Neural Network Renormalization Group

Shuo-Hui Li[1,2] and Lei Wang[1,*]

[1]*Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*
[2]*University of Chinese Academy of Sciences, Beijing 100049, China*

We present a variational renormalization group approach using deep generative model composed of bijectors. The model can learn hierarchical transformations between physical variables and renormalized collective variables. It can directly generate statistically independent physical configurations by iterative refinement at various length scales. The generative model has an exact and tractable likelihood, which provides renormalized energy function of the collective variables and supports unbiased rejection sampling of the physical variables. To train the neural network, we employ probability density distillation, in which the training loss is a variational upper bound of the physical free energy. The approach could be useful for automatically identifying collective variables and effective field theories.

Renormalization group (RG) is one of the central schemes in theoretical physics, whose broad impacts span from high-energy [1] to condensed matter physics [2, 3]. In essence, RG keeps the relevant information while reducing the dimensionality of statistical data. Besides its conceptual importance, practical RG calculations have played important roles in solving challenging problems in statistical and quantum physics [4, 5]. A notable recent development is to perform RG calculation using tensor network machineries [6–17]

The relevance of RG also goes beyond physics. For example, in deep learning applications, the inference process in image recognition resembles the RG flow from microscopic pixels to categorical labels. Indeed, a successfully trained deep neural network extracts a hierarchy of increasingly higher-level of concepts in its deeper layers [18]. In light of such intriguing similarities, References [19–22] drew connections between deep learning and RG. References [23, 24] employed Boltzmann Machines for RG studies of physical problems, and Refs. [25–27] investigated phase transitions from the machine learning perspective. Since the discussions are not totally uncontroversial [20, 22, 23, 28, 29], it remains highly desirable to establish a more concrete, rigorous, and constructive connection between RG and deep learning. Such connection will not only bring powerful deep learning techniques into solving complex physics problems but also benefit theoretical understanding of deep learning from a physics perspective.

In this paper, we present a neural network based variational RG approach (NeuralRG) for statistical physics problems. In this scheme, the RG flow arises from iterative probability transformation in a deep neural network. Integrating latest advances in deep learning including *Normalizing Flows* [30–37] and *Probability Density Distillation* [38] and tensor network architectures, in particular the multi-scale entanglement renormalization ansatz (MERA) [6], the proposed NeuralRG approach has a number of interesting theoretical properties (variational, exact and tractable likelihood, principled structure design via information theory) and high computational efficiency. The NeuralRG approach is closer in spirit to the original proposal based on Bayesian net [19] than more recent discussions on Boltzmann Machines [20, 22, 23] and Principal Component Analysis [21].

Figure 1(a) shows the proposed neural net architecture.

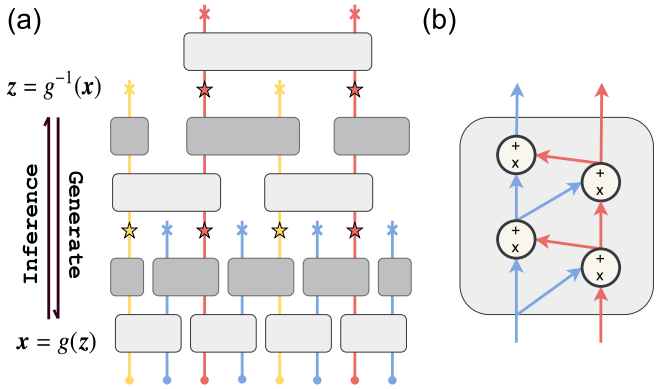

Figure 1. (a) The NeuralRG network is formed by stacking bijector networks into a hierarchical structure. The solid dots at the bottom are the physical variables $\boldsymbol{x}$ and the crosses are the latent variables z. The stars denote the renormalized collective variables at various scales. Each block is a bijective and differentiable transformation parametrized by a bijector neural network. The light gray and the dark gray blocks are the disentanglers and the decimators respectively. The RG flows from bottom to top, which corresponds to inference the latent variables based on physical variables. Conversely, by sampling the latent variables according to the prior distribution and passing them downwards one can generate the physical configuration directly. (b) The internal structure of the bijector block consists of a real-valued non-volume preserving flow [33].

Each building block is a diffeomorphism, i.e., a bijective and differentiable function parametrized by a neural network, denoted by a bijector [39, 40]. Figure 1(b) illustrates a possible realization of the bijector using the real-valued non-volume preserving flow (Real NVP) [33] [41], which is one of the simplest invertible neural networks with efficiently tractable Jacobian determinants known as normalizing flows [30–37].

The neural network relates the physical variables $\boldsymbol{x}$ and the latent variables $\boldsymbol{z}$ via a differentiable bijective map $\boldsymbol{x} = g(\boldsymbol{z})$. Their probability densities are also related [42]

$$\ln q(\boldsymbol{x}) = \ln p(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|, \tag{1}$$

where $q(\boldsymbol{x})$ is the normalized probability density of the physical variables. And $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{1})$ is the prior probability

density of the latent variables chosen to be a normal distribution. The second term of Eq. (1) is the log-Jacobian determinant of the bijective transformation. Since the log-probability can be interpreted as a negative energy function, Eq. (1) shows that the renormalization of the effective coupling is provided by the log-Jacobian at each transformation step.

Since diffeomorphisms form a group, an arbitrary composition the building blocks is still a bijector. This motivates modular design of the network structure shown in Fig. 1(a). The layers alternate between disentangler blocks and decimator blocks. The disentangler blocks shown in light gray reduce correlation between the inputs and pass on less correlated outputs to the next layer. While the decimator blocks in dark gray pass only parts of outputs to the next layer and treat the remaining ones as irrelevant latent variables indicated by the crosses. The RG flow corresponds to inference of the latent variables based on observed physical variables, $z = g^{-1}(\boldsymbol{x})$. The kept degrees of freedom emerge as renormalized collective variables at coarser scales during the inference, denoted by the stars. In the reversed direction, the latent variables are injected into the neural network at different depths. And they affect the physical variables at different length scales.

The bijective property is crucial for learning the RG flow in a controlled way. No matter how complex is the hierarchical transformations performed by the neural network, one can efficiently compute the normalized probability density $q(\boldsymbol{x})$ for either given or sampled physical configuration $\boldsymbol{x}$ by keeping track of the Jacobian determinant of each block locally. One can let the blocks in the same layer share weights due to the translational invariances of the physical problem. Moreover, one can even share weights in the depth direction due to scale invariance emerged at criticality. The scale-invariant reduces the number of parameters to be independent of the system size [43].

The proposed NeuralRG architecture shown in Fig. 1(a) is largely inspired by the MERA structure [6]. In particular, stacking bijectors to transform the probability densities is also analogous to the interpretation of MERA as a reversible quantum circuit. The difference is that the neural network transforms between probability densities instead of quantum states. Compared to the tensor networks, the neural network ,however, has the flexibility that the blocks can be arbitrarily large and long-range connected. Moreover, thanks to the modularity, arbitrary complex NeuralRG architecture can be learned efficiently using standard differential approaches offered in modern deep learning frameworks [44, 45].

