Learning Renormalization Group Flows for Lattices

Jay Shen*
Department of Physics
University of Chicago †

Real space renormalization is a powerful and theoretically fascinating, albeit difficult technique for investigating scale and phase behavior in physical systems. For even the simplest problems, formulating the so-called renormalization group (RG) flow involves incredibly tedious, intuition-dependent work that can drag on for years. However, once accurately described, RG flows have a number of uses, from describing phase behaviors to speeding up simulations. Accordingly, any information at all about their properties is highly valued and sought after by physicists. In this work, we review and assess a novel approach to real space renormalization initially proposed by Hou et al. [1]. The so-called Machine Learning Renormalization Group (MLRG) algorithm automatically determines approximate RG flows of translationally-invariant Ising models, given only the symmetry description of the lattice. It has the potential to effectively characterize a wide range of interesting systems, and also demonstrates an elegant synthesis of both new and old machine learning techniques with statistical physics. In the first section of the review, we will first give some background for real space renormalization and the Ising model. In the second section, we will describe the MLRG algorithm and demonstrate its use. In the third section, we will discuss the algorithmic design space and suggest modifications for greater efficacy and efficiency.

I. BACKGROUND

A. The Gist of Real Space Renormalization

Real-space renormalization is a theoretical framework for understanding scale in physical systems [2, 3]. That is, it tries to understand how the apparent behavior of a system changes with the length scale.

As a toy example, consider the task of modeling some volume of water. At the very smallest scales, we might use field theories or quantum mechanics to describe the microscopic dynamics composing each and every atom. This approach requires a simply enormous amount of information—think positions, momenta, etc. In addition, the relations between these degrees of freedom are, more often than not, impossibly complex and intractable to solve.

However, if we choose to ”zoom out” to a somewhat larger scale, things become substantially simpler and more manageable. At the human scale, for example, fluid mechanics accurately describes the hydrodynamics of the water. But compared to the smaller-scale theories, it is much simpler model, requiring only a few parameters—viscosity, pressures, temperatures, etc.—to fully specify the system. Computation is also completely tractable and comparatively straightforward.

We ask, in what way are all these theories, which model the same system, linked? Why are small scales complex and large scales simple? How is this all connected to the thermodynamic phase behavior of the system? Real space renormalization provides answers to these questions, and more. For simplicity, in this paper we will only describe real space renormalization as it pertains to the Ising model. It is our hope that it is an example rich enough to demonstrate real space renormalization and illustrate its profound importance.

B. The Ising Model

The Ising model is a historically important problem in statistical physics as well as computer science and machine learning [4, 5]. Originally devised as a thermodynamic model of ferromagnetism, it has since been applied to a wide range of problems, including some beyond physics.

The Ising model takes the form of up-down spins arranged on a regular lattice. Each spin is correlated to its

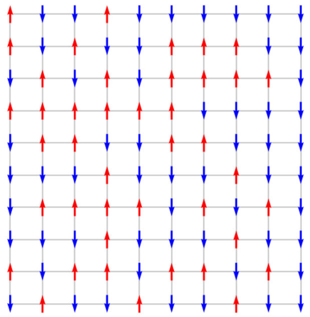

FIG. 1. A random Ising model on a square lattice.

* My deepest gratitude goes out to Professor Kao at NTU for his guidance and hospitality at NTU. I'd also like to thank the NTU physics department buddies and my officemates for their friendship and the cultural exchange we shared. Finally, a huge thanks to all the donors from the UCTS program, the SITG program, the Metcalf Foundation—as well as Professor Chin at UChicago—for making my summer in Taiwan possible in the first place.

† jshe@uchicago.edu

neighboring spins—namely, those adjacent to it on the lattice.

We can model this with a Hamiltonian representing the total energy of the system:

$$H(J,\vec{\sigma}) = -J \sum_{(i,j)\in\mathcal{E}} \sigma_i\sigma_j \tag{1}$$

Here, $\vec{\sigma}$ is a vectorized list of all the spins in the Ising model. $(i,j)\in\mathcal{E}$ means the spins i and j are neighbors on the lattice. $J\in\mathbb{R}$, called the coupling constant, specifies the correlation strength between spins. $\sigma_i,\sigma_j\in\{-1,+1\}$ are the values of spins i and j.

