Learning Interacting Theories from Data

Claudia Merger,[1,2,*] Alexandre René,[1,2,3] Kirsten Fischer,[1,2] Peter Bouss,[1,2] Sandra Nestler,[1,2] David Dahmen,[1] Carsten Honerkamp,[2] and Moritz Helias[1,2]

[1]*Institute of Neuroscience and Medicine (INM-6) and Institute for Advanced Simulation (IAS-6) and JARA-Institute Brain Structure-Function Relationships (INM-10), Jülich Research Centre, Jülich, Germany*

[2]*RWTH Aachen University, Aachen, Germany*

[3]*Department of Physics, University of Ottawa, Ottawa, Canada*

(Received 11 April 2023; revised 5 September 2023; accepted 12 October 2023; published 20 November 2023)

One challenge of physics is to explain how collective properties arise from microscopic interactions. Indeed, interactions form the building blocks of almost all physical theories and are described by polynomial terms in the action. The traditional approach is to derive these terms from elementary processes and then use the resulting model to make predictions for the entire system. But what if the underlying processes are unknown? Can we reverse the approach and learn the microscopic action by observing the entire system? We use invertible neural networks to first learn the observed data distribution. By the choice of a suitable nonlinearity for the neuronal activation function, we are then able to compute the action from the weights of the trained model; a diagrammatic language expresses the change of the action from layer to layer. This process uncovers how the network hierarchically constructs interactions via nonlinear transformations of pairwise relations. We test this approach on simulated datasets of interacting theories and on an established image dataset (MNIST). The network consistently reproduces a broad class of unimodal distributions; outside this class, it finds effective theories that approximate the data statistics up to the third cumulant. We explicitly show how network depth and data quantity jointly improve the agreement between the learned and the true model. This work shows how to leverage the power of machine learning to transparently extract microscopic models from data.

DOI: 10.1103/PhysRevX.13.041033

Subject Areas: Complex Systems, Interdisciplinary Physics, Statistical Physics

I. INTRODUCTION

Models of physical systems are frequently described on the microscopic scale in terms of interactions between their degrees of freedom. Often one seeks to understand the collective behavior that arises in the system as a whole. The interactions can feature symmetries, such as spatial or temporal translation invariance. Prominent examples of these theories can be found in statistical physics, high energy physics, and also in neuroscience. The nature of the interactions is often derived as an approximation of a more complex theory.

The description of systems on the microscopic scale is key to their understanding. In the absence of an underlying theory, the inverse problem has to be solved: one needs to infer the microscopic model by measurements of the collective states. This is typically a hard problem. A recent route toward a solution comes from studies [1–8] that explore the link between the learned features of artificial neural networks and the statistics of the data they were trained on. This inspection yields insights both into the mechanisms by which artificial neural networks achieve stellar performance on many tasks and into the nature of the data. In this study, we make the link between learned parameters and data statistics explicit by studying generative neural networks.

Generative models learn the statistics which underlie the data they are trained on. As such they must possess an internal, learned model of data which is encoded in the network parameters. In this work, we gain insights into the nature of the training data by extracting the model from the network parameters, thus bridging the gap between the learned model and its interpretation.

One class of generative models are invertible neural networks (INNs), also called normalizing flows. INNs are invertible mappings trained to approximate the unknown probability distribution of the training set [9,10]. They can

*c.merger@fz-juelich.de

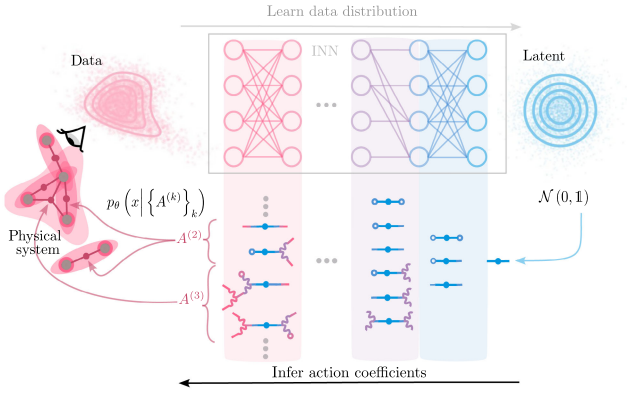


FIG. 1. Learning actions from data. We observe a physical system of interacting degrees of freedom (gray dots), whose precise interactions are unknown (shaded areas). We train a neural network on measurements of the system. The network learns in an unsupervised fashion an estimate of the distribution of training data. We extract the action from the network parameters layer by layer, using a diagrammatic language. The final action coefficients $A^{(k)}$ represent the learned interactions (pink nodes).

be used to generate new samples from the same distribution as the training set, or to manipulate existing data consistent with the features of the training set (for example, transitions between images [11–13]). This is achieved by mapping the highly structured input data to a completely unstructured latent space. The model learned by the network is expressed through the inverse mapping, as this must generate all interactions in the data. However, the network mapping is typically high dimensional and depends on many parameters, which does not allow for a direct interpretation.

In this work, we derive interpretable microscopic theories from trained INNs. We extract an explicit data distribution, formulated in terms of interactions, from the trained network parameters. These interactions form the building blocks of the microscopic theory that describes the distribution of the training data. Furthermore, the process of extracting the microscopic theory makes the relation between the trained network parameters and the learned theory explicit. We show how interactions are hierarchically built through the composition of the network layers. This approach provides an interpretable relation between the network parameters and the learned model.

We illustrate and test this framework on several examples where the underlying theory is exactly known. We find that the networks are able to learn nontrivial interacting theories. Furthermore, we show that theories with higher-order interactions emerge as the network depth increases. Thus, we show how to leverage the power of machine learning to extract interacting models from data.

Solving inverse problems is a well-known challenge, to which many approaches have been developed; these are usually specific to a particular model or use discrete variables, or both. We discuss these approaches in Sec. V. In contrast, our approach considers continuous variables, is not restricted to pairwise interactions, and does not require prior knowledge of the interaction structure. Further, the optimization of the couplings does not require the calculation of correlation functions of the learned theory for optimization but is done implicitly via the mapping encoded by the network.

This paper is structured as follows: in Sec. II we introduce the action as the central object of a theory. We then describe how to extract the action from a trained INN in Sec. III. Subsequently, we test this framework in several settings in Sec. IV. Finally, we summarize and discuss the main findings, compare our work to different previously proposed inference schemes, and provide an outlook on how to extend the framework in Sec. V.

II. ACTIONS IN PHYSICS

Physical theories are often formulated in terms of microscopic interactions of their many constituents, the degrees of freedom $\{x_i\}_{1\le i\le d}$, where d is the number of constituents and x describes the system's state. The degrees of freedom $\{x_i\}_{1\le i\le d}$ can be Ising spins, firing rates of neurons, social agents, field points, or pixels of an image. The interactions of the constituents provide a mechanistic understanding of the system. For example, an interaction between adjacent pixels of an image can induce these pixels to be similar to each other, to create a color patch in the image, or to induce them to be different, and thereby create an edge. The energy or Hamiltonian H of the system can be written as the sum of these interactions. One can then ask how probable it is to observe the system in a specific microscopic state x given an average energy $\langle H\rangle$. The most unbiased (maximum entropy) estimate of the probability density is then

$$p_X(x) = \frac{1}{Z} e^{-\beta H(x)},$$

with Z the normalization factor or partition function. Here p_X is also known as the Boltzmann distribution [14]. In statistical physics, the prefactor β is identified as the inverse temperature.

The action S_X of this system is defined as the log probability; hence, $S_X(x) = \ln p_X(x) = -\beta H(x) - \ln Z$. Therefore, the system is fully characterized by S_X and measurements of the system state x correspond to drawing samples from p_X. Furthermore, up to the constant prefactor $-\beta$ and the constant $\ln Z$, the terms in the action are the same interaction terms as those in the Hamiltonian.

Consider an action of the form

$$S_X(x) = A^{(0)} + \sum_{i=1}^{d} A_i^{(1)} x_i + \sum_{i,j=1}^{d} A_{ij}^{(2)} x_i x_j + \sum_{i,j,k=1}^{d} A_{ijk}^{(3)} x_i x_j x_k + \cdots, \tag{1}$$

where the coefficients $A^{(k)}$ are tensors of rank k and dimension d. Without loss of generality we choose the $A^{(k)}$ to be symmetric tensors, $A^{(k)}_{i_1\ldots i_k} = A^{(k)}_{\mathcal{P}(i_1,\ldots,i_k)}$ for any permutation $\mathcal{P}$ of the indices (for example, a general matrix $A^{(2)}$ and its symmetrized equivalent $\bar{A}^{(2)} = (A^{(2)} + (A^{(2)})^{\mathrm{T}}/2)$ lead to the same quadratic polynomial: $x^{\mathrm{T}}A^{(2)}x = x^{\mathrm{T}}\bar{A}^{(2)}x$ for all $x \in \mathbb{R}^d$). These coefficients encode the coordination between the different degrees of freedom x_i. In general, we refer to a term of type $A^{(k)}_{i_1,\ldots,i_k}x_{i_1}\cdots x_{i_k}$, as a k-point interaction, since this term describes a coaction of k degrees of freedom for $i_1,\ldots,i_k \in \{1,\ldots,d\}$ all unequal. In this work, we focus exclusively on actions of the form of Eq. (1), which are suitable only for describing classical fields x_i, as the fields and the action coefficients are tensors and scalars, not operators. However, we are not limited to equilibrium statistical mechanics: the samples could as well stem from a time-dependent process; in this case, the action describes the measure on a path, which is allowed to come from a nonequilibrium system. Furthermore, even for quantum systems where the interactions are exactly known, a renormalized classical theory is sometimes sought to effectively describe the influence of quantum fluctuations [15,16].

Notation. In the following, we use the notation $u^{\otimes k}$ for the outer product of k instances of a tensor u,

$$u^{\otimes k} = \underbrace{u \otimes u \otimes \cdots \otimes u}_{k \text{ times}},$$

and $T^{(k)} \cdot (u)^{\otimes l}$ for $l \leq k$ to denote the contraction of the first l indices of a rank k tensor $A^{(k)}$ with the first index of each tensor u:

$$\left(A^{(k)} \cdot (u)^{\otimes l}\right)_{\beta_1,\ldots\beta_l,i_{l+1},\ldots,i_k} = \sum_{i_1,\ldots,i_l} A^{(k)}_{i_1,\ldots,i_k} u_{i_1\beta_1}\cdots u_{i_l\beta_l},$$

with multi-indices $\beta_1,\ldots,\beta_l$ whose rank depends on the rank of u. In the special case that u is a vector, the indices $\beta_1,\ldots,\beta_l$ vanish from the expression. If additionally $k = l$, the result is a scalar. Hence, Eq. (1) becomes

$$S_X(x) = A^{(0)} + A^{(1)} \cdot x + A^{(2)} \cdot (x)^{\otimes 2} + A^{(3)} \cdot (x)^{\otimes 3} + \cdots.$$

We symmetrize a tensor by averaging over the set $\mathcal{P}(\alpha)$ of all permutations of the multi-index α:

$$(\mathrm{sym}\, A^{(k)})_\alpha = |\mathcal{P}(\alpha)|^{-1} \sum_{\pi \in \mathcal{P}(\alpha)} A^{(k)}_\pi;$$

this operation does not change the result of polynomial contractions: $A^{(k)} \cdot (x_l)^{\otimes k} \equiv (\mathrm{sym}A^{(k)}) \cdot (x_l)^{\otimes k}$. Thus the choice of symmetric tensors does not restrict the expressivity of S_X.

A typical objective in statistical physics is understanding how the microscopic interactions $A^{(k)}$ determine the macroscopic properties of the system. In this work, we take a different approach: Given samples from a system with unknown microscopic properties, we extract the interactions. Generative models such as INNs are a powerful tool to approximate data distributions p_X [9,10,13,17]. In the next section, we demonstrate how to extract the interaction coefficients $A^{(k)}$ from trained networks.

III. LEARNING ACTIONS WITH INVERTIBLE NEURAL NETWORKS

In this section, we show how to extract an action of the form Eq. (1) by using a special class of generative neural networks, namely invertible neural networks. These neural networks learn a data distribution in an unsupervised manner. Each sample from the dataset is hence viewed as drawn independent identically distributed (IID) from an unknown distribution and each layer of the network is a mapping between different representations of the same random variables. Intuitively, the network bends and stretches the space in which the random variables lie until their distribution in the new space becomes particularly simple, e.g., Gaussian. We will describe this process via a series of action coefficients that characterize the distributions after each network layer. Thus the transformation of the space is encoded in the transformation of the action coefficients. This procedure is illustrated in Fig. 1. We first describe the training objective and architecture of the network, before we move on to the coefficient transforms.

Invertible neural networks are used to learn the data distribution p_X from a dataset $\mathcal{D}$ of training samples [9]. They implement a bijective mapping $f_\theta\colon \mathbb{R}^d \to \mathbb{R}^d$, where the network output $z = f_\theta(x)$ is often referred to as the latent variable. The parameters θ of the network are trained such that the latent variables follow a prescribed target distribution. We follow a common choice for this latent distribution as a set of d uncorrelated centered Gaussian variables with unit variance [9],

$$p_Z(z) = \exp\left(-\frac{1}{2}z^{\mathrm{T}}z - \frac{d}{2}\ln 2\pi\right), \tag{2}$$

where $z^{\mathrm{T}}z = \sum_{i=1}^d z_i^2$ denotes the Euclidean scalar product. Given p_Z, the probability assigned to a specific input is given by the change of variables formula,

$$p_\theta(x) = p_Z[f_\theta(x)]|\det J_{f_\theta}(x)|, \tag{3}$$

which depends only on the network mapping f_θ and its Jacobian J_{f_θ}. The training objective is to minimize the negative log-likelihood,

$$\mathcal{L}(\theta) = -\sum_{x\in\mathcal{D}} \ln p_\theta(x) = -\sum_{x\in\mathcal{D}} S_\theta(x), \tag{4}$$

where we used that $\ln p_\theta$ is precisely the action S_θ of the learned distribution.

In this manner, we optimize Eq. (3) to approximate the unknown underlying data distribution using stochastic gradient descent. Since the target distribution, Eq. (2), is a set of independent Gaussians, the mapping $f_\theta\colon x \mapsto z$ of the network aims to eliminate cross-correlations and higher-order dependencies of the components of the latent. On the level of the action, this means that all k-point interactions with $k \geq 3$ between components of z must vanish. In turn, the inverse mapping defines a generative process that induces interactions in the learned distribution from a noninteracting latent theory.

We now define the architecture that allows us to obtain a polynomial action S_θ from the network parameters θ. The network is composed of multiple layers l; every layer mapping $f_{l,\theta}\colon \mathbb{R}^d \to \mathbb{R}^d$ is an invertible function. For each layer, we define an output action $S_{X,l+1}$, which transforms into the input action $S_{X,l}$ via the change of variables formula. Using Eq. (3) we compute the input action $S_{X,l}$ of each layer given the output action $S_{X,l+1}$,

$$S_{X,l}(x_l) = S_{X,l+1}[f_l(x_l)] + \ln|\det J_{f_l}(x_l)|, \tag{5}$$

starting with the polynomial action of the latent variable $S_Z(y) = \ln p_Z(y)$. We construct all layer mappings f_l such that a polynomial action $S_{X,l+1}$ generates a polynomial action $S_{X,l}$ in Eq. (5) and thus by induction we obtain a polynomial learned input action S_θ.

Each layer mapping f_l is composed of a linear mapping L_l and a nonlinear mapping ϕ_l:

$$f_l(x_l) = \phi_l \circ L_l(x_l). \tag{6}$$

In the overall architecture, linear and nonlinear mappings are therefore stacked alternately (see Fig. 2). After the last nonlinear transform ϕ_L, we add an additional linear transform L_{L+1}, such that the network architecture begins and ends with a linear transform.

The transform of the action via a single layer, Eq. (5), similarly decomposes into two steps, with $h_l = L_l(x_l)$ the intermediate activation:

$$S_{H,l}(h_l) = S_{X,l+1}[\phi_l(h_l)] + \ln|\det J_{\phi_l}(h_l)|, \tag{7}$$

$$S_{X,l}(x_l) = S_{H,l}[L_l(x_l)] + \ln|\det J_{L_l}(x_l)|. \tag{8}$$

The transformations from $S_{X,l+1}$ to $S_{H,l}$, and from $S_{H,l}$ to $S_{X,l}$, are therefore determined by ϕ_l and L_l, respectively, which we express schematically as

$$S_{X,l+1} \xrightarrow{\phi_l} S_{H,l} \xrightarrow{L_l} S_{X,l}. \tag{9}$$

The remainder of this section is concerned with expressing Eq. (9) in terms of transforms of the action coefficients.

