CS230

Deep Convolutional Generative Adversarial Network for Simulating the 2D Ising Model

Shujia Liang
Department of Chemistry
Stanford University
sjliang@stanford.edu

Fangze Liu
Department of Physics
Stanford University
fangzel@stanford.edu

Tianyi Liu
Department of Chemistry
Stanford University
tliu26@stanford.edu

Abstract

Generative adversarial network (GAN) shows superior performance in image generation. In this work we report using GAN to generate configurations of the 2-D Ising model. We found that GAN is able to learn the distribution of the Ising model at critical temperature. Potential applications for using GAN with Ising model include compression of information in Ising model and simulating large clusters, which is computationally expensive otherwise using traditional Monte Carlo.

1 Introduction

To study the 2-D Ising model, we consider a rectangular or square lattice (an $N_x \times N_y$ grid). Each site on the lattice contains one spin that either points up or down. The spin on each site interact with its nearest neighbors (up/down/left/right). It is favorable for two neighboring spins to point in the same direction (parallel) and unfavorable in the opposite direction (antiparallel). Mathematically, the model's Hamiltonian, which can be interpreted as the energy, is given by

$$H = \sum_{\langle \mathbf{i},\mathbf{j} \rangle} -J s_{\mathbf{i}} s_{\mathbf{j}} \tag{1}$$

where $s_{\mathbf{i}} = s(x_i, y_i) = \pm 1$, J sets the scale of interaction between all nearest neighbor spins $s_{\mathbf{i}} s_{\mathbf{j}}$, and the negative sign indicates that parallel spins lowers the energy, which is favorable. The Ising configuration $s_{\mathbf{i}} = s(x_i, y_i)$ can be seen as a 1-channel image with ± 1 on each pixel. This analogy enables us to train GAN with convolutional neural network (CNN), which are widely used in computer graphics, on the Ising model.

We are interested in generating Ising configurations at equilibrium. Statistical mechanics predicts that the Ising model at equilibrium takes various configurations such that its Hamiltonian for a specific configuration satisfies the Boltzmann distribution

$$P(s) = \frac{1}{Z} \exp^{-\frac{H(s)}{k_B T}} \tag{2}$$

where the partition function $Z = \sum_{\{s\}} \exp^{-\frac{H(s)}{k_B T}}$ normalizes the probability distribution, and k_B is a constant. In essence, the Boltzmann distribution requires more spins to be parallel when the Ising system is at low temperature. At high temperature, this requirement is less strict.

The primary goal of this project is to use GAN to generate equilibrium configurations of the Ising model at the critical temperature. The dataset is a collection of Ising spin configurations at the critical

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

temperature, generated with Monte Carlo simulation. Our network consists of one generator and one discriminator, with the same architecture defined in the work by Radford, et al [1]. After training, GAN takes random noise as input and generates 1-channel 2-D images. Then, each pixel value is converted to ± 1 based on the sign: positive number to $+1$ and negative number to -1.

A GAN Ising simulator is an interesting problem to study, as it can show the ability of neural network to capture the underlying physics of model systems. It may also be useful for compressing information of the Ising model. Specifying the configurations of the Ising model using pixel values involves $N_x \times N_y$ values. On the other hand, with GAN, the information is stored in a much smaller vector that is used as the input noise for GAN to generate configurations.

2 Related work

A few machine learning techniques have been used to learn and generate Ising model configurations, such as Restricted Boltzmann Machines [2] [3], deep belief network [3] and GAN [4]. There has also been work on interpretation of a neural network that learns model systems such as the Ising model [2][5]. All of these techniques were able to simulate realistic Ising configurations. Quantities calculated from the Ising states produced by these techniques, such as average energy and average magnetization as a function of temperature, showed similar behavior with those calculated from Monte Carlo simulation. In Liu, et al.'s work, they also showed the distribution of these quantities at a specific temperature, which agreed well with data from Monte Carlo, demonstrating that GAN is a powerful method for generating Ising configurations. In our work, we aim to reproduce the work, with a few modifications, by Liu, et al. on GAN, the code of which is not available. Specifically, we try various loss functions and optimization schemes, as well as using simpler architecture for GAN.

3 Dataset and Features

We collect data from Monte Carlo simulations. Monte Carlo simulation samples the configuration space of the Ising model using detailed balance that satisfies equation (2). It produces reliable Ising configurations with the correct distribution that can be obtained from various other analytical treatment of the Ising model. Therefore, the configurations from Monte Carlo simulations are used as the ground truth. GAN is trained on data from the critical temperature. Some ground truth Ising configurations are shown in figure 1. Since these configurations only takes value ± 1 on each pixel, we did not do any preprocessing.