Compared to ordinary neural networks used in deep learning, the architecture shown in Fig 1(a) has stronger physical and information theoretical motivations. To see this, we consider a simpler reference structure shown in Fig. 2(a) where one uses disentangler blocks at each layer. The resulting structure resembles a time-evolving block decimation network [46]. Since each disentangler block connects only a few neighboring variables, the causal light cone of the physical variables at the bottom can only reach a region of latent variables proportional to the depth of the network. Therefore, the correlation length of the physical variables is limited by the depth of the disentangler layers. The structure Fig. 2(a) is sufficient for physical problems with finite correlation length, i.e. away from the criticality.

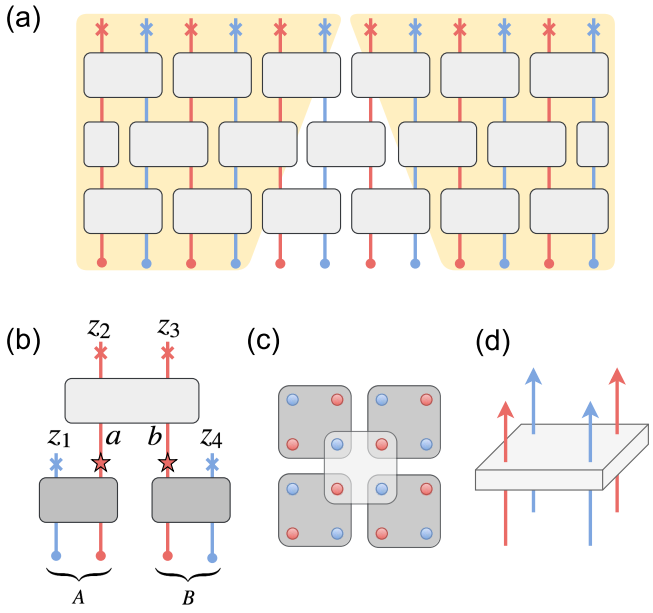

Figure 2. (a) A reference neural network architecture with only disentanglers. The physical variables in the two shaded regions are uncorrelated because their causal light cones do not overlap in the latent space. (b) Mutual information is conserved at the decimation step, see Eq. (2). (c) The arrangement of the bijectors in the two-dimensional space. (d) Each bijector acts on four variables. Disentanglers tries to remove correlations between the variables. While for decimators, only one of its outputs is carried on to the next layer and the others are treated directly as latent variables.

On the other hand, a network formed only by the decimators is similar to the tree tensor network [47]. As shown in Fig. 2(b), the mutual information (MI) between the variables at each decimation step follows

$$I(A:B) = I(z_1 \cup a : b \cup z_4) = I(a:b). \tag{2}$$

The first equality is due to that the mutual information is invariant under invertible transformation of variables within each group. While the second equality is due to the random variables z_1 and z_4 are independent of all other variables. Applying Eq. (2) recursively at each decimation step, one concludes that the MI between two sets of physical variables is limited by the top layer in a neural net of the tree structure. This may not impose a constraint on the expressibility of the network since the MI between two continuous variables can be arbitrarily large in principle. However, a decimator only structure could be limited in practice since it is rather unphysical to carry the MI between two extensive regions with only two variables [48].

The NeuralRG architecture is flexible to handle data in higher dimensional space. For example, one can stack layers of bijectors in the form of Fig. 2(c). These bijectors accept

2×2 inputs as shown in Fig. 2(d). For the decimator, only one of the outputs is passed on to the next layer. In a network with only disentanglers, the depth should scale with the linear system size to capture diverging correlation length at criticality. While the required depth only scales logarithmically with the linear system size if one employs the MERA-like structure. Note that different from the tensor network modeling of quantum states [49], the MERA-like architecture is sufficient to model classical systems with short-range interactions even at criticality since they exhibit the mutual information area law [50].

The neural network shown in Fig. 1 defines a class of generative models with explicit and tractable likelihood Eq. (1). In principle, one can train it using the standard maximum likelihood estimation [42] on a dataset of physical configurations. However, different from typical machine learning tasks, one usually does not have direct access to statistically independent data in physics. Having access to the *unnormalized* probability density $\pi(\boldsymbol{x})$, sampling is generally difficult because of the intractable partition function $Z = \int \mathrm{d}\boldsymbol{x}\, \pi(\boldsymbol{x})$ and exponentially small probability densities in high dimensions [51]. A typical workaround is to employ the Markov chain Monte Carlo (MCMC) approach, in which only ratios between unnormalized probability densities are used to collect samples [52]. However, sampling using MCMC can suffer from long autocorrelation times and result in correlated samples [53].

We train the NeuralRG network by minimizing the *Probability Density Distillation* loss

$$\mathcal{L} = \int \mathrm{d}\boldsymbol{x}\, q(\boldsymbol{x}) \left[\ln q(\boldsymbol{x}) - \ln \pi(\boldsymbol{x})\right], \tag{3}$$

which was recently employed by DeepMind to train parallel WaveNet [38]. The first term of the loss is the negative entropy of the model density $q(\boldsymbol{x})$, which favors diversity in its samples. Since $-\ln \pi(\boldsymbol{x})$ has the physical meaning of energy of the target problem, the second term corresponds to the expected energy evaluated on the model density. This term increases the model probability density at those more probable configurations.

In fact, the loss function Eq. (3) has its origin in the variational approaches in statistical mechanics [51, 54, 55]. To see this, we write

$$\mathcal{L} + \ln Z = \mathbb{KL}\left(q(\boldsymbol{x}) \,\middle\|\, \frac{\pi(\boldsymbol{x})}{Z}\right) \geq 0, \tag{4}$$

where the Kullback-Leibler (KL) divergence measures the proximity between the model and the target probability densities [42, 55]. Equation (4) reaches zero only when the two distributions are identical. One thus concludes that the loss Eq. (3) provides a variational upper bound of the physical free energy of the system, $-\ln Z$.

For the actual optimization of the loss function, we randomly draw a batch latent variables from the prior probability and pass them through the generator network $\boldsymbol{x} = g(\boldsymbol{z})$, an unbiased estimator of the loss Eq. (3) is

$$\mathcal{L} = \mathop{\mathbb{E}}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \left[\ln p(\boldsymbol{z}) - \ln \left|\det\left(\frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right| - \ln \pi(g(\boldsymbol{z}))\right], \tag{5}$$

where the log-Jacobian determinant can be efficiently computed by summing the contributions of each block. Notice that in Eq. (5) all the network parameters are inside the expectation but not in the sampling process, which amounts to the *reparametrization trick* [42]. We perform stochastic optimization of Eq. (5) [56], in which the gradients with respect to the model parameters are computed efficiently using backpropagation. The gradient of Eq. (5) is the same as the one of the KL divergence Eq. (4) since the intractable partition function Z is independent of the model parameter. Learning by optimizing the loss function Eq. (5) can be more efficient compared to the conventional maximum likelihood density estimation [42, 55] and the supervised learning approach [57, 58] since one directly makes use the functional form of the target density and its gradient information. Moreover, since one has an unbiased estimation of the variational loss, it is always better to achieve a lower value of Eq. (5) in the training without the concern of overfitting.