Since the actions are polynomials, at each step we can write $S_{X,l}, S_{H,l}$ in terms of coefficients $\{A_l^{(k)}\}_k, \{B_l^{(k)}\}_k$:

$$S_{X,l}(x_l) = \sum_{k=0}^{K_l} A_l^{(k)} \cdot (x_l)^{\otimes k}, \tag{10}$$

$$S_{H,l}(h_l) = \sum_{k=0}^{K_l} B_l^{(k)} \cdot (h_l)^{\otimes k}, \tag{11}$$

where the sum runs over the ranks k of the coefficient tensors. Here K_l is the degree of the polynomial in h_l, which depends on the layer index l. We show in the following that the rank of the polynomial does not change between $S_{X,l}$ and $S_{H,l}$ in the latter step of Eq. (9). The coefficients further uniquely determine the action. Therefore, Eq. (9) is equivalent to the coefficient mapping:

$$\{A_{l+1}^{(k)}\}_k \xrightarrow{\phi_l} \{B_l^{(k)}\}_k \xrightarrow{L_l} \{A_l^{(k)}\}_k. \tag{12}$$

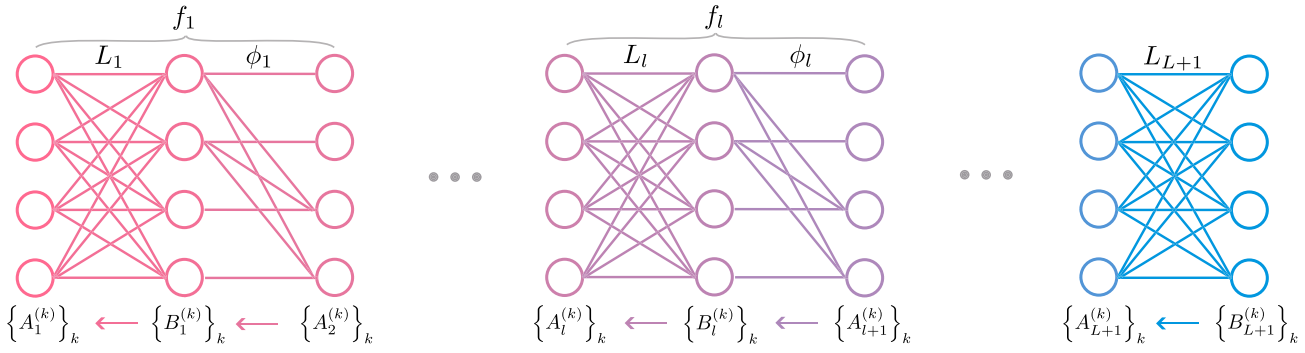


FIG. 2. Architecture and coefficient order. Invertible network composed of multiple layers. $L_1, \ldots, L_{L+1}$ denote linear fully connected layers, $\phi_1, \ldots, \phi_L$ are quadratic nonlinear activation functions. Coefficients $B_l^{(k)}$ of the action $S_{H,l}$ of preactivations h_l in layer l of order k; coefficients $A_l^{(k)}$ of the action $S_{X,l}$ prior to linear layer l of order k.

In Fig. 2 we illustrate the order in which the coefficients are transformed.

We now derive the recursive equations for the coefficient transforms, beginning with the linear mapping.

Linear mapping. The linear mapping is given by

$$h_l = L_l(x_l) = W_l x_l + b_l, \tag{13}$$

with $W_l, b_l \in \theta$. Combining Eqs. (8) and (11) yields

$$S_{X,l}(x_l) = \sum_k B_l^{(k)} \cdot (W_l x_l + b_l)^{\otimes k} + \ln|\det W_l|. \tag{14}$$

We find the transformed coefficients $A_l^{(k)}$ by expanding Eq. (14) and ordering them by powers in x_l. Higher-rank coefficients of $S_{H,l}$ contribute to lower-rank coefficients of $S_{X,l}$ via the contraction with the bias b_l. All remaining indices of $B_l^{(k)}$ must then be contracted with the first index of W_l:

$$A_l^{(k)} = \left[\sum_{k'=0}^{K_l-k} \binom{k+k'}{k'} B_l^{(k+k')} \cdot (b_l)^{\otimes k'}\right] \cdot (W_l)^{\otimes k} + \delta_{k0} \ln|\det W_l|. \tag{15}$$

The combinatorial factor $\binom{k+k'}{k'}$ arises due to the symmetry of the coefficients $B^{(k)}$: since they are symmetric under permutations of the indices, we only need to fix the number of contractions k' with the bias term b_l and count all $\binom{k+k'}{k'}$ possible combinations of k' indices in $B^{(k)}$. Since L_l is linear, $S_{X,l}$ and $S_{H,l}$ both have the same rank K_l.

We illustrate Eq. (15) for the final linear mapping of the network, the first coefficient transform that starts on the known latent space coefficients $\{B_{L+1}^{(k)}\}_k$ (on the far right in Fig. 2). The action of the latent distribution given in Eq. (2) describes a centered Gaussian with unit covariance; its first three coefficients are therefore $B_{L+1}^{(0)} = -(d/2)\ln 2\pi$, $B_{L+1}^{(1)} = 0$, and $B_{L+1}^{(2)} = -\frac{1}{2}\mathbb{1}$, and all coefficients with rank $k \geq 3$ being zero.

The transformed zeroth-rank coefficient $A_{L+1}^{(0)}$, which ensures that the action stays normalized, reads

$$\begin{aligned} A_{L+1}^{(0)} &= -\frac{d}{2}\ln 2\pi + B_{L+1}^{(2)} \cdot (b_{L+1})^{\otimes k} + \ln|\det W_{L+1}| \\ &= -\frac{d}{2}\ln 2\pi - \frac{|b_{L+1}|^2}{2} + \ln|\det W_{L+1}|. \end{aligned} \tag{16}$$

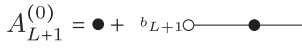
The bias in the linear mapping shifts the mean of the probability distribution, which is induced by the first-order coefficient:

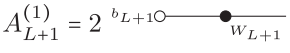
$$\begin{aligned} A_{L+1}^{(1)} &= \left[\binom{2}{1} B_{L+1}^{(2)} \cdot (b_{L+1})^{\otimes 1}\right] \cdot (W_{L+1})^{\otimes 1} \\ &= -W_{L+1}^{\mathrm{T}} b_{L+1}. \end{aligned} \tag{17}$$

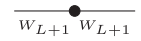
The second-rank coefficient is transformed in accordance with the rotation and scaling of the space due to W_{L+1}:

$$A_{L+1}^{(2)} = -B_{L+1}^{(2)} \cdot (W_{L+1})^{\otimes 2} = -\frac{1}{2} W_{L+1}^{\mathrm{T}} W_{L+1}. \tag{18}$$

In the case of actions with higher-order interactions $k \geq 3$, the coefficient transform Eq. (15) requires the computation of many more terms. To simplify the coefficient transforms, we therefore employ a diagrammatic language. A common practice in statistical physics is to represent kth-order interactions as vertices with k legs [18]. Accordingly, we here express tensors of rank k as vertices with k legs and the contraction between tensors by an attachment of legs. This facilitates the computation of combinatorial factors, which can be read off from the diagram topology. The diagrammatic representations of Eqs. (16), (17), and (18) read:

$A_{L+1}^{(0)} =$  $+ \ln|\det W_{L+1}|$, (19)

$A_{L+1}^{(1)} = 2$  , (20)

$A_{L+1}^{(2)} =$  . (21)

A complete presentation of the diagrammatic method is provided in Appendix A.

Nonlinear mapping. We follow Dinh *et al.* [9] to define an invertible nonlinear mapping: we split the activation vectors and activation functions into two halves (if the dimension d is uneven, we take the first $\lceil d/2 \rceil$ entries of h_l to be in h_l^1), denoting them by $h_l = \binom{h_l^1}{h_l^2}$ and $\phi_l = \binom{\phi_l^1}{\phi_l^2}$. The first half is passed onto the next layer unchanged; we then add a nonlinear function $\tilde{\phi}(h_l^1)$ of the first half onto the second half:

$$x_{l+1} = \phi_l(h_l) = \begin{pmatrix} \phi_l^1(h_l) \\ \phi_l^2(h_l) \end{pmatrix} = \begin{pmatrix} h_l^1 \\ h_l^2 \end{pmatrix} + \begin{pmatrix} 0 \\ \tilde{\phi}_l(h_l^1) \end{pmatrix}. \tag{22}$$

We choose a quadratic nonlinearity,

$$\tilde{\phi}_l(h_l^1) = \tilde{\chi}_l \cdot (h_l^1)^{\otimes 2}, \tag{23}$$

where $\tilde{\chi}_l \in \mathbb{R}^{\lfloor d/2 \rfloor \times \lceil d/2 \rceil \times \lceil d/2 \rceil}$ is a third-order tensor whose coefficients are trained, $\tilde{\chi}_l \in \theta$. In the following, we will use a shorthand notation $\phi_l(h_l) = h_l + \chi_l \cdot (h_l)^{\otimes 2}$ with $\tilde{\chi}_l$ being the nonzero part of χ_l.

Equation (23) is the most elementary nonlinearity which is compatible with a polynomial action. Through the composition of L layer transforms f_l, the network mapping becomes a polynomial of order 2^{L+1}. The advantage of decomposing such a transform in terms of multiple applications of Eqs. (13) and (22) is a particularly simple form of the update equations for the coefficients.

Splitting the nonlinear mapping (22) makes it trivially invertible,

$$h_l = \phi_l^{-1}(x_{l+1}) = \begin{pmatrix} x_{l+1}^1 \\ x_{l+1}^2 \end{pmatrix} - \begin{pmatrix} 0 \\ \tilde{\phi}_l(x_{l+1}^1) \end{pmatrix}, \quad (24)$$

as one can observe by evaluating the composition of Eqs. (22) and (24) on an arbitrary vector h_l.

We compute the action $S_{H,l}$ from $S_{X,l+1}$ using Eq. (8). Since the Jacobian $J_{\phi,l}$ of ϕ_l is a triangular matrix with ones on the diagonal, we have $\ln|\det J_{\phi,l}| = 0$. Therefore, the transform of the action induced by ϕ_l is just the composition

$$\begin{aligned} S_{H,l}(h_l) &= S_{X,l+1}\left(h_l + \chi_l \cdot (h_l)^{\otimes 2}\right) \\ &= \sum_k A_{l+1}^{(k)} \cdot \left(h_l + \chi_l \cdot (h_l)^{\otimes 2}\right)^{\otimes k}. \end{aligned} \quad (25)$$

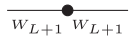
Equation (25) yields a polynomial of order $K_l = 2K_{l+1}$. We expand the products in Eq. (25) and reorder the terms to obtain the transform of the action coefficients. Since each factor of χ_l increases the rank of the resulting tensor by one, lower-order coefficients $A_{l+1}^{(k-k')}$ contribute to the coefficient $B_l^{(k)}$ via

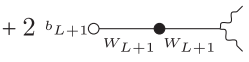
$$\binom{k-k'}{k'} A_{l+1}^{(k-k')} \cdot (\chi_l)^{\otimes k'},$$

with $k > k' \geq 1$. Each contraction of $A_{l+1}^{(k-k')}$ with χ_l consumes one index in $A_{l+1}^{(k-k')}$ and the first index in χ_l, but adds two indices to the resulting tensor. As a result, for each χ_l in the contraction, the rank is raised by one. Therefore, k' factors of χ_l are needed to increase the rank from $k - k'$ to k. The factor $\binom{k-k'}{k'}$ arises because there are $\binom{k-k'}{k'}$ ways of choosing k' of the $k - k'$ indices of the tensor to which to contract the factors of χ_l. However, contractions of this type are no longer symmetric tensors because the resulting kth-order tensor has $2k'$ indices stemming from χ_l and the remaining ones from $A^{(l+1,k-k')}$. To express this result as a symmetric tensor, we symmetrize the result. This yields

$$\begin{aligned} B_l^{(k\leq 1)} &= A_{l+1}^{(k)}, \\ B_l^{(k>1)} &= \mathrm{sym} \sum_{k'=0}^{k} \binom{k-k'}{k'} A_{l+1}^{(k-k')} \cdot (\chi_l)^{\otimes k'}. \end{aligned} \quad (26)$$

Diagrammatically, the contraction with χ_l is represented by splitting the legs of a vertex. By counting the number of splits we can therefore infer the number of factors χ_l of any diagram. To illustrate this, we here show the mapping $\{A_{L+1}^{(k)}\}_k \xrightarrow{\phi_L} \{B_L^{(k)}\}_k$, which generates interactions up to the fourth order. The zeroth- and first-rank coefficients remain unchanged, $B_L^{(0)} = A_{L+1}^{(0)}$ and $B_L^{(1)} = A_{L+1}^{(1)}$, as the $A_{L+1}^{(0)}$ has no legs to split; the splitting of legs in $A_{L+1}^{(1)}$ gives a contribution to $B_L^{(2)}$:

$B_L^{(2)} =$ 

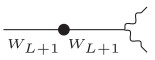
$+ 2$ 

(27)

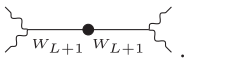
This diagrammatic expression corresponds to

$$\begin{aligned} B_L^{(2)} &= B_{L+1}^{(2)} \cdot (W_{L+1})^{\otimes 2} \\ &\quad + \mathrm{sym}\left(\left[B_{L+1}^{(2)} \cdot b_{L+1}\right] \cdot W_{L+1}\right) \cdot \chi_L. \end{aligned}$$

No further diagrams are generated from $A_{L+1}^{(1)}$, as all legs are split. The higher-order interactions $B_L^{(3)}, B_L^{(4)}$ emerge through the splitting of legs in $A_{L+1}^{(2)}$:

$B_L^{(3)} = \binom{2}{1}$ 

, (28)

$B_L^{(4)} =$ 

. (29)

In tensor notation, the same expressions read

$$\begin{aligned} B_L^{(3)} &= \binom{2}{1} \mathrm{sym}\left[B_{L+1}^{(3)} \cdot (W_{L+1})^{\otimes 2}\right] \cdot \chi_L, \\ B_L^{(4)} &= \mathrm{sym}\left[B_{L+1}^{(3)} \cdot (W_{L+1})^{\otimes 2}\right] \cdot (\chi_L)^{\otimes 2}. \end{aligned}$$

This exemplifies how the interactions are built hierarchically as further layer transforms contribute contractions with $W_{l\leq L}$, $b_{l\leq L}$, and $\chi_{l<L}$, to previous coefficients. See Appendix A for further details.

The degree K_l of the action doubles with each layer, starting from the output action with degree 2. Networks of depth L thus generate actions of degree 2^{L+1}. Through the composition of several nonlinear mappings like Eq. (22),

the prefactors χ_l of later layers will be exponentiated alongside their activations. Increasing the number of layers will therefore generate terms of arbitrarily high degree in both x and in the prefactors. For example, the contribution of χ_1 to x_L will be of order $(\chi_1)^{2^{L-1}}$. As a result, large values in χ_l are unfavorable as they make the activations diverge. In practice, we find that the entries of the tensors χ_l of trained networks are typically small, $|(\chi_l)_{ijk}| \ll 1$.

We note that all coefficients with rank $k > 2$ must contain at least $k-2$ factors of χ_l, where the different factors in general originate from different layers. These terms of rank $k > 2$ constitute the non-Gaussian part of the action. Therefore, for applications where the data can be described as a perturbed Gaussian, small entries in χ_l are sufficient. The higher the rank of the coefficient, the smaller its entries. Consequently, we place a cutoff of two on the number of factors of χ_l in the action coefficients, thus ignoring negligible contributions. This effectively imposes a maximum rank of $k = 4$ onto the action coefficients.

Coefficients of high rank can be numerically intractable for large dimension d, as their size grows as $\mathcal{O}(d^k)$. To mitigate this, we make use of Eq. (23) to write the coefficients in a decomposed form, which speeds up the computations and allows for tractable reductions in the size of the stored tensors. We specify this decomposition in Appendix B.

