Finding discrete symmetry groups via Machine Learning

Pablo Calvo-Barlés,[1,2] Sergio G. Rodrigo,[1,3] Eduardo Sánchez-Burillo,[4] and Luis Martín-Moreno[1,2,*]

[1] *Instituto de Nanociencia y Materiales de Aragón (INMA), CSIC-Universidad de Zaragoza, 50009 Zaragoza, Spain*
[2] *Departamento de Física de la Materia Condensada, Universidad de Zaragoza, 50009 Zaragoza, Spain*
[3] *Departamento de Física Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain*
[4] *PredictLand S.L., 50001 Zaragoza, Spain*

(Dated: September 10, 2024)

We introduce a machine-learning approach (denoted Symmetry Seeker Neural Network) capable of automatically discovering discrete symmetry groups in physical systems. This method identifies the finite set of parameter transformations that preserve the system's physical properties. Remarkably, the method accomplishes this without prior knowledge of the system's symmetry or the mathematical relationships between parameters and properties. Demonstrating its versatility, we showcase examples from mathematics, nanophotonics, and quantum chemistry.

Symmetries play an essential role in the understanding of physical theories. For example, continuous symmetries lead to conservation laws [1], and discrete symmetries (such as parity or time reversal) are key in studying spectral degeneracies in quantum mechanical systems [2].

Unveiling hidden symmetries is thus of immense interest in many fields of physics. Until recently, this task relied solely on human intuition. However, recent advances in Machine Learning (ML), and especially in Neural Networks (NN) [3–7], have opened up new and efficient ways of analyzing data. NNs have already been successfully employed in problems such as learning conservation laws and physically relevant parameters of dynamical systems [8–16]; finding coordinate transformations that reveal hidden symmetries [17, 18]; learning Lie algebras [19–23]; discovering transformations that preserve the statistical distribution of data sets [22, 24]; recognizing symmetries from NN embedding layers [21, 25] and classifying whether pairs of events are related by symmetry or not [12, 26].

A common scenario not addressed in prior studies arises when the physical magnitudes of a system depend on parameters possessing a discrete set of symmetry transformations that leave these magnitudes invariant. To find these transformations, we introduce the Symmetry Seeker Neural Network (SSNN), an NN architecture capable of discovering the matrix representation of all symmetry group elements solely from a dataset of physical parameters and their corresponding measurable magnitudes. Importantly, our approach demands no prior knowledge of the symmetry group or the mathematical relationships between parameters and physical properties. Additionally, the SSNN can tackle inverse design problems with symmetry-related multivalued solutions.

The text is structured as follows: we first introduce the SSNN algorithm and showcase its effectiveness on a simple mathematical problem. Then, we apply this method to examples in nanophotonics and quantum chemistry.

To define the problem in general terms, we consider a function $\vec{y}(\vec{x})$, where the "magnitudes" $\vec{y} \in \mathbb{R}^m$ depend on the "parameters" $\vec{x} \in \mathbb{R}^n$. The "system" may present discrete symmetries, so symmetry-related parameters lead to the same magnitudes. If so, the symmetry transformations form a finite group $G = \{d_1, ..., d_K\}$ of order K with a $n \times n$ matrix representation $\{\hat{D}_1, ..., \hat{D}_K\}$ that satisfies:

$$\vec{y}(\vec{x}) = \vec{y}(\hat{D}_\alpha \vec{x}),\ \forall \vec{x}, \alpha. \tag{1}$$

We assume that all information about the system is represented by N training samples $\{(\vec{x}_i, \vec{y}_i)\}_{i=1}^N$ (where $\vec{y}_i = \vec{y}(\vec{x}_i)$). The goal is to find an accurate prediction $\{\hat{D}_\alpha^{\mathrm{NN}}\}_{\alpha=1}^K$ of the unknown ground truth symmetry matrices $\{\hat{D}_\alpha\}_{\alpha=1}^K$.

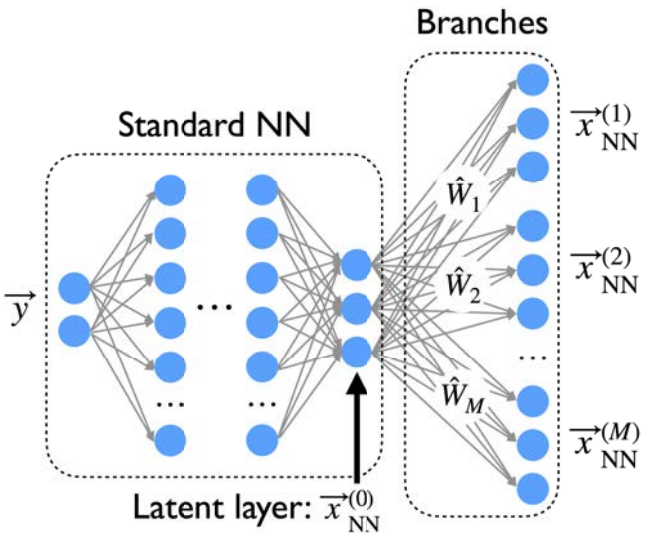

FIG. 1: Schematic representation of the SSNN: Magnitudes $\vec{y}$ enter a standard NN, which yields a vector $\vec{x}^{(0)}_{\mathrm{NN}}$. This NN is complemented with a set of M branches. The α-th branch performs a linear transformation $\hat{W}_\alpha \vec{x}^{(0)}_{\mathrm{NN}}$, providing a prediction $\vec{x}^{(\alpha)}_{\mathrm{NN}}$ for the physical parameters. Without the branches, the presence of symmetries would lead to incorrect predictions. In contrast, the SSNN, trained with the customized loss function given by Eq. (2), predicts parameters accurately from available data. After training, the branches $\hat{W}_\alpha$ provide a representation of the system's symmetry group.

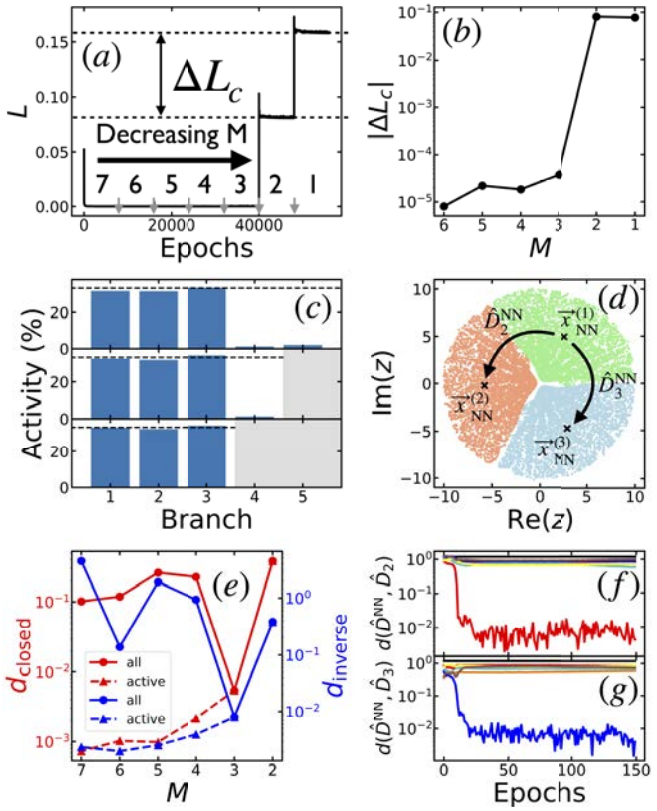

FIG. 2: Iterative SSNN algorithm for $w = z^3$. (a) Learning curve with grey arrows indicating the epoch of branch removal. (b) Loss jump dependence with the number of branches. (c) A_α for $\alpha = 1, ..., M$ at $M = 5$ (above), $M = 4$ (middle), and $M = 3$ (below), with the black dashed line representing $1/K = 1/3$. Grey panels at $M = 4, 3$ indicate removed branches. (d) Predictions of the training data set from SSNN with $M = 3$. Three points highlighted by black crosses represent predictions $\vec{x}^{(\alpha)}_{\rm NN}(\vec{y})$ from a single input data related by $\hat{D}^{\rm NN}_\alpha$. (e) Group metrics showing $d_{\rm closed}$ (red curves) and $d_{\rm inverse}$ (blue curves) for all branches (solid lines) and active branches (dashed lines). (f) Evolution of the distance between the $\hat{D}^{\rm NN}_\alpha$ and the ground truth matrix $\hat{D}_2$ during the first training stage ($M = 7$). (g) Same as (f) for $\hat{D}_3$.

For that, we introduce the SSNN architecture. It is designed to find the inverse function $\vec{x}(\vec{y})$ which, if a symmetry relation presented in Eq. (1) holds, will be K-fold multivalued. This NN depicted in Fig. 1, takes $\vec{y}$ as input and outputs M predictions, denoted as $\{\vec{x}^{(\alpha)}_{\rm NN}\}^M_{\alpha=1}$ (where i indexes training samples, α indexes symmetry matrices and M is an hyperparameter). It comprises two parts. The first part is a neural network producing an n-dimensional vector $\vec{x}^{(0)}_{\rm NN}$, with its last layer termed the "latent" layer. The second part consists of M trainable linear transformations $\hat{W}_\alpha$, referred to as branches, which generate M n-dimensional vectors $\vec{x}^{(\alpha)}_{\rm NN} = \hat{W}_\alpha \vec{x}^{(0)}_{\rm NN}$.