The KL divergence (4) tends to reduce the model probability whenever the target probability density is low [42, 55]. Since one is learning on samples generated by the model itself, it has a potential danger of mode collapse where the model probability covers only one mode of the target probability distribution. Mode collapse corresponds to a local minimum of the loss function Eq. (3) which however can be alleviated by adding regularization or extending the model expressibility.

Different from previous studies using generative models with intractable [20, 23] or implicit [59] likelihood, building up the RG flow using bijectors provides direct access to the exact log-likelihood Eq. (1). One may thus employ the NeuralRG network as a trainable MCMC proposal [51, 60]. To ensure the detailed balance condition, one can accept the proposals from the network according to the Metropolis-Hastings acceptance rule [52, 61]

$$A(\boldsymbol{x} \to \boldsymbol{x}') = \min\left[1, \frac{q(\boldsymbol{x})}{q(\boldsymbol{x}')} \cdot \frac{\pi(\boldsymbol{x}')}{\pi(\boldsymbol{x})}\right]. \tag{6}$$

This realizes a Metropolized Independent Sampler (MIS) [53], where the proposals are statistically independent. Training such proposal policy is different from the previous attempts of training surrogate functions [57, 58, 62, 63] or transition kernels [64–66] for MCMC proposals. In MIS, the only correlation between samples is due to the rejection in Eq. (6). The acceptance rate is a good measure of the quality of the proposals made by the network.

The samples collected according to Eq. (6) can be regarded representatives of the target probability distribution $\pi(\boldsymbol{x})/Z$. We keep a buffer of accepted samples and evaluate the negative log-likelihood [42] on a batch of samples $\mathcal{D}$ randomly drawn from the buffer

$$\mathrm{NLL} = -\frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x}\in\mathcal{D}} \ln q(\boldsymbol{x}). \tag{7}$$

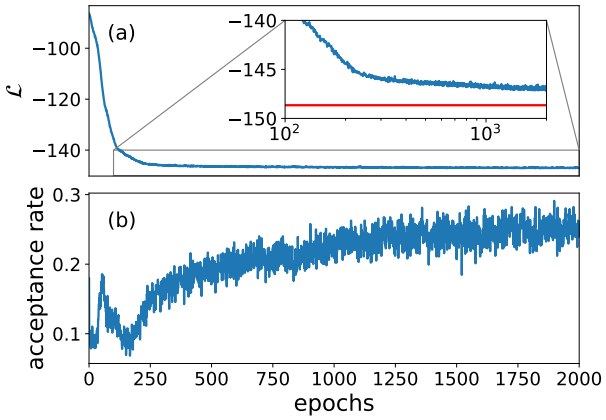

Figure 3. (a) The variational free energy Eq. (3). The inset shows a zoomed-in view, where the solid red line is the exact lower bound from the analytical solution of the Ising model. (b) Average acceptance rate Eq. (6) versus the training steps. The physical problem under consideration is a dual version of the Ising model with continuous field variables Eq. (8) on an 8×8 lattice at critical coupling. An epoch means one optimization step of the network parameters.

The NLL is a proxy of the KL divergence $\mathbb{KL}\left(\frac{\pi(\boldsymbol{x})}{Z}\,\|\,q(\boldsymbol{x})\right)$, which is reversed compared to the one in Eq. (4). When the model probability density approaches to the one of the target density, NLL is also minimized. Since minimizing the NLL amplifies the model probability density on the observed data [42, 55], we add the NLL to the loss function Eq. (3) as a regularization term to prevent the mode collapse [41]. Using a buffer of collected samples can be viewed as an *Experience Replay* [67] in the reinforcement learning of the policy network $q(\boldsymbol{x})$.

As a demonstration, we apply the NeuralRG approach to the two dimensional Ising model, a prototypical model in statistical physics. To conform with the continuous requirement of the physical variables, we employ the continuous relaxations trick of Ref. [68]. We first decouple the Ising spins using a Gaussian integral, then sum over the Ising spins to obtain a target probability density

$$\pi(\boldsymbol{x}) = \exp\left(-\frac{1}{2}\boldsymbol{x}^T\left(K+\alpha I\right)^{-1}\boldsymbol{x}\right)\times\prod_{i=1}^{N}\cosh\left(x_i\right), \qquad (8)$$

where K is an $N\times N$ symmetric matrix, I is an identity matrix and α is constant offset such that $K+\alpha I$ is positive definite [69]. For each of the configuration, one can directly sample the discrete Ising variables $\boldsymbol{s}=\{\pm1\}^{\otimes N}$ according to $\pi(\boldsymbol{s}|\boldsymbol{x}) = \prod_i(1+e^{-2s_ix_i})^{-1}$. It is straightforward to verify that the marginal probability distribution $\int \mathrm{d}\boldsymbol{x}\,\pi(\boldsymbol{s}|\boldsymbol{x})\pi(\boldsymbol{x}) \propto \exp\left(\frac{1}{2}\boldsymbol{s}^T K\boldsymbol{s}\right) \equiv \pi_{\mathrm{Ising}}(\boldsymbol{s})$ restores the Boltzmann weight of the Ising model with the coupling matrix K [70]. Therefore, Equation (8) can be viewed as a dual version of the Ising model, in which the continuous variables $\boldsymbol{x}$ represent the field couple to the Ising spins. We choose K to describe the two-dimensional critical Ising model on a square lattice critical with periodic boundary condition.

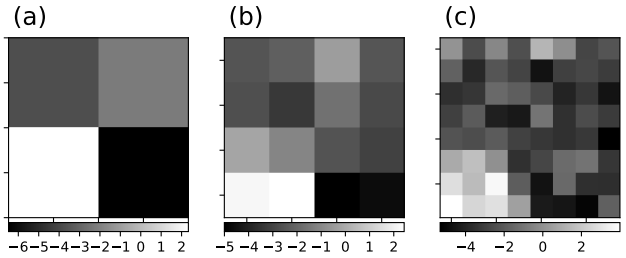

Figure 4. (a, b) Renormalized collective variables [star symbols in Fig. 1(a)] at the 2×2 and 4×4 scales. (c) The physical variables on an $N=8\times8$ lattice.

