Unbiased Monte Carlo Cluster Updates with Autoregressive Neural Networks

Dian Wu
dian.wu@epfl.ch

Riccardo Rossi
riccardo.rossi@epfl.ch

Giuseppe Carleo*
giuseppe.carleo@epfl.ch

Abstract

Efficient sampling of complex high-dimensional probability distributions is a central task in computational science. Machine learning methods like autoregressive neural networks, used with Markov chain Monte Carlo sampling, provide good approximations to such distributions, but suffer from either intrinsic bias or high variance. In this work, we propose a way to make this approximation unbiased and with low variance. Our method uses physical symmetries and variable-size cluster updates which utilize the structure of autoregressive factorization. We test our method for first- and second-order phase transitions of classical spin systems, showing its viability for critical systems and in the presence of metastable states.

1 Introduction

Markov chain Monte Carlo [1] (MCMC) is an unbiased numerical method that allows sampling from unnormalized probability distributions, a central task in many areas of computational science. MCMC is commonly used, for example, in molecular dynamics [2], as well as statistical and quantum physics [3–6]. In addition to fundamental applications, MCMC serves as a physics-inspired approach to solve a variety of computational problems, including combinatorial optimization [7, 8] and computer graphics [9]. While MCMC is a generically applicable technique, its implementation can be plagued by long mixing or autocorrelation time [10]. Various techniques have been proposed to increase the efficiency of MCMC [11], for example, cluster updates [12, 13], parallel tempering [14], the worm algorithm [15], and event-chain Monte Carlo [16]. However, these faster MCMC algorithms rely on details of the physical system considered, and they cannot be applied generically.

Machine learning (ML) methods, given their intrinsic flexibility in addressing problems in computational physics [17], are being intensively investigated as a way to improve MCMC. Applications in this direction include, for example, self-learning Monte Carlo methods [18–23], enhanced sampling driven by neural networks [24, 25], and neural importance sampling [26]. Strongly rooted in the principles of statistical physics, variational sampling techniques are among the most promising ML-driven approaches. Generative neural samplers (GNS) [27–29] are a chief example of ML-driven variational methods. These approaches build on the idea of constructing approximate representations of the original probability distribution at hand. The resulting variational approximations can efficiently perform sampling by construction, thus completely bypassing MCMC. A particularly interesting aspect of this approach is its systematic improvability when using the free energy bound minimization as the guiding principle to gauge the approximation accuracy. The main drawback of the variational approach, however, is that the estimators of expectation values are intrinsically biased by the representation error of the approximated distribution. As unbiased estimators are of central importance in many fundamental applications in physics, recent research has started addressing the key problem of removing the bias induced by ML variational representations, for example, through importance sampling, and incorporating again MCMC strategies [26, 30, 31].

*Institute of Physics, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

1.1 Bias in variationally trained generative neural samplers

Consider a system of V classical Ising spins $\mathbf{s} := (s_1, \dots, s_V)$, $s_i \in \{-1, 1\}$, at inverse temperature β. We use a generative neural sampler (GNS) q_θ with parameters θ that variationally approximates the Boltzmann probability distribution $p(\mathbf{s}) \propto \tilde{p}(\mathbf{s}) := \mathrm{e}^{-\beta E(\mathbf{s})}$ by minimizing a free energy upper bound [27], which is equivalent to minimizing the Kullback–Leibler (KL) divergence $D_{\mathrm{KL}}(q_\theta \,\|\, p) := \sum_{\mathbf{s}} q_\theta(\mathbf{s}) \ln \frac{q_\theta(\mathbf{s})}{p(\mathbf{s})}$. To construct an expressive q_θ, we use an autoregressive neural network (ARNN) to factorize it into a product of conditional probabilities $q_\theta(\mathbf{s}) =: \prod_{i=1}^{V} q_{\theta;i}(s_i \mid \mathbf{s}_{<i})$, where $\mathbf{s}_{<i} := (s_1, \dots, s_{i-1})$, which also allows us to efficiently sample from q_θ by sampling from $\{q_{\theta;i}\}$ sequentially.

However, the fact that the two distributions are only approximately equal, $q_\theta(s) \approx \tilde{p}(\mathbf{s})$, implies that the samples $\{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(N)}\}$ drawn from the network carry an intrinsic bias. When these samples are used to compute the expectation value of a physical observable, the resulting estimator $\bar{O} = \frac{1}{N}\sum_{i=1}^{N} O(\mathbf{s}^{(i)})$ is biased from the true value $\mathbb{E}_p[O]$ and it is impossible in general to reliably estimate the direction and the magnitude of such bias.