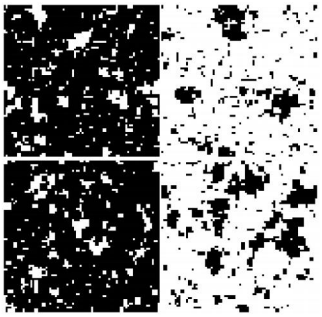


Figure 1: Some real configurations of the 2D Ising model

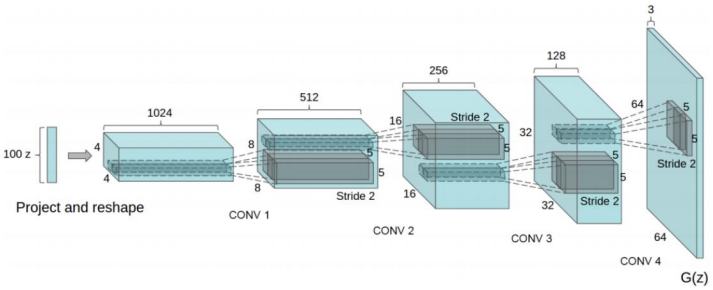


Figure 2: Generator architecture

4 Methods

We use a generative adversarial network with one generator and one discriminator, following the original DCGAN architecture [1]. The generator consists of 4 layers, each containing a transposed convolution (also known as a fractionally-strided convolution), a batch normalization and a ReLU activation. The transposed convolution is used to upsample from the low dimensional latent space to high dimensional Ising configuration space. The last layer is followed by a tanh activation to set the value on each pixel to within $(-1, 1)$. The discriminator network shares a similar architecture,

with transposed convolution replaced by regular convolution and ReLU activation replaced by leaky ReLU. The regular convolution downsamples the Ising configuration space, and leaky ReLU helps the learning process of discriminator by avoiding zero gradient. The last layer is followed by a sigmoid activation to classify the input as real/fake. A schematic drawing of the generator architecture is shown in figure 2 [1].

The basic principle of a generative adversarial network is to train the generator to output realistic images, in our case the Ising spin configurations, and to train the discriminator to classify an input image as real or fake. As training proceeds, the generator produces more realistic images and the discriminator is more capable of identifying fake images. Formally, this procedure can be expressed as [6]

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{3}$$

According to this, the discriminator tries to maximize the probability that it classifies real images $x \sim P_{data}$ as real and fake images $G(z)_{z\sim p_z(z)}$ as fake, while the generator tries to minimize the probability that fake images are classified as fake. In practice, this is realized by minimizing the loss of the discriminator. The loss function we have tried includes binary cross entropy (BCE), Wasserstein [7] and hinge loss [8].

During training, minibatches of real Ising configurations with real labels (1) and fake configurations with fake label (0) are fed into the discriminator and their weights updated according to discriminator loss. Then the discriminator is fed with fake images again, but this time with real label (1). After the discriminator loss is back propagated, the weights of only the *generator* are updated. This process essentially tells the generator to adjust its weights so that the images it produces are more likely to be classified as real by the discriminator. For BCE loss, this is equivalent to maximizing $\log(D(G(z)))$ instead of minimizing $\log(1-D(G(z)))$. The latter suffers from small gradients in early stage training, so it is beneficial to use the former.

We have also experimented with the Wasserstein loss. In this case the basic idea is that, provided that the discriminator is 1-Lipschitz, the output is not the probability of input being real/fake, but a score the discriminator gives to the input images. Higher score means more realistic image. To implement this, the sigmoid activation in the last layer of the discriminator is removed. Therefore, the loss functions that the discriminator and generator need to maximize are, respectively

$$L_D^{WGAN} = -\mathbb{E}_{x\sim p_{data}}[D(x)] + \mathbb{E}_{\hat{x}\sim p_{gen}}[D(\hat{x})] \tag{4}$$

$$L_G^{WGAN} = -\mathbb{E}_{\hat{x}\sim p_{gen}}[D(\hat{x})] \tag{5}$$

and the weights of the discriminator are manually "clipped" (only takes on a range of values) to satisfy the 1-Lipschitz constraint.

Besides weight clipping, we also have tried various other regularization techniques, such as gradient penalty [9] and spectral normalization [8]. The loss function of the Wasserstein GAN with gradient penalty is

$$L_D^{WGAN-GP} = -\mathbb{E}_{x\sim p_{data}}[D(x)] + \mathbb{E}_{\hat{x}\sim p_{gen}}[D(\hat{x})] + \lambda\mathbb{E}_{z\sim p_z}[(||\nabla_z D(z)||_2 - 1)^2] \tag{6}$$

where z is the random sample. It penalizes gradient deviating from the target norm 1. Also, batch normalization in the discriminator, which relates all the samples in the same batch, can hurt the performance of gradient penalty. So when gradient penalty is applied, batch norm should be absent in the discriminator.