We began this section by equating the transform of the action through the network to the transform of its coefficients, decomposed as alternating linear and nonlinear transforms. We then make these transforms explicit in Eqs. (15) and (26). Given a trained network, these expressions allow us to extract the learned action through the iterative application of the coefficient transforms from the last layer to the first. In this way, we can describe the data distribution constructively, by tracking how the latent distribution is transformed through successive network layers.

Equation (26) shows how higher-rank coefficients hierarchically emerge through repeated contractions with the parameters $\chi_L, \ldots, \chi_1$ of the nonlinear mappings and $W_L, \ldots, W_1, b_L, \ldots, b_1$ of the linear mappings of different layers. In the data space, these coefficients correspond to interactions; therefore this approach establishes an explicit relation between network parameters θ and the characteristic properties of the learned distribution p_θ. In the next section, we test this method in several cases with known ground-truth distributions.

IV. EXPERIMENTS

In this section, we will test the learning of actions in four different settings. In Sec. IV A, we use a randomly initialized teacher network to generate samples. The teacher network has the same architecture as the one described in Sec. III, enabling us to compute the ground-truth action. In Sec. IV B, the ground-truth action coefficients themselves are generated randomly, leading to multimodal data distributions. In Sec. IV C, we move to a physics-inspired model system with interactions on a square lattice of $d = 10^2$ sites. Finally, in Sec. IV D, we apply our method to the MNIST dataset, a standard benchmark dataset in image classification.

A. In-class distributions

First, we test whether we can recover the action coefficients of a known action from samples drawn from the respective distribution. We initialize a teacher network with random weights and compute the corresponding action coefficients $\{T^{(k)}\}_{k\leq 4}$ with the method outlined in Sec. III. We then generate a training set $\mathcal{D}$ by sampling Gaussian random variables $z \sim p_Z$ and apply the inverse network transform of the teacher on them; the elements of $\mathcal{D}$ are therefore samples from the teacher distribution. Since the teacher distribution is by construction part of the set of learnable student distributions, we refer to this as an in-class distribution. We then initialize a student network to identity $W_l = \mathbb{1}$, $b_l = 0$, $\chi_l = 0 \ \forall\, l$ and train it on the training set $\mathcal{D}$. This choice for the initialization ensures that all variability in the trained result is due to the training data and the sequence of random batches drawn during training. After training, we extract the student coefficients $\{A^{(k)}\}_{k\leq 4}$ as described in Sec. III. The student network has learned the teacher distribution if the associated action coefficients match, $T^{(k)} = A^{(k)} \ \forall\, k$.

Note that $T^{(k)} = A^{(k)} \ \forall\, k$ does not imply that the parameters θ of the teacher and student network are equal. Because of the rotational invariance of the latent space, an additional linear transform which rotates the latent space z does not result in a change in the learned action. Hence, $T^{(k)} = A^{(k)} \ \forall\, k$ implies only that the two networks learn the same statistics.

We compare teacher and student coefficients in Fig. 3 for two different training set sizes $|\mathcal{D}|$. For a sufficiently large dataset, the student learns the teacher coefficients arbitrarily well: In Figs. 3(a)–3(d), the coefficient entries coincide while the network parameters θ do not align (see Appendix C for a comparison between network parameters). This confirms that the extracted coefficients are indeed characteristic of what the network has learned, as opposed to the parameters. Given sufficient samples, we therefore find that the method recovers the correct action coefficients.

For a smaller training set with $|\mathcal{D}| = 10^3$, we find that the student network overfits the training set. To see this, we compute the test loss $\mathcal{L}_{\text{test}}(\theta)$ on a test set of 10^4 samples. In Fig. 3(e) the test loss is significantly larger than the training loss. This is reflected in deviating coefficient entries in

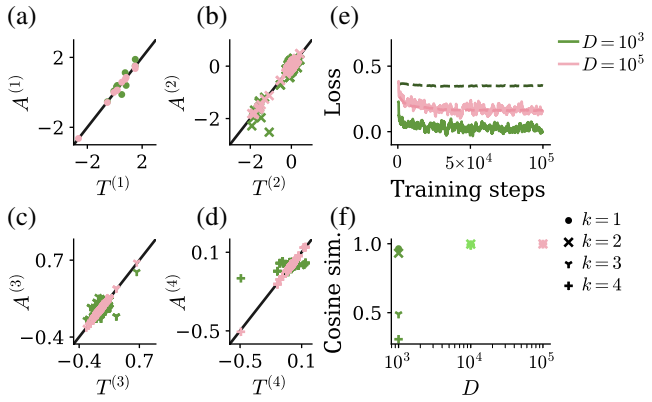


FIG. 3. Teacher-student coefficient comparison for varying training set sizes $D = |\mathcal{D}|$. Both teacher and student have depth $L = 1$. (a)–(d) Student coefficients $A^{(k)}$ over teacher coefficients $T^{(k)}$ up to fourth order for $D = 10^3$ in green and $D = 10^5$ in pink. (e) Training loss (full lines) and test loss (dashed lines) over training steps. (f) Cosine similarity of coefficients over number of training samples.

Figs. 3(a)–3(d). To quantify this disparity, we compute the cosine similarity between the tensors,

$$\cos\angle(T^{(k)}, S^{(k)}) = \frac{|\sum_\alpha T_\alpha^{(k)} S_\alpha^{(k)}|}{\sqrt{\sum_\alpha (S_\alpha^{(k)})^2 \sum_\alpha (T_\alpha^{(k)})^2}}, \quad (30)$$

where the sum runs over all independent indices α, excluding duplicate tensor entries which are equal due to the symmetry. The cosine similarity ranges between zero and one; for perfect alignment it is equal to one. Figure 3(f) shows how the cosine similarity between teacher and student coefficients increases with the training set size. The lower-order coefficients $T^{(k\leq 2)}$ are approximated well even in the case of little training data. The learned higher-order coefficients clearly deviate from the teacher coefficients in this case [see Figs. 3(c)–3(f)], indicating that the higher-order coefficients, corresponding to higher-order interactions in the teacher distribution, can only be conveyed through larger datasets.

Learning rules in coefficient space. Higher-order interactions depend on higher-order statistics of the data distribution, which must be expressed through a limited amount of samples. We train the network using stochastic gradient descent (SGD), which updates all parameters of the network at training time t according to the dependence of the loss $\mathcal{L}$ on a subset of training data $\mathcal{D}_t \subset \mathcal{D}$. In SGD, the update of a single weight $\Delta\theta_i = \theta_i(t+1) - \theta_i(t)$ is

$$\Delta\theta_i = -\eta \frac{\partial}{\partial\theta_i} \mathcal{L}_{\mathcal{D}_t} = \eta \frac{\partial}{\partial\theta_i} \langle S_\theta(x)\rangle_{\mathcal{D}_t}, \quad (31)$$

where η is the learning rate and $\langle\cdot\rangle_{\mathcal{D}_t}$ denotes the average over the current training batch $\mathcal{D}_t$. In Appendix D, we show that this leads to a noisy update in the coefficients $A^{(k)}$ for $k \geq 1$ of

$$\begin{aligned}\Delta A_\alpha^{(k)} = {} & \eta\left(\xi_t^{(k)} + \langle (x^{\otimes k})_\alpha\rangle_{\mathcal{D}} - \langle (x^{\otimes k})_\alpha\rangle_A\right)\sum_i \left(\frac{\partial A_\alpha^{(k)}}{\partial\theta_i}\right)^2 \\ & + \eta \sum_{l,\alpha_l\neq\alpha} \left(\xi_{\alpha_l,t}^{(l)} + \langle (x^{\otimes l})_{\alpha_l}\rangle_{\mathcal{D}} - \langle (x^{\otimes l})_{\alpha_l}\rangle_A\right) \\ & \times \sum_i \frac{\partial A_\alpha^{(k)}}{\partial\theta_i}\frac{\partial A_{\alpha_l}^{(l)}}{\partial\theta_i} + \mathcal{O}(\Delta\theta^2), \end{aligned} \quad (32)$$

where $\langle\cdot\rangle_{\mathcal{D}}$ is the empirical average over all samples in the full training set $\mathcal{D}$, and $\langle\cdot\rangle_A$ is the expectation with regard to the current estimate of the density depending on learned coefficients $\{A^{(k)}\}_k$. The random variable $\xi_t^{(k)}$ encodes the difference between the mean estimated on the whole training set and a training batch. One can show that on average over all batches, the noise vanishes, $\langle\xi^{(k)}\rangle = 0$, and the variance also decreases with the training set size:

$$\langle\langle \xi^{(k)} \rangle\rangle = (\langle x^{\otimes 2k}\rangle_{\mathcal{D}} - (\langle x^{\otimes k}\rangle_{\mathcal{D}})^{\otimes 2})|\mathcal{D}_t|^{-1}$$

(see, e.g., Ref. [19]). Smaller batch sizes $|\mathcal{D}_t|$ therefore increase the noise in the updates of the action coefficients.

The expected update $\langle\Delta A_\theta^{(k)}\rangle$ vanishes on average over all batches when the learned moments and the moments on the training set match: $\langle x^{\otimes k}\rangle_{\mathcal{D}} = \langle x^{\otimes k}\rangle_A$. However, for any finite training set, there will furthermore be a deviation between $\langle x^{\otimes k}\rangle_{\mathcal{D}}$ and the true moment $\langle x^{\otimes k}\rangle_T$ of the teacher network. This expected difference scales as

$$\langle x^{\otimes k}\rangle_{\mathcal{D}} - \langle x^{\otimes k}\rangle_T \propto \sqrt{[\langle x^{\otimes 2k}\rangle_T - (\langle x^{\otimes k}\rangle_T)^{\otimes 2}]|\mathcal{D}|^{-1}}.$$

Therefore, there is a batch-size-dependent variability in the coefficient updates as well as a bias induced by the limited amount of training data. This drift introduces a bias in the training, leading to overfitting. For many distributions, both $\langle x^{\otimes k}\rangle_{\mathcal{D}} - \langle x^{\otimes k}\rangle_T$ and $\langle\langle \xi^{(k)}\rangle\rangle$ increase with k [20,21]. In this case, both the bias and the variability of the training procedure increase with k, explaining why it is harder to learn higher-order statistics.

B. Out-of-class distributions

In the previous section, we demonstrated that invertible networks accurately learn any distribution generated by the image set of inverse mappings f_θ^{-1}. However, it is interesting to investigate how our approach deals with distributions outside this set. A first step to this end lies in understanding the nature of mappings employed by the invertible network.

The proposed network architecture belongs to the class of volume-preserving networks [9,10]: the additive nature of the nonlinearity in Eq. (22) leads to a constant Jacobian determinant $\det J_{f_\theta}(x)$. The Jacobian determinant

$\det J_{f_\theta}(x)$ of a mapping states how the image of the mapping is locally stretched. Since the only contribution to $|\det J_{f_\theta}(x)|$ comes from the linear transform, Eq. (13), the Jacobian determinant is constant in x. Therefore, this stretch is homogeneous everywhere. As a result, an invertible network with $|\det J_{f_\theta}(x)| = \text{const}$ and a Gaussian target distribution p_Z can only learn unimodal distributions p_θ, i.e., distributions with only one maximum. To see this, we compute the gradient of Eq. (3) with respect to x:

$$\nabla_x p_\theta(x) = 0 \overset{|\det J_{f_\theta}(x)|=\text{const}>0}{\iff} \nabla_{f_\theta} p_Z[f_\theta(x)] = 0.$$

This shows that the learned input distribution $p_\theta(x)$ has an extremum at x_0 if and only if the target distribution has an extremum at $f_\theta(x_0)$. Since p_Z has a single extremum, so does p_θ. This limitation is not caused by the choice of a polynomial activation function, but a consequence of the special structure of the Jacobian of the nonlinearity Eq. (22) and the latent distribution p_Z.

The method can be generalized to incorporate multimodal distributions by the choice of a multimodal latent distribution. For example, for p_Z a Gaussian mixture model with m components, the networks learn an interacting theory with m sets of coefficients $A^{(k,m)}$, one set for each of the m modes of the learned distribution. Alternatively, we can directly parametrize $\ln p_Z$ by a multimodal polynomial. Then the computation of the coefficients simply starts with a different set of coefficients for the latent distribution, but the same update equations as derived for the single mode Gaussian case apply.

An effective unimodal model, however, may also prove useful. While volume-preserving invertible networks with unimodal latent distribution p_Z cannot learn a multimodal distribution p_θ exactly, they can learn an approximation. In this section we show how volume-preserving networks can therefore be used to extract an effective theory in the multimodal case.

To avoid tying results to a particular choice of distribution, we generate actions S_R with random coefficients $\{R^{(k)}\}_{k\leq 3}$. A diagonal negative action coefficient $R^{(4)}$ is then added to obtain a normalizable distribution. We ensure that the corresponding distributions are multimodal and that their terms are balanced in strength using a sampling method for the coefficients that is detailed in Appendix E. Although the action S_R is an unnormalized log-probability [we do not compute the constant term in S_R which ensures $\int dx \exp[S_R(x)] = 1$ as it is not needed for the sampling method], we can then sample a training set $\mathcal{D}$ using a Markov chain Monte Carlo (MCMC) sampler; for this work we used a Hamiltonian Monte Carlo [22–24] sampler implemented in PyMC3 [25]. (For details see Appendix F.)

Given a known random action S_R and a corresponding training set $\mathcal{D}$ of generated samples, we train networks of different depths and compare their action coefficients. In Fig. 4(a) we show a two-dimensional example of such a randomly generated distribution as well as the learned monomodal approximation. Figures 4(b)–4(d) show comparisons of the true compared to the learned coefficients in the $d = 10$ dimensional case. As expected, some action coefficients cannot be learned correctly. The largest deviations from the true coefficients occur in the diagonal entries $A_i^{(1)}, A_{ii}^{(2)}, \ldots$. However, we observe that many action coefficient entries that have at least two different indices [shown as $A^{(2)}_{\text{off-diag}}$ in Fig. 4(d)] recover approximately the correct value.

Using the network to generate samples, which hence belong to the learned distribution p_θ, we compare the cumulants of the two distributions. The cumulants of a distribution can be computed from its moments and vice versa. For example, the first three cumulants are equal to the mean, the variance, and the centered third moment of a distribution. Cumulants are better suited than moments to compare two different distributions because they contain only independent statistical information. We distinguish kth-order cumulants from moments by using single brackets $\langle x^k \rangle$ for moments and double brackets $\langle\langle x^k \rangle\rangle$ for cumulants.

Despite the disparity in the coefficients, we find that the cumulants agree up to third order [compare Figs. 4(e)–4(g)]. The learned distribution is therefore an effective theory that reproduces the statistics of the system beyond the Gaussian order, since in a Gaussian model the third-order cumulants are zero. Equation (32) shows that the action coefficient $A^{(k)}$ converges in expectation either when the moments of the training set and the learned distribution coincide, $\langle x^{\otimes k} \rangle_{\mathcal{D}} = \langle x^{\otimes k} \rangle_A$, or when the network cannot tune the coefficients in the relevant direction. Therefore the training aims to match the moments $\langle x^{\otimes k} \rangle_{\mathcal{D}}$, $\langle x^{\otimes k} \rangle_A$ (and, thereby, the cumulants) within the bounds of the flexibility allowed by the network architecture. We find that the higher-order cumulants are learned later in training [see Fig. 4(h)].

Mulitmodality often appears as a result of symmetry breaking. Consider the classical example of an Ising model [18]. The action of this system is symmetric under a global flipping of all spins. Below the critical temperature, two modes appear, one for positive and one for negative net magnetization. However, in a physical system, this multimodality cannot be observed, because the probability of a global sign flip approaches zero as the system size increases. Furthermore, external factors such as coupling to the environment or a measurement device will also break the symmetry. In such a setting, the network can nevertheless find an informative theory, characterizing the observed monomodal distribution.