1.2 Neural importance sampling and global updates

Refs. [30, 31] have proposed two closely related solutions to the bias problem. The first method, which we denote neural importance sampling (NIS) in the following, consists of using the modified unbiased estimator $\bar{O} = \sum_{i=1}^{N} w(\mathbf{s}^{(i)}) O(\mathbf{s}^{(i)})$, where $w(\mathbf{s}^{(i)}) := \frac{\tilde{w}(\mathbf{s}^{(i)})}{\sum_{j=1}^{N} \tilde{w}(\mathbf{s}^{(j)})}$ and $\tilde{w}(\mathbf{s}^{(i)}) := \frac{\tilde{p}(\mathbf{s}^{(i)})}{q_\theta(\mathbf{s}^{(i)})}$ are the normalized and the unnormalized weights respectively. The second proposed solution, which we denote neural global updates (NGU) hereafter, consists of using the GNS as a Markov chain Monte Carlo (MCMC) proposer: if $\mathbf{s}$ is the current Markov chain state, a proposed state $\mathbf{s}'$ is drawn from the GNS and accepted with the Metropolis probability $P_{\mathrm{acc}}(\mathbf{s} \to \mathbf{s}') := \min\left(1, \frac{\tilde{w}(\mathbf{s}')}{\tilde{w}(\mathbf{s})}\right)$.

However, as shown in our numerical experiments, those methods can be plagued by the ergodicity issue, which produces impractically long autocorrelation time and high variance of the estimators, due to the generic presence of "exponentially suppressed configurations".

2 Methods

2.1 Exponentially suppressed configurations

We point out an elementary property of the KL divergence: the cost of allowing a single bad approximation $q_\theta(\mathbf{s})$ scales only logarithmically with the ratio $q_\theta(\mathbf{s})/p(\mathbf{s})$. Therefore, even when the free energy is well approximated after the variational training, $q_\theta(\mathbf{s})$ is still exponentially smaller than $p(\mathbf{s})$ for a small portion p_{ESC} of configurations, which we call exponentially suppressed configurations (ESC). A well-trained network has $p_{\mathrm{ESC}} \ll 1$, and those ESC have a limited effect on the training loss, but rather a strong effect on the autocorrelation time.

Let us consider a Markov chain evolution using NGU, and suppose that the current state $\mathbf{s}$ is an ESC. The ratio $\tilde{w}(\mathbf{s}')/\tilde{w}(\mathbf{s})$ in P_{acc} will be exponentially small for almost any other configuration $\mathbf{s}'$; therefore, the Markov chain will be essentially stuck in $\mathbf{s}$ for a long time before accepting any new proposal, and the autocorrelation time of the whole chain will be impractically large. A similar argument applies when considering the variance of the NIS method.

2.2 Neural cluster updates with symmetries

To solve the generic ergodicity problem of neural global update methods, we propose an enhanced MCMC method with the use of the physical symmetries and the autoregressive structure, which we denote as neural cluster updates with symmetries (NCUS), as described in Alg. 1 for a 2D lattice with translational and reflectional symmetries, and sketched in Fig. 1.

Both the random symmetry operations and the cluster updates help escape from ESC. Assume that the current configuration $\mathbf{s}$ is an ESC, and we use a random symmetry operation to change $\mathbf{s}$ to another configuration $\mathbf{s}^*$ in the equivalence class $\mathcal{C}$. The probability that all configurations in $\mathcal{C}$ are

ESC is on the order of $p_{\text{ESC}}^{\#\mathcal{C}}$, so it is extremely unlikely to get stuck within the whole equivalence class. Note that the occurrence of ESC does not depend on the physical symmetries, which can be checked in numerical experiments, but rather on the structure of the network.

In a cluster update, the weight ratio in P_{acc} becomes $\frac{\tilde{w}(\mathbf{s}')}{\tilde{w}(\mathbf{s})} = \frac{\tilde{p}(\mathbf{s}')}{\tilde{p}(\mathbf{s})} \prod_{i=V-k+1}^{V} \frac{q_{\theta;i}(s_i|\mathbf{s}_{<i})}{q_{\theta;i}(s_i'|\mathbf{s}'_{<i})}$, which is not too far from 1 when k is small. Therefore, the new configuration is closer to the old one and is easier to be accepted. Although the cluster size k can be sampled from an arbitrary distribution $P_{\text{cluster}}(k)$, numerical experiments have shown that the uniform distribution $P_{\text{cluster}}(k) \equiv 1/V$ already works better than many other cases we have explored.