Spectral Normalization is another technique for imposing 1-Lipschitz and stabilizing the training of the discriminator by normalizing the weights of the discriminator [8]. Experiments show that spectral normalization performs better if the used with hinge loss

$$V_D(\hat{G}, D) = \mathbb{E}_{x\sim p_{data}}[min(0, -1 + D(x))] + \mathbb{E}_{z\sim p(z))}[min(0, -1 - D(\hat{G}(z)))] \tag{7}$$

$$V_G(G, \hat{D}) = -\mathbb{E}_{z\sim p(z)}[\hat{D}(G(z))] \tag{8}$$

Furthermore, the difference in a physical observable between the real and generated sample proved to be important (see experiment section). We choose the average magnetization, which is simply the sum of all the pixel values divided by lattice size. Therefore, the following may be added to generator loss

$$L_{m_z} = \mathbb{E}[(\langle m_z\rangle_{real} - \langle m_z\rangle_{gen})^2] \tag{9}$$

where $\langle m_z\rangle = \frac{1}{Nx\times Ny}\sum_{i,j} s_{i,j}$

5 Experiments/Results/Discussion

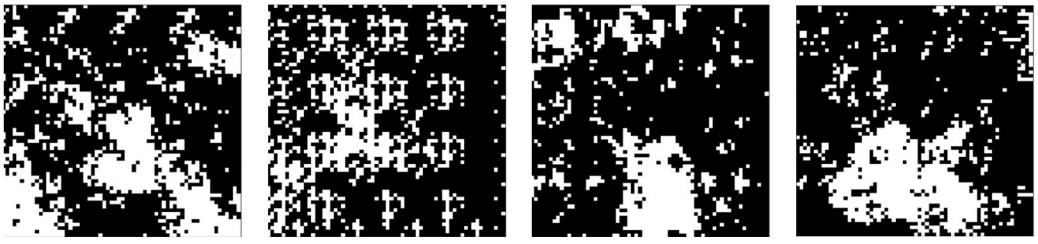
The hyperparameters are primarily learning rate and batch size. The choice of optimizers are Adam and RMSProp, and the current state-of-the-art regularization techniques include weight clipping, gradient penalty and spectral normalization. Since our goal is to generate realistic spin configurations, the primary standard by which we tune the hyperparameters and choose optimizers is to compare generated figures with real figures from simulation. We train GAN with Ising samples at the so-called critical temperature. The spin configurations at critical temperature is very random, with fluctuating domains (i.e. ferromagnetic domains) of aligned spins. This means that in the short range the spins want to be parallel, as dictated by eq (1), but in the long range the propensity to line up is disrupted by thermal fluctuations, which are prominent at critical temperature. Therefore, we manually examine the generated pictures and look for two features: is the configuration random enough and are there many ferromagnetic domains? Pictures satisfying these two standards are considered realistic.

First, we have tried all combinations of various loss functions and optimizers, coupled with regularization technique empirically shown to work well with the specific loss. The configurations generated are shown in figure 3. All figures suffer from a similar symptom: they are not random enough. For example, with BCE loss, the spin configuration has four major clusters and does not have many black pixels in regions with predominantly white pixels. Wasserstein loss with gradient penalty or spectral normalization generate configurations that seem to have an underlying periodic structure, which is not true for Ising model at critical temperature. Hinge loss with spectral normalization seems to generate the more realistic figures.

(a) BCE (b) WGAN-GP (c) SNGAN - W loss (d) SNGAN - Hinge loss

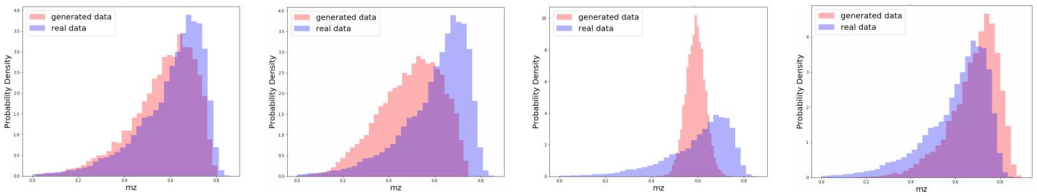
Figure 3: Generated Ising samples using different loss functions and regularization tecniques: (a) Binary cross entropy, (b) Wasserstein with gradient penalty, (c) spectral normalization GAN with Wasserstein loss, (d) spectral normalization GAN with Hinge loss

In addition, similar to applications of GAN in other areas such as image generation, our GAN also suffers from mode collapse. For a model with any specific loss, images similar to the shown images were given out by the generator even when fed with different random noises. To address this, we note that the high randomness of the Ising model at critical temperature is manifested in physical observables easy to calculate, and so we believe that using one of the observables as a criterion, and adding the difference in such observable between real and fake samples to the loss function, might be beneficial. Since the observable in each real sample configuration is a random variable relating to a well-defined probability distribution (see eq (2)), by incorporating this observable we in effect help GAN learn the distribution and generate samples with sufficient randomness and great diversity.