The SSNN's performance is guided by the loss function, given by:

$$L = \frac{1}{2N} \sum_i \min_\alpha L_i^{(\alpha)}, \tag{2}$$

where the loss associated to the α-th branch is $L_i^{(\alpha)} = ||\vec{x}_i - \vec{x}^{(\alpha)}_{\rm NN}(\vec{y}_i)||^2$, i.e., the mean square error between the branch prediction $\vec{x}^{(\alpha)}_{\rm NN}(\vec{y}_i)$ and the training sample $\vec{x}_i$. For each training data point i, only the branch with the lowest $L_i^{(\alpha)}$ (termed the "winner" branch for point i) contributes to the loss L. To evaluate how each branch contributes, we introduce the branch activity $A_\alpha = N_\alpha/N$, where N_α counts the number of times branch α produced winning predictions across the entire training dataset.

Multibranched NNs have already been used in the past to tackle multivalued inverse designs [27]. However, the SSNN incorporates two modifications: first, the latent layer matches the parameter space's dimension, and second, branches are defined by linear transformations. These changes are crucial for establishing a link between branches and symmetry representation matrices, as shown in the subsequent analysis.

To find the symmetry group, we employ an iterative training method, where the least active branch is sequentially eliminated. The process is as follows: we begin by initializing the SSNN with a sufficiently large number of branches, denoted as $M_{\rm max}$ (which should exceed the group order K). The SSNN is then trained until the loss L converges. At this point we compute the branch activities and remove the branch with the lowest A_α. This training and pruning cycle repeats until the network retains only one branch. Each cycle is termed a training stage, and we record the converged loss $L_c(M)$ at each stage.

In a postprocessing stage, we use $L_c(M)$ and other measures (that will be defined later on) to extract both the group order and the group representation. Notice that although this paper emphasizes finding the discrete symmetry group, the SSNN also solves the inverse problem. When $M = K$, the branches of the trained SSNN provide all possible parameters $\vec{x}$, each one satisfying the multivalued equation $\vec{y}(\vec{x})$. Since all parameters $\vec{x}$ are mathematically related by the group symmetry identified by the SSNN, our method operates without the need for direct feed-forward networks to validate the results, differing from the approach in Ref. [27].

For a practical demonstration, we apply our method to a mathematical example: the complex cube root $w = z^3$, where z and w are complex numbers. This problem exhibits a symmetry group of rotations of order $K = 3$ in the complex plane, as $z^3 = (e^{i2\pi/3}z)^3 = (e^{i4\pi/3}z)^3$, $\forall z$. Using the habitual terminology in Machine Learning, we associate $\vec{x} = ({\rm Re}(z), {\rm Im}(z))^T$ and $\vec{y} = ({\rm Re}(w), {\rm Im}(w))^T$, where T states for transposition. The 2×2 symmetry

matrices are:

$$\hat{D}_\alpha = \begin{pmatrix} \cos\theta_\alpha & \sin\theta_\alpha \\ -\sin\theta_\alpha & \cos\theta_\alpha \end{pmatrix}, \tag{3}$$

with $\theta_\alpha = 2\pi(\alpha-1)/3$, with $\alpha = 1, 2, 3$.

In order to find the symmetry relations using the SSNN, we train the model by randomly selecting 6000 values of z with $|z| < 10$. The main results are presented here, while the details of the training/validation data set generation, SSNN hyperparameters, and other examples for different root powers can be found in Appendix A.

The training process is illustrated in Fig. 2. In Fig. 2(a), we show the evolution of L as M decreases during the training epochs, starting from an initial value of $M_{\max} = 7$. Figure 2(b) shows the jump in the converged loss at consecutive stages, defined as $\Delta L_c(M) = L_c(M) - L_c(M+1)$. These plots indicate the presence of a symmetry group of order $K = 3$. If $(\vec{y})^{-1}$ is a K-fold symmetry-multivalued function, $M \geq K$ predictions are required to fit all possible $(\vec{y})^{-1}$ solutions and achieve $L_c \simeq 0$. However, when $M < K$, there are not enough branches to predict all possible solutions, leading to incorrect parameter predictions for certain data points, resulting in a significantly higher L_c. This is apparent in Fig. 2(b), where $\Delta L_c(M)$ leaps from 10^{-5} (for $M \geq 3$) to 10^{-1} (for $M < 3$), indicating that the group order is $K = 3$.

The branch activities further confirm this observation. In Fig. 2(c) we depict $\{A_\alpha\}_{\alpha=1}^{M}$ at the end of stages $M = 5, 4, 3$. In all cases, at the end of each training stage, precisely K active branches exhibit $A_\alpha \simeq 1/3 = 1/K$, while the remaining $M - K$ branches are inactive and have $A_\alpha \simeq 0$. This occurs because, during the training process, the activity of K branches is sufficient to provide all solutions to $(\vec{y})^{-1}$. In contrast, the remaining branches become less active, contributing less often to the loss and ultimately becoming detached from the training refinement.

Let's examine the active branches. Fig. 2(d) shows the SSNN predictions $\vec{x}_{\rm NN}^{(\alpha)}(\vec{y}_i)$ $(\alpha = 1, ..., M)$ for every training sample $\vec{y}_i$ at the end of the stage $M = 3$. Each branch offers predictions within a distinct, non-overlapping irreducible region of the symmetry group, collectively covering the entire parameter space. Additional plots of predictions from inactive branches for $M > K$, and a visualization of activities A_α, are provided in Appendix A.

Notably, we observe that 3-fold rotations transform each irreducible region into another, indicating a relationship between symmetry transformations and branch predictions. In fact, we can extract the predicted symmetry matrices $\hat{D}_\alpha^{\rm NN}$ from the active branches $\hat{W}_\alpha$. However, a degeneracy issue arises: any invertible matrix $\hat{A}$ fulfills $\hat{D}_\alpha \vec{x}_{\rm NN}^{(0)} = (\hat{D}_\alpha \hat{A}^{-1})(\hat{A}\vec{x}_{\rm NN}^{(0)})$, and thus there are infinite sets of $\{\hat{W}_\alpha\}$ that yield the same loss. To resolve this, we exploit the fact that any symmetry group includes the identity as an element. Hence, among the branches identified by the SSNN, we choose a reference matrix $\hat{W}_{\rm ref}$ (for definiteness, we choose $\hat{W}_{\rm ref} = \hat{W}_1$) and compute the predicted symmetry matrices as:

$$\hat{D}_\alpha^{\rm NN} = \hat{W}_\alpha \hat{W}_{\rm ref}^{-1}. \tag{4}$$

Apart from including the identity, a group must satisfy two other fundamental properties [28]: closure under multiplication ($\forall \hat{D}_\alpha, \hat{D}_\beta \in G,\ \exists \hat{D}_\gamma \in G$ such that $\hat{D}_\alpha \hat{D}_\beta = \hat{D}_\gamma$), and the inverse of every group element must also be a group element ($\forall \hat{D}_\alpha \in G,\ \exists \hat{D}_\gamma \in G$ such that $\hat{D}_\alpha \hat{D}_\gamma = \hat{D}_\gamma \hat{D}_\alpha = \hat{1}$). Since these two properties are not enforced in our calculation, we can employ them to validate the found symmetry matrices. For this purpose, we define two group metrics as functions of $\{\hat{D}_\alpha^{\rm NN}\}$ that quantify how well these matrices satisfy the group properties. For that, we first define a distance measure between two arbitrary matrices $\hat{A}$ and $\hat{B}$:

$$d(\hat{A}, \hat{B}) = \frac{1}{n^2} \sum_{kl} |a_{kl} - b_{kl}|, \tag{5}$$

where k and l index the matrix elements and n is the dimension of the matrices. The metric for determining whether the set $\{\hat{D}_\alpha^{\rm NN}\}$ is closed under multiplication ("closed metric") is defined as,

$$d_{\rm closed}(\{\hat{D}_\alpha^{\rm NN}\}) = \frac{1}{M^2} \sum_{\alpha\beta} \min_\gamma d(\hat{D}_\alpha^{\rm NN} \hat{D}_\beta^{\rm NN}, \hat{D}_\gamma^{\rm NN}), \tag{6}$$

Similarly, the "inverse metric" quantifies whether the set $\hat{D}_\alpha^{\rm NN}$ has an inverse belonging to the set:

$$d_{\rm inverse}(\{\hat{D}_\alpha^{\rm NN}\}) = \frac{1}{M} \sum_{\alpha} \min_\gamma d(\left[\hat{D}_\alpha^{\rm NN}\right]^{-1}, \hat{D}_\gamma^{\rm NN}). \tag{7}$$

The matrices $\{\hat{D}_\alpha^{\rm NN}\}$ form a group if and only if $d_{\rm closed} = d_{\rm inverse} = 0$.

In Fig. 2(e), we evaluate the existence of a symmetry group using two different approaches.

In the first approach, we compute the group metrics by considering all predicted matrices, both from active and inactive branches. For $M = K = 3$, both closed and inverse metrics approach zero, confirming that these matrices indeed form a group. However, when $M < K$, the predicted matrices do not fulfill the group properties after multiplication or inversion, causing an increase in the group metrics. Similarly, for $M > K$, inactive branches are not sufficiently optimized, yielding matrices that do not form a group representation, causing a rise in the group metrics.