We train the NeuralRG network of the structure shown schematically in Fig. 1(a) with two modifications. First, the layout of the blocks is shown in Fig. 2(c), where we use two layers of disentanglers between every other decimator [71]. Second, the bijectors are of the size 2×2, as shown in Fig. 2(d). The results in Fig. 3 shows that the variational free-energy continuously decreases during the training. The red solid line in the inset shows the exact lower bound $-\ln Z = -\ln Z_{\mathrm{Ising}} - \frac{1}{2}\ln\det(K+\alpha I) + \frac{N}{2}[\ln(2/\pi)-\alpha]$, where $Z_{\mathrm{Ising}} = \sum_{\boldsymbol{s}}\pi_{\mathrm{Ising}}(\boldsymbol{s})$ is known from the Onsager's exact solution [72]. The average acceptance rate Eq. (6) increases with training, indicating that a better variational approximation of the target density also provides better Monte Carlo proposals. Since the proposals are independent, increased acceptance rate directly implies decreased autocorrelation times between samples. We have also checked that the accepted samples yield unbiased physical observables. After training, we generate physical configurations by passing independent Gaussian variables through the network [41]. Figure 4 shows the collective variables together with the physical variables, where the decimators learned to play an analogous role of the block spin transformation [73]. Although capturing coarse-grained features in the deeper layers of the network was widely observed in deep learning, the NeuralRG approach puts it into a quantitative manner since dependence of the collective variables and their effective coupling are all tractable through Eq. (1).

The minimalist implementation of NeuralRG [74] can be further improved in several aspects. First, One could even use a different prior density instead of the uninformative Gaussian distribution so that the RG does not always flow towards the infinite temperature fixed point. Second, the MERA inspired network structure can nevertheless be generalized by following considerations in tensor network architecture design [75, 76]. Lastly, one can improve each bijector using more expressive normalizing flows [31, 32, 34–37]. Since the size of each block is independent of the system size in NeuralRG, one may even employ more general bijectors [77] compared to the current choices in deep learning. Any of this improvement is likely to further to improve the variational upper bound of Fig. 3(a). Currently, the proposed NeuralRG frame-

work is limited to problems with continuous variables since it relies on the probability transformation Eq. (1) and the differential learnability of the bijectors. Employing ideas from deep learning to generalize the approach to discrete variables is an interesting direction [78, 79].

The NeuralRG approach provides an automatic way to identify collective variables and their effective couplings [80, 81]. The application is particularly relevant to off-lattice molecular simulations which involve a large number of continuous degrees of freedom. The NeuralRG essentially performs a learnable change-of-variables from the physical space to the less correlated latent space $Z = \int \mathrm{d}\boldsymbol{x}\, \pi(\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{z}\, \pi(g(\boldsymbol{z})) \left|\det\left(\frac{\partial g(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right|$ [41]. Instead of the Metropolized independent sampling Eq. (6), it may also be advantageous to perform ordinary Metropolis [52] or hybrid Monte Carlo [82] sampling in the latent space. We focused on physical systems with translational invariance in this paper, where it makes sense to use a predetermined homogenous architecture. For physical systems with disorders [83, 84] or realistic dataset in machine learning, it would be interesting to learn the network structure based on the mutual information pattern of the problem [85, 86].

Besides calling a revived attention to the probabilistic [87] and information theory [88] perspectives on the RG flow, the NeuralRG also arouses a few mathematical questions. Conventional RG is a semigroup since the process is irreversible. However, the NeuralRG networks built from bijectors form a group. Taking the deep learning perspective [42], one tempts to view the RG as a continuous transformation of the data manifolds. However, since diffeomorphism function keeps the topology of the manifolds of physical and latent space, it is yet to be seen how does such assumption interplay with the topological features in statistical configurations.

Despite the similarity between the disentanglers to the convolutional kernel and decimators to the pooling layers, the proposed NeuralRG architecture Fig. 1(a) is different from the convolutional neural network. A crucial difference is that each bijector performs *nonlinear bijective* transformation instead of linear transformations. We believe that it is the large-scale structural similarities between MERA and dilated convolutions [35, 36, 38] and factor out layers [33] used in modern deep generative models underline their successes in modeling quantum states and classical data. It is interesting to compare the performance of the NeuralRG architecture to conventional deep learning models in machine learning tasks [41] and study its general implications for neural network architecture design.

The authors thank Yang Qi, Yi-Zhuang You, Pan Zhang, Jin-Guo Liu, Lei-Han Tang, Chao Tang, Lu Yu, Long Zhang, Guang-Ming Zhang, and Ye-Hua Liu for discussions and encouragement. We thank Wei Tang for providing the exact free energy value of the 2D Ising model used in the inset of Fig. 3(a). The work is supported by the Ministry of Science and Technology of China under the Grant No. 2016YFA0300603 and the National Natural Science Foundation of China under Grant No. 11774398.

* wanglei@iphy.ac.cn

[1] M. Gell-Mann and F. E. Low, "Quantum electrodynamics at small distances," Physical Review **95**, 1300–1312 (1954).
[2] Kenneth G. Wilson, "Renormalization Group and Critical Phenomena. I. Renormalization Group and the Kadanoff Scaling Picture," Physical Review B **4**, 3174 (1971).
[3] Kenneth G Wilson, "Renormalization Group and Critical Phenomena. II. Phase-Space Cell Analysis of Critical Behavior," Physical Review B **4**, 3184 (1971).
[4] Kenneth G. Wilson, "The renormalization group: Critical phenomena and the Kondo problem," Reviews of Modern Physics **47**, 773–840 (1975).
[5] Robert H. Swendsen, "Monte Carlo Renormalization Group," Physical Review Letters **42**, 859 (1979).
[6] G. Vidal, "A class of quantum many-body states that can be efficiently simulated," Physical Review Letters **101**, 110501 (2008), arXiv:0610099.
[7] Michael Levin and Cody P. Nave, "Tensor renormalization group approach to two-dimensional classical lattice models," Physical Review Letters **99**, 120601 (2007), arXiv:0611687.
[8] Zheng-Cheng Gu, Michael Levin, and Xiao-Gang Wen, "Tensor-entanglement renormalization group approach as a unified method for symmetry breaking and topological phase transitions," Physical Review B **78**, 205116 (2008), arXiv:0807.2010.
[9] Zheng-Cheng Gu and Xiao-Gang Wen, "Tensor-entanglement-filtering renormalization approach and symmetry-protected topological order," Physical Review B **80**, 155131 (2009), arXiv:0903.1069.
[10] Z. Y. Xie, H.C. Jiang, Q.N. Chen, Z. Y. Weng, and T. Xiang, "Second Renormalization of Tensor-Network States," Physical Review Letters **103**, 160601 (2009).
[11] H. H. Zhao, Z. Y. Xie, Q. N. Chen, Z. C. Wei, J. W. Cai, and T. Xiang, "Renormalization of tensor-network states," Physical Review B **81**, 174411 (2010), arXiv:1002.1405.
[12] Z. Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang, "Coarse-graining renormalization by higher-order singular value decomposition," Physical Review B **86**, 045139 (2012), arXiv:1201.1144.
[13] Efi Efrati, Zhe Wang, Amy Kolan, and Leo P. Kadanoff, "Real-space renormalization in statistical mechanics," Reviews of Modern Physics **86**, 647–667 (2014), arXiv:1301.6323v1.
[14] G. Evenbly and G. Vidal, "Tensor Network Renormalization," Physical Review Letters **115**, 180405 (2015), arXiv:1412.0732.
[15] G. Evenbly and G. Vidal, "Tensor Network Renormalization Yields the Multiscale Entanglement Renormalization Ansatz," Physical Review Letters **115**, 200401 (2015), arXiv:1502.05385.
[16] M. Bal, M. Mariën, J. Haegeman, and F. Verstraete, "Renormalization Group Flows of Hamiltonians Using Tensor Networks," Physical Review Letters **118**, 250602 (2017), arXiv:1703.00365.
[17] Markus Hauru, Clement Delcamp, and Sebastian Mizera, "Renormalization of tensor networks using graph-independent local truncations," Phys. Rev. B **97**, 045111 (2018).
[18] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision* (Springer, 2014) pp. 818–833.

