Simulate Quantum Annealing Ising Model with Conditional Generative Adversarial Network

Pengyuan Zhai[1,2]

[1]*Department of Civil and Environmental Engineering, University of California, Berkeley, CA, United States*

[2]*Department of Industrial Engineering and Operations Research, University of California Berkeley, CA, United States*

Abstract: *This study aims to create a GAN-based Ising simulator (SQA-GAN) to produce realistic Ising states given a target transverse field strength. The Ising state generator will implicitly learn the data distribution from conventional Path-integral Monte Carlo simulations for a three-dimensional quantum annealing Ising model. We tested and compared the SQA-GAN's generative capabilities based on a* 32×32 *spin glass system and found that (add more when all experiments are done)...*

1 Background & Motivations

Replicating complex system behaviour has been an increasing challenge in the data explosion era...

Deep learning has shown great potential in learning and describing complicated physical systems in quantum physics and astronomical sciences (REF)...

Recent development in computer vision has revealed the great potential of convolutional neural networks (CNNs), which were successful in learning many body systems (REF) and predicting phase transitions and calculating system Hamiltonians under supervised learning frameworks...

Additionally, generative models were used to simulate natural and human behaviours through hidden Markov model, variational autoencoders and reinforcement learning under unsupervised learning frameworks. Generative Adversarial Networks (GANs) learn the distribution of training data in a zero-sum game-theoretical framework, allowing to construct models to imitate realistic stochastic physical systems (REF)...

2 Related Work

The concept of GAN was first introduced by Goodfellow et al. (2014a), which is a generative model that generates new data from the learned distribution. Unlike conventional DL models, GAN consists of two networks, namely the *generator* and the *discriminator*, where the generator creates synthetic data and the discriminator classifies an input sample as "real" or "synthetic." GAN uses adversarial training, where each network aims to minimize the gain of the opposite side while maximizing its own. Ideally, both the generator and the discriminator converge to the Nash equilibrium (Goodfellow, 2016), where the discriminator gives equal predictive probabilities to real and synthesized samples. Until now, GAN has been applied to many computer vision (CV) tasks, e.g., fake image generation (Radford et al., 2015), image-to-image translation (Isola et al., 2017), and medical imaging synthesis, reconstruction, and classification (Yi et al., 2019).

Liu and Rodrigues (REF) developed an Ising simulator built by a GAN framework to simulate the 2D Ising model near critical temperature and verified its effectiveness in generating realistic Ising states compared to those generated by a Monte Carlo (MC) simulation... (more previous studies to be added)

The Ising model is often used to describe natural systems in physics, and many optimization problems can be mapped into physical systems described by the Ising model whose ground states provide the solutions to the optimization problems. Examples include travel- ing salesman problem, portfolio optimization, integer factoring, social economics network, protein folding, protein modeling and statistical genetics. See Irback, Peterson and Potthast (1996), Majewski, Li and Ott (2001), McGeoch (2014) and Stauffer (2008).

3 Ising Model Monte Carlo Simulations

3.1 Classical Ising Model

In statistical physics, the Ising model is described as an ensemble of binary spins with coupling interactions in some lattices. For A classic 2-dimensional lattice Ising model with n rows and m columns, there are in total $b = n * m$ sites. At each lattice site, site variable s_i stands for a binary random variable indicating the spin position (pointing up or down), i.e., $s_i \in \{+1, -1\}, \forall i = 1, ..., b$. The Hamiltonian of the classical Ising model is given by

$$H_I^c(s) = -\sum_{<i,j>} J_{ij} s_i s_j - \sum_j h_j s_j \tag{1}$$

where J_{ij} stands for the coupling interaction between sites i and j, and h_j describes an external magnetic field on site j. For a given configuration $\mathbf{s}$, the energy of the Ising model is equal to $\mathbf{H_I^c}(\mathbf{s})$.