C. Interaction on a lattice

Physical theories often feature a local structure of the interactions, for example, a lattice structure. We here

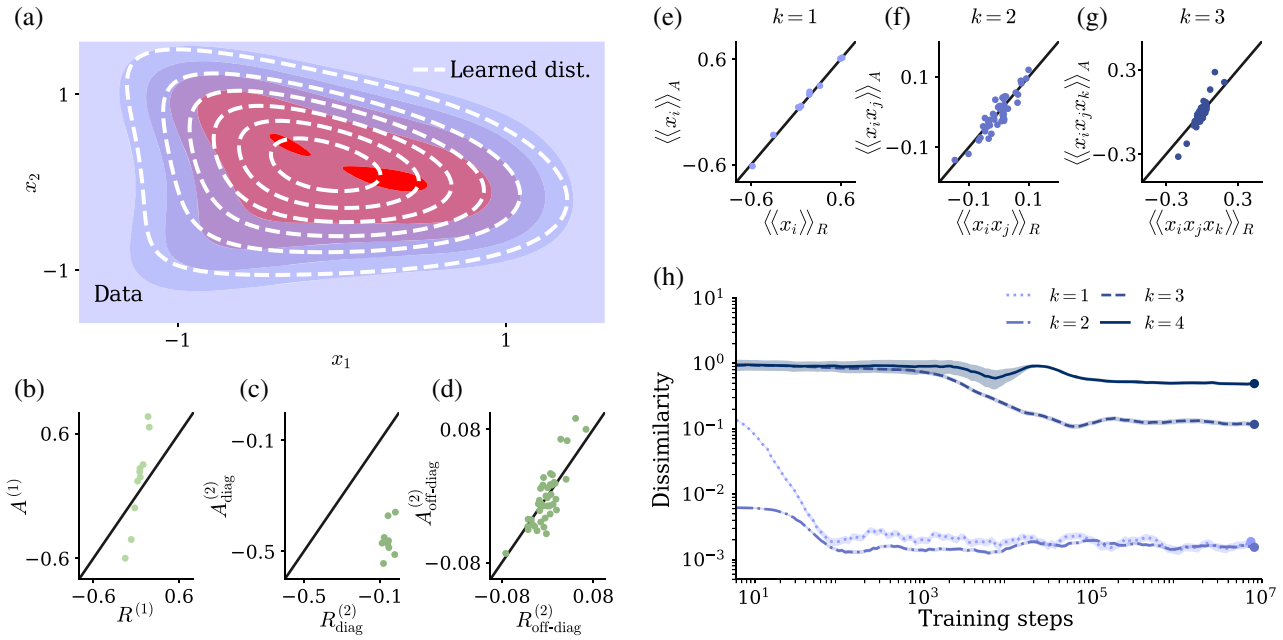


FIG. 4. Learning an effective monomodal theory. (a) Two-dimensional example of random density with multiple maxima. White lines are level lines of learned distribution for a five layer network. All other panels show results from a $d = 10$ dimensional dataset. (b)–(d) Learned over true coefficients for a three layer network on a $d = 10$. We distinguish diagonal tensor entries from off-diagonal ones, where at least two indices differ. (e)–(g) Learned over true cumulants, computed from samples. Error bars are typically smaller than marker size. (h) Dissimilarity of true and learned cumulants: $1 - \cos\angle(\langle\langle x^{\otimes k}\rangle\rangle_A, \langle\langle x^{\otimes k}\rangle\rangle_R)$ over training steps. We record the cumulants at logarithmically spaced intervals during training. The curves are then smoothed by averaging over ten adjacent recording steps. Shaded areas show the variation due to the estimation of the cumulants from samples. Dots indicate training stage of cumulants shown in (e)–(g).

construct such a system by introducing nearest-neighbor couplings on a square lattice of $d = 10 \times 10$ sites with periodic boundary conditions. Furthermore, we introduce self-interaction terms of second and fourth order. The resulting action is symmetric under a global sign change and under translations along the lattice. In an experiment, such symmetries may be broken by the coupling of the system to an external environment. We model this breaking of both symmetries by introducing a heterogeneous external field which introduces a bias to each degree of freedom.

The action therefore reads

$$S_I(x) = -\beta\left[\frac{1}{2}\sum_{i,j} x_i(r_0\delta_{ij} - \Lambda_{ij})x_j + \sum_i (h_i x_i + u x_i^4)\right]$$
$$=: I^{(1)}\cdot x + I^{(2)}\cdot (x)^{\otimes 2} + I^{(4)}\cdot (x)^{\otimes 4}, \tag{33}$$

where the $I^{(k)}$ are the coefficients of the true distribution and we have omitted the normalization. Here β serves as an inverse temperature. The matrix Laplacian $\Lambda_{ij} = -\delta_{ij}\deg(i) + a_{ij}$ with a_{ij} the adjacency matrix ($a_{ij} = 1$ if i, j are connected and $a_{ij} = 0$ else) constitutes an interaction with the $\deg(i) = 4$ nearest neighbors on the lattice. Both the diagonal part of Λ and r_0 encode a second-order self-interaction.

The fourth-order term is another self-interaction $-\beta u x_i^4$. This model can be considered the lattice version of the effective long distance theory of an Ising model in two dimensions [18]. We illustrate the network topology and external field in Fig. 5(a).

As in Sec. IV B, we sample from this distribution using an MCMC sampler (see Appendix F for details) and train networks of different depths L. In Figs. 5(b)–5(d) we compare the learned action coefficients $A^{(k)}$ to the corresponding target values $I^{(k)}$. We find good agreement for $A^{(1)}$ and the off-diagonal values $A^{(2)}_{ij}$, $i \neq j$, independent of network depth. The external field and nearest-neighbor coupling is therefore recovered accurately. The self-interaction $A^{(2)}_{\text{diag}}$ is typically lower than the target value while the fourth-order self-interaction $A^{(4)}_{\text{diag}}$ is typically larger. Since both parameters control the widths of the distributions, the slightly lower $A^{(2)}_{\text{diag}}$ can compensate for the too small magnitude of $A^{(4)}_{\text{diag}}$, producing an effective theory. Nevertheless, the higher-order coefficients $A^{(k\geq 3)}$ of the learned theory are relevant. We show in Figs. 5(f)–5(h) that the first, second, and third cumulants of the learned and true distribution approach each other as the network depth increases. A Gaussian approximation of S_I would only tune $A^{(1)}$ and $A^{(2)}$ with $A^{(k\geq 3)} = 0$ to match the first and

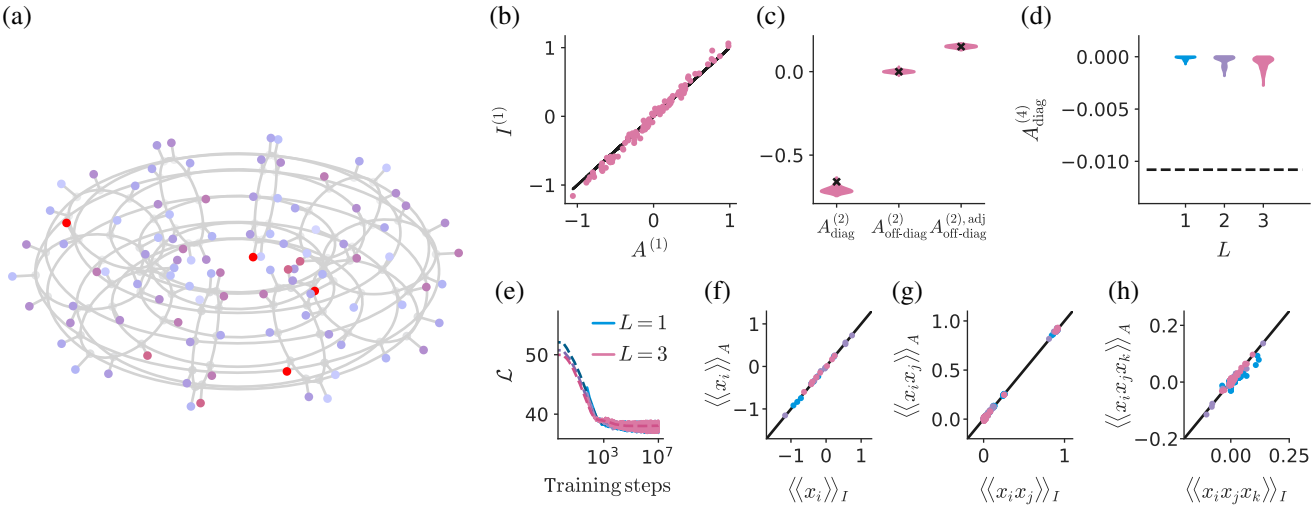


FIG. 5. Symmetry broken lattice model for networks of varying depth trained on a $d = 10^2$ dimensional dataset with $D = 10^6$ samples. (a) Sites of square lattice with periodic boundary conditions distributed on a two-dimensional torus. Colored dots show strength of external field h at connected lattice sites. (b) Learned over true first-order coefficients for network depth $L = 3$. (c) Distribution of learned coefficient entries $A^{(2)}$ compared to target values (black crosses) for network depth $L = 3$. We distinguish self-interaction terms $A^{(2)}_{\text{diag}}$ from off-diagonal entries $A^{(2)}_{\text{off-diag}}$. From the off-diagonal entries $A^{(2)}_{\text{off-diag}}$, we further separate those entries belonging to adjacent lattice sites $A^{(2),\text{adj}}_{\text{off-diag}}$. (d) Training loss (solid curves) and test loss (dashed curves). Colors distinguish different network depths L. (e) Distribution of learned fourth-order self-interactions as function of network depth. The dashed line marks the target value. (f)–(h) Learned over true cumulants of up to third order. Cumulants were computed on a subset of 10 randomly chosen lattice sites. Colors distinguish different network depths L.

second cumulants shown in Figs. 5(f) and 5(g). As in the multimodal case, therefore, we learn an effective non-Gaussian theory. The effective theory may result from the depth of the network being small in comparison to the dimensionality of the system to tune all higher-order coefficients $A^{(k\geq 3)}$. The number of independent entries in the action coefficients up to the fourth order is $\mathcal{O}(d^4)$, roughly 4.6×10^6 for the case of $d = 10^2$, with by far the most number of entries in the highest-order coefficient $A^{(4)}$. In contrast, the number of free parameters of a single layer is $\lceil d/2 \rceil^3$ in $\tilde{\chi}_l$, d^2 in W_l, and d in b_l, so all in all $\lceil d/2 \rceil^3 + d(d+1)$. Although the coefficients do not depend linearly on the network parameters, this gives a rough estimate of the required depth $L = 34$ of the network, at which the number of free parameters in the network and in the coefficients coincide. Since the number of entries in the coefficients grows with $\mathcal{O}(d^4)$, but the number of free parameters in the network is only $\mathcal{O}(Ld^3)$, the depth required to tune all coefficient entries grows with d. [Analogously, one can work out that one needs a network of depth $L = \mathcal{O}(d^{k-3})$ to learn all interaction coefficients of order k exactly by using quadratic nonlinearities as defined here.] Below this depth, it may well be that the flexibility of the network is too small to tune all fourth-order action coefficients. Even though $\langle (x^{\otimes k})_\alpha \rangle_\mathcal{D} - \langle (x^{\otimes k})_a \rangle_A$ in Eq. (32) is likely nonzero then, the coefficients may still reach stationary values by the combination of the terms on the right-hand side of Eq. (32) vanishing. Indeed we find in Appendix G that, in lower dimensions, smaller network depths are sufficient to tune the higher statistical orders. Furthermore, the learned coefficients $A^{(4)}$ approach the target value as the depth increases. In all examples studied here, the alignment between learned and true statistics improves with depth, which increases the network flexibility. For shallow networks, however, although the learned action is not equal to the true one, it effectively describes the statistics of the true distribution beyond the Gaussian order as indicated by the good agreement of the cumulants. This behavior is equivalent to that of renormalized theories [18], which feature the same statistical correlations while changing the interaction strengths in a consistent manner.

D. Inferring pixel interactions in MNIST

We will now exemplify the learning of higher-order interactions on the MNIST dataset [26]. MNIST is a dataset of gray scale images with 28×28 pixels. Each image shows a handwritten digit between zero and nine. To show the effect of higher-order interactions, we train both a linear and a nonlinear network. A linear network may only learn a Gaussian approximation of the data distribution, corresponding to coefficients $A^{(1)}$ and $A^{(2)}$ in Eq. (1), whereas a nonlinear network is able to learn higher-order coefficients $A^{(3)}, A^{(4)} \neq 0$ as well. The authors of Refs. [5,6] showed that nonlinear classifiers can make use of the higher-order

statistics of MNIST and other datasets; therefore it is useful to gain an understanding of the higher-order statistics also in the classification setting.

We train networks on single digit datasets, since the whole dataset is likely multimodal. (One could equivalently train with a Gaussian mixture as a latent and then treat each mixture component separately as outlined in Sec. IV B.) For brevity, we here focus on the digit three, examples of which are shown in Fig. 6(a); comparable results for other digits are given in Appendix H. We then sample new digits from both the linear and the nonlinear models, uncurated (randomly selected) samples of which are shown in Figs. 6(b) and 6(c). We show the test and training loss in Fig. 6(i). The more flexible nonlinear model overfits the training data slightly, after around 10^3 optimization steps. We take the training stage yielding the best test performance to compute the coefficients. We then first compute the self-interacting action coefficients, namely the diagonal entries $A^{(2)}_{ii}$, $A^{(3)}_{iii}$, $A^{(4)}_{iiii}$ of both models and visualize them on the same 28×28 grid, such that the action coefficient entries lie in the same position as the pixels whose statistics they characterize [see Fig. 6(h) for the linear network and Figs. 6(k)–6(m) for the nonlinear network]. We further visualize $\mu_A = -\frac{1}{2}(A^{(2)})^{-1}A^{(1)}$ in the same way for both models in Figs. 6(g) and 6(j); in the case of the linear model, this is simply the mean of the associated Gaussian theory. For the nonlinear model, the mean cannot be computed exactly from $A^{(1)}, \ldots, A^{(4)}$; we nevertheless depict the same quantity μ_A for comparability. The coefficients $A^{(1)}, A^{(2)}$ are very similar across models; however, the higher-order coefficients of the nonlinear model expose a further structure in the data: we find that the third-order coefficients are large near the edges of the digit. The distributions of the pixels at the digit edges are typically skewed, which are better approximated by including a third-order term. We exemplify this behavior on three pixels of the images in Figs. 6(d)–6(f). Pixels on the border of the image are typically black, corresponding to a single peak at zero. Pixels in the middle of the "three" can be either black or white, which is modeled by both networks via a broader distribution. Finally, pixels at the typical location of edges are either black or gray scale, resulting in the skewed distribution in Fig. 6(f). Therefore, the diagonal terms in $A^{(3)}$ are good indicators of typical edge locations.

We further examine pixel-pixel interactions. To showcase the higher-order statistics, we focus on three-point interactions. To this end, we compute those entries in the

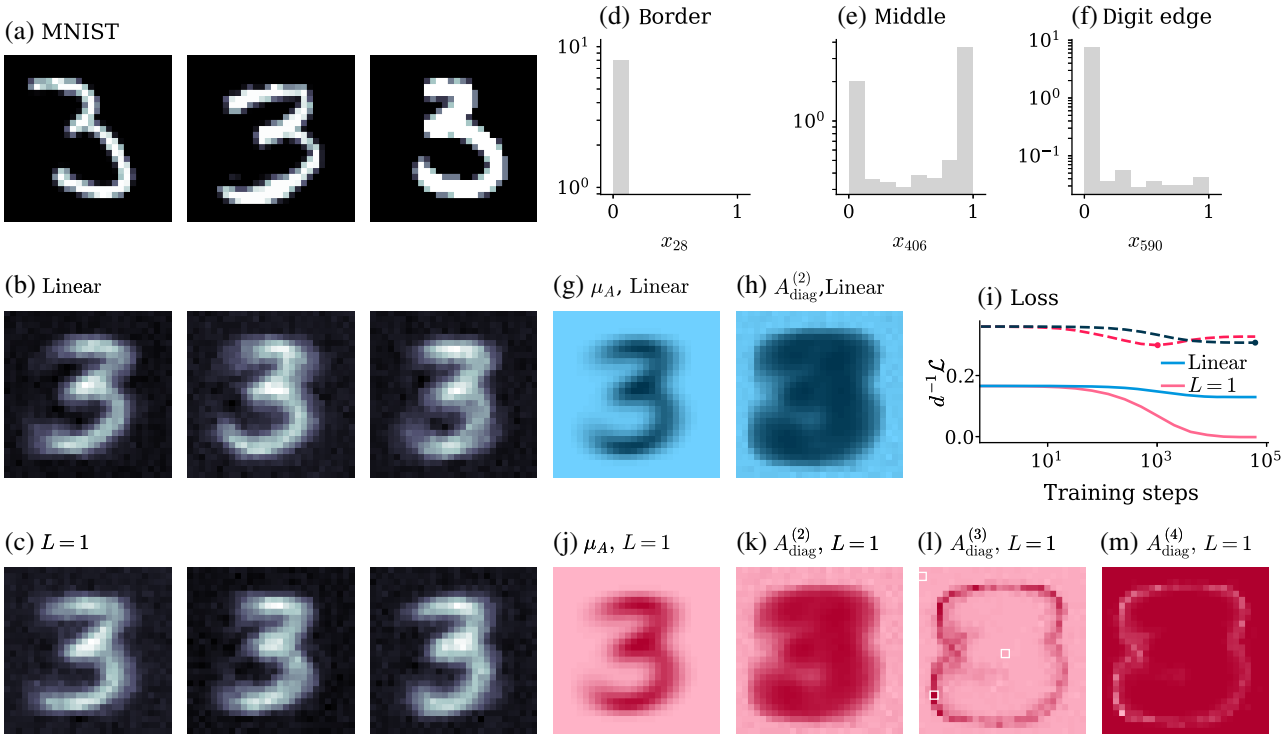


FIG. 6. Inference of interactions on MNIST for digit three. (a)–(c) Images from the dataset, the linear model, and an $L = 1$ layer nonlinear model, respectively. (d)–(f) Single pixel activation statistics from three distinct locations in the image. (g) Entries of the mean μ_A of the Gaussian theory (linear model). (h) Entries on the diagonal of the second-order coefficient $A^{(2)}_{\text{diag}}$ of the linear model. (i) Training loss (full lines) and test loss (dashed lines) over training steps. Dots mark the training stages from which the coefficients of both models were extracted. (j) Mean μ_A for the nonlinear model if $A^{(3)}, A^{(4)}$ were not present. (k)–(m) Entries on the diagonals of the remaining coefficients of the $L = 1$ layer nonlinear model. White squares in (l) mark the locations of the single pixel statistics shown in (d)–(f).