In the second approach, even for $M > K$, we only consider predicted matrices from the active branches, excluding the inactive branches. The closed and inverse metrics remain small indicating that, even when M exceeds the

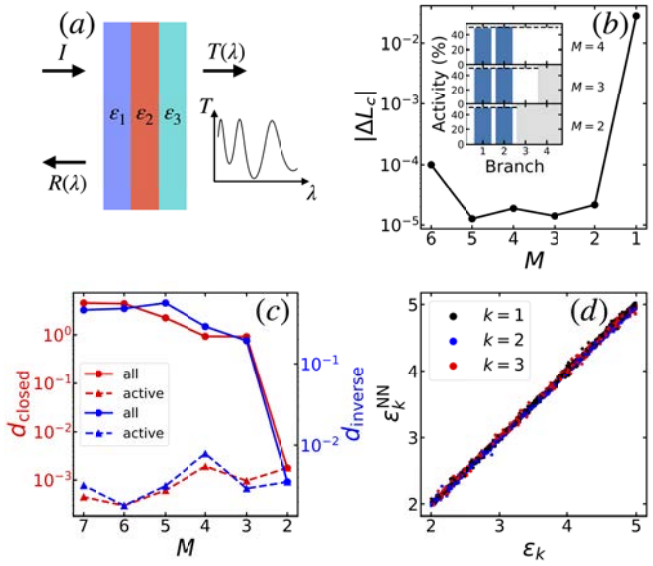

FIG. 3: Optical transmittance spectra symmetries in a multilayer stack: (a) Schematic representation of the considered photonic system. (b) Loss jump variation with the number of branches. The inset displays A_α for $\alpha = 1, ..., M$ at three different stages: $M = 4, 3, 2$. The black dashed line represents $1/K = 1/2$. The grey regions at $M = 3, 2$ indicate the removed branches. (c) Group metrics depicted as a function of M. (d) SSNN predictions as a function of the true values of ε_1, ε_2, and ε_3.

group order, the subset of active branches accurately predict the group representation.

In this simple example of the cubic root, where we know the ground truth symmetry matrices $\{\hat{D}_\alpha\}$, we can monitor the optimization progress of the predicted $\hat{D}_\alpha^{\rm NN}$ during the initial training stage. Figure 2(f) displays the distances between each predicted $\hat{D}_\alpha^{\rm NN}$ and $\hat{D}_2$ (see Eq. (3)), while Figure 2(g) does the same for $\hat{D}_3$ (we omit the curve for the identity transformation since $\hat{D}_1 = \hat{D}_1^{\rm NN}$ by design). In each panel, one of the curves (corresponding to a particular active branch) reaches a value significantly smaller than that of the other curves. This confirms that the set $\{\hat{D}_\alpha^{\rm NN}\}$ from active branches provide an excellent approximation to the exact $\{\hat{D}_\alpha\}$.

To show the generality of the method, let us now apply the SSNN to two physical examples. The first one involves a dielectric three-layer stack (Fig. 3(a)), each layer having a width $d = 250$ nm, and dielectric constants ε_i $(i = 1, 2, 3)$ ranging from 2 to 5. An electromagnetic plane wave with wavelength λ impinges the stack at normal incidence. By solving Maxwell's equations, we compute the transmittance spectrum $T(\lambda)$ for a given set $\{\varepsilon_i\}$ (see Appendix B for details). The mapping here relates the parameters $\vec{x} = (\varepsilon_1, \varepsilon_2, \varepsilon_3)^T$ to a transmittance spectrum evaluated at discrete wavelengths $\vec{y} = [T(\lambda_1), ..., T(\lambda_k), ..., T(\lambda_m)]^T$. We use $m = 100$ λ's, evenly distributed between 500 nm and 900 nm. Time reversal symmetry implies that the transmittance remains invariant when the stack order is inverted [29]. Thus, the problem presents a symmetry group consisting of the identity $(\hat{D}_1)$ and the inversion in the ordering of $\{\varepsilon_i\}$ $(\hat{D}_2)$.

Figure 3(b) shows $\Delta L_c(M)$, with an initial SSNN setup of $M_{\rm max} = 7$ branches. The substantial jump of $\Delta L_c(M)$ by several orders of magnitude for $M < 2$ strongly indicates that the group order is $K = 2$. Further evidence is seen in the inset of that panel, showing that, even for $M > 2$, there are two active branches with $A_\alpha \simeq 1/2$. The closed and inverse metrics, as depicted in Fig. 3(c) affirm that the matrices corresponding to the active branches indeed form a group (Appendix B shows that $\hat{D}_2^{\rm NN}$ converges to $\hat{D}_2$).

Finally, we showcase the potential of the proposed method in inverse design. To achieve this, we employ the SSNN prediction from the winner branch denoted as $(\varepsilon_1^{\rm NN}, \varepsilon_2^{\rm NN}, \varepsilon_3^{\rm NN})$ for each data point.

Figure 3(d) displays the comparison between the ground truth ε_k and the corresponding $\varepsilon_k^{\rm NN}$ for all multilayer stacks in the validation set. The results show an excellent agreement, confirming the method's effectiveness in providing inverse design solutions for all symmetry-related cases.

Lastly, we apply our method to molecular systems in which discrete symmetry groups play an important role. We analyze two different scenarios of increasing complexity. In both cases, molecules are described by a tight-binding model with one orbital per atom.

In the first scenario, we consider molecules with 4 atoms, each one of them with an orbital energy e_i $(i = 1, ..., 4)$. We assume electron hopping only between nearest-neighbor atoms, with a hopping parameter $J = 1$, serving as our energy unit. The molecule's geometry defines the hopping terms (see Appendix C for details). The molecular electronic energies are denoted as E_i, for $i = 1, ..., 4$. Here, the mapping $\vec{y}$ relates a tuple of atomic energies, $\vec{x} = (e_1, e_2, e_3, e_4)^T$, to a tuple of molecular eigenvalues, $\vec{y} = (E_1, E_2, E_3, E_4)^T$. We explore three different molecular geometries: rectangular, square, and tetrahedral [see insets in Figs. 4(a,c,e)]. For each geometry, we generate 100,000 "molecules" by randomly selecting values for $\{e_i\}$ (ranging from 1 to 6) and solving the Schrödinger equation to determine $\{E_i\}$. The symmetry transformations of each molecule preserve the set of eigenvalues and include rotations, reflections, and improper rotations. The group orders are $K = 4$ for the rectangle, $K = 8$ for the square, and $K = 24$ for the tetrahedron. These transformations imply that certain permutations of the atomic energies $\{e_i\}$ result in the same $\{e_i\}$. For instance, in the square molecule, a $\pi/2$ rotation is given by the transformation $\hat{D}$ such that $\hat{D}(e_1, e_2, e_3, e_4)^T = (e_4, e_1, e_2, e_3)^T$.

For each molecular geometry, we train a different SSNN. The corresponding $\Delta L_c(M)$ is presented in Fig-

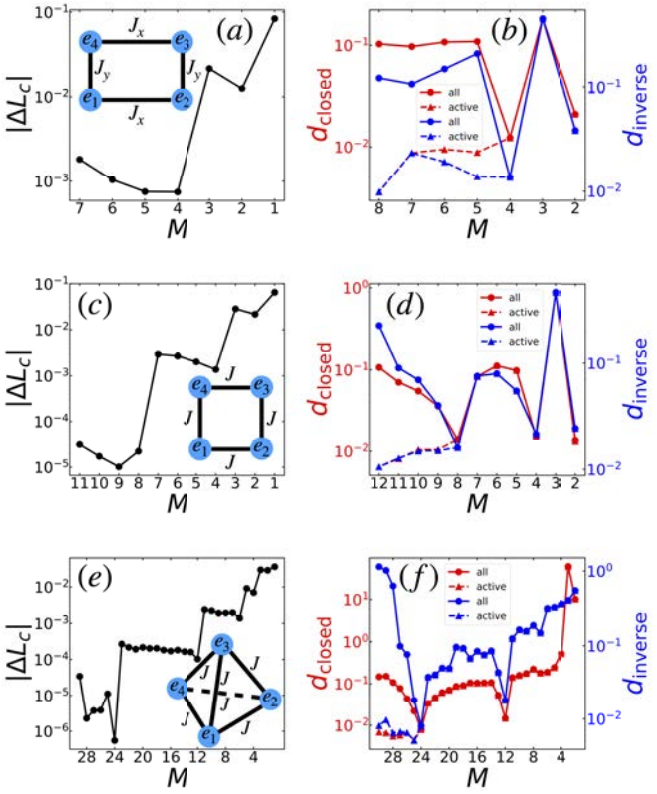

FIG. 4: Applying SSNN to discover symmetries in a 4-atom molecular spectra. For each molecular geometry (rectangle, square, tetrahedron), we present the dependence with the number of branches of both the loss jump and the group metrics (circles: all branches are considered; triangles: only active branches are included in the calculation). Figures (a-b), (c-d), and (e-f) correspond to the rectangular, square, and tetrahedral molecules, respectively.

ures 4(a,c,e), while Figures 4(b,d,f) display the corresponding closed and inverse group metrics, considering either all branches or only the active ones. The presence of jumps in $\Delta L_c(M)$ and the behavior of the group metrics as M changes clearly indicate that the SSNN consistently identifies the correct group order and symmetry matrices. This conclusion is further supported by the convergence of the predicted matrices to the ground truth matrices, as shown in Appendix C.

Notably, for each molecular geometry, deep minima appear in both $d_{\text{closed}}(M)$ and $d_{\text{inverse}}(M)$ at values $M < K$. These minima correspond to the set $\{\hat{D}^{\text{NN}}_\alpha\}$ forming a subgroup. For instance, for the tetrahedron, for $M = 12$ the set $\{\hat{D}^{\text{NN}}_\alpha\}$ represents the subgroup consisting of the identity, 3 C_2 rotations, and the 8 C_3 rotations.