If $J>0$, the system is ferromagnetic, and aligned spins lower the energy while misalign spins raise it. If $J<0$, the system is antiferromagnetic and the reverse is true: aligned spins raise the energy and misaligned spins lower it. If $J=0$, the system is non-interacting and the spins have no correlation.

Using this Hamiltonian, we can define a Gibbs probability distribution over all the states of an Ising model:

$$P(J,\vec{\sigma}) = \frac{1}{Z}e^{-H(J,\vec{\sigma})} \tag{2}$$

Implicitly, this assigns higher probabilities to lower energy states.

C. Real Space Renormalization of the Ising Model

Say we have some Ising model defined by a specific lattice and a coupling constant J. We would like to “zoom out” and understand what this Ising model looks like from some larger length scale.

To apply real space renormalization, we must first engineer a *coarse-graining* transformation to remove some degrees of freedom from the system. This is necessary to obtain a large-scale description of the system that models macroscopic, rather than microscopic, features.

For example, given a square lattice, we might choose Kadanoff blocking [6] as our coarse-graining transformation. Kadanoff blocking removes degrees of freedom by replacing a block of microscopic spins with a single macroscopic spin.

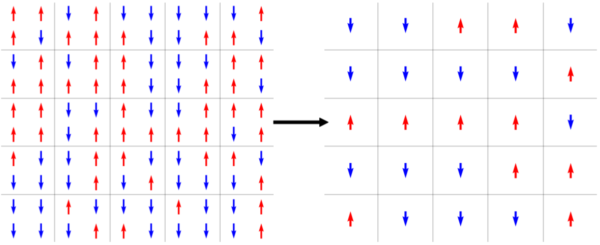

FIG. 2. A Kadanoff blocking procedure.

Note that this transformation has the required property of *scale invariance*. In other words, the coarse-grained system on the new set of spins $\vec{\sigma}'$ is still an Ising model. This means it can be modeled with the same Hamiltonian. Only the coupling constant J will change to some J' to reflect the change in scale.

$$H(J',\vec{\sigma}') = -J' \sum_{(i,j)\in\mathcal{E}'} \sigma_i\sigma_j \tag{3}$$

This constitutes one real space renormalization step.

Note that for each J there is only one unique J'. This means we can write down a function modelling the change in coupling constant due to the coarse-graining transformation:

$$J' - J = f(J) : \mathbb{R} \mapsto \mathbb{R} \tag{4}$$

We call this function the renormalization group (RG) flow. It turns out that the asymptotic behaviors of this function correspond directly to the thermodynamic phase behaviors of the system. Formulating this function across all values of J, is extremely difficult, however. We will now describe an algorithm that learns it automatically.

II. THE MLRG ALGORITHM

A. Algorithm Description

The Ising model we use in this algorithm will be defined on the Lieb lattice. It is similar to the square lattice, but incorporates ”hidden” spins which we can ignore when considering the visible behavior of the system. A single coupling constant, J, still mediates correlations between neighbors.

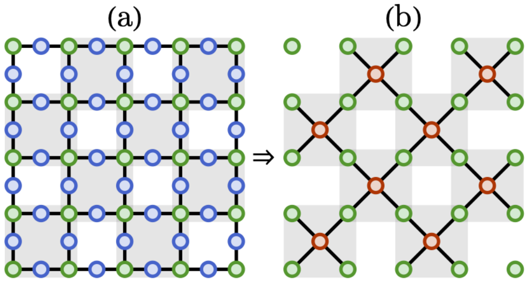

FIG. 3. (a) The fine-grain Lieb lattice. Visible spins are green and hidden spins are blue. (b) The coarse-grain Lieb lattice. The new hidden spins are red.

To coarse-grain, we use the Lieb blocking transformation visualized in Figure 3. Real space renormalization is applicable, as the system exhibits scale-invariance—the coarse-grained lattice is a Lieb lattice rotated by 45 degrees, with hidden and visible spins swapped.

Let’s note two niceties of the Lieb lattice important to the algorithm.

1. Since the Lieb lattice is *translationally invariant*, we don’t need to work with an entire lattice, only

the composite block that repeats to form it (Figure 4). Mathematically, this means we only need the Gibbs distributions over the block, not over the entire lattice, which would be completely unmanageable.

