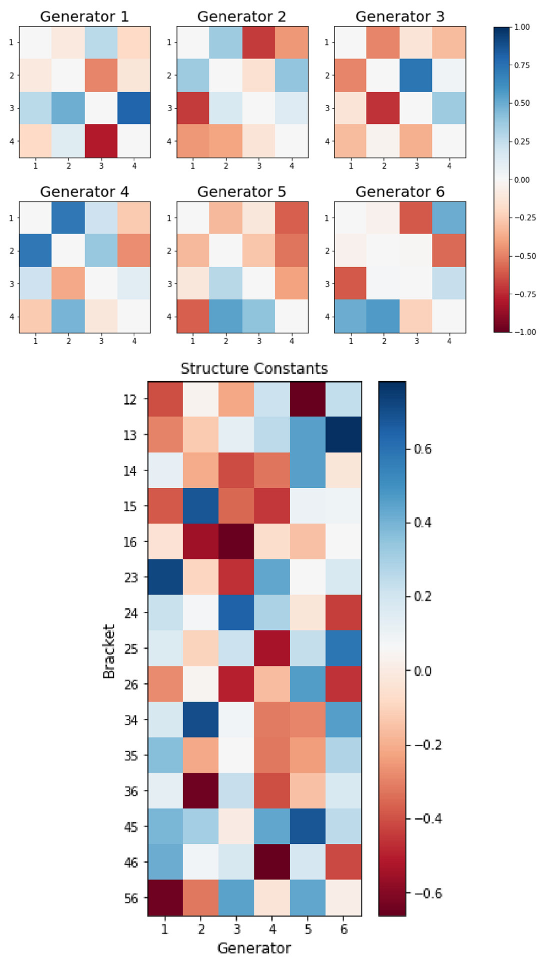

Figure 23. Top row: the successfully learned generators for the $n = 4$, $N_g = 6$ exercise considered in section 6.4, using the pseudo-Euclidean oracle (36). Bottom panel: the corresponding structure constants.

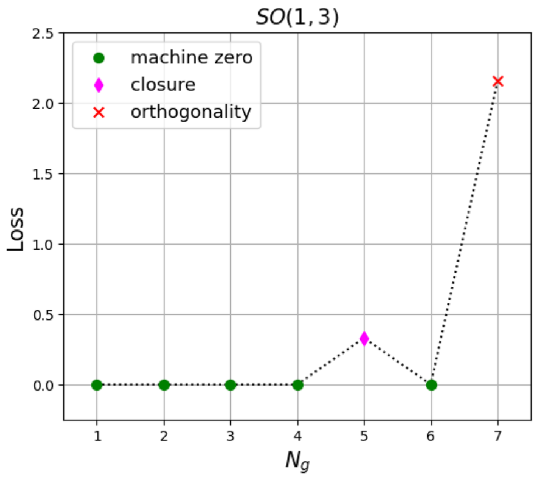

Figure 24. The final value of the overall loss function as a function of the requested number of generators N_g for the $SO(1,3)$ example of section 6. The colored symbols identify the dominant contribution to the overall loss: magenta diamonds for the closure term (14) and red crosses for the orthogonality term (13). For the green circles the total loss is zero to within machine precision. The learning rate was 0.005 and $\varepsilon = 0.001$.

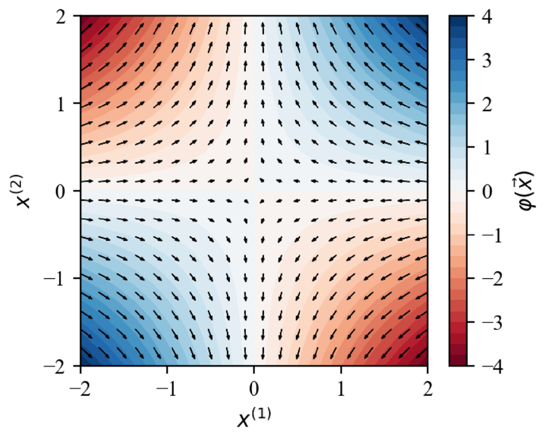

Figure 25. Color heatmap showing the values of the oracle (56) in the $(x^{(1)}, x^{(2)})$ Cartesian plane. The superimposed vector field represents the symmetry found by our method.