Algorithm 1 A step of NCUS on a 2D lattice with $T_x \times T_y \times D_4 \times \mathbb{Z}_2$ symmetry.

1: Input the current configuration $\mathbf{s}$
2: Sample an integer $k \in \{1, \ldots, V\}$ from P_{cluster}
3: Sample the last k spins and propose the configuration $\mathbf{s}'$
4: Accept $\mathbf{s} \leftarrow \mathbf{s}'$ with probability $P_{\text{acc}}(\mathbf{s} \to \mathbf{s}')$
5: Translate $\mathbf{s}$ by a random displacement
6: Reflect $\mathbf{s}$ along the x axis, the y axis and the diagonal, each with 50% probability
7: Reflect $\mathbf{s}$ along the z axis (flip all spins) with 50% probability
8: Output $\mathbf{s}$ as a sample in the Markov chain

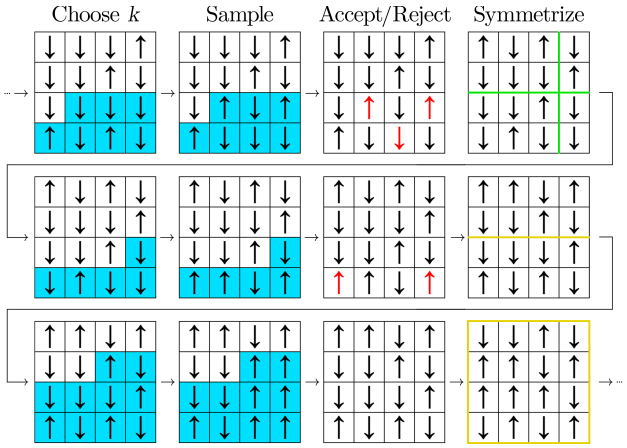

Figure 1: Example of three steps of NCUS applied to a 4×4 spin model. The columns correspond to different lines in Alg. 1. The last k spins that can be flipped are highlighted in blue. If a proposal is accepted, the spins actually flipped are shown in red. For translations, the original borders of the lattice are shown in green. For reflections, the plane of reflection is shown in yellow, and yellow borders around the lattice indicate a reflection along the z axis (across the xy plane).

3 Numerical experiments

3.1 Ising model

We start to demonstrate the effectiveness of NCUS on the conventional 2D Ising model $E(\mathbf{s}) := \sum_{i,j=1}^{L} s_{i,j}(s_{i+1,j} + s_{i,j+1})$, with periodic boundary conditions $s_{L+1,j} = s_{1,j}$, $s_{i,L+1} = s_{i,1}$. We use a lightweight convolutional ARNN with only 3 convolutional layers and approximately 4×10^3 parameters, and use the same network to compare the different sampling methods. Thanks to the MCMC bias removal, we do not need the network to approximate the true distribution to an extremely high precision, which will be increasingly difficult for larger lattices. Details of the network structure, training, and sampling are described in Appendix A.

From Fig. 2 (a), we see that both NGU and NIS have pathologically high autocorrelation times in the critical region. An inspection of their autocorrelation function in Fig. 2 (b) shows that the