In the second scenario, we consider a more complex system, where the parameter space has more degrees of freedom, and the symmetry group exhibits a more complicated structure. The molecular geometry is depicted in the inset to Fig. 5(a). The 5 atoms represented in red have an atomic energy μ, while the energy of the 8 atoms in blue is taken randomly. Only nearest-neighbor hoppings are considered; in all cases, its value is J, except for the 4 blue atoms forming the central ring (atoms 3-6) where the hopping is J'. For definiteness, in this example, we have taken the arbitrary values $\mu = -1$, $J = 3$, $J' = 0.5$. As in the previous example, we generate 100,000 "molecules" by randomly selecting values for $\{e_i\}$ within the interval [1,6]. In this case, $\vec{x} = (e_1, ..., e_8)^T$ and the magnitudes are the 13 molecular eigenvalues, $\vec{y} = (E_1, ..., E_{13})^T$.

We implement the SSNN's training procedure starting with $M_{\text{max}} = 80$ branches (further details of the SSNN architecture can be found in Appendix C). Figure 5(a) shows the loss jump $\Delta L_c(M)$ and Figure 5(b) shows the group metrics, both as a function of the decreasing M. The jump in ΔL_c and the simultaneous minimum of the group metrics when all branches are considered that occur at $M = 64$ strongly indicate that the system presents a symmetry group of order $K = 64$. An analysis of the branch activities reinforces this result; there are always 64 active branches for $M \geq K$ (see Appendix C). Notice that the combination of all considered metrics allows the unambiguous detection of the underlying symmetry, even when the group order is larger than in the previously considered examples and, thus, the relative variation of the metrics is smaller when one branch is removed away from the optimum architecture $M = K$.

In this study, we introduced the SSNN, a neural network architecture that can uncover discrete symmetries in parameter space that leaves the system invariant, providing both the group order and the group representation.

Importantly, the SSNN method relies solely on a training dataset without requiring prior knowledge of either the system symmetry, the function $\vec{y}(\vec{x})$, or an oracle approximation of $\vec{y}$. Additionally to the unveiling of symmetries, the SSNN can be used to perform inverse design in symmetric systems, identifying all sets of symmetry-related parameters that yield specific magnitudes in the target physical system.

We acknowledge Project PID2020-115221GB-C41, financed by MCIN/AEI/10.13039/501100011033, and the Aragon Government through Project Q-MAD. The code for this paper can be found online at [30].

Appendix A: n-th root problem

In this section, we present the technical details of the SSNN training algorithm and data generation for the n-th complex root problem $w = z^n$. We extend our exploration to various root problems, including the complex third root (shown in the main text), the real second root, and the complex fourth and fifth roots. Additionally, we provide visual representations to enhance the understanding of the obtained activity values.

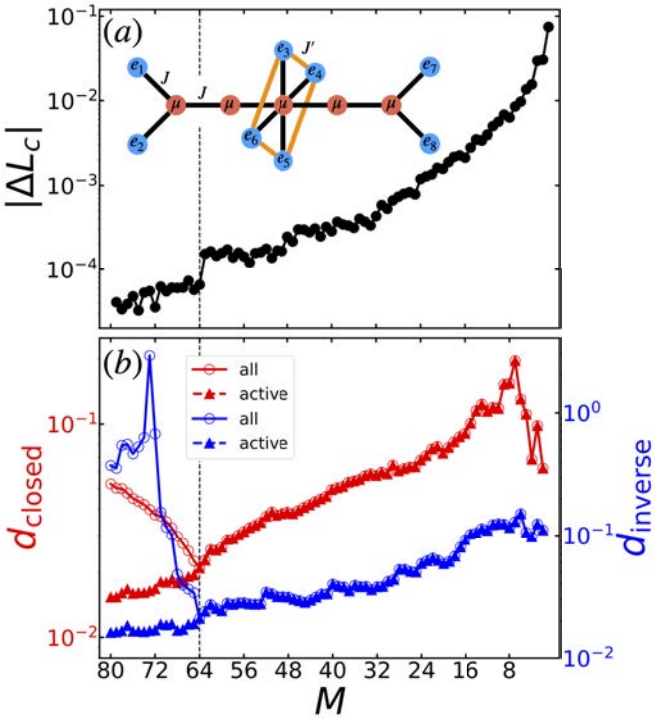

FIG. 5: Application of SSNN to discover symmetries in spectra of a complex chain-like molecule with randomly coupled atoms. The inset to (a) shows a schematic representation of the molecule geometry. The atoms in blue have random atomic energy $\epsilon_i \in [1, -6]$, while those in red have energy $\mu = -1$. Only nearest-neighbor hoppings are considered; in all cases, its value is $J = 3$, except for the 4 blue atoms forming the central ring (atoms 3-6) where the hopping is $J' = 0.5$. (a) Loss jump as a function of the decreasing number of branches. (g) Group metrics vs. the number of branches, when either all branches or only the active ones are included in the calculation.

SSNN training

For all the NN calculations in this paper, we utilized Tensorflow [31] and Keras [32] libraries, with Adam [33] as the gradient-descent optimizer for the loss function. Each n-th root problem was trained with 5000 training samples and 1000 validation samples. The z values were randomly generated from a uniform distribution within the region $V = \{z \in \mathbb{C} \mid |z| \leq 10\}$ ($\mathbb{R}$ for the second root). Data was normalized as $z_i^{\text{norm}} = z_i/10$ and $w_i^{\text{norm}} = z_i/10^n$.

The standard NN architecture consisted of an input layer (2 neurons for the third, fourth, and fifth roots, and 1 neuron for the second), followed by two dense hidden layers with 10 neurons each using a sigmoid activation function. The latent layer had no activation function, only weights and biases. A learning rate of 0.01 and a mini-batch size of 128 samples were used. The SSNN was trained for 8000 epochs at each M-stage to ensure loss convergence. The maximum number of branches was $M_{\max} = 7$ for each n-th root problem. Supplementary results for the second, fourth, and fifth roots can be observed in Fig. 6.

Branch activities

We now show the visual representation of predictions and activities from active and inactive branches. Fig. 7 corresponds to the second root problem. We represent, as function of the training input samples $\{\vec{y}_i\}$, the predictions of each branch: $\{\vec{x}^{(0)}_{\text{NN}}(\vec{y}_i)\}, ..., \{\vec{x}^{(\alpha)}_{\text{NN}}(\vec{y}_i)\}, ..., \{\vec{x}^{(M)}_{\text{NN}}(\vec{y}_i)\}$. Three different M-stages are shown: $M = K$ (Fig. 7a), $M = K + 1$ (Fig. 7b) and $M = K + 2$ (Fig. 7c). The inset displays the activities of the branches with the same color as the curves. Active branches accurately fit the function $x = \pm\sqrt{y}$ throughout the full range of the parameter space V, used in the training, while inactive branches do not provide a correct approximation, consistent with the A_α values. This is in accordance with what is expected from the A_α values.

Figs. 8, 9 and 10 correspond to the third, fourth and fifth root, respectively. In these figures, we only represent the parameter space (complex plane z). Two types of scatter plots are shown for each training stage $M = K$, $M = K+1$ and $M = K+2$. The first type (upper row in the figures) displays $\{\vec{x}^{(\alpha)}_{\text{NN}}(\vec{y}_i)\}$ for $\alpha = 1, ..., M$. As expected, the active branches cover the irreducible regions maximally. However, small regions in V, particularly near the boundaries between irreducible regions, are not fully covered by the active branches. Notice that, close to boundaries, two branches provide predictions with similar losses. This makes the inactive branches specialize in covering these small regions, explaining why A_α is not exactly zero for the inactive branches. The second type (lower row) plots the output samples $\{\vec{x}_i\}$ from the training data set (uniformly distributed). The color of each point represents the branch with the minimum MSE prediction for that sample, indicating the area of the parameter space covered by each branch, directly related to its activity. The corresponding activities are shown in the insets.

Appendix B: Optical transmission in multilayer stack

In this section, we outline the calculation details of the optical transmission problem discussed in the main text. First, we show how to calculate the transmission spectrum from a given set of dielectric materials. Subsequently, we present the SSNN training specifics, along with the convergence of the predicted symmetry matrices to the ground truth matrices.

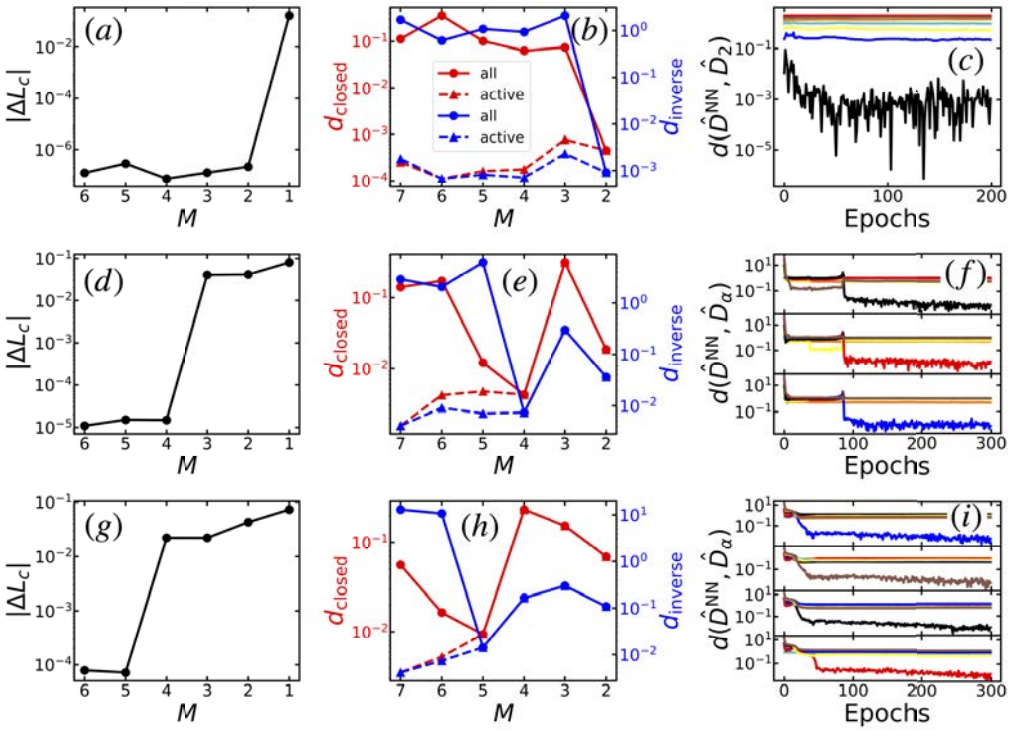

FIG. 6: Symmetries in $w = z^n$: For each n-th root problem, we present three results: the loss variation as a function of the number of branches, the group metrics (for all branches and the active ones), and the distances between the predicted and ground truth symmetry matrices. Specifically, we show the real second root in (a-c), the complex fourth root in (d-f), and the complex fifth root in (g-i).