These results are precisely what one would expect. The symmetry which preserves the oracle (56) is the squeeze mapping

$$\mathbb{F} = \begin{pmatrix} \frac{1}{\ell} & 0 \\ 0 & \ell \end{pmatrix}, \tag{58}$$

where ℓ is a scale factor. Considering infinitesimal transformations (18) with $\ell = 1 + \varepsilon$ immediately leads to the generator (57).

8. Discontinuous oracles

8.1. Piecewise linear oracle

Our setup is not limited to only continuous oracle functions like the ones discussed so far in the preceeding sections. Our method can also work with piecewise-defined functions like

$$\varphi(\mathbf{x}) = \begin{cases} -x^{(2)}, & \text{for } x^{(1)} < 0, \\ +x^{(1)}, & \text{for } x^{(1)} \geqslant 0. \end{cases} \tag{59}$$

The resulting oracle function is illustrated with the color map in figure 26. Since the oracle is now a function which is not continuously differentiable, our parametrization of the symmetry transformation needs to be properly generalized.

The advantage of using a NN as a universal function approximator is highlighted in figure 26. In the top panel we use no hidden layers, while in the bottom panel we use a deep learning architecture with three hidden layers. The found symmetry transformation in each case is then shown with the vector field and superimposed on the oracle color map. The results are noticeably different, particularly near the locations of discontinuity in the oracle function. The deep-learning approach in the bottom panel correctly identifies a transformation which preserves the oracle values everywhere within the considered domain. In contrast, the shallow approach in the top panel is unable to adjust the transformation near the discontinuity, which leads to locations near the boundary $x^{(1)} = 0$ where the arrows run across the equipotential contours, violating the conservation law.

8.2. Manhattan distance oracle

In this subsection we consider one more example of an oracle in two dimensions ($n = 2$), which, while continuous, is not continuously differentiable:

$$\varphi(\mathbf{x}) = |x^{(1)}| + |x^{(2)}|. \tag{60}$$

The results from our procedure are shown in figure 27 in complete analogy to figure 26. In the top panel we use no hidden layers, while in the bottom panel we use a deep learning architecture with three hidden layers and bias. We observe that the deep-learning approach can again correctly handle the discontinuities, always generating transformations along, but never across, the contours of equal oracle function values.

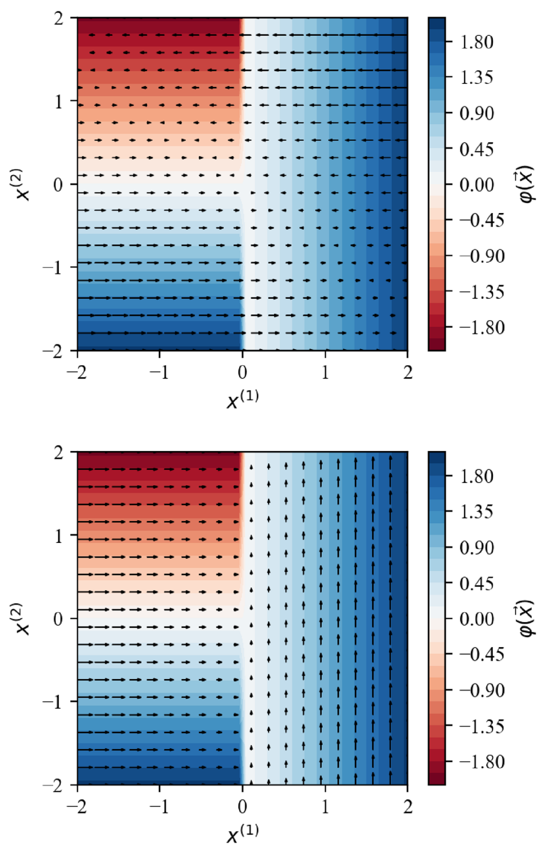

Figure 26. The symmetry generated by the oracle (59) with a shallow method with no hidden layers and no bias (top panel) or a deep method with three hidden layers and bias (bottom panel).