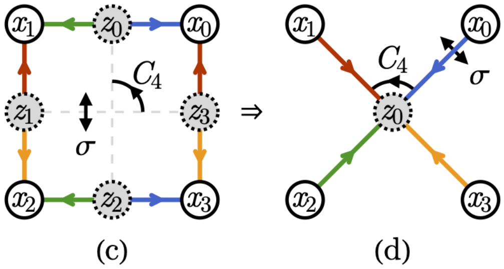

FIG. 4. (c) The fine-grained Lieb block is a square graph. (d) The coarse-grained Lieb block is a cross graph.

2. Since behavior of the hidden spins doesn't matter, we only need to consider the behavior of the visible spins. This will allow us to interpret the Lieb lattice as a Restricted Boltzmann Machine (RBM) [7].

So how are we going to formulate $f(J)$? The MLRG algorithm proposes approximating [8, 9] it using a neural network:

$$f_\theta(J) : \mathbb{R} \mapsto \mathbb{R} \tag{5}$$

Here, θ denotes the parameters of our network which will be tuned during training.

To train $f_\theta(J)$, Hou et al. [1] propose the following learning procedure:

1. **Sample a value of J**

   We want our learned function $f_\theta(J)$ to be highly accurate around its zeroes, because their locations specify important thermodynamic fixed and critical points. Elsewhere, it is fine if $f_\theta(J)$ is only reasonably accurate.

   To this end, we choose to use Hamiltonian Monte Carlo (HMC) [10, 11]. Using Hamiltonian dynamics, HMC samples from the unnormalized probability distribution:

   $$\tilde{P}(J) = e^{-U(J)} \tag{6}$$

   Here, $U(J)$ is a potential energy function which is minimized by the samples. We set it to:

   $$U(J) = \beta||f_\theta(J)||^2 \tag{7}$$

   $U(J)$ is annealed in during training by increasing β from $\beta_0 = 0$ to a $\beta_T > 0$ according to some schedule. This ensures the sampling is more or less random at the beginning of training, and we can get a general idea of the function shape. Later on, as the full value of $U(J)$ kicks in, the sampler begins to sample heavily only around the points of interest, thus refining their values.

2. **Forward pass to get $J' = J + f_\theta(J)$**

   Remember, the $f_\theta(J)$ represents the change in the coupling constant J after the coarse-graining transformation. Thus we can compute J', which we will use in the next step.

3. **Evaluate the error between fine and coarse-grained models**

   J is the coupling constant for the fine-grained model (Figure 4c). J', calculated in step 2, is the coupling constant for the coarse-grained model (Figure 4d). Ideally, if $f_\theta(J)$ does a good job, these two models should behave as similarly as possible.

   Now, the distributions over the visible spins provide our understanding of the model behaviors. These are given by marginalizing hidden spins out of the original Gibbs distribution:

   $$P_{fine}(J, \vec{v}) = \sum_{\vec{h}} P(J, \vec{v}, \vec{h}) \tag{8}$$

   $$P_{coarse}(J', \vec{v}) = \sum_{\vec{h}'} P(J', \vec{v}, \vec{h}') \tag{9}$$

   We can then use a Kullback-Liebler (KL) divergence to measure the difference between these distributions:

   $$D_{KL} = \sum_{\vec{v}} P_{fine}(\vec{v}) \log\Big(\frac{P_{fine}(\vec{v})}{P_{coarse}(\vec{v})}\Big) \tag{10}$$

   Alternatively, we can use the common contrastive approximation [12] to the KL divergence:

   $$D_C = F_{coarse}(\vec{v}_{fine}) - F_{coarse}(\vec{v}_{coarse}) \tag{11}$$

   Here, F_{coarse} is the free energy function of the coarse-grained model. $\vec{v}_{fine}$ and $\vec{v}_{coarse}$ are configurations of visible spins Gibbs-sampled from the fine and coarse-grained models, respectively.

   Either of these divergences is an explicit expression measuring the difference between the fine and coarse-grained distributions, and, by proxy, how well $f_\theta(J)$ predicts J'.