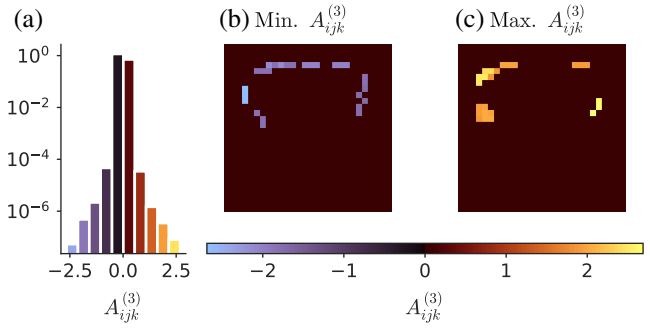


FIG. 7. Three-point interactions in MNIST for digit three. (a) Histogram of all entries of the third-order coefficient $A^{(3)}_{ijk}$ for $i \neq j$, $j \neq k$, and $i \neq k$, color coded according to their value. (b),(c) Triplets corresponding to the ten most negative (b) or most positive (c) values of $A^{(3)}_{ijk}$. For each triplet, we color pixels i, j, and k, according to the value of the interaction coefficient in $A^{(3)}_{ijk}$. Thus triplets of pixels corresponding to the same entry in $A^{(3)}$ have the same color.

third coefficient which correspond to an interaction between three distinct pixels $A^{(3)}_{ijk}$ for $i \neq j$, $j \neq k$, and $i \neq k$. Since the inputs are all positive, $0 \leq x_i \leq 1$, a positive value of $A^{(3)}_{ijk}$ favors images where all pixels are jointly brighter, whereas a negative value of $A^{(3)}_{ijk}$ favors that at least one of the pixels have value zero, i.e., that at least one pixel is black. Out of all distinct triplets i, j, k, we pick those which have largest magnitude in $A^{(3)}_{ijk}$ and show those with the most negative value in Fig. 7(b) and those with the most positive value in Fig. 7(c). For both cases, we find that the three-point interactions are predominantly *local*, coupling adjacent pixels. They further align with the edges of the digit. The largest values are found in the upper half of the digit; the lower half of the digit shows a qualitatively similar enhancement of third-order coefficients, only to a slightly lower extent. Most entries in $A^{(3)}_{ijk}$ are, however, close to zero [see Fig. 7(a)]. This can be readily understood from the fact that there are many more entries in $A^{(3)}_{ijk}$ which couple pixels that are distant from each other, and can therefore be assumed to be mostly independent. Thus we find that the higher-order statistics express nontrivial interactions localized at digit edges. In Appendix H we provide details on the training process and show the same analysis for the digit two, as well as the strongest 10^2 three-point interactions for both digits. We find that the results do not depend on the choice of the digit. Thus the third-order interactions encode the digit edges in a localized manner.

V. DISCUSSION

A. Summary of main findings

We have developed a method to learn a microscopic theory from data—concretely, we learn a classical action that assigns a probability to each observed state. For this data-driven approach, we employ a specific class of deep neural networks that are invertible and that can be trained in an unsupervised manner, without the need of labeled training data. Such networks have been used before as generative models [9,10,13], but are generally considered a black box: after training the learned information is stored in a large number of parameters in an accessible, yet distributed and generally incomprehensible manner.

The diagrammatic formalism developed here allows us to extract the data statistics from the trained network in terms of an underlying set of interactions—a common formulation used throughout physics. To achieve this, we designed the network architecture as a trade-off between flexibility and analytical tractability. The choice of a quadratic polynomial, along with a volume-preserving invertible architecture, allows us to obtain explicit expressions for the interaction coefficients. This formalism shows how the interplay between linear and nonlinear mappings in the network composes non-Gaussian statistics, and hence higher-order interactions, in a hierarchical manner. As a consequence of the quadratic interaction constituting the fundamental building block, higher-order interactions are decomposed into this simplest possible form of nonlinear interplay. As a result, the order of interaction in the data directly maps to the required depth of the network in an understandable manner, thus providing an explanation of why deep networks are required to learn higher-order interactions.

We complement this study with a characterization of the training process, confirming the expectation that both larger training set size and network depth improve learning. Larger datasets in general decrease the bias of the learned distribution due to undersampling of the true distribution. This point is most severe for higher-order statistics, while the first two orders of the statistics are typically learned robustly also from limited data. We provide an approximate expression, Eq. (32), to investigate the convergence properties of statistics of different orders. While deeper networks are required to offer sufficient flexibility to learn higher-order statistics, the larger number of trainable parameters at the same time requires more data to learn the statistics accurately. Alternatively, the network flexibility can be increased by raising the order of the polynomial activation function. Finding the optimal trade-off between local nonlinearity and depth is an interesting point of future research.

B. Related work

The general problem of inferring models from data discussed in this work is a well-known challenge. In the dynamical systems setting, the authors of Refs. [27–30] use regression to infer the right-hand side of the governing differential equation of a system from a set of basis functions. Other studies [31,32] infer rules for the time

dependence of couplings (synaptic plasticity) using regression and genetic programming. These approaches produce interpretable models, but require a predetermined set of basis functions or operations, through the combination of which the system dynamics are approximated. Inference of parameters of stochastic processes [33–36] also relies on the specific form of the update equations. Prior knowledge about likely terms in the dynamical equations or their exact functional form is therefore needed in these works.

Perhaps the most prominent example of an inference problem is the Ising model, which is also at the heart of training Boltzmann machines [37]. Here, a set of pairwise couplings between binary degrees of freedom is inferred. The difficulty in inferring pairwise couplings stems from the need to maximize the likelihood, which requires computing the correlation functions of the inferred model at every optimization step. Nevertheless, many algorithms now exist for inferring models with pairwise interactions, such as the inverse Ising or XY model. However, these algorithms deal with discrete data and are limited to pairwise interactions.

Other than Boltzmann machines, a range of techniques for the Ising or XY models first solve the forward problem, namely the statistics given the couplings—using variations of mean-field theory [38–41] or the Thouless-Anderson-Palmer (TAP) equations [42–44]—and then invert these relations explicitly or iteratively [44–47]. Maximum likelihood methods or the TAP equations have also been used to infer the patterns stored in Hopfield models [48,49]. In the special case of a treelike, known network topology, or translationally invariant higher-order couplings along a linear chain, the inverse problem can be solved exactly [50]. Further works maximize the likelihood of the network model given the data, by using belief propagation to reconstruct the network structure from infection cascades [51], or Monte Carlo sampling to infer amino acid sequences in proteins [52]. A series of works use the pseudo-likelihood [53,54] and interaction screening objective [55], or derive optimal objective functions [56,57] using the cavity method and replica trick. The authors of Ref. [58] impose a special factorization of the distribution of discrete variables. In contrast to these works, in this study we consider continuous rather than discrete variables. Furthermore, we are not restricted to pairwise interactions and do not require prior knowledge on the structure of interactions.

Zache *et al.* [59] also solve the forward problem: they approximate a higher-order interacting theory to tree level or one-loop order in the effective action, and thereby obtain an invertible relation between interactions and correlations. This approach relies on the validity of the approximations, namely for the typically difficult step from interactions to correlation functions. These approximations are not necessary to train INNs, as the correlation functions are implicitly generated by the network mapping.

Neural networks have also previously been used to treat inverse problems. They are trained to infer the posterior probability of characteristic parameters given data [60,61], and to compute renormalized degrees of freedom that are maximally informative about the global state of a system [62]. For systems of interacting identical particles, Cranmer *et al.* [63] use symbolic regression on trained graphical neural networks to derive interpretable interactions. However, these approaches make no lucid connection between the learned model and the parameters of the neural network as we do in this study.

C. Extensions

Exponential distributions with polynomial actions can model a broad class of phenomena, including highly skewed distributions (such as the pixel statistics in MNIST), but they are not universal: in particular, they have finite support everywhere. This limitation, however, is not specific to this approach, since learning heavy tails in general is not possible without strong parametric priors. We expect that a model learning on a finite dataset drawn from a heavy-tailed distribution would adjust its coefficients to broaden the distribution to incorporate the data of the training set. In the absence of prior knowledge on the heavy-tailed nature of the data distribution, our method therefore yields interpretable statistics—and therefore an effective theory—for the data within the high probability region. Invertible neural networks which are not volume preserving can yield higher performance and broader tails but lack interpretability.

To provide the most transparent setting, we have here chosen the simplest but common case of a latent Gaussian distribution, which has the aforementioned advantage of mapping to a noninteracting theory. A consequence is that the latent distribution only has a single maximum. Since for invertible volume-preserving networks the number of modes in data space and in latent space are identical, these network architectures therefore learn only monomodal distributions. For many settings of interest this is sufficient: multimodal distributions in physical systems often occur together with nonergodic behavior such as spontaneous symmetry breaking, selecting one of the modes of the distribution. As presented, our approach necessarily learns the statistics of the mode selected by nature, and thus obtains an effective theory of the single observed phase of the system. The simplest way to learn genuine multimodal distributions is the use of a multimodal latent distribution, such as a Gaussian mixture. Our framework would then provide one set of action coefficients for each mixture component, correspondingly offering one effective theory for each phase.

The analytical framework we developed can also readily be extended to higher-order nonlinearities: In terms of the diagrammatic language, the quadratic activations used in this work amount to the splitting of "legs" in the Feynman diagrams into pairs. Likewise one obtains a threefold splitting from a cubic term, a fourfold splitting from a quartic term, and so on. Such higher-order nonlinearities

would allow the composition of more complex interactions with fewer layers.

Several studies have highlighted the role of the stochasticity of the training algorithm for networks performing classification [64–66]. A possible starting point for understanding SGD for generative models could be Eq. (32), which relates the trajectory of the learned distribution in coefficient space to the training algorithm. Equation (32) closely resembles the training of restricted Boltzmann machines, where the pairwise coupling matrix between hidden and visible layers is updated according to the difference between learned and observed pairwise correlations [37]. Studying Eq. (32) could shed light on the dynamics of unsupervised learning, for which the architecture used in this work is a fully tractable prototype.

Another challenge in learning interacting theories of higher order is the necessarily large size of the action coefficients which grows with the dimension, irrespective of how these interactions are inferred. However, it is plausible that not all terms in these tensors are equally important: For spatially or temporally extended systems, interactions between distant degrees of freedom may be irrelevant. The framework explicitly shows how higher-order interactions are composed of lower-rank coefficients. This may be leveraged to extract the most relevant contributions in a tractable manner (see Appendix B).

Actions such as Eq. (1) also appear in dense associative memory models [67,68], which store patterns of binary variables. There, the number of patterns and the robustness of the storage increase [69] with the interaction order. Here, we observe that more flexible models yield higher interaction orders, but require a larger amount of data to train, and are more susceptible to noise. We believe that exploring the link between associative memory models and generative models as presented here in the context of noisy data could hence be a fruitful future direction of research.

From a physics point of view, one may regard the trained network as a device to solve an interacting classical field theory in a data-driven manner: once the network has been trained, it maps each configuration of the interacting theory in data space to samples in latent space that follow a Gaussian theory, hence a noninteracting theory. Such a mapping allows one to compute arbitrary connected correlation functions of the interacting theory. The framework offers two routes to this end. The traditional route uses common rules of diagrammatic perturbation theory to obtain controlled approximations of connected correlation functions in terms of connected diagrams constructed from propagators and interaction vertices of the inferred action. An alternative route directly constructs connected correlation functions hierarchically across the layers of the network, ultimately reduced to pairwise interactions on the level of the latent Gaussian. For example, the second-order correlations read $\langle\langle x_i x_j \rangle\rangle_{p_\theta} = \langle\langle f^{-1}_{\theta,i}(z) f^{-1}_{\theta,j}(z) \rangle\rangle_{z\sim\mathcal{N}(0,\mathbb{1})}$. For the nth-order correlations $\langle\langle x_{i_1} \cdots x_{i_n} \rangle\rangle_{p_\theta}$ one can therefore either work out the coefficients of the polynomial $f^{-1}_{\theta,i_1}(z) \cdots f^{-1}_{\theta,i_n}(z)$, in a similar manner to the action transform, and then average the resulting function over p_Z, or estimate the correlations by drawing samples from the generative network. An open avenue to explore further in this regard is the link between the presented framework and asymptotically free theories, where an interacting theory becomes noninteracting at high energy (UV) scales. In that case, different scales are connected by a renormalization group (RG) flow. It would be interesting to investigate whether the change of couplings described by the RG flow can be related to the transformations performed by the network. More broadly, the ability to learn an interacting theory by the network can be considered an alternative to asymptotic freedom, as the flow across layers does not have to correspond to a change of length scale.

With the here proposed extraction method for the action of a physical system at hand, one can now proceed to extract hitherto unknown interacting models from data. One interesting application of this approach is to learn actions for systems for which a microscopic or mesoscopic description is not known, for example, in biological neuronal networks: the inferred coefficients would determine the importance of nonlinear interactions in biological information processing. The approach may also be fruitful when applied to systems in physics where the microscopic theory may be known but an effective theory is sought that captures an observed macroscopic phenomenon.

ACKNOWLEDGMENTS

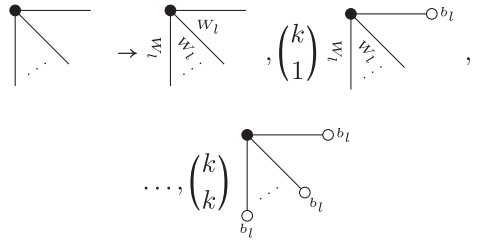
We are grateful to Thorben Finke, Sebastian Goldt, Christian Keup, Michael Krämer, and Alexander Mück for helpful discussions. This work was partially supported by the German Federal Ministry for Education and Research (BMBF Grant No. 01IS19077A to Jülich and BMBF Grant No. 01IS19077B to Aachen) and funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—368482240/GRK2416, the Excellence Initiative of the German federal and state governments (ERS PF-JARA-SDS005), and the Helmholtz Association Initiative and Networking Fund under Project No. SO-092 (Advanced Computing Architectures, ACA). Open access publication funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—491111487.

APPENDIX A: DIAGRAMMATIC UPDATE EQUATIONS

We here describe the diagrammatic rules to compute the action coefficient transforms in Eqs. (15) and (26).