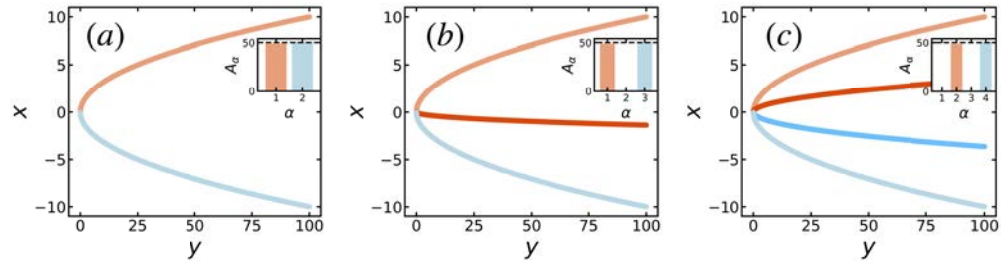

FIG. 7: Visualization of predictions made by the active/inactive branches in the second root problem. The plots display the branch predictions as a function of the training data $\{y_i\}$, with each branch represented by a different color. The activities A_α for $\alpha = 1, ..., M$ are shown in an inset using the same colors as the curves. (a) $M = 2$ branches. (b) $M = 3$ branches. (c) $M = 4$ branches.

Transmission spectrum calculation

We study a photonic system described in the third section of the main text. The system consists of three stacked layers (labeled as 1, 2, and 3) with the same width d and dielectric constants ε_1, ε_2, and ε_3 (see Fig. 11). In our calculations, we set $d = 250$ nm, and ε_i varies within the range $[2, 5]$. Media 0 and $0'$ are treated as vacuum ($\varepsilon_0 = \varepsilon_{0'} = 1$).

An incident electromagnetic wave incident, with wavelength λ, impinging normally into the system. This incident wave is transmitted and reflected at each interface. We express the magnitude of the electric field in each medium as follows:

$$E_0 = e^{ikz} + re^{-ikz}, \tag{B1}$$

$$E_j = A_j e^{ik_j(z-(j-1)d)} + B_j e^{-ik_j(z-(j-1)d)}, \tag{B2}$$

$$E_{0'} = te^{ik(z-3d)}. \tag{B3}$$

In the expressions above, A_j and B_j are the complex amplitudes of upgoing and downgoing waves, respectively, at each medium j, while t and r are are the complex transmission and reflection coefficients of the whole system. Besides, $k = 2\pi/\lambda$ is the wavevector at the vacuum and

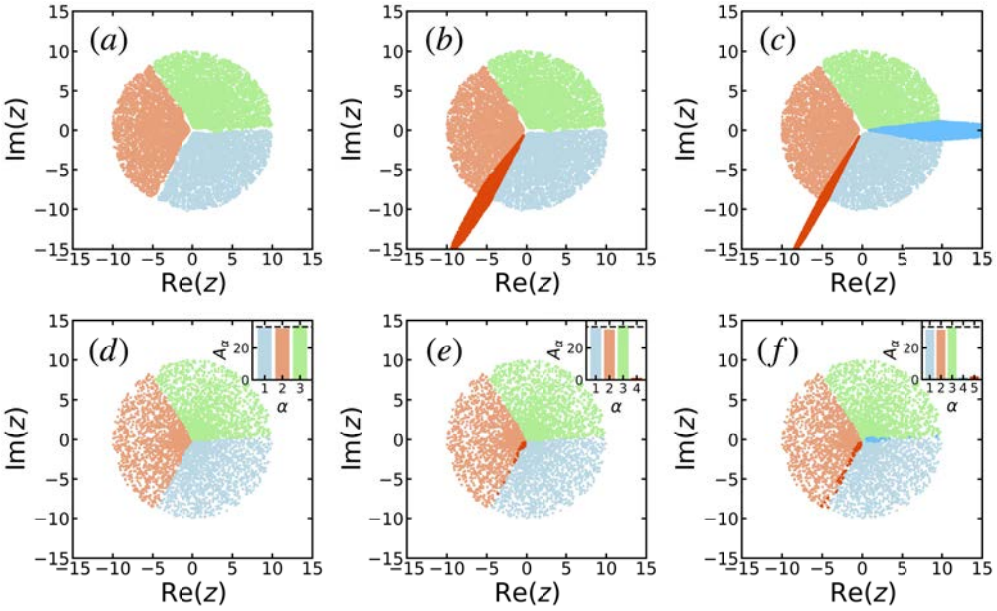

FIG. 8: Visualization of predictions made by the active/inactive branches in the third root problem. The upper row plots display the branch predictions in the parameter space, with each branch represented by a different color. The lower row plots show the output training samples$\{\vec{x}_i\}$ in the parameter space, where each color corresponds to the branch with the minimum-MSE prediction. An inset shows the activities A_α for $\alpha = 1, ..., M$ with the same colors as the points. (a,d) $M = 3$ branches. (b,e) $M = 4$ branches. (c,f) $M = 5$ branches.

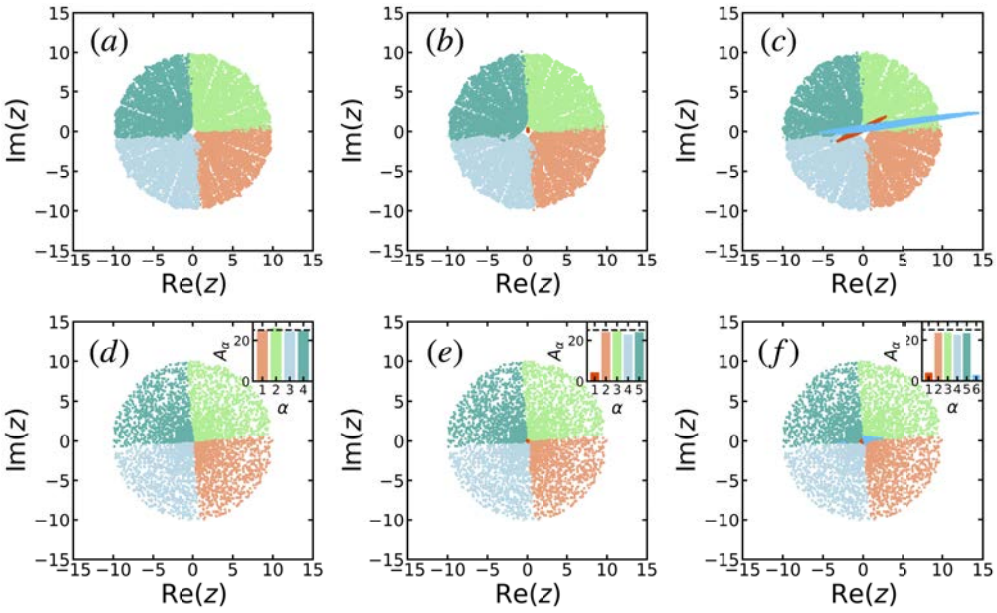

FIG. 9: Visualization of predictions made by the active/inactive branches in the fourth root problem. (a,d) $M = 4$ branches. (b,e) $M = 5$ branches. (c,f) $M = 6$ branches.