4. **Backpropagate error to θ**

   Once we compute either D_{KL} or D_C, we use it as a loss function and backpropagate [13] it to adjust the parameters θ. This is fairly easy using auto-differentiation [14], since both D_{KL} and D_C are computed using the forward pass value $f_\theta(J)$. As training progresses, $f_\theta(J)$ will get better and better at predicting J' given J.

B. Demonstration of Results

Following this procedure, we trained a simple two-layer multi-layer perceptron (MLP) with ReLU activations as $f_\theta(J)$:

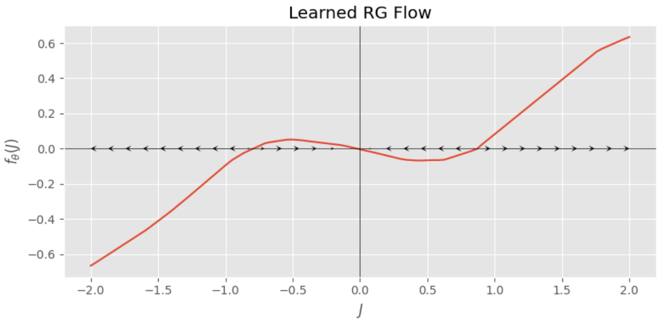

FIG. 5. Learned $f_\theta(J)$ function in red. The equivalent 1-dimensional vector field is shown on the x-axis.

At a glance, this $f_\theta(J)$ exhibits the general features we would expect:

- Anti-symmetric about $J = 0$
- Fixed points at $J = 0, -\infty, +\infty$
- Finite critical points where flow direction switches.

But how precise is it? From analytically solutions to the Ising model, we know the critical point should be located at:

$$J_C = \frac{1}{2}\text{arccosh}(1 + \sqrt{2}) \approx 0.7643 \tag{12}$$

Using Newton's root-finding method, we can numerically solve for the critical point predicted by $f_\theta(J)$. We can estimate the distribution of these predictions by training multiple instances of $f_\theta(J)$. A boxplot of this distribution is shown in Figure 6. We observe that, indeed, the algorithm is successful at identifying critical points near the correct value. In III we will discuss how to improve these results.

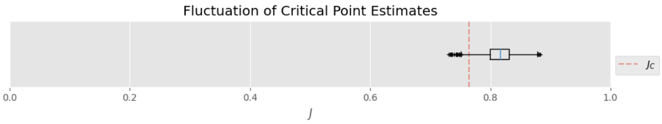

FIG. 6. Boxplot of predictions for J_C. Analytical target value shown by dotted red line.

III. THE MLRG DESIGN SPACE

A. Bias Reduction Using Higher-Dimensional Representations

The distribution of predictions in Figure 6 are visibly biased from the real J_C. Why? In Section III E and III D, we show some ways to slightly reduce this bias by adjusting algorithm hyperparameters. However, Hou et al. [1] conjecture that most of the bias comes from excessive information loss of relevant microscopic effects, which occurs during the coarse-graining transformation.

To ameliorate this, they propose increasing the representational capacity of the Ising model, reasoning that the more complex the Ising model is, the more information it can retain past coarse-graining. This can be done by using *higher-dimensional* lattices—in short, decorating the base Ising model with more spins and wiring additional correlations between those spins (See Figure 7) Now, these higher-dimensional lattices must have the same symmetries as the original, base Ising model. We can ensure this by referencing the representations of the lattice symmetry group. See Hou et al. [1] for an in-depth discussion of this process.

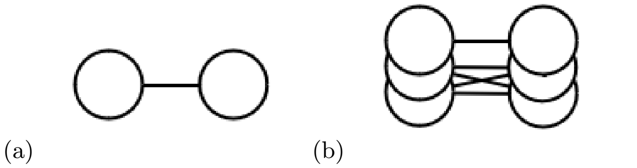

FIG. 7. (a) A bond in a base 1-dimension Ising model (A_1 representation) (b) A bond in a 5-dimension Ising model ($A_1 \oplus E$ representation)

The same general learning algorithm can be applied to these higher-dimensional lattices. However, since we added additional correlations between spins, we will have more coupling constants that need to change during the coarse-graining transformation. If we have n coupling constants, we can express the coupling constants by a vector $\vec{J} \in \mathbb{R}^n$. The RG flow, which we want to model, is also higher-dimension now:

$$\vec{J}' - \vec{J} = \vec{f}(J) : \mathbb{R}^n \mapsto \mathbb{R}^n \tag{13}$$

We can verify Hou et al.'s conjecture, observing in Figure 8 that increasing the lattice dimensionality does in fact reduce the prediction bias.