[19] Cédric Bény, "Deep learning and the renormalization group," arXiv (2013), arXiv:1301.3124.

[20] Pankaj Mehta and David J. Schwab, "An exact mapping between the Variational Renormalization Group and Deep Learning," arXiv (2014), arXiv:1410.3831.

[21] Serena Bradde and William Bialek, "PCA Meets RG," Journal of Statistical Physics **167**, 462–475 (2017), arXiv:1610.09733.

[22] Satoshi Iso, Shotaro Shiba, and Sumito Yokoo, "Scale-invariant Feature Extraction of Neural Network and Renormalization Group Flow," arXiv (2018), arXiv:1801.07172.

[23] Maciej Koch-Janusz and Zohar Ringel, "Mutual Information, Neural Networks and the Renormalization Group," arXiv (2017), arXiv:1704.06279.

[24] Yi-Zhuang You, Zhao Yang, and Xiao-Liang Qi, "Machine learning spatial geometry from entanglement features," Phys. Rev. B **97**, 045153 (2018).

[25] Juan Carrasquilla and Roger G. Melko, "Machine learning phases of matter," Nature Physics **13**, 431–434 (2017), arXiv:1605.01735.

[26] Evert P.L. Van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber, "Learning phase transitions by confusion," Nature Physics **13**, 435–439 (2017), arXiv:1610.02048.

[27] Lei Wang, "Discovering phase transitions with unsupervised learning," Physical Review B **94**, 195105 (2016), arXiv:1606.00318.

[28] Henry W. Lin, Max Tegmark, and David Rolnick, "Why Does Deep and Cheap Learning Work So Well?" Journal of Statistical Physics **168**, 1223–1247 (2017), arXiv:1608.08225.

[29] David J. Schwab and Pankaj Mehta, "Comment on "Why does deep and cheap learning work so well?" [arXiv:1608.08225]," arXiv (2016), arXiv:1609.03541.

[30] Laurent Dinh, David Krueger, and Yoshua Bengio, "NICE: Non-linear Independent Components Estimation," arXiv (2014), 1410.8516, arXiv:1410.8516.

[31] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," arXiv (2015), arXiv:1502.03509.

[32] Danilo Jimenez Rezende and Shakir Mohamed, "Variational Inference with Normalizing Flows," arXiv (2015), arXiv:1505.05770.

[33] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio, "Density estimation using Real NVP," arXiv (2016), 1605.08803, arXiv:1605.08803.

[34] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling, "Improving Variational Inference with Inverse Autoregressive Flow," arXiv (2016), arXiv:1606.04934.

[35] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu, "Pixel Recurrent Neural Networks," in *International Conference on Machine Learning I(CML)*, Vol. 48 (2016) pp. 1747—-1756, arXiv:1601.06759.

[36] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," arXiv (2016), arXiv:1609.03499.

[37] George Papamakarios, Theo Pavlakou, and Iain Murray, "Masked Autoregressive Flow for Density Estimation," arXiv (2017), arXiv:1705.07057.

[38] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis, "Parallel WaveNet: Fast High-Fidelity Speech Synthesis," arXiv (2017), arXiv:1711.10433.

[39] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous, "TensorFlow Distributions," arXiv (2017), arXiv:1711.10604.

[40] Pyro Developers, "Pyro," (2017).

[41] See the Supplementary Materials for details of the training algorithm, the implementation of the Real NVP bijectors, samples from the model, effects of the latent space and results on the MNIST dataset.

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning* (MIT Press, 2016).

[43] In this case, one can iterate the training process for increasingly larger system size and reuse the weights from the previous step as the initial value.

[44] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv (2016), arXiv:1603.04467.

[45] Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito, "Automatic differentiation in PyTorch," in *NIPS 2017 Workshop Autodiff* (2017).

[46] Guifre Vidal, "Efficient classical simulation of slightly entangled quantum computations," Physical Review Letters **91**, 147902 (2003), arXiv:0301063.

[47] Y. Y. Shi, L. M. Duan, and G. Vidal, "Classical simulation of quantum many-body systems with a tree tensor network," Physical Review A **74**, 022320 (2006), arXiv:0511070.

[48] The decimator only tree structure NeuralRG network is sufficient to model one dimensional physical systems with short-range interactions since the mutual information is constant [50].

[49] Thomas Barthel, Martin Kliesch, and Jens Eisert, "Real-space renormalization yields finite correlations," Physical Review Letters **105**, 010502 (2010), arXiv:1003.2319.

[50] Michael M. Wolf, Frank Verstraete, Matthew B. Hastings, and J. Ignacio Cirac, "Area laws in quantum systems: Mutual information and correlations," Physical Review Letters **100**, 070502 (2008), arXiv:0704.3906.

[51] David J C MacKay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, 2005).

[52] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller, "Equation of state calculations by fast computing machines," Journal Chemical Physics **21**, 1087–1092 (1953), arXiv:5744249209.

[53] Jun S. Liu, *Monte Carlo Strategies in Scientific Computing* (Springer, 2001).

[54] R. P. Feynman, *Statistical Mechanics: A Set of Lectures* (W. A. Benjamin, Inc., 1972).

[55] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, 2006).

[56] Diederik P Kingma and Jimmy Lei Ba, "Adam: A Method for Stochastic Optimization," arXiv (2014), arXiv:arXiv:1412.6980v9.

[57] Li Huang and Lei Wang, "Accelerated Monte Carlo simulations with restricted Boltzmann machines," Physical Review B **95**, 035105 (2017), arXiv:1610.02746.

[58] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu, "Self-learning Monte Carlo method," Physical Review B **95**, 041101 (2017), arXiv:1610.03137.

[59] Zhaocheng Liu, Sean P. Rodrigues, and Wenshan Cai, "Simulating the Ising Model with a Deep Convolutional Generative Adversarial Network," arXiv (2017), arXiv:1710.04987.

[60] Nando de Freitas, Højen Sørensen Rensen, Michael I Jordan, and Stuart Russell, "Variational MCMC," arXiv (2013), arXiv:1301.2266.

[61] W K Hastings, "Monte Carlo sampling methods using Markov chains and their applications," Biometrica **57**, 97 (1970).

[62] C. E. Rasmussen, "Gaussian Processes to Speed up Hybrid Monte Carlo for Expensive Bayesian Integrals," Bayesian Statistics 7 , 651–659 (2003).