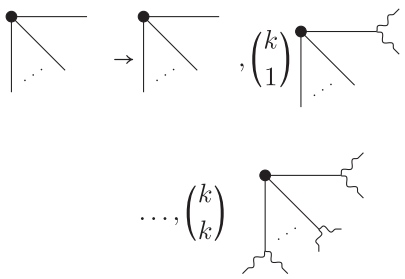
Following the structure in Sec. III, we first treat the linear transform Eq. (13). Each action coefficient $A^{(k)}_{l+1}$ is represented by a vertex, where the number of outgoing lines, also called legs, is equal to the rank of the coefficient. Each

leg is assigned an index i_l corresponding to the indices $(i_1, \ldots, i_k)$ of the coefficient entry of $A^{(k)}_{l+1}$ it represents. Equation (15) shows that each index of the coefficient must either be contracted with W_l, and therefore remains a leg of the resulting vertex from the second index of W_l, or it must be contracted with b_l, and therefore drops out. We represent the contraction with W_l by an elongated line decorated with W_l, and the contraction with b_l as a leg ending on an empty circle. Thus, to compute the transformed action coefficients, we must add empty circles to the previous vertices in all possible ways and then elongate the remaining legs using W_l. A diagram with k legs therefore produces the following diagrams:



where the combinatorial factors $\binom{k}{l}$ arise due to the different ways of choosing legs which are contracted with b_l. For example, there are $\binom{k}{1} = k$ ways of choosing a single leg from a vertex with k legs which is then contracted with b_l. We first compute the new diagrams for all k, then sum up all diagrams that have equal numbers of legs to one coefficient. This illustrates how higher-order action coefficients, by the contraction of their indices with the biases, i.e., the attachment of empty circles to their legs, contribute to lower-order coefficients.

For the nonlinear transform, we have the opposite effect: each index in the coefficient either remains or is contracted with χ_l, which increases the rank of the transformed diagram by one. Therefore, either the legs of vertices remain as they are or they must be split into two to signify the contraction with χ_l, which increases the number of legs of the vertex by one. We keep the split legs distinguishable from three-point vertices by using curved lines for the split legs and sum over all possible ways to split legs. A diagram with k legs therefore produces the following diagrams:



Here, the combinatorial factors arise due to the number of ways in which to choose the split legs. The number of factors χ_l in any diagram can then be read off from the number of leg splits. As in the linear transform, to compute the transformed action coefficients of rank k, we must therefore sum over all diagrams with equal numbers of legs. This illustrates how higher-order action coefficients arise through the splitting of legs by factors of χ_l.

APPENDIX B: DECOMPOSED TENSORS

Higher-order tensors $T^{(k)}$ of rank k can become numerically intractable for large dimension d as the number of entries in $T^{(k)}$ grows as $\mathcal{O}(d^k)$. Specifically, two challenges arise. First, to store the entries of the tensors. Second, to compute contractions with matrices W such as

$$T^{(k)} \cdot (W)^{\otimes k}, \tag{B1}$$

which arise due to the linear coefficient transform Eq. (15). For these contractions, we must compute the sum over all entries,

$$\left(T^{(k)} \cdot (W)^{\otimes k}\right)_{i_{1,\ldots i_k}} = \sum_{j_1,\ldots j_k=1}^{d} T^{(k)}_{j_1,\ldots j_k} W_{j_1,i_1} \cdots W_{j_k,i_k};$$

therefore, without further simplification this entails the computation of d^k entries of $T^{(k)} \cdot (W)^{\otimes k}$ from d^k terms each, so the total number of floating point operations scales as $\mathcal{O}(d^{2k})$. Even though the number of steps required therefore only grows polynomially with d, for realistic dataset sizes and $k = 4$, this number increases very fast.

To facilitate the computation of coefficients with rank $k = 4$, we exploit that they are built from coefficients of lower rank to write the tensors in a decomposed form. As a first step, we decompose the network parameters χ_l. Without loss of network expressivity, we may choose χ_l to be symmetric in its latter two indices $(\chi_l)_{\mu jk} = (\chi_l)_{\mu kj}$. In the following, we drop the layer index l for brevity, as the structure of the computation is the same for any layer. We then rearrange the tensor to be a list of d symmetric matrices $\bar{\beta}^\mu, \mu = 1, \ldots, d$ such that $\chi_{\mu kj} = \bar{\beta}^\mu_{kj}$. Using the eigendecomposition of these matrices $\bar{\beta}^\mu$, we may write

$$\chi = \sum_{\mu,\nu=1}^{d} \gamma^{\mu,\nu} \otimes \beta^{\mu,\nu} \otimes \beta^{\mu,\nu}, \tag{B2}$$

where $\gamma^{\mu,\nu}, \beta^\nu$ are vectors and $\gamma^{\mu,\nu}_\tau = \delta_{\tau,\mu}\lambda^\mu_\nu$ has only one nonzero entry, namely the νth eigenvalue of the μth matrix $\bar{\beta}^\mu$. To store this object we require $2d^2$ vectors of length d, namely d^2 vectors $\beta^{\mu,\nu}$ and d^2 vectors $\gamma^{\mu,\nu}$. The magnitude of entries in χ is directly related to the magnitude of the eigenvalues λ^μ_ν, which is typically small for trained networks. We show distributions of eigenvalues from trained networks in Fig. 8. The distributions broaden with increasing depth; however, the peak of the distribution remains at

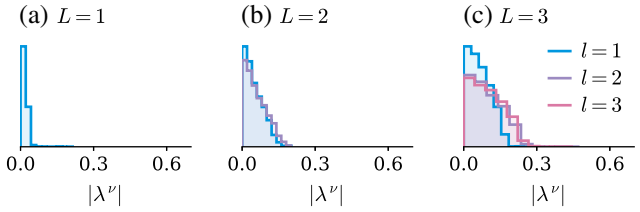


FIG. 8. Eigenvalue distributions of decomposed χ_l for networks of different depths. We decompose trained network parameters χ_l from Sec. IV C to the form of Eq. (B2) and distinguish eigenvalues from the decomposed form of different layers l.

$|\lambda_\nu^\mu| = 0$. It is therefore possible to reduce the space required to store χ and all tensors related to it by placing a cutoff $\bar{\lambda} \geq 0$ on the eigenvalues, keeping only the $\bar{n} \leq d^2$ largest eigenvalues which have $|\lambda_\nu^\mu| \geq \bar{\lambda}$. Then the number of entries required to store χ scales as $\mathcal{O}(2\bar{n}d)$, as again we need $2\bar{n}$ vectors of length d each. To further simplify the expression, we absorb the sum over μ, ν into a single index $\tau = 1, \ldots, \bar{n}$.

An alternative way to achieve the decomposition of χ into a reduced number of components would be to use the decomposed form Eq. (B2) directly during training and limit the number of independent vectors β^τ. This approach effectively trades the network expressivity for the tractability of the action coefficient transforms.

In addition to reduced storage requirements, the decomposed form (B2) also has the advantage that the decomposition translates to all tensors computed via contraction with χ, which is how higher-order coefficients are originally generated [compare with Eq. (26)]. The contraction between a rank k symmetric tensor $T^{(k)}$ and χ is

$$T^{(k)} \cdot \chi = \sum_\tau (T^{(k)} \cdot \gamma^\tau) \otimes \beta^\tau \otimes \beta^\tau.$$

If $k = 1$, the result is just a matrix. If $k = 2$, then $T^{(k)} \cdot \gamma^\tau =: \alpha^\tau$ is a vector; therefore, $T^{(k)} \cdot \chi$ can be written as a sum of outer products between three vectors. If $k = 3$, the result is a sum of outer products between a matrix $T^{(k)} \cdot \gamma^\tau = \bar{\alpha}^\tau$ and two vectors:

$$T^{(3)} \cdot \chi = \sum_\tau \bar{\alpha}^\tau \otimes \beta^\tau \otimes \beta^\tau. \tag{B3}$$

The matrices $\bar{\alpha}^\tau$ are symmetric since

$$\bar{\alpha}^\tau_{ab} = \sum_c T^{(k)}_{abc} \gamma^\tau_c = \sum_c T^{(k)}_{bac} \gamma^\tau_c = \bar{\alpha}^\tau_{ba}.$$

The case $k \geq 4$ does not arise, as any coefficient with degree $k \geq 4$ must already contain at least two factors χ.

To store the factors of Eq. (B3) we therefore require $\bar{n}$ matrices $\bar{\alpha}^\tau$ and $\bar{n}$ vectors β^τ. The number of matrix and vector entries required to store this object is therefore $\mathcal{O}(\bar{n}d^2)$. The contraction with matrices W along all indices is

$$(T^{(3)} \cdot \chi) \cdot (W)^{\otimes 4} = \sum_\tau (W^T \bar{\alpha}^\tau W) \otimes (W^T \beta^\tau) \otimes (W^T \beta^\tau),$$

which corresponds to $\bar{n}$ matrix-vector products $W^T\beta^\tau$ and $2\bar{n}$ matrix-matrix products for $\bar{\alpha}^\tau W$ and $W^T(\bar{\alpha}^\tau W)$. Each term in the matrix-matrix product is computed from d terms; therefore, the number of terms required to compute $W^T\bar{\alpha}^\tau W$ is $\mathcal{O}(2d^\omega)$ with the matrix multiplication exponent ω, which depends on the concrete algorithm used for matrix multiplication, e.g., $\omega \approx 2.8$ for the Strassen algorithm [70]. To evaluate the contraction, we therefore need to compute $\mathcal{O}(2\bar{n}d^\omega)$ terms. Even in the case of no cutoff $\bar{\lambda} = 0 \Rightarrow \bar{n} = d^2$, this approach significantly reduces the required computations compared to the naive implementation.

In Ref. [71] it was shown that the number of floating point operations needed to compute general contractions of the type of Eq. (B1) can be reduced by exploiting the symmetry of the tensors. They propose a simple scheme to reduce the number of floating point operations (using $\omega = 3$) to roughly $\mathcal{O}(d^{k+1})$, and a more complex structure of saving these tensors, which further speeds up the computations at the expense of storing more intermediate entries. In the absence of any cutoff, we find a scaling of our algorithm roughly equal to the simpler scheme proposed in Ref. [71]. In our experiments, in Secs. IV A–IV C, we have used no cutoff $\bar{\lambda} = 0$. For the MNIST dataset with $d = 784$, we used a cutoff of $\lambda = 10^{-2}$. The combination of a cutoff, more efficient storing of symmetric tensors as suggested in Ref. [71], or restricting the number of free components in χ directly, facilitates the extension of the coefficient transforms to higher dimension d.

APPENDIX C: DISSIMILARITY OF PARAMETERS

We here show that although the learned statistics of two networks may be the same, their parameters do not need to align. To this end, we use the teacher-student setup presented in Sec. IV A, and compute the cosine similarities between pairs of network parameters b_l, W_l, χ_l—both between the teacher and the student and between the teacher and a network of the same architecture with random Gaussian weights. We then average over the cosine similarities of the different parameters to obtain the average network cosine similarity. We show in Fig. 9. that although the teacher and student coefficients approach each other for increasing dataset sizes D, their network parameters remain dissimilar. The appropriate object to compare the learned statistics is therefore the action, not the parameters of the network.

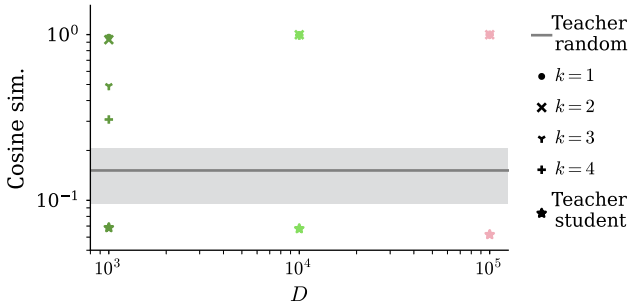


FIG. 9. Dissimilarity of parameters. Stars show the cosine similarities of the teacher and student network parameters trained on varying dataset sizes D. The average cosine similarity between the teacher and 10^2 randomly generated random networks is marked by the gray line, the shaded area encompasses one standard deviation. The remaining markers display the cosine similarity between the teacher coefficients $T^{(k)}$ and student coefficients $S^{(k)}$.

APPENDIX D: LEARNING ACTION COEFFICIENTS WITH SGD

The learned action $S_\theta(x)$ depends on the parameters θ only through the coefficients $A^{(k)}$. We can therefore also view the training as a nonlinear optimization of the action coefficients via the parameters. However, we cannot freely move in this coefficient space: we must ensure that the action stays normalized, $\int \exp[S_\theta(x)]x = 1$. To this end, we fix the constant term in the action:

$$A^{(0)} = -\ln \int \exp\left(\sum_{k\geq 1} A^{(k)} \cdot (x)^{\otimes k}\right) dx. \tag{D1}$$

Given this constraint, we rewrite the updated equation (31) in terms of the coefficients using the chain rule,

$$\Delta\theta_i = \eta \sum_{k\geq 1} \sum_{\alpha_k} \frac{\partial}{\partial A^{(k)}_{\alpha_k}} \langle S_\theta(x)\rangle_{\mathcal{D}_t} \frac{\partial A^{(k)}_{\alpha_k}}{\partial \theta_i},$$

where $\sum_{\alpha_k}$ runs over all possible (multi)indices of $A^{(k)}$. The update in the parameters induces a change in the action coefficients. We use this to approximate the update step for the coefficient entry $A^{(k)}_\alpha$ to linear order in the parameter updates:

$$\begin{aligned}\Delta A^{(k)}_\alpha &\approx \sum_i \frac{\partial A^{(k)}_\alpha}{\partial \theta_i} \Delta\theta_i \\ &= \eta \frac{\partial}{\partial A^{(k)}_\alpha} \langle S_\theta(x)\rangle_{\mathcal{D}_t} \sum_i \left(\frac{\partial A^{(k)}_\alpha}{\partial \theta_i}\right)^2 \\ &\quad + \eta \sum_{l,\alpha_l \neq \alpha} \frac{\partial}{\partial A^{(l)}_{\alpha_l}} \langle S_\theta(x)\rangle_{\mathcal{D}_t} \sum_i \frac{\partial A^{(k)}_\alpha}{\partial \theta_i} \frac{\partial A^{(l)}_{\alpha_l}}{\partial \theta_i}.\end{aligned} \tag{D2}$$

The update step $\Delta A^{(k)}_\alpha$ therefore depends on the network architecture through the derivatives $\partial A^{(k)}_\alpha / \partial \theta_i$. The derivative of the expectation of the action with respect to the coefficients in the former factor in Eq. (D2) is

$$\begin{aligned}\frac{\partial}{\partial A^{(k)}_\alpha} \langle S_\theta(x)\rangle_{\mathcal{D}_t} &= \langle (x^{\otimes k})_\alpha \rangle_{\mathcal{D}_t} + \frac{\partial A^{(0)}}{\partial A^{(k)}_\alpha} \\ &= \langle (x^{\otimes k})_\alpha \rangle_{\mathcal{D}_t} - \langle (x^{\otimes k})_\alpha \rangle_A,\end{aligned} \tag{D3}$$

where $\langle\cdot\rangle_A$ denotes the current average of the learned distribution and we used Eq. (D1) in the second line. This term induces a variability in the coefficient updates, as $\langle x^{\otimes k}\rangle_{\mathcal{D}_t}$ will vary from batch to batch due to the finite size of each batch. We define the random variable $\xi^{(k)}_t = \langle x^{\otimes k}\rangle_{\mathcal{D}_t} - \langle x^{\otimes k}\rangle_{\mathcal{D}}$ to be the deviation between the moment estimated on the current batch $\mathcal{D}_t$ and the moment estimated on the entire dataset $\mathcal{D}$. Combining Eqs. (D3) and (D2) yields Eq. (32).

APPENDIX E: RANDOM GENERATION OF MULTIMODAL ACTIONS

1. Coefficient distributions for random actions

In Sec. IV B we use multimodal actions S_R constructed from randomly drawn coefficients $R^{(k)}$. A basic condition these coefficients must satisfy is that the resulting action be normalizable: $\int S_R(x)dx < \infty$. We note that for large enough x, the action is dominated by the highest-order terms:

$$S(x) \underset{\|x\to\infty\|}{\longrightarrow} (R^{(4)}) \cdot x^{\otimes 4}.$$

It is therefore necessary and sufficient for normalizability that $R^{(4)}$ be negative definite, which we ensure by choosing $R^{(4)}$ to be a diagonal tensor with negative coefficients:

$$R^{(4)}_{i_1 i_2 i_3 i_4} = \begin{cases} -\frac{x_r^{-4}}{d} & \text{if } i_1 = i_2 = i_3 = i_4 \\ 0 & \text{otherwise.} \end{cases} \tag{E1}$$

Here d is the dimensionality of the data, and $x_r \in \mathbb{R}$ is a length scale which we are free to choose; one can view $R^{(4)}$ as a regulator term, and x_r as the value for which it becomes strongly suppressing. For our experiments we used $x_r = 1.0$.