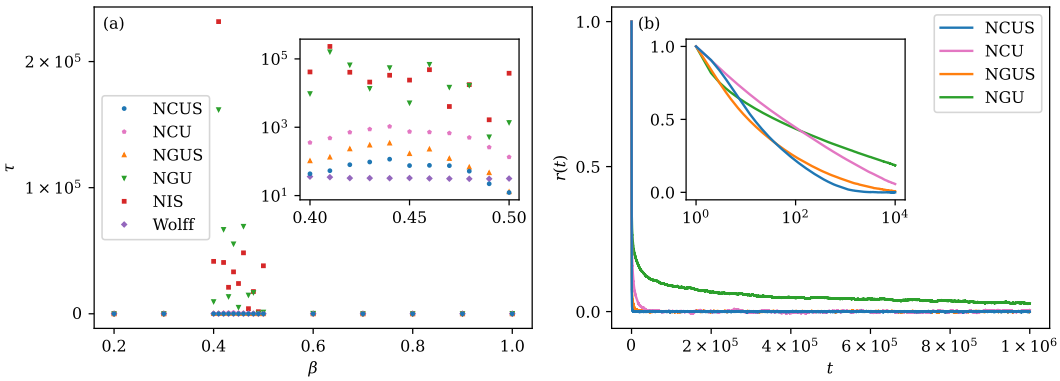

Figure 2: (a) Integrated autocorrelation time τ as a function of temperature on the 16×16 Ising model. For NIS, we use the increased variance from the reweighting procedure as the effective autocorrelation time. The inset focuses on their behaviors near the critical point and uses the logarithmic scale on the y axis. (b) Autocorrelation functions $r(t)$ on the 16×16 Ising model at $\beta = 0.44$. The inset uses the logarithmic scale on the x axis to focus on their behaviors at small t.

Markov chain of NGU is essentially non-ergodic in the available simulation time. By contrast, our proposed method NCUS has no issue in the critical region. A closer inspection of the inset of Fig. 2 (a) shows that the autocorrelation time of NCUS still increases in the critical region, and the sampling efficiency is improved typically by two orders of magnitude compared with the global update methods. The performance of NCUS is also comparable to the celebrated Wolff cluster update method [13], which is specifically tailored for the Ising model. The ablation studies of neural cluster updates without symmetries (NCU) and neural global updates with symmetries (NGUS) are also shown, and the uncertainties of τ are analyzed in Appendix B.

3.2 Frustrated plaquette model

We now study another model that presents a richer physics than the Ising model, and for which, to our knowledge, no traditional cluster update method is applicable. We consider a classical spin-1/2 system with nearest-neighbor J_1, next-next-nearest-neighbor J_3, and plaquette K interactions $E(\mathbf{s}) := J_1 \sum_{i,j=1}^{L} s_{i,j}(s_{i+1,j} + s_{i,j+1}) + J_3 \sum_{i,j=1}^{L} s_{i,j}(s_{i+2,j} + s_{i,j+2}) + K \sum_{i,j=1}^{L} s_{i,j}\, s_{i+1,j}\, s_{i,j+1}\, s_{i+1,j+1}$, with periodic boundary conditions, which we denote as the frustrated plaquette model (FPM). In this work, we set $J_1 = J_3 = -1$, $K = 2$, where a first-order transition between the ferrimagnetic (fM) and the paramagnetic (PM) phases is conjectured from an analogy to the $q = 8$ Potts model [32, 33]. The comparison of autocorrelation times from different sampling methods is presented in Fig. 3, which provides numerical evidence that NCUS greatly alleviates the metastability issue expected near first-order phase transitions [34].

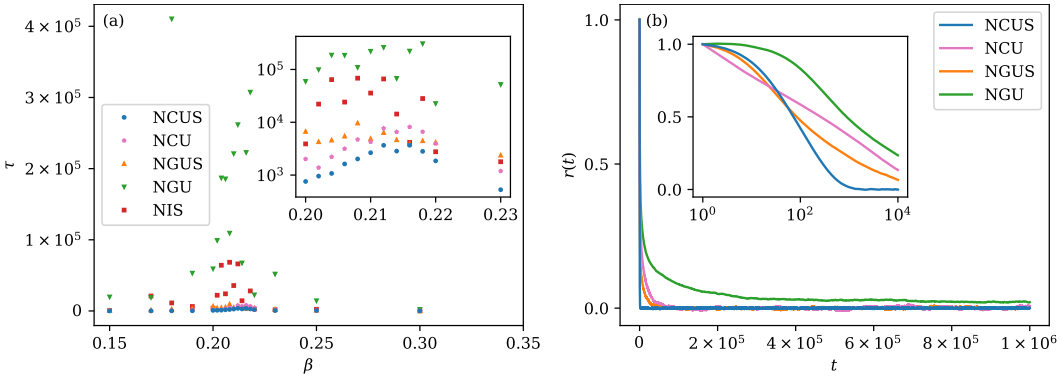

Figure 3: (a) Integrated autocorrelation time τ as a function of temperature on the 32×32 FPM. (b) Autocorrelation functions $r(t)$ on the 32×32 FPM at $\beta = 0.2$.