Hou et al. [1] further show that as the number of dimensions increases, that is as $n \to \infty$, the bias seems to go to 0. Unfortunately, due in part to the curse of dimensionality, this comes at the significant cost of degraded training stability and increased compute time. Nevertheless, this is reassuring, as it provides a loose scaling law for the algorithm's predictive accuracy.

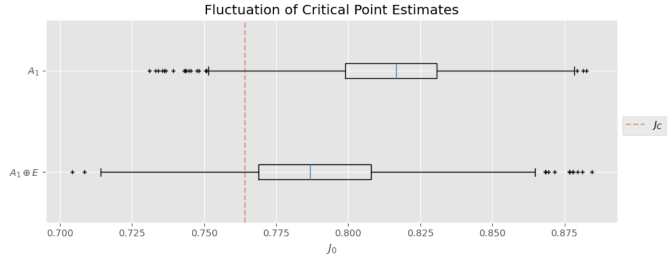

FIG. 8. Boxplot of predictions for J_C using an A_1 representation (1 dimension) versus an $A_1 \oplus E$ representation (5 dimensions). Analytical target value shown by dotted red line.

B. Learning the Flow versus the Monotone

In all our results so far, we have learned the RG flow function $\vec{f}(\vec{J}) = \vec{J}' - \vec{J}$. In their paper, however, Hou et al. learn instead the RG monotone function:

$$C(\vec{J}) : \mathbb{R}^n \mapsto \mathbb{R} \tag{14}$$

According to the C-theorem [15], the monotone $C(\vec{J})$ has the important property of always decreasing monotonically under the Ising model RG flow, and is guaranteed to exist. Learning $C(\vec{J})$ is attractive because of this theoretical soundness.

On the other hand, learning $\vec{f}(\vec{J})$ is theoretically shaky. It is related to the monotone by:

$$\vec{f}(\vec{J}) = \vec{\nabla} C(\vec{J}) \tag{15}$$

However, the $\vec{f}_\theta(\vec{J})$ we learn might not be, and usually is not, the gradient of any function.

Qualitative observations seem to suggest, however, that learning $\vec{f}(\vec{J})$ is good enough for our purposes, and we see no significant degradation in predictive abilities and accuracy.

Moreover, learning $\vec{f}(\vec{J})$ is significantly faster, not to mention easier to implement. This is because learning $C(\vec{J})$ requires computing $\vec{\nabla} C(\vec{J})$ in the forward pass for $\vec{J}' = \vec{J} + \vec{\nabla} C(\vec{J})$, as well as $\vec{\nabla}||\vec{\nabla} C(\vec{J})||^2$ during Hamiltonian Monte Carlo sampling. These first and second-order derivatives are very expensive for auto-differentiation to compute and backpropagate. Learning $\vec{f}(\vec{J})$, on the other hand, only requires computing at worst the first-order quantity $\nabla||\vec{f}(\vec{J})||^2$. The compute time difference is shown in Figure 9.

As a side note, learning the flow $\vec{f}(\vec{J})$ also permits the use of non-smooth non-linearities like ReLU in the neural network, which wouldn't be compatible with Newton's method if learning the monotone.

More work is needed to investigate any possible consequences of learning the flow instead of the monotone.

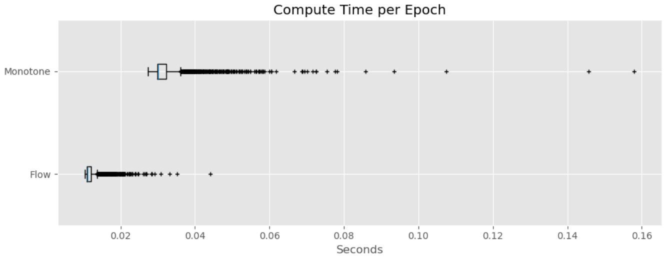

FIG. 9. Compute time per training epoch for learning the monotone $C(\vec{J})$ versus learning the flow $\vec{f}(\vec{J})$.