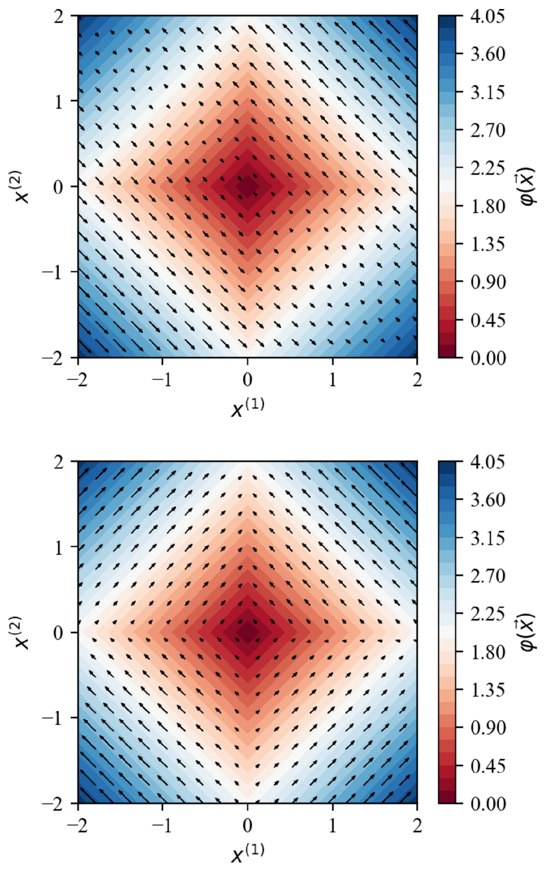

Figure 27. The symmetry generated by the L1 oracle (60) with a shallow method with no hidden layers and no bias (top panel) or a deep method with three hidden layers and bias (bottom panel).