Theoretically, a first-order phase transition occurs when the distribution of energy $p(E) \propto N(E)\,\mathrm{e}^{-\beta E}$ has two peaks with the same size, as shown in Fig. 4, where $N(E)$ is the number of configurations with energy E. A GNS-based sampling method has equal probabilities to generate a sample from the two peaks, and the probability to accept that proposal will be close to 1, if the network is ideally trained and there is no problem of ESC. Meanwhile, for traditional local-update MCMC methods, they can only move small horizontal steps in Fig. 4, so it takes more steps ($\sim L^2$) and exponentially lower probability ($\sim \mathrm{e}^{-\beta\delta E L^2}$) for them to walk from the low-energy peak to the high-energy one, where δE is the typical energy difference in a local update, which does not scale with L. In other words, the exponentially large number of configurations in the high-energy peak will not make it easier for local-update MCMC methods to sample from that peak because it is exponentially hard for the walker to walk between those configurations in locally connected paths. NCUS reaches a balance between the two extremes, which solves the problem of ESC and keeps the autocorrelation time practically low, even if the network is lightweight and cannot ideally approximate the true distribution.

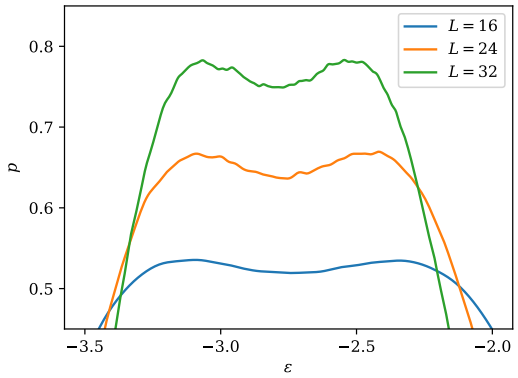

Figure 4: Probability distribution of the energy per site $p(\varepsilon)$ for the FPM with different lattice sizes L at their respective phase transition temperatures, obtained by NCUS.

4 Conclusions

In this work, we have shown a method to systematically remove the bias and keep a low variance in GNS-based sampling, using symmetries of the physical system and cluster updates of the autoregressive factorization, which is viable for critical systems and in the presence of metastable states. While we have been mainly concerned with the metric of autocorrelation time, we recognize that the wall-clock time is another important metric for practical computations. In this respect, when computing the energy of the system has a negligible computational cost, current neural network-based methods are not yet competitive with traditional MCMC methods. It can then be argued that the ideal application scenario for ML-based methods are those cases where evaluating the integrand is expensive, for example, in determinant quantum Monte Carlo [35] and lattice field theory [36]. In future work, computational efficiency can be addressed on multiple fronts, for example, by introducing techniques such as hierarchy and sparsity of the neural network models, to reduce the computation time and scale up the lattice size by orders of magnitude. After that, we expect that the slow asymptotic growth of the autocorrelation time of GNS will eventually make them outperform traditional MCMC methods in terms of wall-clock time.

Acknowledgments and Disclosure of Funding

We acknowledge insightful comments and suggestions from Lei Wang and Pan Zhang. Support from the Swiss National Science Foundation is acknowledged under Grant No. 200021_200336. The computing power is supported with Cloud TPUs from Google's TensorFlow Research Cloud (TFRC).

References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[2] K. Binder, *Monte Carlo and Molecular Dynamics Simulations in Polymer Science* (Oxford University Press, 1995).

[3] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics* (Springer, 2010).

[4] W. Krauth, *Statistical Mechanics: Algorithms and Computations* (Oxford University Press, 2006).

[5] J. Gubernatis, N. Kawashima, and P. Werner, *Quantum Monte Carlo Methods* (Cambridge University Press, 2016).

[6] F. Becca and S. Sorella, *Quantum Monte Carlo Approaches for Correlated Systems* (Cambridge University Press, 2017).

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).

[8] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning* (Springer, 2004).

[9] R. L. Cook, ACM Trans. Graph. **5**, 51 (1986).

[10] H. Müller-Krumbhaar and K. Binder, J. Stat. Phys. **8**, 1 (1973).

[11] J. S. Liu, *Monte Carlo Strategies in Scientific Computing* (Springer, 2004).

[12] J.-S. Wang and R. H. Swendsen, Physica A **167**, 565 (1990).

[13] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[14] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **57**, 2607 (1986).

[15] N. V. Prokof'ev, B. V. Svistunov, and I. S. Tupitsyn, Phys. Lett. A **238**, 253 (1998).