[63] Li Huang, Yi Feng Yang, and Lei Wang, "Recommender engine for continuous-time quantum Monte Carlo methods," Physical Review E **95**, 031301(R) (2017), arXiv:1612.01871.

[64] Jiaming Song, Shengjia Zhao, and Stefano Ermon, "A-NICE-MC: Adversarial Training for MCMC," arXiv (2017), arXiv:1706.07561.

[65] Daniel Levy, Matthew D. Hoffman, and Jascha Sohl-Dickstein, "Generalizing Hamiltonian Monte Carlo with Neural Networks," arXiv (2017), arXiv:1711.09268.

[66] Marco F. Cusumano-Towner and Vikash K. Mansinghka, "Using probabilistic programs as proposals," arXiv (2018), arXiv:1801.03612.

[67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, "Human-level control through deep reinforcement learning," Nature **518**, 529 (2015), arXiv:1604.03986.

[68] Yichuan Zhang, Charles Sutton, and Amos Storkey, "Continuous Relaxations for Discrete Hamiltonian Monte Carlo," Advances in Neural Information Processing Systems 25 , 3194—-3202 (2012).

[69] We choose α such that the lowest eigenvalue of $K + \alpha I$ equals to 0.1.

[70] Note that Eq. (8) is not a unique. By diagonalizing $(K + \alpha I)$ one obtains a model where each continuous variable couples to one Fourier component of the original Ising spins [68]. Performing NeuralRG on such problem would correspond to the momentum space RG.

[71] Isaac H. Kim and Brian Swingle, "Robust entanglement renormalization on a noisy quantum computer," arXiv (2017), arXiv:1711.07500.

[72] Lars Onsager, "Crystal statistics. I. A two-dimensional model with an order-disorder transition," Physical Review **65**, 117–149 (1944).

[73] Leo P. Kadanoff, "Scaling Law for Ising models near T_c," Physics **2**, 263–272 (1966).

[74] See https://github.com/li012589/NeuralRG for a PyTorch implementation of the code.

[75] L. Tagliacozzo, G. Evenbly, and G. Vidal, "Simulation of two-dimensional quantum systems using a tree tensor network that exploits the entropic area law," Physical Review B **80**, 235127 (2009), arXiv:0903.5017.

[76] G. Evenbly and G. Vidal, "Class of highly entangled many-body states that can be efficiently simulated," Physical Review Letters **112**, 240502 (2014), arXiv:1210.1895.

[77] Scott Shaobing Chen and Ramesh A. Gopinath, "Gaussianization," Advances in Neural Information Processing Systems 13 , 423–429 (2001).

[78] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical Reparameterization with Gumbel-Softmax," arXiv (2016), arXiv:1611.01144.

[79] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh, "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables," arXiv (2016), arXiv:1611.00712.

[80] Alessandro Barducci, Massimiliano Bonomi, and Michele Parrinello, "Metadynamics," Wiley Interdisciplinary Reviews: Computational Molecular Science **1**, 826–843 (2011).

[81] Michele Invernizzi, Omar Valsson, and Michele Parrinello, "Coarse graining from variationally enhanced sampling applied to the Ginzburg-Landau model," Proceedings of the National Academy of Sciences **114**, 3370–3374 (2017).

[82] Simon Duane, A D Kennedy, Brian J Pendleton, and Duncan Roweth, "Hybrid Monte Carlo," Physics Letters B **195**, 216–222 (1987).

[83] Shang Keng Ma, Chandan Dasgupta, and Chin Hun Hu, "Random antiferromagnetic chain," Physical Review Letters **43**, 1434–1437 (1979).

[84] Daniel S. Fisher, "Random transverse field Ising spin chains," Physical Review Letters **69**, 534–537 (1992).

[85] C. K. Chow and C. N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," IEEE Transactions on Information Theory **14**, 462–467 (1968).

[86] Katharine Hyatt, James R. Garrison, and Bela Bauer, "Extracting Entanglement Geometry from Quantum States," Physical Review Letters **119**, 140502 (2017), arXiv:1704.01974.

[87] G. Jona-Lasinio, "The renormalization group: A probabilistic view," Il Nuovo Cimento B Series 11 **26**, 99–119 (1975).

[88] S. M. Apenko, "Information theory and renormalization group flows," Physica A: Statistical Mechanics and its Applications **391**, 62–77 (2012), arXiv:0910.2097.

[89] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," arXiv (2015), arXiv:1511.07289.

[90] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges, "Mnist handwritten digit database," (2010).

Supplemental Materials: Neural Network Renormalization Group

Training Algorithm

Algorithm 1 shows the training procedure of the NeuralRG network [41]. The experience buffer has a fixed maximum size. The samples in the buffer approach to the target probability density during training. When building up the buffer, we perform data argumentation by appending symmetry related configurations of the same weight. We prevent mode collapse using the NLL (7) evaluated on samples drawn from the buffer. The MCMC rejection is done by comparing samples drawn from the buffer and sample drawn from the network.

Algorithm 1 NeuralRG Training Algorithm

Require: Normalized prior probability density `p(z)`
Require: Unnormalized target probability density `pi(x)`
Ensure: A bijector neural network `x=g(z)` with normalized probability density `q(x)`
  Initialize a bijector `g`
  Initialize an experience buffer $\mathcal{B} = \varnothing$
  while `Stop Criterion Not Met` **do**
    Sample a batch of latent variables `z` according to the prior `p(z)`
    Obtain physical variables `x=g(z)` and their densities `q(x)` ▷ Eq. (1)
    `loss = mean{ln[q(x)]-ln[pi(x)]}` ▷ Eq. (5)
    Push accepted samples to the buffer $\mathcal{B}$ ▷ Eq. (6)
    Sample a batch of data from the buffer $\mathcal{B}$, evaluate the `NLL` ▷ Eq. (7)
    `loss += NLL`
    Optimization step for the `loss`
  end while

Details about the Real NVP Bijector

To implement the bijector we use the real-valued non-volume preserving (Real NVP) net [33], which belongs to a general class of bijective neural networks with tractable Jacobian determinant [30–37]. Real NVP is a generative model with explicit and tractable likelihood. One can efficiently evaluate the model probability density $q(\boldsymbol{x})$ for any sample, either given externally or generated by the network itself. This feature is important for integrating with an unbiased Metropolis sampler.