$k_j = \sqrt{\varepsilon_j}k$ is the wavevector at the medium j. Additionally, $k = 2\pi/\lambda$ denotes the wavevector in vacuum, and $k_j = \sqrt{\varepsilon_j}k$ is the wavevector in medium j. Using these electric field magnitudes, we can calculate the magnetic field modules H_j by applying the following relations derived from Maxwell's equations:

$$-\vec{u}_z \times \vec{H}_j = \pm y_j \vec{E}_j, \tag{B4}$$

In the expressions above, $\vec{u}_z$ represents the unitary vector in the z direction, and $y_j = \sqrt{\varepsilon_j}$ is the modal admittance. The positive sign is used for the transmitted modes, and the negative sign is used for the reflected modes. Once we

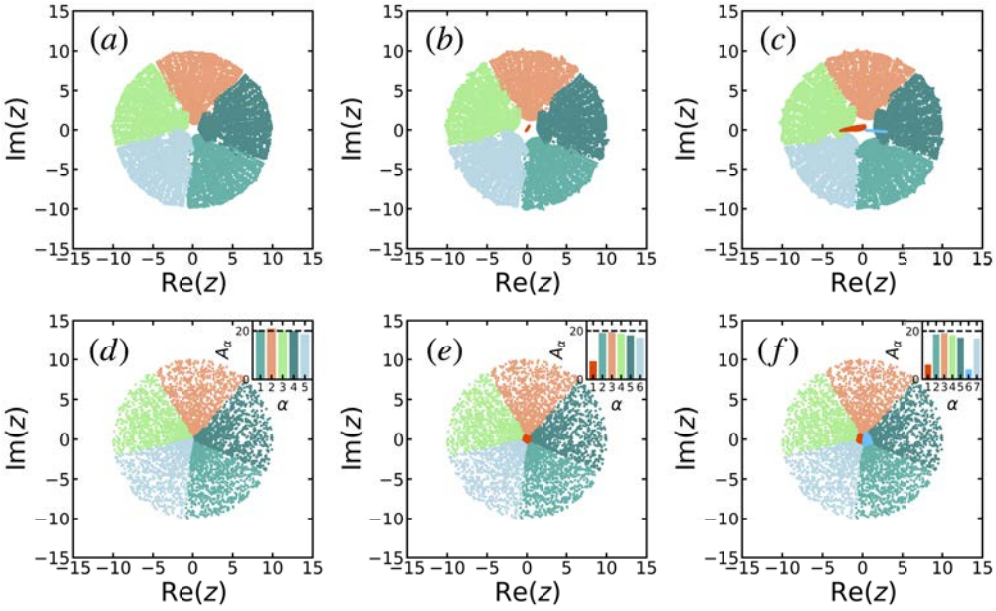

FIG. 10: Visualization of predictions made by the active/inactive branches in the fifth root problem. (a,d) $M = 5$ branches. (b,e) $M = 6$ branches. (c,f) $M = 7$ branches.

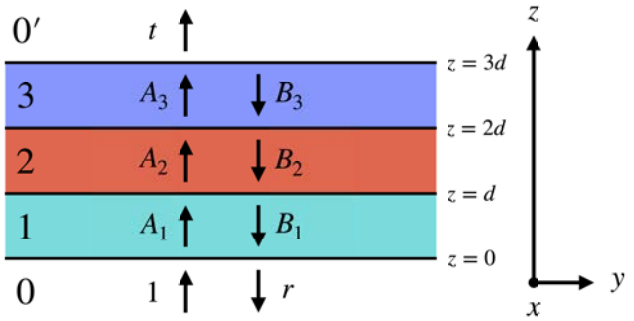

FIG. 11: Schematic representation of the dielectric multilayer stack. The direction of light propagation is along the z-axis. The amplitudes of the transmitted modes are denoted as A_j, while the amplitudes of the reflected modes are denoted as B_j. The transmission coefficient is represented by t, and the reflection coefficient by r.

have obtained the electric and magnetic fields, we apply the boundary conditions at each interface:

$$E_j(jd) = E_{j+1}(jd), \tag{B5}$$

$$H_j(jd) = H_{j+1}(jd), \tag{B6}$$

for $j = 0, 1, 2, 3$. We then obtain a linear system of 8 equations with 8 unknowns, $r, A_1, B_1, A_2, B_2, A_3, B_3$ and t:

$$\begin{cases} r - A_1 - B_1 = -1 \\ -r - y_1 A_1 + y_1 B_1 = -1 \\ A_1 e^{ik_1 d} + B_1 e^{-ik_1 d} - A_2 - B_2 = 0 \\ y_1 e^{ik_1 d} A_1 - y_1 e^{-ik_1 d} B_1 - y_2 A_2 + y_2 B_2 = 0 \\ e^{ik_2 d} A_2 + e^{-ik_2 d} B_2 - A_3 - B_3 = 0 \\ y_2 e^{ik_2 d} A_2 - y_2 e^{-ik_2 d} B_2 - y_3 A_3 + y_3 B_3 = 0 \\ -t + e^{ik_3 d} A_3 + e^{-ik_3 d} B_3 = 0 \\ -t + y_3 e^{ik_3 d} A_3 - y_3 e^{-ik_3 d} B_3 = 0. \end{cases} \tag{B7}$$

To calculate the transmission coefficient t for a given wavelength λ, we solve this linear system. After this, the transmittance is obtained as denoted as $T(\lambda) = |t|^2$. For the SSNN training data set, we calculate transmittance spectra by considering a discretized set of $m = 100$ equidistant wavelengths between $\lambda_{\min} = 500$ nm and $\lambda_{\max} = 900$ nm. An illustrative example is presented in Fig. 12.

SSNN training

We have trained the SSNN with 9000 training samples and 1000 validation samples. For the SSNN, we have normalized the input samples $\vec{\varepsilon}_i$ and output samples $\vec{T}_i$ as follows: $\vec{\varepsilon}_i^{\,\text{norm}} = \left(\vec{\varepsilon}_i - (2,2,2)^T\right)/3$ and $\vec{T}_i^{\,\text{norm}} = \left(\vec{T}_i - (T_{\min}, ..., T_{\min})^T\right)/(T_{\max} - T_{\min})$, being $T_{\max}$ and $T_{\min}$ the highest and lowest transmission values in the training data set, respectively. The standard section of the SSNN architecture consists of an input layer with

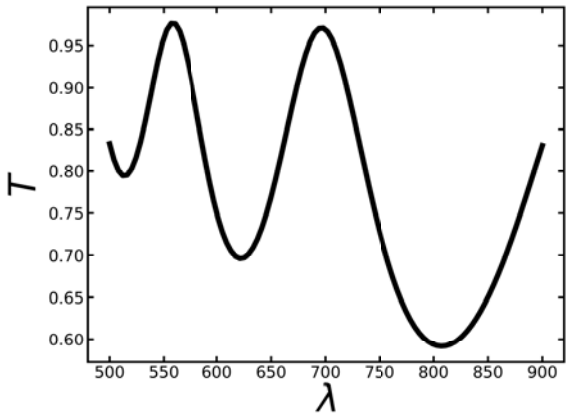

FIG. 12: Example of the transmittance spectrum. The corresponding dielectric constants of the materials are $\varepsilon_1 = 2.5$, $\varepsilon_2 = 3.5$ and $\varepsilon_3 = 4.8$.

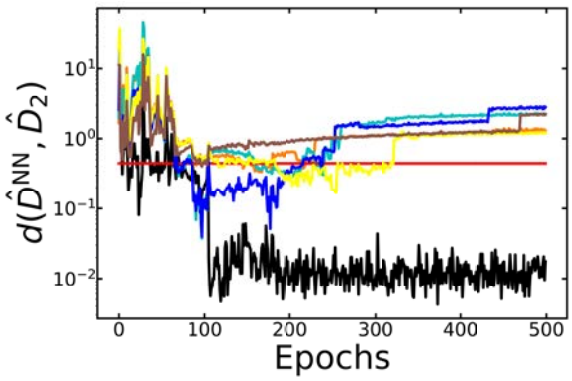

FIG. 13: Evolution of the distance between the predicted matrices and the ground truth matrix $\hat{D}_2$ during the first training stage (M = 7) in the optical transmission problem.

$m = 100$ neurons and two non-linear hidden layers, each with 200 neurons and sigmoid activation function. The latent layer is linear (i.e., we have not used an activation function). During training, we used a learning rate of 0.01 and a mini-batch size of 100 samples. The SSNN was trained for 2000 epochs for each M-stage, and the maximum number of branches was set to $M_{\max} = 7$.

In Fig. 13, we present the evolution of matrix distances between each learned matrix $\hat{D}_\alpha^{\rm NN}$ and the inversion transformation in the order of the dielectric constants $\hat{D}_2$ (a symmetry in parameter space originating from the time reversal) during the first training stage ($M = 7$). Interestingly, only one curve decays to 10^{-2}.

Appendix C: Energy spectrum in molecules

In this final section, we provide the calculation details of the molecule eigenspectrum problems as discussed in the main text. Firstly, we outline the method to calculate the eigenenergies of each molecular system (first and second scenario). Subsequently, we present the SSNNs training details, including the network hyperparameters and the branch activities at different M-stages. For the first scenario, we further demonstrate the convergence of the predicted symmetry matrices to the ground truth ones for each molecular geometry.

Eigenvalues calculation

In the first scenario, we consider the three types of 4-atom molecules introduced in the main text: rectangular, squared, and tetrahedral. We describe these systems with a tight-binding model with one atomic level per atom. Hopping is assumed to occur only between nearest neighbors. The Hamiltonians of each system are:

$$\hat{H}_R = \begin{pmatrix} e_1 & J_x & 0 & J_y \\ J_x & e_2 & J_y & 0 \\ 0 & J_y & e_3 & J_x \\ J_y & 0 & J_x & e_4 \end{pmatrix}, \tag{C1}$$

$$\hat{H}_S = \begin{pmatrix} e_1 & J & 0 & J \\ J & e_2 & J & 0 \\ 0 & J & e_3 & J \\ J & 0 & J & e_4 \end{pmatrix}, \tag{C2}$$

$$\hat{H}_T = \begin{pmatrix} e_1 & J & J & J \\ J & e_2 & J & J \\ J & J & e_3 & J \\ J & J & J & e_4 \end{pmatrix}, \tag{C3}$$

where $\{e_i\}$ are the atomic-energies and J, J_x, J_y are the hopping terms. The eigenvalues E_i are obtained by diagonalizing these matrices. For the training data set, we generate random uniform atomic energies within the range $[1, 6]$, while fixing the values of the hopping terms ($J = J_x = 1$ and $J_y = 0.75$). The data set of atomic energies is the same for each molecule, while the eigenvalues are different on each case.