[16] E. P. Bernard, W. Krauth, and D. B. Wilson, Phys. Rev. E **80**, 056704 (2009).

[17] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).

[18] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, in *International Conference on Learning Representations* (2018).

[19] J. Song, S. Zhao, and S. Ermon, in *Advances in Neural Information Processing Systems, Long Beach, CA* (2017) pp. 5140–5150.

[20] M. Medvidovic, J. Carrasquilla, L. E. Hayward, and B. Kulchytskyy, (2020), arXiv:2012.01442 .

[21] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, Phys. Rev. B **95**, 041101 (2017).

[22] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).

[23] H. Shen, J. Liu, and L. Fu, Phys. Rev. B **97**, 205140 (2018).

[24] L. Bonati, Y.-Y. Zhang, and M. Parrinello, Proc. Natl. Acad. Sci. **116**, 17641 (2019).

[25] F. Noé, S. Olsson, J. Köhler, and H. Wu, Science **365**, 10.1126/science.aaw1147 (2019).

[26] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, ACM Trans. Graph. **38**, 1 (2019).

[27] D. Wu, L. Wang, and P. Zhang, Phys. Rev. Lett. **122**, 080602 (2019).

[28] M. S. Albergo, G. Kanwar, and P. E. Shanahan, Phys. Rev. D **100**, 034515 (2019).

[29] S.-H. Li and L. Wang, Phys. Rev. Lett. **121**, 260601 (2018).

[30] K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel, Phys. Rev. E **101**, 023304 (2020).

[31] B. McNaughton, M. V. Milošević, A. Perali, and S. Pilati, Phys. Rev. E **101**, 053312 (2020).

[32] H. Arisue and K. Tabata, in *Non-Perturbative Methods and Lattice QCD* (World Scientific, 2001) pp. 233–241.

[33] V. Gorbenko, S. Rychkov, and B. Zan, J. High Energy Phys. **2018** (10), 1.

[34] R. Gheissari and E. Lubetzky, (2016), arXiv:1607.02182 .

[35] R. Blankenbecler, D. J. Scalapino, and R. L. Sugar, Phys. Rev. D **24**, 2278 (1981).

[36] D. C. Hackett, C.-C. Hsieh, M. S. Albergo, D. Boyda, J.-W. Chen, K.-F. Chen, K. Cranmer, G. Kanwar, and P. E. Shanahan, (2021), arxiv:2107.00734 .

[37] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, in *International Conference on Machine Learning* (2016).

[38] S. Elfwing, E. Uchibe, and K. Doya, Neural Netw. **107**, 3 (2018).

[39] F. Yu and V. Koltun, in *International Conference on Learning Representations* (2016).

[40] D. P. Kingma and J. Ba, in *International Conference on Machine Learning* (2015).

[41] A. Gelman and D. B. Rubin, Stat. Sci. **7**, 457 (1992).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] This is rather fundamental physics.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Details of numerical experiments

Our network has 3 convolutional layers, each with kernel size 5. The convolutions are masked to implement the autoregressive property, as introduced in PixelCNN [37]. The numbers of input, hidden, and output channels are $1 \to 16 \to 16 \to 1$. SiLU activations [38] are applied after the first and the second convolutional layers, which are reported to produce lower loss than ReLU. Sigmoid activation is applied after the third convolutional layer to restrain the output into $(0, 1)$.

We keep the network to be lightweight, so we can generate a large number of samples and show the effect of the MCMC bias removal. To capture the long-range correlations in the physical system, the receptive field of the network should be able to approximately cover the whole lattice. We use dilated convolutions [39] to expand the receptive field, and increase the dilation rate in each convolutional layer by a step size. The receptive field radius can be calculated by

$$\text{Receptive field radius} = \frac{1}{2} D \left((D-1)d + 2 \right) \frac{s-1}{2}, \tag{1}$$

where $D = 3$ is the number of convolutional layers, $s = 5$ is the convolution kernel size, and d is the dilation step size. For lattice sizes $L = 8, 16, 24, 32$, we use $d = 1, 2, 3, 4$ respectively. The network has $3,761$ non-masked parameters in total, regardless of the lattice size. It is possible to use larger networks with the similar structure to obtain better variational free energies before the MCMC bias removal, as discussed in Refs. [27, 30].