The probability of a given configuration $\mathbf{s}$ at a given absolute temperature T follows a Boltzmann (or Gibbs) distribution

$$P_\beta(\mathbf{s}|\mathbf{T}) = \frac{\mathrm{e}^{-\beta \mathbf{H_I^c}(\mathbf{s})}}{\mathrm{Z}_\beta}, \mathrm{Z}_\beta = \sum_{\mathbf{s}} \mathrm{e}^{-\beta \mathrm{H_I^c(s)}}, \beta = (\mathrm{k_B T})^{-1}, \quad (2)$$

where $\beta = (k_B T)^{-1}$ is the inverse temperature, where k_B is the Boltzmann constant and T is a given absolute temperature. In this work, *temperature* refers to $k_B T$ which is the fundamental temperature of the system with units of energy, and β is simply its reciprocal.

3.2 Classical Annealing

A combinatorial optimization problem can be cleverly mapped to an Ising model, whose ground state (a configuration that minimizes the energy function $\mathbf{H}_I^c(\mathbf{s})$) corresponds the optimal objective. In a complex system where exhaustively searching for the minimum is computationally prohibitive, annealing approaches such as Simulated Annealing (SA) are used to probabilistically explore the search space with repeated Markov Chain Monte Carlo (MCMC) iterations. The Metropolis-Hastings algorithm is a popular MCMC method that generates configuration samples (Ising states) from the Boltzmann distribution at slowly-decreasing temperatures.

Steps of the SA algorithm:

(1) Randomly and independently initializes the spins with +1 and -1.

(2) At the kth sweep (one sweep is a complete update of all spins), for spin i, attempt to flip its state, keeping other spins unchanged. Calculated the energy change, $\triangle E_i^{(k)}$.

(3) The new state for spin i is accepted with probability $min\{1, \exp(-\triangle E_i^{(k)}/T_k)\}$

3.3 Quantum Ising Model

3.3.1 Quantum System Representation

Computers rely on physical systems to represent digits. Classical computers encode bits 0 and 1 by low and high voltages. Analog to bits 0 and 1 in classic computation, quantum computations rely on qubits $|0\rangle$ and $|1\rangle$. Quantum superposition allows qubits to encode ones and zeros simultaneously. While a classical bit can only be either 0 or 1, a qubit can be a superposition of both $|0\rangle$ and $|1\rangle$, which is realized by a particle's quantum spin where $|0\rangle$ and $|1\rangle$ correspond to the up spin and down spin respectively. A superposition qubit is $\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 are two complex numbers satisfying $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Thus, a qubit can be represented by a unit vector $[\alpha_0, \alpha_1]^T$ in $\mathbb{C}^2$, and $|0\rangle$ and $|1\rangle$ are the orthonormal basis or the computational basis. Due to the non-observability of qubits, we can only observe 0 with probability $|\alpha_0|^2$, or 1 with probability $|\alpha_1|^2$.

The complex space increases exponentially with respect to the number of qubits. In the case of a b-qubit system, the computational basis takes the form $|x_1 x_2 ... x_b\rangle, x_j \in \{+1, -1\}, \forall j \in \{1, ..., b\}$, eg., when $b = 2$, the computational basis are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. A unit vector $|\psi\rangle = [\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}]^T$, with $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$, $|\psi\rangle \in \mathbb{C}^{2^2}$, represents a specific superposition state of this 2-qubit system.

3.3.2 Quantum Annealing

As mentioned in Section 3.2.1, the quantum state of a b-qubit quantum system is represented by a unit vector $|\psi\rangle \in \mathbb{C}^{2^b}$. The continuous time evolution of $\psi(t)\rangle$ is governed by the famous Schrödinger Equation:

$$|\psi(t)\rangle = \mathrm{e}^{-\sqrt{-1}\mathbf{H_t}}|\psi(0)\rangle, \quad (3)$$

where the quantum Hamiltonian, $\mathbf{H}_t \in \mathbb{C}^{2^b \times 2^b}$ is a time-dependent Hermitian matrix. The possible energies of the quantum system corresponds to the eigenvalues of the quantum Hamiltonian, and the ground state is the eigenvector corresponding to the smallest eignvalue.