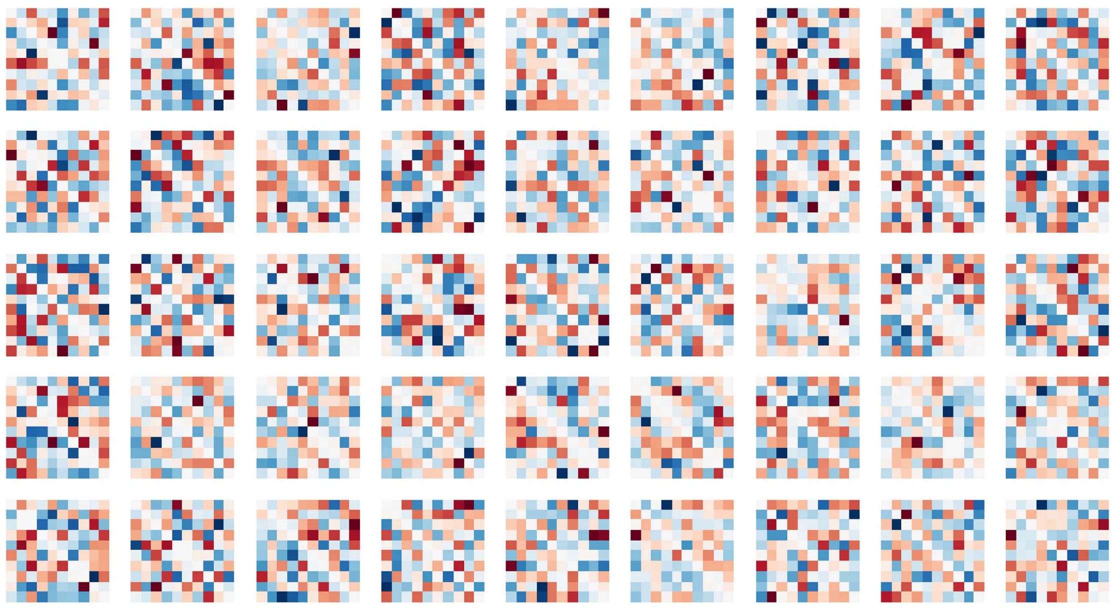

Figure 28. The set of $N_g = 45$ generators found by our method in the case of the $SO(10)$ group (length-preserving rotations in $n = 10$ dimensions). Due to the complexity of the problem, the learning rate for this example was reduced to 0.01.

9. Conclusions

In this paper, we studied a fundamental problem in data science which is commonly encountered in many fields: what is the symmetry of a labeled dataset, and how to identify its group structure? For this purpose, we designed a deep-learning method which models the generic symmetry transformation and its generators with a fully connected NN. We then constructed suitable loss functions which ensure that the applied transformations are symmetries and that the corresponding set of generators forms a closed (sub)algebra. An important advantage of our approach is that we do not require any advance knowledge of what symmetries can be expected in the data, i.e. instead of testing for symmetries from a predefined list of possibilities, we learn the symmetry directly.

Our procedure is very general and is universally applicable in a wide variety of situations (see, e.g. [54]). The centerpiece of our analysis is an oracle $\varphi(\mathbf{x})$ which defines the conserved quantity. The oracle can be completely general, as illustrated with several examples in the paper. For example, it can be a continuous bilinear function, as in the case of the Euclidean or Minkowski distances and the squeeze mapping; the corresponding symmetries were discussed in sections 5–7. It can also be a discontinuous function (section 8.1) or a function which is not smooth and continuously differentiable (section 8.2).

In the process of deriving the full set of symmetries, the method also allows us to analyze the complete subgroup structure of the symmetry group. In the paper we worked out explicitly the subgroup structure of the rotation groups $SO(2)$ (section 5.1), $SO(3)$ (section 5.2) and $SO(4)$ (section 5.3), as well as the Lorentz group $SO(1,3)$ ((section 6)). As one last *tour de force* example, we successfully applied our method to the case of Euclidean length-preserving rotations in $n = 10$ dimensions. The resulting 45 generators of the corresponding group $SO(10)$ commonly used in grand unification [55] are shown in figure 28.

The symmetries discussed in this paper have important implications for simulation-based inference, and in particular parameter retrieval from observations. In a typical inverse problem scenario, the (possibly multi-dimensional) labels y play the role of observed variables, and the features $\mathbf{x}$ play the role of input parameters. The parameter retrieval task is to infer $\mathbf{x}$ given y. In the presence of a symmetry this inversion is not unique, but results in a whole family of valid input parameters, all related by a symmetry. Therefore, knowing that there is a symmetry in the data to begin with can be very useful in parameter retrieval algorithms, especially when the data is very high dimensional and the symmetry is difficult to see by the human eye.

Note that the same approach can be extended or modified to solve other common problems and tasks in group theory and/or Beyond-Standard-Model (BSM) model building not discussed in this paper.

- *Derivation of the Cartan subalgebra.* In our approach, this can be trivially accomplished by setting $a_{[\alpha\beta]\gamma} = 0$ in (15), which will force all the learned generators to commute.

- *Direct product decompositions.* We can also look for symmetries whose algebras can be expressed as direct sums of two separate subalgebras—we just have to include closure terms in the loss function for each of the two individual subalgebras.
- *Internal symmetries.* Note that our approach can be readily generalized to look for internal symmetries [41], which can prove useful in model building in high energy theory.
- *Canonical basis for the generators.* Since our method is basis-independent, the generators are obtained in a generic basis, where the structure constants may be difficult to recognize. We could, however, aid the program in finding a canonical basis of generators by including a term in the loss function which encourages sparsity among the structure constants [56].