In the second scenario, we consider the molecule introduced in the final part of the main text. Similarly, the system is described with a tight-binding model with one atomic level per atom. The Hamiltonian of the system is given by:

$$\hat{H} = \begin{pmatrix} e_1 & 0 & J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e_2 & J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ J & J & \mu & J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & J & \mu & J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & J & \mu & J & J & J & J & J & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J & e_3 & J' & 0 & J' & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J & J' & e_4 & J' & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J & 0 & J' & e_5 & J' & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J & J' & 0 & J' & e_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & J & 0 & 0 & 0 & 0 & \mu & J & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J & \mu & J & J \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J & e_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & J & 0 & e_8 \end{pmatrix}, \tag{C4}$$

where $\{e_i\}$ are the orbital-energies of the "free" atoms, μ is the orbital-energy of the "fixed" atoms and J, J' are the hopping terms. The eigenvalues E_i are obtained by diagonalizing these matrices. For the training data set, we generate random uniform atomic energies within the range $[1, 6]$, while fixing the values of the hopping terms and the fixed atoms energy ($J = 3$, $J' = 0.5$ and $\mu = -1$).

SSNN training

For each molecule in the first scenario, we have trained the SSNN with 70000 training samples and 30000 validation samples. The standard section of the SSNN is composed by an input layer with $m = 4$ neurons and 2 non-linear hidden layers with 100 neurons each, using sigmoid activation functions. The latent layer is linear. The learning rate was set to 10^{-2}, and a mini-batch size of 200 samples was used. We trained the SSNN for 4000 epochs for each M-stage. The maximum number of branches was set to $M_{\max} = 8$ for the rectangle, $M_{\max} = 12$ for the square, and $M_{\max} = 30$ for the tetrahedron.

Fig. 14 displays the branch activities for each molecular system at three different stages: $M = K + 2$, $M = K + 1$ and $M = K$. In all cases, the values A_α indicate the presence of K active branches and $M - K$ inactive branches.

Figs. 15, 16 and 17 depict the evolution of the distance between each learned matrix $\hat{D}_\alpha^{\rm NN}$ and each ground truth transformation $\hat{D}_\beta$ during the first training stage $M = M_{\max}$. As expected, these plots demonstrate that each ground truth symmetry matrix is discovered by a different active branch.

In the second scenario, we have also used 70000 training samples and 30000 validation samples. The standard section of the SSNN is composed by an input layer with $m = 13$ neurons and 2 non-linear hidden layers with 300 neurons each, using sigmoid activation functions. The latent layer is linear. A mini-batch size of 512 samples was used. In the first training stage, we trained the SSNN with $M_{\max} = 80$ branches during 30000 epochs (in order to ensure loss convergence) using a learning rate of 10^{-3}. In the remaining stages $M = 79, ..., 1$, the SSNN was trained with a learning rate of 10^{-5} with 150 epochs per stage.

Fig. 18 displays the branch activities in the latest molecular system at three different stages: $M = K + 2$, $M = K + 1$ and $M = K$. In all cases, the values A_α indicate the presence of K active branches and $M - K$ inactive branches.

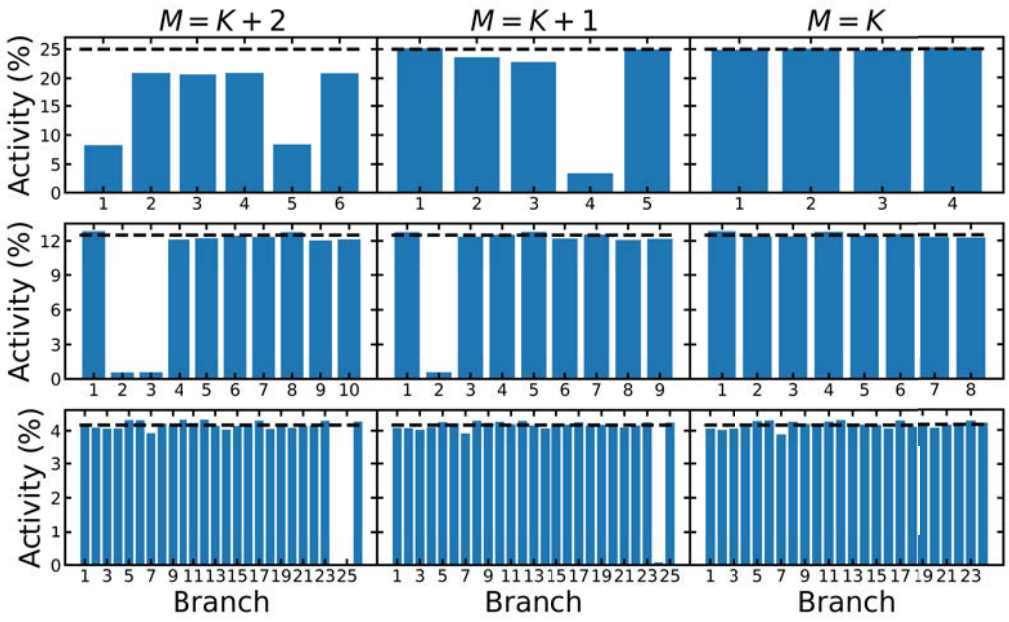

FIG. 14: Branch activities A_α for $\alpha = 1, ..., M$ at three different stages: $M = K + 2$, $M = K + 1$ and $M = K$. Each row corresponds to a different molecule: the upper row corresponds to the rectangular molecule, the middle row to the square molecule, and the lower row to the tetrahedral molecule.

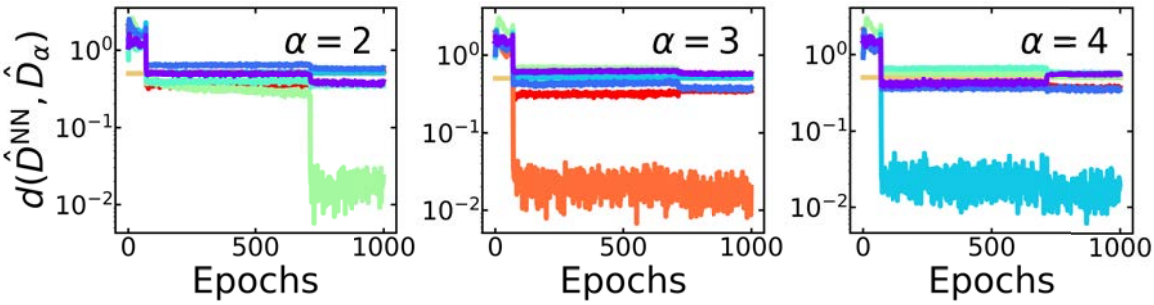

FIG. 15: Evolution of the distance between the predicted matrices and the ground truth ones during the first training stage (M = 8) for the rectangular molecule. Each panel corresponds to a specific ground truth symmetry $\hat{D}_\alpha$.

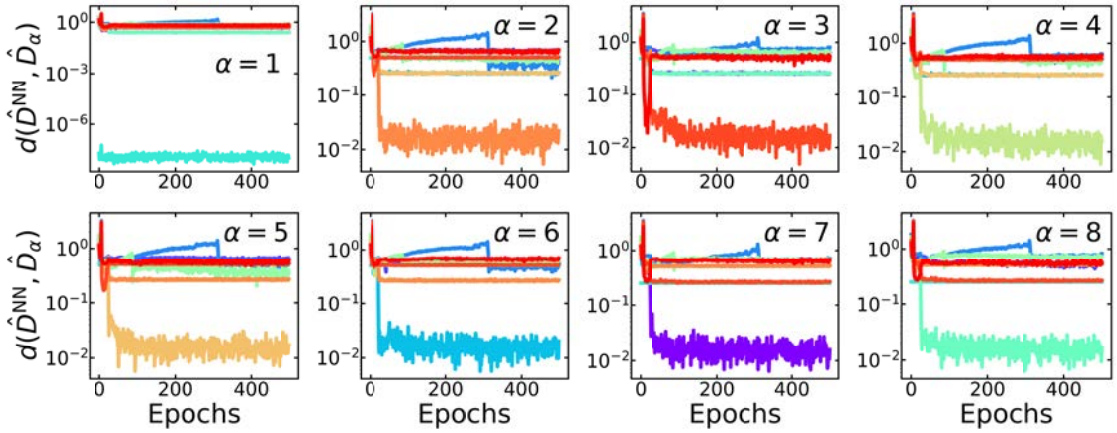

FIG. 16: Evolution of the distance between the predicted matrices and the ground truth ones during the first training stage (M = 12) for the square molecule. Each panel corresponds to a specific ground truth symmetry $\hat{D}_\alpha$.