To analogously represent a quantum Ising model using the classical Ising model idea in Section 3.1, we replace each lattice position varible $s_i = \pm 1$ by a Pauli matrix

$$\sigma_j^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (4)$$

The quantum Hamiltonian of the quantum Ising model thus becomes:

$$\mathbf{H}_I^q = -\sum_{<i,j>} J_{ij}\sigma_i^z\sigma_j^z - \sum_j h_j\sigma_j^z, \quad (5)$$

where $J_i j$ is the Ising coupling of lattice positions i and j, and h_j is the local field on jth lattice position. It is worth noting that $\sigma_i^z\sigma_j^z$ denotes a tensor product, which makes the first term in (5) a diagonal matrix. $\mathbf{H}_I^q$ is thus also a diagonal matrix (the second term of (5) is diagonal).

The eigenvalues of $\mathbf{H}_I^q$ are its diagonal entries, which actually corresponds all the 2^b possible values of a classical Hamiltonian $\mathbf{H}_I^c(s)$ with b total spins (REF). Finding the minimal energy of the quantum Hamiltonian is equivalent to finding the minimal energy of the classical Hamiltonian.

However, unlike the classical Ising model, an additional transverse magnetic field orthogonal to the Ising axis is introduced to drive the transitions between the up and down states of each spin, this added field turns the system behaviour from classical to quantum (REF). The transverse magnetic field is governed by a quantum Hamiltonian

$$\mathbf{H}_x = -\sum_j \sigma_j^x, \quad (6)$$

where σ_j^x is a Pauli matrix in the x axis:

$$\sigma_j^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{7}$$

During quantum annealing, the system evolves from the initial Hamiltonian $\mathbf{H}_X$ to the final target Hamiltonian through annealing schedules $A(t)$ and $B(t)$, which is realized by turning on a off magnetic fields adiabatically (as in the D-wave quantum computer). (REF)

$$\mathbf{H}_D(t) = A(t)\mathbf{H}_X + B(t)\mathbf{H}_I^q. \tag{8}$$

According to the quantum adiabatic theorem, the system tends to remain in ground states of the instantaneous Hamiltonian through quantum tunneling. Thus at the end of quantum annealing, if the system is in a ground state of the final Hamiltonian, an optimal solution is obtained by measuring the system (REF):

3.3.3 Simulated Quantum Annealing

The quantum Hamiltonian's size increases exponentially with the number of qubits in the system. Simulating the quantum state evolution requires to exponentiate such exponentially large, time-dependent and non-commutable Hamiltonian matrices, which is prohibitive by classical computing. Simulated Quantum Annealing (Martonák, Santoro and Tosatti (2002)) approximates the partition function for the quantum annealing Hamiltonian through the path-integral technique using the Trotter formula. Specifically, SQA maps the transverse field quantum Ising model to a classical (2+1)-dimensional anisotropic Ising model with Hamiltonian

$$\mathbf{H}_{aI}^c(s) = -\sum_{l=1}^{\tau}[B(t)\sum_{<i,j>} J_{ij}s_{il}s_{jl} + J(t)\sum_j s_{jl}s_{j,l+1}], \tag{9}$$

where $s_{jl} = \pm 1$, $\tau \in \mathbb{Z}$, l is the index for an extra imaginary-time dimension, and J_{ij} are the couplings between the spins in the original 2-dimensional Ising model. Additionally, $J(t)$ is the coupling along the imaginary-time dimension:

$$J(t) = -\frac{\tau T}{2}\ln[\tanh(\frac{A(t)}{\tau T})], \tag{10}$$

where $A(t)$ and $B(t)$ are the same annealing schedules in the original quantum annealing formulation.

Due to the extra imaginary-time dimension, the MCMC method should run the Metropolis-Hastings algorithm in two directions: 1. the local update of b spins at a fixed imaginary-time index, i.e., updating $\mathbf{s}_l = \{s_{il}, i = 1, ..., b\}$ with a fixed l, where $\mathbf{s}_l$ is called the lth Trotter slice. 2. the global update of the same spin position in all Trotter slices, i.e., fix i and update all s_{il} for each l value.

The SQA Algorithm:

1. Initialize spins in all Trotter slice with $+1$ and -1 randomly and independently. Burn-in simulations can be added.

2. Locally and globally update spins one by one for each trotter slice. A complete update of all spins locally and globally is one sweep.