In this paper we focused on idealized examples with no noise or statistical fluctuations in the determination of the labels. It will be instructive to test our method on noisy datasets, including real experimental data. We postpone this study to a future publication.

In conclusion, our study opens the door for using a ML approach in the study of Lie groups and their properties and further bridges the fields of formal mathematics and theoretical computer science.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/royforestano/Deep_Learning_Symmetries [57].

Acknowledgment

We thank A Davis, S Gleyzer, K Kong, S Mrenna, H Prosper and P Shyamsundar for useful discussions. We thank P Ramond for group theory insights and inspiration. This work is supported in part by the U.S. Department of Energy Award Number DE-SC0022148.

ORCID iDs

Roy T Forestano https://orcid.org/0000-0002-0355-2076
Konstantin T Matchev https://orcid.org/0000-0003-4182-9096
Katia Matcheva https://orcid.org/0000-0003-3074-998X
Alexander Roman https://orcid.org/0000-0003-2719-221X
Eyup B Unlu https://orcid.org/0000-0002-6683-6463
Sarunas Verner https://orcid.org/0000-0003-4870-0826

References

[1] Gross D J 1996 The role of symmetry in fundamental physics *Proc. Natl Acad. Sci.* **93** 14256–9
[2] Csáki C, Lombardo S and Ofri Telem T A S I 2018 Lectures on Non-supersymmetric BSM Models *Proc. Theoretical Advanced Study Institute in Elementary Particle Physics : Anticipating the Next Discoveries in Particle Physics (TASI 2016)(Boulder, CO, USA, 6 June–1 July 2016)* ed R Essig and I Low (WSP) pp 501–70
[3] Bourilkov D 2020 Machine and deep learning applications in particle physics *Int. J. Mod. Phys.* A **34** 1930019
[4] Calafiura P, Rousseau D and Terao K 2022 *Artificial Intelligence for High Energy Physics* (Singapore: World Scientific)
[5] Plehn T, Butter A, Dillon B and Krause C 2022 Modern machine learning for LHC physicists (arXiv:2211.01421 [hep-ph])
[6] Dersy A, Schwartz M D and Zhang X 2022 Simplifying polylogarithms with machine learning (arXiv:2206.04115 [cs.LG])
[7] Alnuqaydan A, Gleyzer S and Prosper H 2022 SYMBA: symbolic computation of squared amplitudes in high energy physics with machine learning (arXiv:2206.08901 [hep-ph])
[8] Choi S 2011 Construction of a kinematic variable sensitive to the mass of the standard model higgs boson in $H \to WW^* \to l^+ \nu l^- \bar{\nu}$ using symbolic regression *J. High Energy Phys. vol.* JHEP08(2011)110
[9] Udrescu S-M and Tegmark M 2020 AI feynman: a physics-inspired method for symbolic regression *Sci. Adv.* **6** eaay2631
[10] Lample G and Charton Fçois 2019 Deep learning for symbolic mathematics (arXiv:1912.01412 [cs.SC])
[11] Cranmer M, Sanchez-Gonzalez A, Battaglia P, Rui X, Cranmer K, Spergel D and Shirley H 2020 Discovering symbolic models from deep learning with inductive biases (arXiv:2006.11287 [cs.LG])
[12] Butter A, Plehn T, Soybelman N and Brehmer J 2021 Back to the formula—LHC edition (arXiv:2109.10414 [hep-ph])
[13] Arechiga N, Chen F, Chen Y-Y, Zhang Y, Iliev R, Toyoda H and Lyons K 2021 Accelerating understanding of scientific experiments with end to end symbolic regression (arXiv:2112.04023 [cs.LG])
[14] Matchev K T, Matcheva K and Roman A 2022 Analytical modeling of exoplanet transit spectroscopy with dimensional analysis and symbolic regression *Astrophys. J.* **930** 33
[15] d'Ascoli S, Kamienny P-A, Guillaume L and François C Deep symbolic regression for recurrent sequences 2022 (arXiv:2201.04600 [cs.LG])
[16] Lemos P, Jeffrey N, Cranmer M, Shirley H and Battaglia P2022 Rediscovering orbital mechanics with machine learning (arXiv:2202.02306 [astro-ph.EP])
[17] Kamienny P-A, d'Ascoli S, Lample G and Charton Fçois 2022 End-to-end symbolic regression with transformers (arXiv:2204.10532 [cs.LG])