The Real NVP block divides the inputs into two groups $\boldsymbol{z} = \boldsymbol{z}_< \cup \boldsymbol{z}_>$, and updates only one of them with information of another group

$$\begin{cases} \boldsymbol{x}_< = \boldsymbol{z}_<, \\ \boldsymbol{x}_> = \boldsymbol{z}_> \odot e^{s(\boldsymbol{z}_<)} + t(\boldsymbol{z}_<), \end{cases} \tag{S1}$$

where $\boldsymbol{x} = \boldsymbol{x}_< \cup \boldsymbol{x}_>$ is the output. $s(\cdot)$ and $t(\cdot)$ are two arbitrary functions parametrized by neural networks. In our implementation, we use multilayer perceptrons with 64 hidden neurons of exponential linear activation [89]. The output activation of the scaling s-function is a tanh with learnable scale. While the output of the translation t-function is a linear function. The $\odot$ symbol denotes element-wise product. The transformation Eq. (S1) is easy to invert by reversing the basic arithmetical operations. Moreover, the transformation has a triangular Jacobian matrix, whose determinant can be computed efficiently by summing over each component of the outputs of the scaling function $\ln\left|\det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right)\right| = \sum_i [s(\boldsymbol{z}_<)]_i$. The transformation Eq. (S1) can be iterated so that each group of variables is updated. In our implementation, we update each of the two groups four times in an alternating order. The log-Jacobian determinant of the bijector block is computed by summing up contributions of each layer. To construction NeuralRG network, we use the same blocks in each layer with shared parameters. The log-Jacobian determinant is computed by summing up contributions of each block.

Samples from the model during training

Figure S1 shows the first two components of the physical variables sampled during the training. The setup is the same as Fig. 3 of the main text. Starting from an initial Gaussian distribution the model quickly learned about positive correlations between the neighboring physical variables.

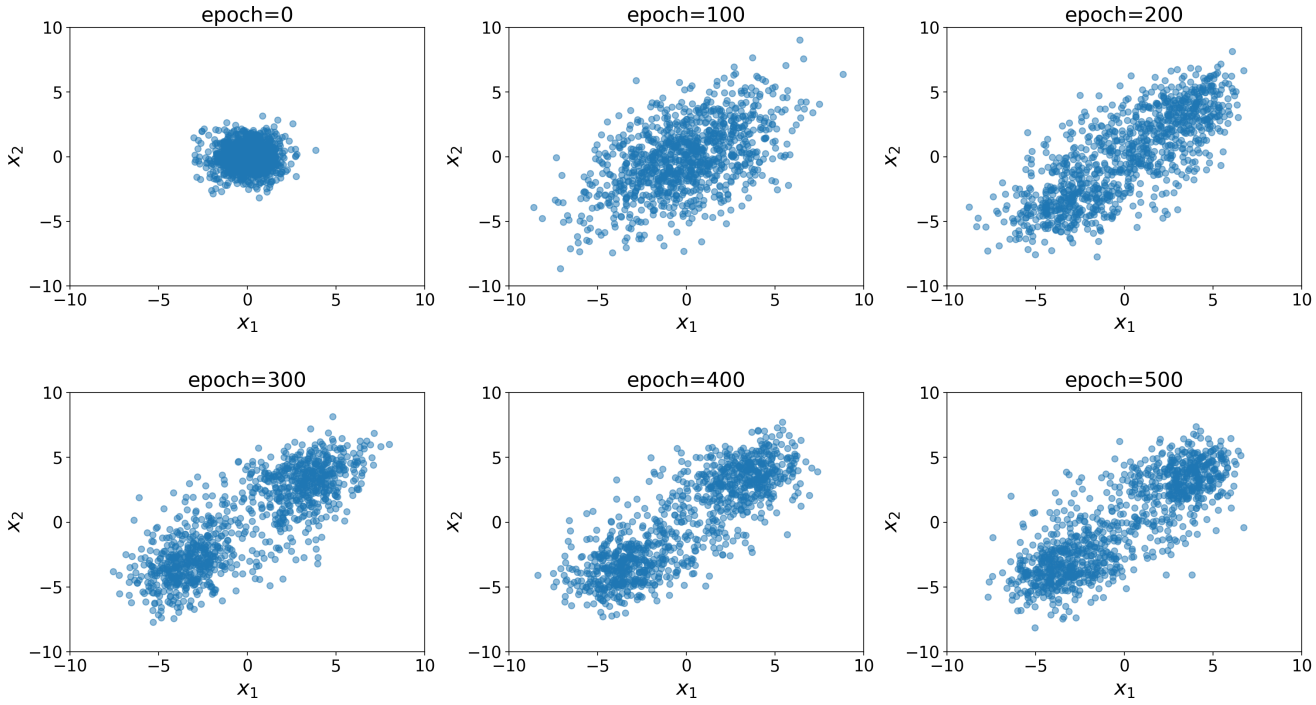

Figure S1. Samples generated by the model at different training steps.

Wander in the latent space

Figures S2-S3 show the renormalized collective variables by flipping one component of the latent vector in Fig. 4 of the main text. Figure S4 shows the corresponding physical variables. One sees that flipping most of the latent vector components result in local changes to the physical variables. However, a few latent components control the physical variables globally.

Experiments on MNIST dataset

We train the NeuralRG network on the MNIST hand-written digit dataset [90]. We pad the MNIST images to be 32×32 and construct a NerualRG network of 5 layers. We use 4-in-4-out Real NVP networks for the disentanglers and decimators. In each of these Real NVP networks, we use 4-layers of multilayer perceptrons with 64 hidden neurons for both $s(\cdot)$ and $t(\cdot)$ functions.

For the training, we optimize the NLL loss (7) using the Adam optimizer with a learning rate 0.001. After 500 epochs of training, the NLL reaches around -4900 on the test set. Figures S5 shows the sampled images at each layer of the NeuralRG network. One sees the NerualRG network successfully learned about the distribution and yield coarse-grained features as collective variables, despite that the periodic boundary condition and translational invariance do not apply to the MNIST images.