C. KL vs Contrastive Divergence

So far, we have used the contrastive divergence as our loss function, instead of the KL divergence, mostly to adhere to convention—the KL divergence is typically intractable to compute. However, in the case of the A_1 representation (1-dimensional) Lieb Ising model, the KL divergence is actually tractable to evaluate—there are only a few hundred spin configurations to sum over.

In the machine learning literature, there has been skeptism about the validity of the contrastive approximation [16–18], including assertions that it can introduce bias. We wish to elucidate whether or not this affects our learned model.

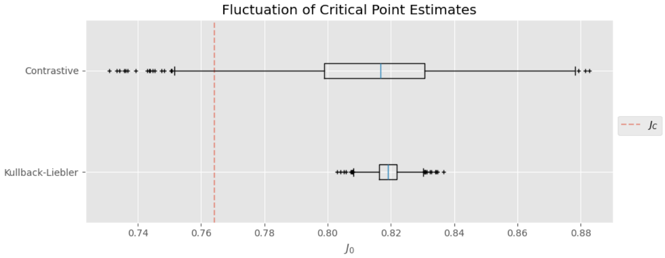

FIG. 10. Distributions of critical point predictions using the contrastive versus KL divergences as our loss function.

We observe in Figure 10 that, in fact, using the KL divergence only really affects the variance of our predictions, not the bias. Since there are other ways to reduce variance, such as decreasing learning rate, and variance is not a particularly pressing issue, we conclude that the extra compute cost needed to use the KL divergence is not worth it. Furthermore, the KL divergence does become intractable when using higher-dimension representations.

D. Divergence Hyperparameters

If we decide to use the contrastive divergence as our loss function, we want to be sure to compute it correctly and efficiently. There a few hyperparameters to control for:

- The number of samples to take from the models: the batch size, more or less
- The number of Gibbs sampling steps for the fine-grained model
- The number of Gibbs sampling steps for the coarse-grained model

The effect on A_1 critical point predictions from varying each of these hyperparameters, while holding all others constant, is shown in Figure 11.

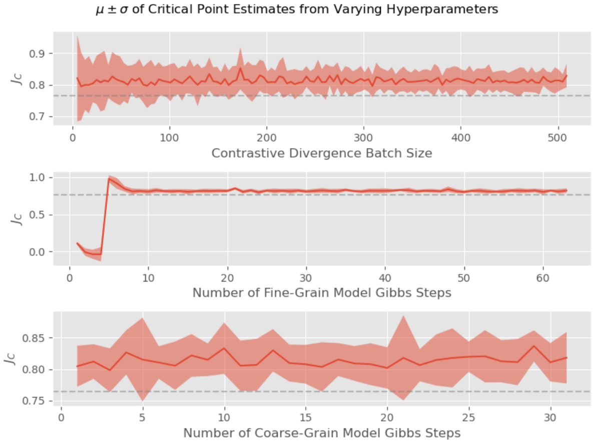

FIG. 11. Hyperparameter space explorations for three contrastive divergence hyperparameters.

We conclude that:

- The batch size should be fairly large to control variance, though there are diminishing returns past a few hundred samples.
- The number of fine-grained model Gibbs steps should be larger than a minimum threshold to control bias
- The number of coarse-grained model Gibbs steps can be minimal—1 is plenty

E. Sampler Hyperparameters

There are a number of hyperparameters to set for the HMC sampler. Here, we do a limited hyperparameter space exploration for the following:

- The number of trajectories: basically the sample batch size
- Number of solve steps: controls for how long the numerical integrator solves the Hamiltonian equations.
- Number of sampling steps: controls how many repeated, refining Monte Carlo iterations are performed.

The effect on A_1 critical point predictions from varying each hyperparameter, while holding all others constant, is shown in Figure 12.