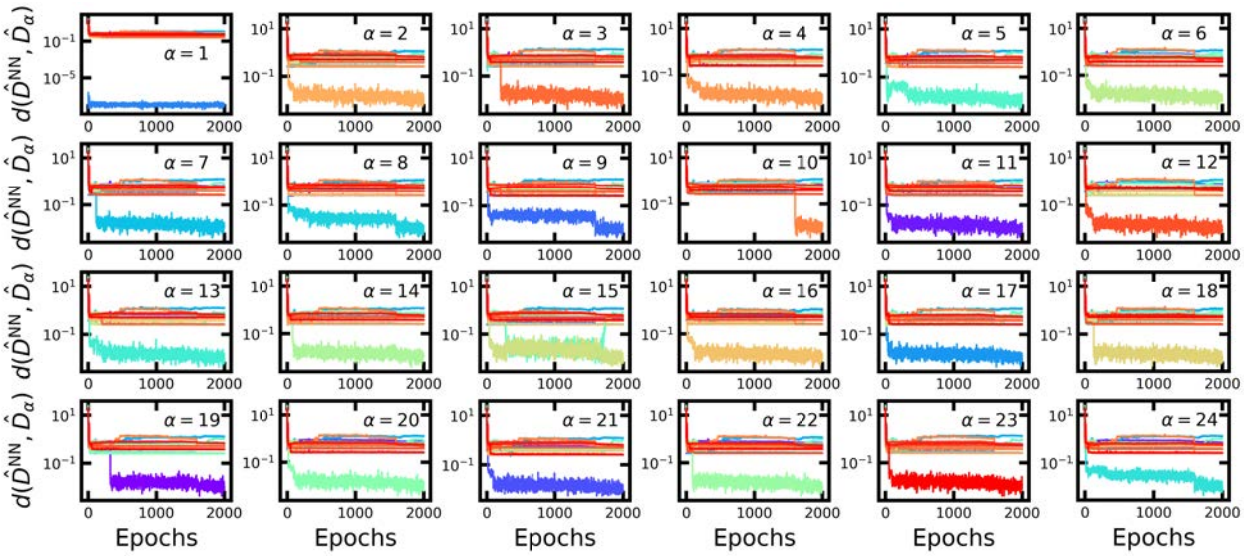

FIG. 17: Evolution of the distance between the predicted matrices and the ground truth ones during the first training stage (M = 30) for the tetrahedral molecule. Each panel corresponds to a specific ground truth symmetry $\hat{D}_\alpha$.

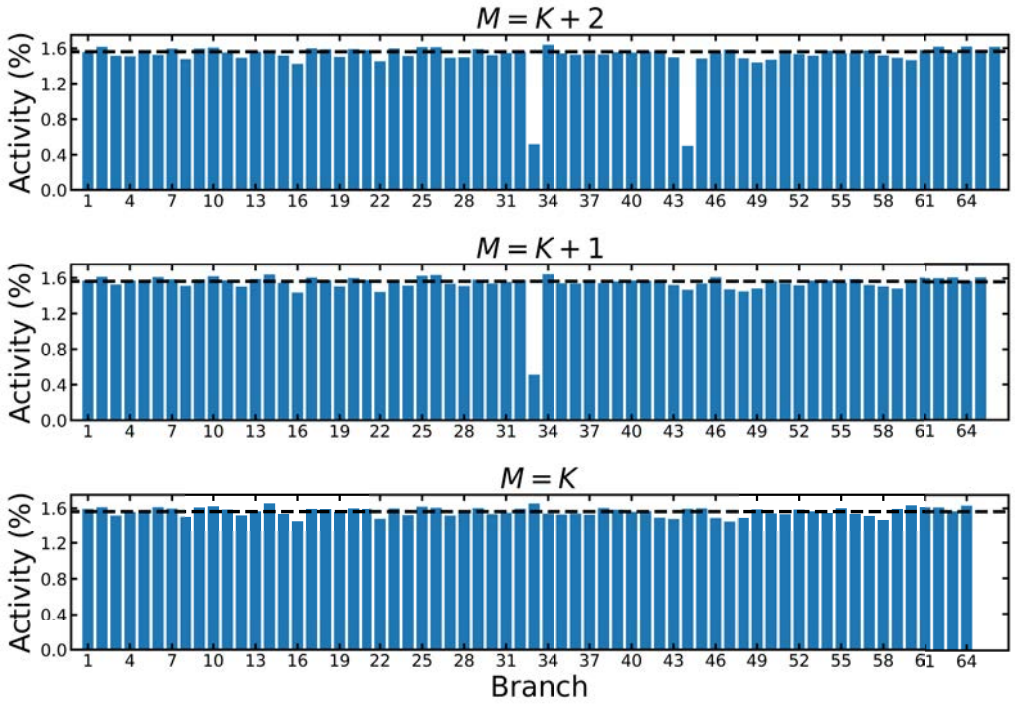

FIG. 18: Branch activities A_α for $\alpha = 1, ..., M$ in the chain molecule at three different stages: $M = K + 2$, $M = K + 1$ and $M = K$.

* lmm@unizar.es

[1] E. Noether, Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse **1918**, 235 (1918), URL http://eudml.org/doc/59024.

[2] C. Cohen-Tannoudji, B. Diu, and F. Laloë, *Quantum mechanics; 1st ed.* (Wiley, New York, NY, 1977), trans. of : Mécanique quantique. Paris : Hermann, 1973, URL https://cds.cern.ch/record/101367.

[3] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015), URL https://doi.org/10.1038%2Fnature14539.

[4] G. Cybenko, Mathematics of Control, Signals, and Systems **2**, 303 (1989), URL https://doi.org/10.1007/bf02551274.

[5] K. Hornik, Neural Networks **4**, 251 (1991), URL https://doi.org/10.1016/0893-6080(91)90009-t.

[6] M. Nielsen, *Neural Networks and Deep Learning* (Determination Press, 2015), URL https://books.google.es/books?id=STDBswEACAAJ.

[7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019), URL https://link.aps.org/doi/10.1103/RevModPhys.91.045002.

[8] Z. Liu, V. Madhavan, and M. Tegmark, Phys. Rev. E **106**, 045307 (2022), URL https://link.aps.org/doi/10.1103/PhysRevE.106.045307.

[9] Z. Liu and M. Tegmark, Phys. Rev. Lett. **126**, 180604 (2021), URL https://link.aps.org/doi/10.1103/PhysRevLett.126.180604.

[10] Z. Liu, P. O. Sturm, S. Bharadwaj, S. Silva, and M. Tegmark, *Discovering new interpretable conservation laws as sparse invariants* (2023), 2305.19525.

[11] S. Ha and H. Jeong, Phys. Rev. Res. **3**, L042035 (2021), URL https://link.aps.org/doi/10.1103/PhysRevResearch.3.L042035.

[12] S. J. Wetzel, R. G. Melko, J. Scott, M. Panju, and V. Ganesh, Phys. Rev. Res. **2**, 033499 (2020), URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.033499.

[13] S. Greydanus, M. Dzamba, and J. Yosinski, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), vol. 32, URL https://proceedings.neurips.cc/paper_files/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf.

[14] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, *Lagrangian neural networks* (2020), 2003.04630.

[15] P. Y. Lu, R. Dangovski, and M. Soljačić, *Discovering conservation laws using optimal transport and manifold learning* (2022), 2208.14995.

[16] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, Discovering physical concepts with neural networks, Phys. Rev. Lett. **124**, 010508 (2020)

[17] Z. Liu and M. Tegmark, Phys. Rev. Lett. **128**, 180201 (2022), URL https://link.aps.org/doi/10.1103/PhysRevLett.128.180201.

[18] R. Bondesan and A. Lamacraft, in *ICML 2019 Workshop on Theoretical Physics for Deep Learning* (2019), 1906.04645.

[19] S. Craven, D. Croon, D. Cutting, and R. Houtz, Phys. Rev. D **105**, 096030 (2022), URL https://link.aps.org/doi/10.1103/PhysRevD.105.096030.

[20] R. T. Forestano, K. T. Matchev, K. Matcheva, A. Roman, E. B. Unlu, and S. Verner, Machine Learning: Science and Technology **4**, 025027 (2023), URL https://dx.doi.org/10.1088/2632-2153/acd989.

[21] S. Krippendorf and M. Syvaeri, Machine Learning: Science and Technology **2**, 015010 (2020), URL https://dx.doi.org/10.1088/2632-2153/abbd2d.

[22] J. Yang, R. Walters, N. Dehmamy, and R. Yu, *Generative adversarial symmetry discovery* (2023), 2302.00236.

[23] N. Dehmamy, R. Walters, Y. Liu, D. Wang, and R. Yu, in *Advances in Neural Information Processing Systems*, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (Curran Associates, Inc., 2021), vol. 34, pp. 2503–2515, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/148148d62be67e0916a833931bd32b26-Paper.pdf.

[24] K. Desai, B. Nachman, and J. Thaler, Phys. Rev. D **105**, 096031 (2022), URL https://link.aps.org/doi/10.1103/PhysRevD.105.096031.

[25] G. Barenboim, J. Hirn, and V. Sanz, SciPost Phys. **11**, 014 (2021), URL https://scipost.org/10.21468/SciPostPhys.11.1.014.

[26] A. Decelle, V. Martin-Mayor, and B. Seoane, Phys. Rev. E **100**, 050102(R) (2019), URL https://link.aps.org/doi/10.1103/PhysRevE.100.050102.

[27] C. Zhang, J. Jin, W. Na, Q.-J. Zhang, and M. Yu, IEEE Transactions on Microwave Theory and Techniques **66**, 3781 (2018), URL https://doi.org/10.1109%2Ftmtt.2018.2841889.

[28] G. B. Arfken, H. J. Weber, and F. E. Harris, in *Mathematical Methods for Physicists (Seventh Edition)*, edited by G. B. Arfken, H. J. Weber, and F. E. Harris (Academic Press, Boston, 2013), pp. 815–870, seventh edition ed., ISBN 978-0-12-384654-9, URL https://www.sciencedirect.com/science/article/pii/B9780123846549000177.

[29] R. G. Newton, *Scattering Theory of Waves and Particles* (Springer Berlin Heidelberg, 1982), URL https://doi.org/10.1007/978-3-642-88128-2.

[30] P. Calvo-Barlés, S. G. Rodrigo, E. Sánchez-Burillo, and L. Martín-Moreno (2023), URL https://github.com/pablocalvo7/Symmetry_Seeker_NN.

[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), software available from tensorflow.org, URL https://www.tensorflow.org/.

[32] F. Chollet et al., *Keras*, https://keras.io (2015).

[33] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization* (2017), 1412.6980.