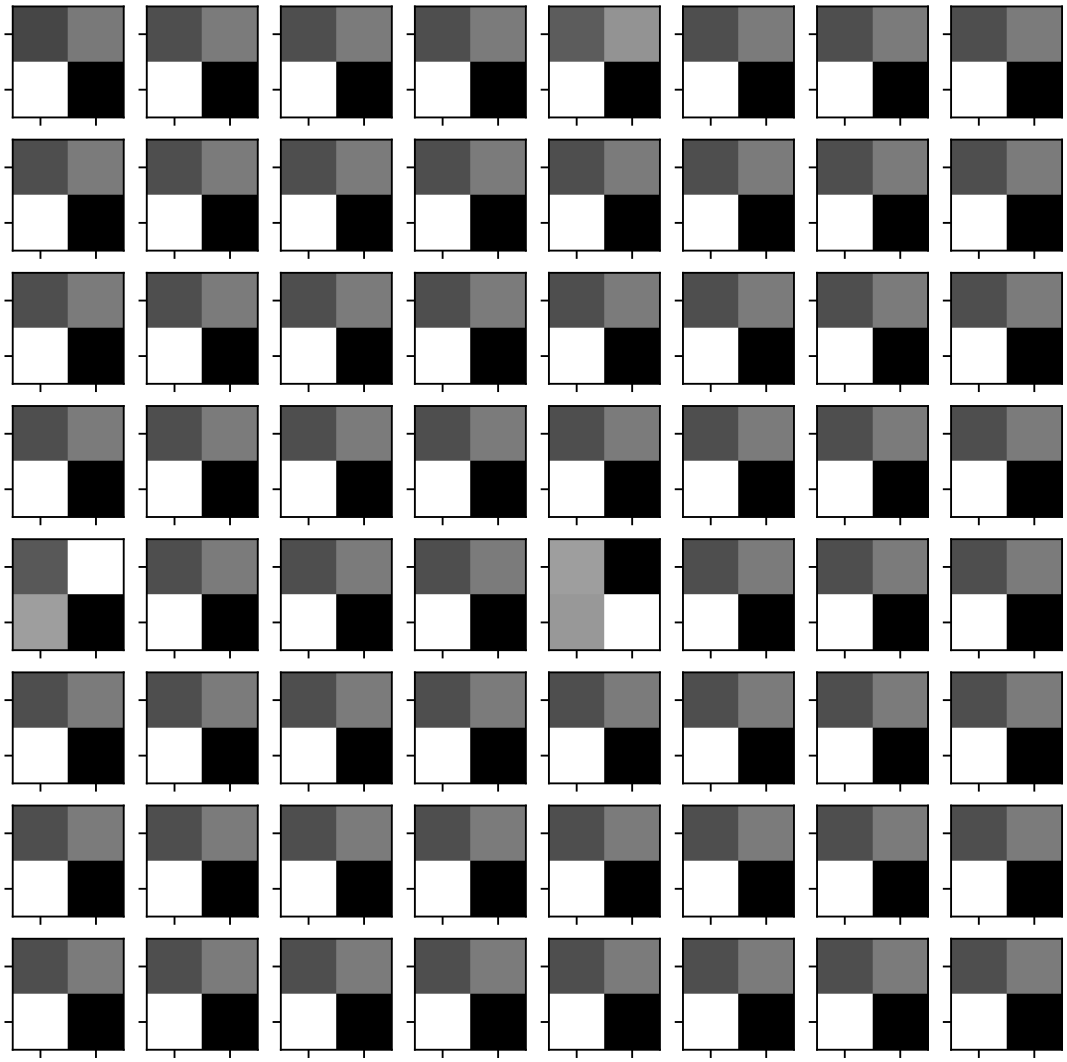

Figure S2. Sampled renormalized collective variables in the same scale as Fig. 4(a) by flipping one component in the latent space.

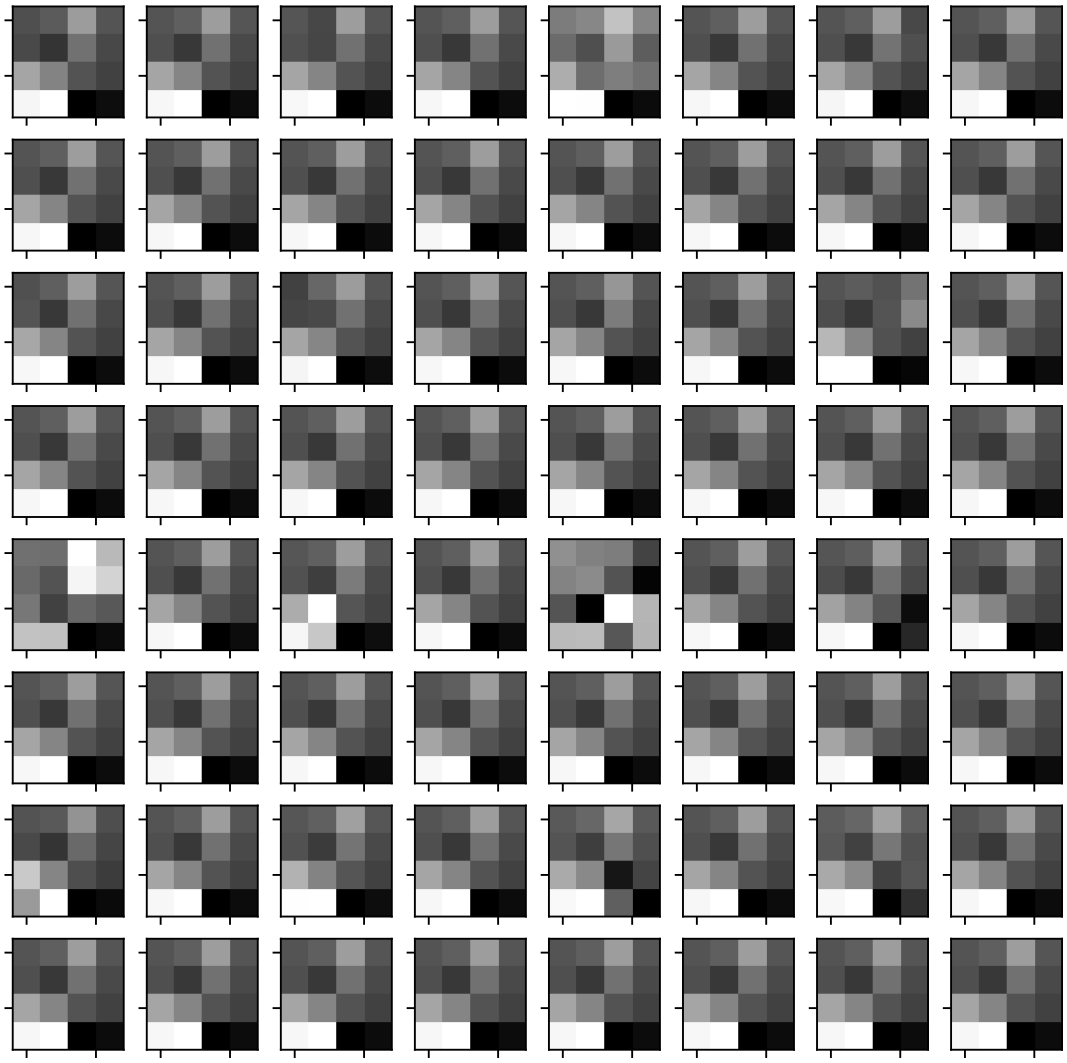

Figure S3. Sampled renormalized collective variables in the same scale as Fig. 4(b) by flipping one component in the latent space.

Figure S4. Sampled physical variables by flipping one component in the latent space.

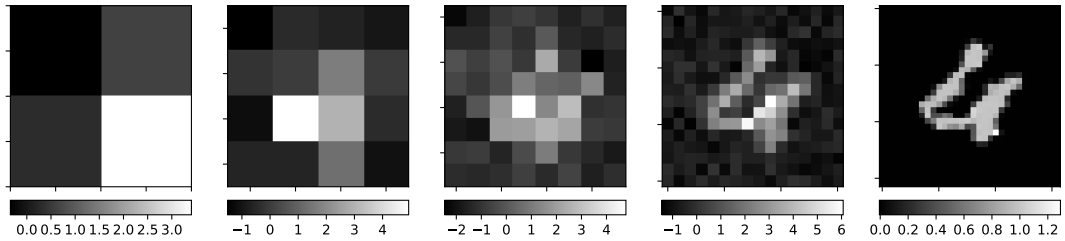

Figure S5. Generative modeling of the MNIST dataset using the NeuralRG network. Similar RG phenomenon appears in collective variables.