(a) BCE (b) WGAN-GP (c) WGAN (d) SNGAN - Hinge loss

Figure 4: Generated Ising samples using different loss functions and regularization techniques, and also an added penalty term for average magnetization in the generator loss

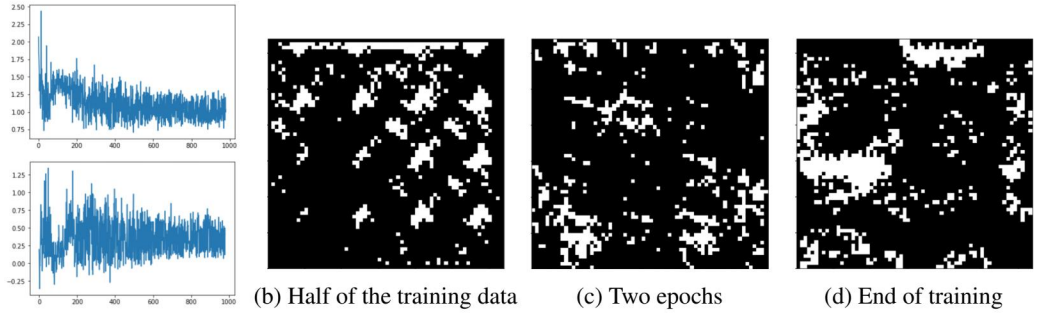


(a) BCE (b) WGAN-GP (c) WGAN (d) SNGAN - Hinge loss

Figure 5: Histogram of the average magnetization of a collection of generated samples using different loss, regularization and with added term that take into account average magnetization in the losses

Indeed, when L_{m_z} from eq (9) was added to the generator loss, the quality of the generated samples was greatly improved, as shown in figure 4. To quantitatively evaluate our model, we also plot the histograms of the average magnetization $\langle m_z \rangle$ of the generated and real samples, in figure 5. GAN with hinge loss and spectral normalization shows superior performance over the others in that the configuration it generates is noticeably the most random and it produces a distribution of $\langle m_z \rangle$ in good agreement with that of the real data. We believe that this is due to the robustness of hinge loss towards outliers in the data, and so it performabce well for Ising model, which does have highly random configurations.

We have also monitored how the loss of the generator and discriminator change with training iterations. As seen in other GAN training, the losses do not provide much useful information about the convergence of the model. As an example, consider the losses of the generator and discriminator when hinge loss is used (figure 6). Both losses fluctuate about some values and show no sign of converging, while in fact as GAN is trained with more iterations, it produces more realistic images.



(a) Loss (b) Half of the training data (c) Two epochs (d) End of training

Figure 6: (a) Uppder: discriminator loss, lower: generator loss; (b) Generated sample after half of the training set was used; (c) Generated sample after two epochs; (d) Generated sample at the end of training

6 Conclusion/Future Work

We have successfully trained GAN with hinge loss and spectral normalization on spin configurations of the Ising model at critical temperature. The generated configurations are visually similar to real samples, and the distribution of average magnetization is highly similar to that of the real Ising model. Adding to generator loss a penalizing term for the difference in average magnetization between fake and real samples shows great improvement. In the future, the model should be tested at various other temperatures, and may be used for compressing information of the Ising model.

7 Contributions

S. L. made the poster and edited the report. F. L. and T. L. edited the poster and wrote the report. All authors wrote part of the GAN and Monte Carlo code. GAN and Monte Carlo code can be found at `https://github.com/tliu26/Ising_dcgan`

References

[1] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[2] Guido Cossu, Luigi Del Debbio, Tommaso Giani, Ava Khamseh, and Michael Wilson. Machine learning determination of dynamical parameters: The ising model case. *arXiv preprint arXiv:1810.11503*, 2018.

[3] Alan Morningstar and Roger G Melko. Deep learning the ising model near criticality. *The Journal of Machine Learning Research*, 18(1):5975–5991, 2017.

[4] Zhaocheng Liu, Sean P Rodrigues, and Wenshan Cai. Simulating the ising model with a deep convolutional generative adversarial network. *arXiv preprint arXiv:1710.04987*, 2017.

[5] Sebastian J Wetzel and Manuel Scherzer. Machine learning of explicit order parameters: From the ising model to su (2) lattice gauge theory. *Physical Review B*, 96(18):184410, 2017.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[8] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.