At the time of the kth sweep (denoted by t_k):

(i). Local Update: For each spin i in each Trotter slice l, attempt to flip from its old state $s_{il}^{(k-1)}$ to the new state $s_{il}^{(k)} = -s_{il}^{(k-1)}$, keeping all other spins unchanged. The change of energy is: (formulation subject to change)

$$\begin{aligned} \triangle E_{1il}^{(k)} = &-B(t_k)[\sum_{j=1}^{i-1} J_{ij}s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)}) \\ &+ \sum_{j=i+1}^{b} J_{ij}s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)})] \\ &- J(t_k)[s_{il}^{(k)}s_{i,l+1}^{(k)} + s_{i,l-1}^{(k)}s_{il}^{(k)} \\ &\quad - s_{il}^{(k-1)}s_{i,l+1}^{(k-1)} - s_{i,l-1}^{(k-1)}s_{il}^{(k-1)}]. \end{aligned} \tag{11}$$

The local update accepts the new state $s_{il}^{(k)}$ with probability $min\{1, exp[-E_{1il}^{(k)}/(\tau T)]\}$

(ii). Global Update: Once the local update is done for all spins in all Trotter slices, iterate though each spin position i and attempt to flip states (for given i) $\{s_{il}^{(k-1)}, l = 1, ..., \tau\}$ to new states $\{s_{il}^{(k)} = -s_{il}^{(k-1)}, l = 1, ..., \tau\}$, keeping all other spins unchanged. Calculate the change of energy as: (formulation subject to change)

$$\begin{aligned} \triangle E_{2i}^{(k)} = -\sum_{l=1}^{\tau} B(t_k)[&\sum_{j=1}^{i-1} J_{ij}s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)}) \\ &+ \sum_{j=i+1}^{b} J_{ij}s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)})]. \end{aligned} \tag{12}$$

The global update accepts the new states $\{s_{il}^{(k)}, l = 1, ..., \tau\}$ with probability $min\{1, exp[-\triangle E_{2i}^{(k)}/(\tau T)]\}$.

3. When all sweeps are complete, evaluate the original classical Hamiltonian, use the first Trotter slice at the last sweep and obtain $\mathbf{s}^{(k)} = \{s_i^{(k)}, i = 1, ..., b\}$, and evaluate $\mathbf{H}_I^c(\mathbf{s}^{(k)})$.

4 GAN-based Implicit Learning Models

4.1 Basics of GAN

GAN consists of a minimax game between the generator and the discriminator. Let $x \in \mathbb{R}^d$ be a sample, then $x_r \sim p_{data}$ is a sample from the the real data distribution and $x_g \sim p_g$ is a generated sample from the GAN-learned, synthetic data distribution. The generator G with parameters θ_G is trained to synthesize samples that mimic the real sample distribution, p_{data}, by mapping the noise vector (latent variable), $z \sim p_z$, to a synthesized sample $x_g = G(z; \theta_G)$, $x_g \sim p_g$. The discriminator D with parameters θ_D takes in a sample $x \in \mathbb{R}^d$ (either real or synthesized) and outputs $D(x; \theta_D)$, which is the predictive probability that x comes from p_{data} rather than p_g.

During the training, G and D compete with each other according to:

$$\min_G \max_D E_{x\sim P_{data}(x)}[\log(D(x))]+E_{z\sim P_z(x)}[\log(1-D(G(z)))] \tag{13}$$

In Equation (13), the first term is the negated cross-entropy between $p_{data}(x)$ and $D(x)$, whose value is positively associated with D's ability of correctly predicting real samples as from the real data distribution $p_{data}(x)$; the second term is the negated cross-entropy between $p_z(z)$ and $1-D(G(z))$, where $1-D(G(z))$ is D's predictive probability that a synthesized sample $x_g = G(z)$ is indeed considered as "synthetic," i.e., $x_g \sim p_g$. D aims to maximize its discriminative power characterized by both terms, while the generator G tries to undermine D's performance by synthesizing realistic samples to trick D (minimizing the second term).

Both D and G can be parametrized by deep neural networks or CNNs, and they are trained and optimized alternatively according to Equation (13) until reaching the optima or designated number of iterations.