During training, we use Adam optimizer [40] with a conventional learning rate 10^{-3}, batch size 64, and take 2×10^4 training steps. To avoid being trapped in local minima, especially at low temperatures, in the first 10^4 steps we linearly anneal β from 0 to the desired value, which is reported to produce a lower loss than exponential annealing. We do not use weight regularization or gradient clipping, because the network is shallow and there is no significant instability in training.

For NCUS, NCU, and NGUS sampling, we generate 10^3 Markov chains in parallel, each containing 10^5 samples. The chains are initialized by samples from the network. The first 10^4 samples in each chain are discarded to ensure that only the samples after thermalization are taken into account. For each experiment of NCUS up to $L = 32$, the Gelman–Rubin diagnostic [41] is less than 1.1, which confirms that the chains are thermalized. The integrated autocorrelation time (IAT) is less than 4×10^3, which is shorter than the remaining chain length by orders of magnitude. For NGU and NIS sampling, we generate 10^2 chains with length 10^6, and discard the first 10^5 samples for NGU. The effective autocorrelation time of NIS is computed individually for each "chain". On an NVIDIA V100 GPU, the training for the 32×32 FPM takes two hours, and the sampling of NCUS takes a day.

Our code is available at `https://github.com/wdphy16/neural-cluster-update`

B Uncertainty analysis

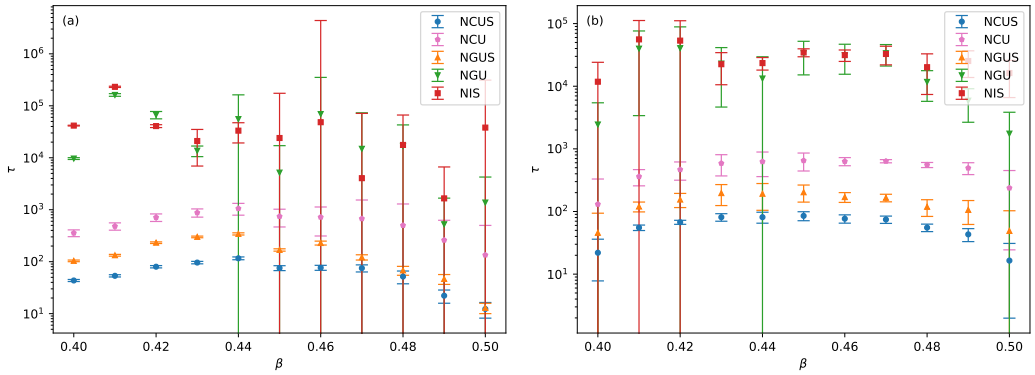

Figure 5: The uncertainties of the IAT in the inset of Fig. 2 (a). (a) The mean values are computed using a typical random seed, and the error bars are the standard deviations over all chains that we generate in parallel. (b) The mean values are computed over 5 random seeds, and the error bars are the standard deviations of the mean values from each random seed. The IAT of NCUS from different chains and random seeds are consistent within the same order of magnitude, while those of NGU and NIS are generally impractical to converge.

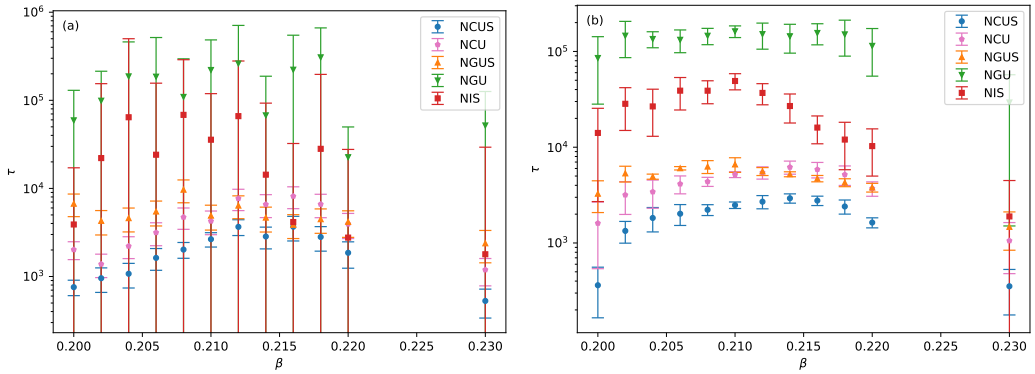

Figure 6: The uncertainties of the IAT in the inset of Fig. 3 (a).