Having ensured that the action is normalizable, we can define the probability $p_R(x|\{R^{(k)}\}_{k\leq 4}) = \exp[S_R(x)]/\int \exp[S_R(x)]dx$. We choose the S_R such that the data can be described as a perturbation of a Gaussian theory—we therefore also choose $R^{(2)}$ to be negative definite (i.e., a valid precision matrix). Since any d-dimensional multivariate Gaussian can be written as a linear combination of d independent Gaussian variables, we define $R^{(2)}$ as follows:

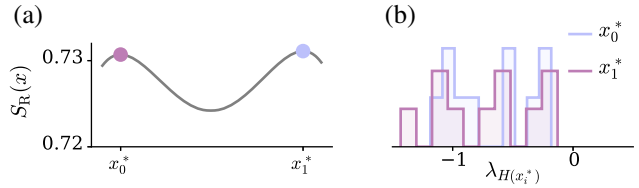


FIG. 10. Multiple local maxima in S_R. (a) S_R along the straight line connecting two local maxima x_0^*, x_1^* of S_R found by the optimization algorithm. (b),(c) Eigenvalues of the Hessian H of S_R at local maxima x_0^*, x_1^*. All eigenvalues $\lambda_{H(x_i^*)}$ are negative; therefore the action is convex down in all directions.

$$W_{ij} \sim \mathcal{N}(0, 1/d^2), \quad i, j = 1, \ldots d,$$
$$R^{(2)}_{ij} := c - \frac{1}{2}\sum_a W_{ia} W_{ja}, \tag{E2}$$

with $c = -0.1$ in our experiments. This is equivalent to transforming a Gaussian variable $z \sim \mathcal{N}(0, \mathbb{1})$ by a linear transform $x = W^{-1}z$ and then computing the action of x [compare to Eq. (18)].

Finally, the coefficients $R^{(1)}$ and $R^{(3)}$ are chosen as follows $(i, j, k = 1, \ldots d)$:

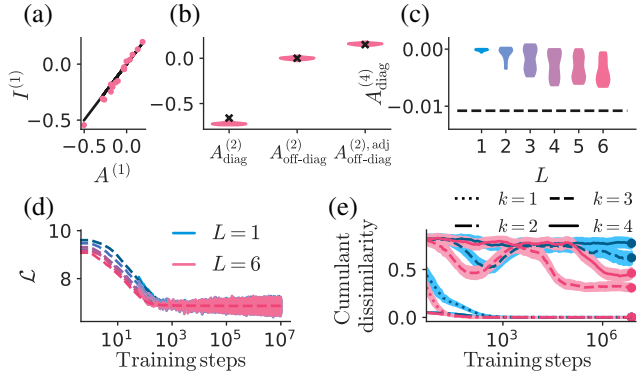


FIG. 11. Coefficients of lattice model for networks of varying depth trained on a $d = 16$ dimensional dataset with $D = 10^5$ samples. (a) Learned $(L^{(1)})$ over true $(A^{(1)})$ first-order coefficients. (b) Distribution of learned coefficient entries $A^{(2)}$ compared to target values (black crosses). Self-interaction terms are labeled $A^{(2)}_{\text{diag}}$, off-diagonal entries $A^{(2)}_{\text{off-diag}}$. Among the off-diagonal entries $A^{(2)}_{\text{off-diag}}$, entries belonging to adjacent lattice sites $A^{(2),\text{adj}}_{\text{off-diag}}$ are shown separately. (c) Training loss (full curves) and test loss (dashed curves). Colors distinguish different network depths L. (d) Distribution of learned fourth-order self-interactions over network depth. The dashed line marks the target value. (e) Dissimilarity of true and learned cumulants: $1 - \cos\angle(\langle\langle x^{\otimes k}\rangle\rangle_A, \langle\langle x^{\otimes k}\rangle\rangle_R)$ over training steps. We record the cumulants at logarithmically spaced intervals during training. The curves are smoothed by averaging over ten adjacent recording steps. Shaded areas show the variation due to the estimation of the cumulants from samples. Dots indicate training stage of coefficients shown in (a) and (b).

$$R^{(1)}_i \sim \mathcal{N}(0, \sigma_i^2), \qquad \sigma_i = \frac{x_r^{-1}}{d},$$
$$R^{(3)}_{ijk} \sim \mathcal{N}(0, \sigma_{ijk}^2), \qquad \sigma_{ijk} = \frac{x_r^{-3}}{s_{ijk}\gamma_{ijk}}.$$

The scaling with respect to x_r^{-1} and x_r^{-3} ensures that neither linear nor cubic terms are negligible within the region where the regulator term $R^{(4)}$ is nonsuppressing. The variable $\gamma_\alpha = |\mathcal{P}(\alpha)|$ in the denominator of σ_α is the multiplicity of the index α. This is the number of times the component $R^{(3)}_{ijk}$ appears in $R^{(3)}$; since coefficients are symmetric, it is equal to the number of distinct permutations of (i, j, k). We scale the multiplicity by s_α, the number of different components which have the same number of permutations—for example, the permutations of the indices (2, 1, 1) and (5, 3, 3) appear $\gamma_{ijj} = 3$ times each, and there are $s_{ijj} = d(d-1)$ distinct entries of such indices. Scaling σ_α by γ_α and s_α ensures that both the on-diagonal and off-diagonal components of $R^{(2)}, R^{(3)}$ are significant.

2. Multimodality of random actions

We here provide evidence that the distribution in Sec. IV B is indeed multimodal. To do so, we initialize an optimization algorithm at random points and attempt to

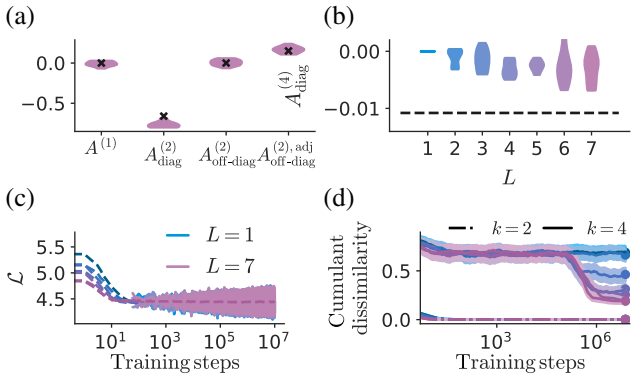


FIG. 12. Coefficients of lattice model without external field for networks of varying depth trained on a $d = 9$ dimensional dataset with $D = 10^5$ samples. (a) Distribution of learned coefficient entries $A^{(1)}, A^{(2)}$ compared to target values (black crosses). Self-interaction terms $A^{(2)}_{\text{diag}}$ are shown separately from off-diagonal entries $A^{(2)}_{\text{off-diag}}$. Among the off-diagonal entries $A^{(2)}_{\text{off-diag}}$, those entries belonging to adjacent lattice sites $A^{(2),\text{adj}}_{\text{off-diag}}$ are shown separately. (b) Distribution of learned fourth-order self-interactions compared to network depth. The dashed line marks the target value. (c) Training loss (full curves) and test loss (dashed curves). Colors distinguish different network depths L. (d) Dissimilarity of true and learned cumulants: $1 - \cos\angle(\langle\langle x^{\otimes k}\rangle\rangle_A, \langle\langle x^{\otimes k}\rangle\rangle_R)$ over training steps. We record the cumulants at logarithmically spaced intervals during training. The curves are smoothed by averaging over ten adjacent recording steps. Shaded areas show the variation due to the estimation of the cumulants from samples. Dots indicate training stage of coefficients shown in (a) and (b).

find the maximum of $S_R(x^*)$. We initialized at 10^3 different values. In Fig. 10 we show the maximal values of $S_R(x^*)$ found by the algorithm as well as the eigenvalues of the Hessian for selected, distinct final values x^*. Since all eigenvalues are below zero, the action S_R is locally convex down in all directions at both points. The action S_R therefore has at least two local maxima.

APPENDIX F: SAMPLING FROM ACTIONS WITH A MCMC SAMPLER

Given ground truth coefficients $T^{(k)}$, we create a dataset $\mathcal{D}$ by drawing samples according to the probability $p_C(x|\{T^{(k)}\}_{k\leq 4})$. Markov chain Monte Carlo sampling is well suited to this task since it requires only the unnormalized log probability, i.e., $S_C(x)$, and is guaranteed to converge to the true distribution (in contrast to variational methods like automatic differentiation variational inference [72]). To generate $\mathcal{D}$, we used the no-U-turn sampler [73] implementation provided by PyMC3 [25]; sampler parameters mostly followed recommended defaults, with 10^3 tuning steps and a mass matrix initialized to unity. The target acceptance rate was increased to 0.95 to increase the sensitivity to small features of the probability distribution.

APPENDIX G: LATTICE MODEL IN LOW DIMENSIONS

We here show a lower-dimensional version of the lattice model introduced in Sec. IV C.

Here, the combined number of independent entries in the first four action coefficients is only 4844, which corresponds to roughly 6 network layers. Figure 11 shows a comparison of true vs learned coefficients. Independent of network depth, we find that $A^{(1)}$ and the off-diagonal entries $A^{(2)}_{ij}$ with $i \neq j$ are recovered correctly [see Figs. 11(a) and 11(b)]. For shallow networks, the diagonal entries in the fourth-order coefficient $A^{(4)}_{\text{diag}}$ are approximately zero. Their magnitudes increase with L [see Fig. 11(d)]. Increasing the depth L also speeds up learning [see Fig. 11(c)]. Furthermore, we find that up to the fourth order, the cumulants of the learned distribution increasingly align with those of the true distribution as we increase network depth. Therefore we can conclude that increasing the depth of the network increases the accuracy of the learned distribution, both in terms of its coefficients and of its cumulants.

We repeated the experiment for $d = 9$ without any heterogeneous external field, therefore $h = 0$. Again, we find an alignment of most entries in $A^{(1)}, A^{(2)}$, with $A^{(2)}_{\text{diag}}$

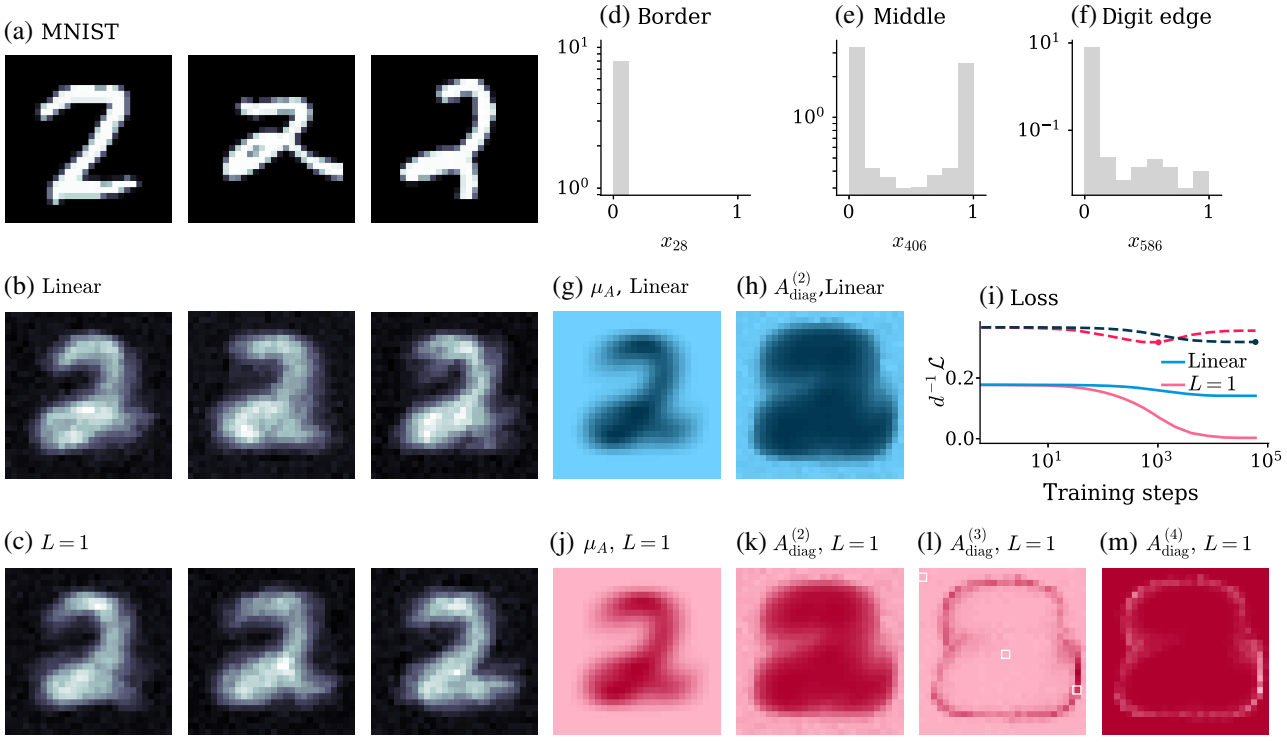


FIG. 13. Inference of interactions on MNIST for digit two. (a)–(c) Images from the dataset, the linear model, and an $L = 1$ layer nonlinear model, respectively. (d)–(f) Single pixel activation statistics from three distinct locations in the image. (g) Entries of the mean μ_A of the Gaussian theory (linear model). (h) Entries on the diagonal of the second-order coefficient $A^{(2)}_{\text{diag}}$ of the linear model. (i) Training loss (full lines) and test loss (dashed lines) over training steps. Dots mark the training stages from which the coefficients of both models were extracted. (j) Mean μ_A for the nonlinear model if $A^{(3)}, A^{(4)}$ were not present. (k)–(m) Entries on the diagonals of the remaining coefficients of the $L = 1$ layer nonlinear model. White squares in (l) mark the locations of the single pixel statistics shown in (d)–(f).

slightly lower than expected and $A^{(4)}_{\text{diag}}$ larger than expected [see Figs. 12(a) and 12(b)]. In this setting, the symmetry of the action causes the first and third cumulant to vanish. We therefore only compute the alignment of the second and fourth cumulants. As in the previous cases, this alignment increases with depth, as shown in Fig. 12(d). Therefore, the effective nature of the learned theory does not depend on the system's symmetry being broken.

APPENDIX H: TRAINING ON MNIST DIGITS

It is well known that the MNIST data lie on a lower-dimensional manifold, as a principal component analysis of the MNIST dataset confirms: several eigenvalues of the covariance matrices of the MNIST dataset are small $\lambda^{\text{MNIST}}_{\text{min}} \sim 10^{-28}$. This aspect of the dataset can lead to diverging eigenvalues in the Jacobian of the INN (see, e.g., Refs. [74,75] for a detailed treatment). However, we find that this problem can be mitigated in two steps. In the first step, we add a small, IID Gaussian noise ξ with mean zero and variance $\sigma^2_\xi = 10^{-2}$ to each pixel value in the dataset. The small noise ensures that the eigenvalues of the covariance matrix are all of order σ^2_ξ or larger. In the second step, we perform a full principal component (PC) decomposition on this noised dataset, retaining all principal components. We then train a network on the data in PC space. As PC decomposition is a linear mapping, we can compute the action coefficients by including this linear transform as a final step in the transformation of coefficients obtained from the trained network. As we have a limited number of samples, we also add a small IID Gaussian noise ξ_{train} with variance $\sigma^2_{\xi_{\text{train}}} = 10^{-2}$ to each training batch to prevent overfitting. We find that this procedure both prevents divergences of the training loss during training as well as speeds up the training process.

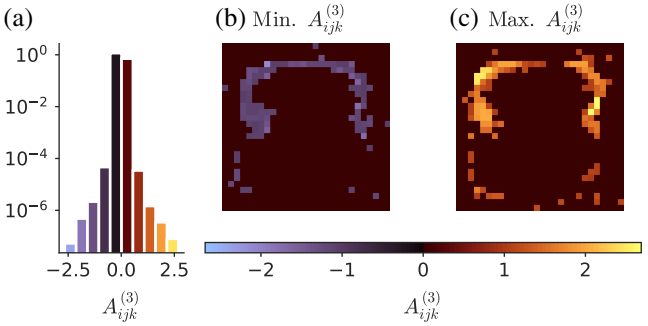


FIG. 15. Three-point interactions in MNIST for digit three. (a) Histogram of all entries of the third-order coefficient $A^{(3)}_{ijk}$ for $i \neq j$, $j \neq k$, and $i \neq k$, color coded according to their value. Panels (b) and (c) show triplets corresponding to the 10^2 most negative (b) or most positive (c) values of $A^{(3)}_{ijk}$. For each triplet, we color pixels i, j, and k, according to the value of the interaction coefficient in $A^{(3)}_{ijk}$. Thus triplets of pixels corresponding to the same entry in $A^{(3)}$ have the same color.