[18] Jiachen Li, Yuan Y and Shen H-B 2022 Symbolic expression transformer: a computer vision approach for symbolic regression (arXiv:2205.11798 [cs.CV])
[19] Matsubara Y, Chiba N, Igarashi R, Taniai T and Ushiku Y 2022 Rethinking symbolic regression datasets and benchmarks for scientific discovery (arXiv:2206.10540 [cs.LG])
[20] Dong Z, Kong K, Matchev K T and Matcheva K 2023 Is the machine smarter than the theorist: deriving formulas for particle kinematics with symbolic regression *Phys. Rev. D* **107** 055018 (arXiv:2211.08420 [hep-ph])
[21] Iten R, Metger T, Wilming H, Lídia del R and Renner R 2020 Discovering physical concepts with neural networks *Phys. Rev. Lett.* **124** 10508
[22] Dillon B M, Kasieczka G, Olischlager H, Plehn T, Sorrenson P and Vogel L 2022 Symmetries, safety and self-supervision *SciPost Phys.* **12** 188
[23] Krippendorf S and Syvaeri M 2020 Detecting symmetries with neural networks (arXiv:2003.13679 [physics.comp-ph])
[24] Butter A, Kasieczka G, Plehn T and Russell M 2018 Deep-learned top tagging with a lorentz layer *SciPost Phys.* **5** 028
[25] Gurtej Kanwar M S Albergo D B, Kyle Cranmer D C Hackett S Rère, Rezende D J and Shanahan P E 2020 Equivariant flow-based sampling for lattice gauge theory *Phys. Rev. Lett.* **125** 121601
[26] Bogatskiy A, Brandon Anderson J T Offermann M R, Miller D W and Kondor R 2020 Lorentz group equivariant neural network for particle physics (arXiv:2006.04780 [hep-ph])
[27] Gong S, Meng Q, Zhang J, Qu H, Li C, Qian S, Du W, Ma Z-M and Liu T-Y 2022 An efficient Lorentz equivariant graph neural network for jet tagging *J. High Energy Phys.* JHEP07(2022)030
[28] Bogatskiy A *et al* 2022 Symmetry Group Equivariant Architectures for Physics *2022 Snowmass Summer Study* (arXiv:2203.06153 [cs.LG])
[29] Congqiao Li, Huilin Q, Qian S, Meng Q, Gong S, Zhang J, Liu T-Y and Li Q 2022 Does Lorentz-symmetric design boost network performance in jet physics? (arXiv:2208.07814 [hep-ph])
[30] Hao Z, Kansal R, Duarte J and Chernyavskaya N 2022 Lorentz group equivariant autoencoders (arXiv:2212.07347 [hep-ex])
[31] Gruver N, Finzi M, Goldblum M and Wilson A G 2022 The lie derivative for measuring learned equivariance (arXiv:2210.02984 [cs.LG])
[32] Fenton M J, Shmakov A, Ta-Wei H, Hsu S-C, Whiteson D and Baldi P 2022 Permutationless many-jet event reconstruction with symmetry preserving attention networks *Phys. Rev.* D **105** 112008
[33] Shmakov A, Fenton M J, Ta-Wei H, Hsu S-C, Whiteson D and Baldi P 2022 SPANet: generalized permutationless set assignment for particle physics using symmetry preserving attention *SciPost Phys.* **12** 178
[34] Tombs R and Lester C G 2022 A method to challenge symmetries in data with self-supervised learning *J. Instrum.* **17** 08024
[35] Lester C G and Tombs R 2021 Using unsupervised learning to detect broken symmetries, with relevance to searches for parity violation in nature (Previously: 'Stressed GANs snag desserts') (arXiv:2111.00616 [hep-ph])