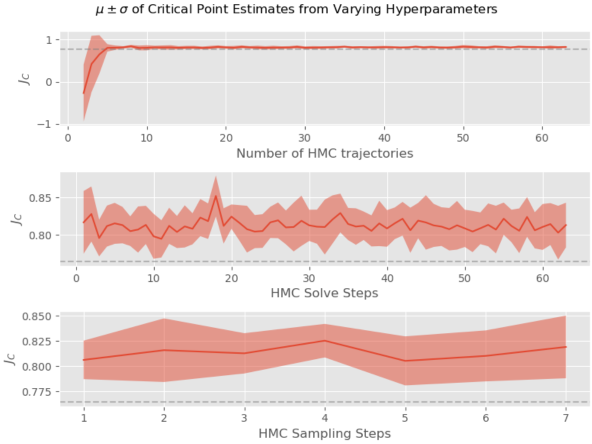

FIG. 12. Hyperparameter space explorations for three HMC divergence hyperparameters.

We conclude that:

- The number of trajectories should be set above the minimal threshold to reduce heavy biasing error as well as variance.
- The number of solve steps can be small.
- The number of sampling steps can be small.

IV. CONCLUSION

In this review, we introduced the concept of real-space renormalization, presented the Ising model problem, and demonstrated how to apply real-space renormalization to it. We then described a neural learning algorithm to approximate the renormalization group flow function over a Lieb lattice Ising model. We demonstrated its success at learning an accurate approximation, and observed its predictive abilities for critical points. We then discussed some aspects of the algorithmic design space, notably the reduction of bias via higher-dimensional representations, as well as the choice of loss function, whether to learn the flow or the monotone, and the role of sampler and contrastive divergence hyperparameters.

All in all, the MLRG algorithm represents both a new way to automatically determine renormalization group flows, as well as an exciting new paradigm of machine learning. In the future, it will be interesting to see further research clarifying aspects such as flow versus monotone learning, the effect of contrastive approximations on the true gradient, and the significance of increasing the representational dimension. There is also much follow-up work to do regarding applying this algorithm to other systems, such as different Ising models and maybe even non-Ising models as well as improving the algorithm to stably and efficiently handle higher-dimension representations.

For reference, our implementations can be found at the repository https://github.com/jshe2304/dlrg.

[1] W. Hou and Y.-Z. You, Machine learning renormalization group for statistical physics (2023), arXiv:2306.11054 [cond-mat.stat-mech].

[2] M. E. Fisher, The renormalization group in the theory of critical behavior, Rev. Mod. Phys. **46**, 597 (1974).

[3] K. G. Wilson, Renormalization group and critical phenomena. i. renormalization group and the kadanoff scaling picture, Phys. Rev. B **4**, 3174 (1971).

[4] S. G. Brush, History of the lenz-ising model, Rev. Mod. Phys. **39**, 883 (1967).

[5] E. Ising, Contribution to the Theory of Ferromagnetism, Z. Phys. **31**, 253 (1925).

[6] L. P. Kadanoff, Scaling laws for ising models near T_c, Physics Physique Fizika **2**, 263 (1966).

[7] G. E. Hinton, A practical guide to training restricted boltzmann machines, in *Neural Networks: Tricks of the Trade: Second Edition*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012) pp. 599–619.

[8] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359 (1989).

[9] Z. Shen, H. Yang, and S. Zhang, Optimal approximation rate of relu networks in terms of width and depth, Journal de Mathématiques Pures et Appliquées **157**, 101 (2022).

[10] M. Betancourt, A conceptual introduction to hamiltonian monte carlo (2018), arXiv:1701.02434 [stat.ME].

[11] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid monte carlo, Physics Letters B **195**, 216 (1987).

[12] G. E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Computation **14**, 1771 (2002).

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature **323**, 533 (1986).

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Pytorch: An imperative style, high-performance deep learning library (2019), arXiv:1912.01703 [cs.LG].

[15] A. B. Zomolodchikov, "Irreversibility" of the flux of the renormalization group in a 2D field theory, Soviet Journal of Experimental and Theoretical Physics Letters **43**, 730 (1986).

[16] Y. Bengio and O. Delalleau, Justifying and generalizing contrastive divergence, Neural Computation **21**, 1601 (2009).

[17] I. Sutskever and T. Tieleman, On the convergence properties of contrastive divergence, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010) pp. 789–795.

[18] M. A. Carreira-Perpinan and G. Hinton, On contrastive divergence learning, in *International workshop on artificial intelligence and statistics* (PMLR, 2005) pp. 33–40.