We showcase the detection of edges also for the digit two in Fig. 13. Again, we find an increased magnitude in $A^{(3)}_{\text{diag}}$ and $A^{(4)}_{\text{diag}}$ along the location of the typical digit edges in Fig. 13. The three-point interactions in $A^{(3)}_{ijk}$ for the digit two are shown in Fig. 14.

Finally, in Figs. 15 and 16, we show the first 100 most positive and most negative three-point interactions of digits two and three. This shows that the higher-order interactions consistently trace the edges of the digits.

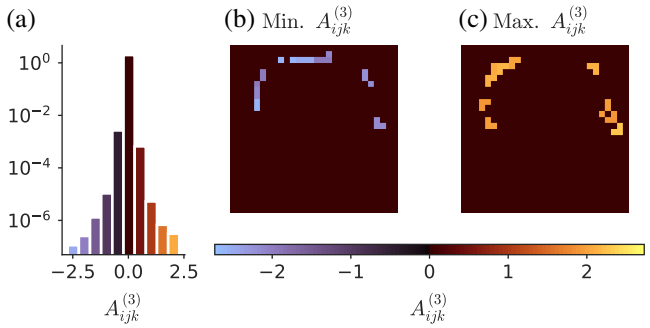


FIG. 14. Three-point interactions in MNIST for digit two. (a) Histogram of all entries of the third-order coefficient $A^{(3)}_{ijk}$ for $i \neq j$, $j \neq k$, and $i \neq k$, color coded according to their value. (b), (c) Triplets corresponding to the ten most negative (b) or most positive (c) values of $A^{(3)}_{ijk}$. For each triplet, we color pixels i, j, and k, according to the value of the interaction coefficient in $A^{(3)}_{ijk}$. Thus triplets of pixels corresponding to the same entry in $A^{(3)}$ have the same color.

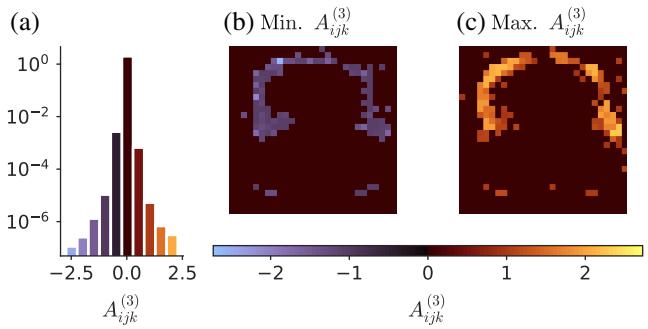


FIG. 16. Three-point interactions in MNIST for digit two. (a) Histogram of all entries of the third-order coefficient $A^{(3)}_{ijk}$ for $i \neq j$, $j \neq k$, and $i \neq k$, color coded according to their value. Panels (b) and (c) show triplets corresponding to the 10^2 most negative (b) or most positive (c) values of $A^{(3)}_{ijk}$. For each triplet, we color pixels i, j, and k, according to the value of the interaction coefficient in $A^{(3)}_{ijk}$. Thus triplets of pixels corresponding to the same entry in $A^{(3)}$ have the same color.

[1] S. Lee and J. Jo, *Scale-invariant representation of machine learning*, Phys. Rev. E **105**, 044306 (2022).

[2] U. Cohen, S. Chung, D. D. Lee, and H. Sompolinsky, *Separability and geometry of object manifolds in deep neural networks*, Nat. Commun. **11**, 746 (2020).

[3] O. Duranthon, M. Marsili, and R. Xie, *Maximal relevance and optimal learning machines*, J. Stat. Mech. (2021) 033409.

[4] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová, *Modeling the influence of data structure on learning in neural networks: The hidden manifold model*, Phys. Rev. X **10**, 041044 (2020).

[5] M. Refinetti, A. Ingrosso, and S. Goldt, *Neural networks trained with SGD learn distributions of increasing complexity*, arXiv:2211.11567.

[6] K. Fischer, A. René, C. Keup, M. Layer, D. Dahmen, and M. Helias, *Decomposing neural networks as mappings of correlation functions*, Phys. Rev. Res. **4**, 043143 (2022).

[7] J. Tubiana and R. Monasson, *Emergence of compositional representations in restricted Boltzmann machines*, Phys. Rev. Lett. **118**, 138301 (2017).

[8] L. Xiao and J. Pennington, *Synergy and symmetry in deep learning: Interactions between the data, model, and inference algorithm*, in *Proceedings of the 39th International Conference on Machine Learning, Baltimore* (PMLR, 2022), pp. 24347–24369.

[9] L. Dinh, D. Krueger, and Y. Bengio, *NICE: Non-linear independent components estimation*, arXiv:1410.8516.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using Real NVP*, arXiv:1605.08803.

[11] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, *Analyzing inverse problems with invertible neural networks*, arXiv:1808.04730.

[12] R. E. Odstrcil, P. Dutta, and J. Liu, *LINES: Log-probability estimation via invertible neural networks for enhanced sampling*, J. Chem. Theory Comput. **18**, 6297 (2022).

[13] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible* 1×1 *convolutions*, in *Advances in Neural Information Processing Systems*, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., Montréal, 2018).

[14] N. Goldenfeld, *Lectures on Phase Transitions and the Renormalization Group* (Perseus Books, Reading, MA, 1992).

[15] S. Wessel, S. Trebst, and M. Troyer, *A renormalization approach to simulations of quantum effects in nanoscale magnetic systems*, Multiscale Model. Simul. **4**, 237 (2005).

[16] A. Cuccoli, R. Giachetti, V. Tognetti, R. Vaia, and P. Verrucchi, *The effective potential and effective Hamiltonian in quantum statistical mechanics*, J. Phys. Condens. Matter **7**, 7891 (1995).

[17] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, J. Mach. Learn. Res. **22**, 2617 (2021), https://jmlr.org/papers/v22/19-1028.html.

[18] D. J. Amit, *Field Theory, The Renormalization Group, and Critical Phenomena* (World Scientific, Singapore, 1984).

[19] H. Cramér, Mathematical Methods of Statistics (PMS-9), Vol. **9** (Mon, 04/12/1999–12:00), https://www.jstor.org/stable/j.ctt1bpm9r4.

[20] A. Krasilnikov, V. Beregun, and O. Harmash, *Analysis of estimation errors of the fifth and sixth order cumulants*, in *Proceedings of the IEEE 39th International Conference on Electronics and Nanotechnology (ELNANO), 2019* (IEEE, Kiev, 2019), pp. 754–759, https://ieeexplore.ieee.org/document/8783910.

[21] V. Beregun and O. Harmash, *Application of cumulant coefficients for solving the problems of testing and diagnostics in control systems*, in *Proceedings of the IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), 2018* (IEEE, Kiev, 2018), pp. 210–213, https://ieeexplore.ieee.org/document/8576176.

[22] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid Monte Carlo*, Phys. Lett. B **195**, 216 (1987).

[23] R. M. Neal, *MCMC using Hamiltonian dynamics*, arXiv:1206.1901.

[24] M. Betancourt, *A conceptual introduction to Hamiltonian Monte Carlo*, arXiv:1701.02434.

[25] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, *Probabilistic programming in PYTHON using PyMC3*, PeerJ Comput. Sci. **2**, e55 (2016).

[26] L. Deng, *The MNIST database of handwritten digit images for machine learning research*, IEEE Signal Process. Mag. **29**, 141 (2012).

[27] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proc. Natl. Acad. Sci. U.S.A. **113**, 3932 (2016).

[28] J. Casadiego, M. Nitzan, S. Hallerberg, and M. Timme, *Model-free inference of direct network interactions from nonlinear collective dynamics*, Nat. Commun. **8**, 2192 (2017).

[29] J. Miller, S. Tang, M. Zhong, and M. Maggioni, *Learning theory for inferring interaction kernels in second-order interacting agent systems*, arXiv:2010.03729.

[30] F. Lu, M. Maggioni, and S. Tang, *Learning interaction kernels in heterogeneous systems of agents from multiple trajectories*, arXiv:1910.04832.

[31] J. Jordan, M. Schmidt, W. Senn, and M. A. Petrovici, *Evolving interpretable plasticity for spiking networks*, eLife **10**, e66273 (2021).

[32] B. Confavreux, F. Zenke, E. Agnes, T. Lillicrap, and T. Vogels, *A meta-learning approach to (re)discover plasticity rules that carve a desired function into a neural network*, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., Vancouver, 2020), pp. 16398–16408.

[33] M. Opper and G. Sanguinetti, *Variational inference for Markov jump processes Advances in Neural Information Processing Systems*, Vol. 20 (Curran Associates, Inc., Vancouver, 2007).

[34] M. Opper, *Variational inference for stochastic differential equations*, Ann. Phys. (Berlin) **531**, 1800233 (2019).

[35] M. D. Vrettas, M. Opper, and D. Cornford, *Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations*, Phys. Rev. E **91**, 012148 (2015).

[36] A. Ruttor and M. Opper, *Efficient statistical inference for stochastic reaction processes*, Phys. Rev. Lett. **103**, 230601 (2009).
[37] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *A learning algorithm for Boltzmann machines*, Cogn. Sci. **9**, 147 (1985).
[38] M. Mezard and J. Sakellariou, *Exact mean field inference in asymmetric kinetic Ising systems*, J. Stat. Mech. (2011) L07002.
[39] E. Schneidman, M. J. Berry, R. Segev, and W. Bialek, *Weak pairwise correlations imply strongly correlated network states in a neural population*, Nature (London) **440**, 1007 (2006).
[40] H.-L. Zeng, E. Aurell, M. Alava, and H. Mahmoudi, *Network inference using asynchronously updated kinetic Ising model*, Phys. Rev. E **83**, 041135 (2011).
[41] E. Agliari, P. J. Sáez, A. Barra, M. Piel, P. Vargas, and M. Castellana, *A statistical inference approach to reconstruct intercellular interactions in cell migration experiments*, Sci. Adv. **6**, eaay2103 (2020).
[42] A. N. Vasiliev and R. A. Radzhabov, *Legendre transforms in the Ising model*, Theor. Math. Phys. **21**, 963 (1974).
[43] D. J. Thouless, P. W. Anderson, and R. G. Palmer, *Solution of 'solvable model of a spin glass'*, Philos. Mag., **35**, 593 (1977).
[44] J. Sakellariou, Y. Roudi, M. Mezard, and J. Hertz, *Effect of coupling asymmetry on mean-field solutions of direct and inverse Sherrington-Kirkpatrick model*, Philos. Mag. **92**, 272 (2012).
[45] H.-L. Zeng, E. Aurell, M. Alava, and H. Mahmoudi, *Network inference using asynchronously updated kinetic Ising model*, Phys. Rev. E **83**, 041135 (2011).
[46] Y. Roudi and J. Hertz, *Mean field theory for nonequilibrium network reconstruction*, Phys. Rev. Lett. **106**, 048702 (2011).
[47] C. Baldassi, F. Gerace, L. Saglietti, and R. Zecchina, *From inverse problems to learning: A statistical mechanics approach*, J. Phys. Conf. Ser. **955**, 012001 (2018).
[48] A. Decelle, S. Hwang, J. Rocchi, and D. Tantari, *Inverse problems for structured datasets using parallel TAP equations and restricted Boltzmann machines*, Sci. Rep. **11**, 19990 (2021).
[49] S. Cocco, R. Monasson, and V. Sessak, *High-dimensional inference with the generalized Hopfield model: Principal component analysis and corrections*, Phys. Rev. E **83**, 051123 (2011).
[50] I. Mastromatteo, *Beyond inverse Ising model: Structure of the analytical solution*, J. Stat. Phys. **150**, 658 (2013).
[51] A. Braunstein, A. Ingrosso, and A. P. Muntoni, *Network reconstruction from infection cascades*, J. R. Soc. Interface **16**, 20180844 (2019).
[52] T. Mora, A. M. Walczak, W. Bialek, and C. G. Callan, *Maximum entropy models for antibody diversity*, Proc. Natl. Acad. Sci. U.S.A. **107**, 5405 (2010).
[53] E. Aurell and M. Ekeberg, *Inverse Ising inference using all the data*, Phys. Rev. Lett. **108**, 090201 (2012).
[54] A. Abbara, Y. Kabashima, T. Obuchi, and Y. Xu, *Learning performance in inverse Ising problems with sparse teacher couplings*, J. Stat. Mech. (2020) 073402.
[55] A. Y. Lokhov, M. Vuffray, S. Misra, and M. Chertkov, *Optimal structure and parameter learning of Ising models*, Sci. Adv. **4**, e1700791 (2018).
[56] J. Berg, *Statistical mechanics of the inverse Ising problem and the optimal objective function*, J. Stat. Mech. (2017) 083402.
[57] L. Bachschmid-Romano and M. Opper, *A statistical physics approach to learning curves for the inverse Ising problem*, J. Stat. Mech. (2017) 063406.
[58] A. Murugan, T. Mora, A. M. Walczak, and C. G. Callan, *Statistical inference of the generation probability of T-cell receptors from sequence repertoires*, Proc. Natl. Acad. Sci. U.S.A. **109**, 16161 (2012).
[59] T. V. Zache, T. Schweigler, S. Erne, J. Schmiedmayer, and J. Berges, *Extracting the field theory description of a quantum many-body system from experimental data*, Phys. Rev. X **10**, 011020 (2020).
[60] P. J. Gonçalves, J.-M. Lueckmann, M. Deistler, M. Nonnenmacher, K. Öcal, G. Bassetto, C. Chintaluri, W. F. Podlaski, S. A. Haddad, T. P. Vogels, D. S. Greenberg, and J. H. Macke, *Training deep neural density estimators to identify mechanistic models of neural dynamics*, eLife **9**, e56261 (2020).
[61] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, *Analyzing inverse problems with invertible neural networks*, arXiv:1808.04730.
[62] D. E. Gökmen, Z. Ringel, S. D. Huber, and M. Koch-Janusz, *Statistical physics through the lens of real-space mutual information*, Phys. Rev. Lett. **127**, 240603 (2021).
[63] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, *Discovering symbolic models from deep learning with inductive biases*, in *Advances in Neural Information Processing Systems* (Ref. [32]), pp. 17429–17442.
[64] Q. Li, C. Tai, and W. E, *Stochastic modified equations and adaptive stochastic gradient algorithms*, in *Proceedings of the 34th International Conference on Machine Learning, Sydney* (PMLR, 2017), pp. 2101–2110.
[65] P. Chaudhari and S. Soatto, *Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks*, arXiv:1710.11029.
[66] F. Mignacco, F. Krzakala, P. Urbani, and L. Zdeborová, *Dynamical mean-field theory for stochastic gradient descent in Gaussian mixture classification*, J. Stat. Mech. (2021) 124008.
[67] D. Krotov and J. J. Hopfield, *Dense associative memory for pattern recognition*, in *Advances in Neural Information Processing Systems*, Vol. 29 (Curran Associates, Inc., Barcelona, 2016).
[68] P. Baldi and S. S. Venkatesh, *Number of stable points for spin-glasses and neural networks of higher orders*, Phys. Rev. Lett. **58**, 913 (1987).
[69] E. Agliari and G. D. Marzo, *Tolerance versus synaptic noise in dense associative memories*, Eur. Phys. J. Plus **135**, 883 (2020).
[70] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. **13**, 354 (1969).
[71] M. D. Schatz, T. M. Low, R. A. van de Geijn, and T. G. Kolda, *Exploiting symmetry in tensors for high performance: multiplication with symmetric tensors*, SIAM J. Sci. Comput. **36**, C453 (2014).

[72] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, *Automatic differentiation variational inference*, J. Mach. Learn. Res. **18**, 1 (2017), https://proceedings.mlr.press/v130/morningstar21b.html.

[73] M. D. Hoffman and A. Gelman, *The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo*, J. Mach. Learn. Res. **15**, 1593 (2014), https://jmlr.org/papers/volume15/hoffman14a/hoffman14a.pdf.

[74] G. Loaiza-Ganem, B. L. Ross, J. C. Cresswell, and A. L. Caterini, *Diagnosing and fixing manifold overfitting in deep generative models*, arXiv:2204.07172.

[75] R. Cornish, A. L. Caterini, G. Deligiannidis, and A. Doucet, *Relaxing bijectivity constraints with continuously indexed normalising flows*, in *Proceedings of the International Conference on Machine Learning, Long Beach, 2019* (PMLR, 2019).