[36] Birman M, Nachman B, Sebbah R, Sela G, Turetz O and Bressler S 2022 Data-directed search for new physics based on symmetries of the SM *Eur. Phys. J.* C **82** 508
[37] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D and Shirley H 2020 Lagrangian neural networks (arXiv:2003.04630 [cs.LG])
[38] Liu Z and Tegmark M 2021 Machine learning conservation laws from trajectories *Phys. Rev. Lett.* **126** 180604
[39] Tailin W and Tegmark M 2019 Toward an artificial intelligence physicist for unsupervised learning *Phys. Rev.* E **100** 3
[40] Barenboim G, Hirn J and Sanz V 2021 Symmetry meets AI *SciPost Phys.* **11** 014
[41] Craven S, Croon D, Cutting D and Houtz R 2022 Machine learning a manifold *Phys. Rev.* D **105** 096030
[42] Wetzel S J, Melko R G, Scott J, Panju M and Ganesh V 2020 Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks *Phys. Rev. Res.* **2** 033499
[43] Chen H-Y, Yang-Hui H, Lal S and Zaid Zaz M 2020 Machine learning etudes in conformal field theories (arXiv:2006.16114 [hep-th])
[44] Yang-Hui H 2017 Machine-learning the string landscape *Phys. Lett.* B **774** 564–8
[45] Carifio J, Halverson J, Krioukov D and Nelson B D 2017 Machine learning in the string landscape *J. High Energy Phys.* JHEP09(2017)157
[46] Ruehle F 2020 Data science applications to string theory *Phys. Rept.* **839** 1–117
[47] Desai K, Nachman B and Thaler J 2022 Symmetry discovery with deep learning *Phys. Rev.* D **105** 096031
[48] Chen H-Y, Yang-Hui H, Lal S and Majumder S 2021 Machine learning Lie structures & applications to physics *Phys. Lett.* B **817** 136297
[49] Liu Z and Tegmark M 2022 Machine learning hidden symmetries *Phys. Rev. Lett.* **128** 180201
[50] Moskalev A, Sepliarskaia A, Sosnovik I and Smeulders A 2022 Liegg: studying learned lie group generators (arXiv:2210.04345 [cs.LG])
[51] Paszke A *et al* 2019 Pytorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32 (Curran Associates, Inc.) pp 8024–35
[52] Hornik K, Stinchcombe M and White H 1989 Multilayer feedforward networks are universal approximators *Neural Netw.* **2** 359–66
[53] Hladik J and J.M. Cole yr1999 *Spinors in Physics* (*Graduate Texts in Contemporary Physics*) (New York: Springer) (available at: https://books.google.com/books?id=25eTtXqLW8UC)
[54] Alexander Roman R T Forestano K T Matchev K M and Unlu E B 2023 Oracle-preserving latent flows (arXiv:2302.00806 [cs.LG])
[55] Gell-Mann M, Ramond P and Slansky R 1979 Complex spinors and unified theories *Conf. Proc.* C **790927** 315–21 (arXiv:1306.4669 [cs.LG])
[56] Forestano R T, Matchev K T, Matcheva K, Roman A, Unlu E B and Verner S 2023 Discovering sparse representations of lie groups with machine learning (arXiv:2302.05383 [hep-ph])
[57] Forestano Roy *et al* University of Florida 2023 Deep learning symmetries (available at: https://github.com/royforestano/Deep_Learning_Symmetries) (Accessed 2 June 2023)