4.2 Conditional GAN

Conditional GAN (CGAN) was introduced by Mirza et al in 2014 (REF). A piece of additional information y (such as class labels, or data from different modality) is fed into the generator and discriminator in order to direct the data generation process (Figure 1). CGAN aims to tackles two challenges: (i) the difficulty of training GANs in cases of extremely large numbers of predicted output categories. (ii) learning one-to-one mappings from input to output, eg., learning different tags that could appropriately be assigned to a given image.

The two-player minimax game objective function of CGAN is:

$$\min_G \max_D E_{x\sim P_{data}(x)}[\log(D(x|y))]+E_{z\sim P_z(x)}[\log(1-D(G(z|y)))] \tag{14}$$

5 Learning Simulated Quantum Annealing with GAN

5.1 Dataset

This study aims to establish a conditional GAN-based framework that implicitly learns the distribution of the quantum Ising states output by the simulated quantum annealing (SQA) algorithm at a given transverse field strength Γ, which is controlled by the annealing schedule $A(t)$. We focus on the the spin glasses example (REF Rieger and Young, 1996), whose transverse field Ising model is defined by the Hamiltonian

$$\mathbf{H} = -\sum_{\langle ij\rangle} J_{ij}\sigma_i^z\sigma_j^z - \Gamma\sum_i \sigma_i^x, \tag{15}$$

which is mapped to a (2+1)-dimensional anistropic Ising model with SQA:

$$\begin{aligned}\mathbf{H}_{aI}^c(s) = -\sum_{l=1}^{\tau}[\sum_{<i,j>} J_{ij}s_{il}s_{jl} + J(t)\sum_j s_{jl}s_{j,l+1}],\\ J(t) = -\frac{\tau T}{2}\ln[\tanh(\frac{A(t)}{\tau T})].\end{aligned} \tag{16}$$

At each sweep k, the transverse field strength Γ is equal to $A(t_k)$, and the SQA algorithm outputs an Ising "cube" of size n by n by τ, where n is the number of rows or columns in a Trotter slice and τ is the total number of Trotter slice. The total number of spins in each Trotter slice is thus $b = n^2$. We herein represent each of such simulated Ising "cubes" as $\mathbf{x}$, corresponding to the real data samples for GAN, i.e., $\mathbf{x} \sim P_{data}(\mathbf{x})$.

The Ising cubes output by the SQA model is analogous to the image data for computer vision (CV). The numbers of rows and columns correspond to the image height and width, and the number of Trotter slices is the number of "image" channels. Therefore, a Convolutional Neural Network (CNN) can be readily applied to the input data, which extracts the high-level features of each Ising cube and forms a feature map subsequently learnt by a neural network. Such CNN is the basic form of the GAN discriminator (REF?).

5.2 GAN Input Conditional Embedding

The classical GAN generator uses a dense layer and multiple deconvolutional layers to map the input noise vector $z \sim p_z$ to a synthetic data sample $x_g = G(z;\theta_G)$. The input z vector first goes through a dense layer, whose output is reshaped to a 3-dimensional (width, height, channel) feature map $fm_1(z)$. Then, multiple deconvolutional layers transform $fm_1(z)$ into a synthetic sample $G(z)$.

Inspired by CGAN, we supply additional information about the target transverse field strength Γ to the generator to direct its data generation process. The generator transforms the input Γ value by feeding it to a single dense layer with enough neurons so that its output can be reshaped to the same dimensions as $fm_1(z)$ (call it $fm_y(\Gamma)$). Finally, the concatenated feature map, i.e., $fm_1(z)\bigoplus fm_y(\Gamma)$, is then fed through the subsequent deconvolutional layers to generate the conditional synthetic sample $G(z,\Gamma)$

5.2.1 Discriminator Loss

As a preliminary study, we combine loss functions across existing literature for learning the SQA Ising data.

Supervised Regression Loss The data labels of interest (transverse field strength, Γ) is continuous. Converting continuous labels to some number of discrete classes not only increases the quantization error of training, but also requires large discriminator outputs, decreasing the classification and

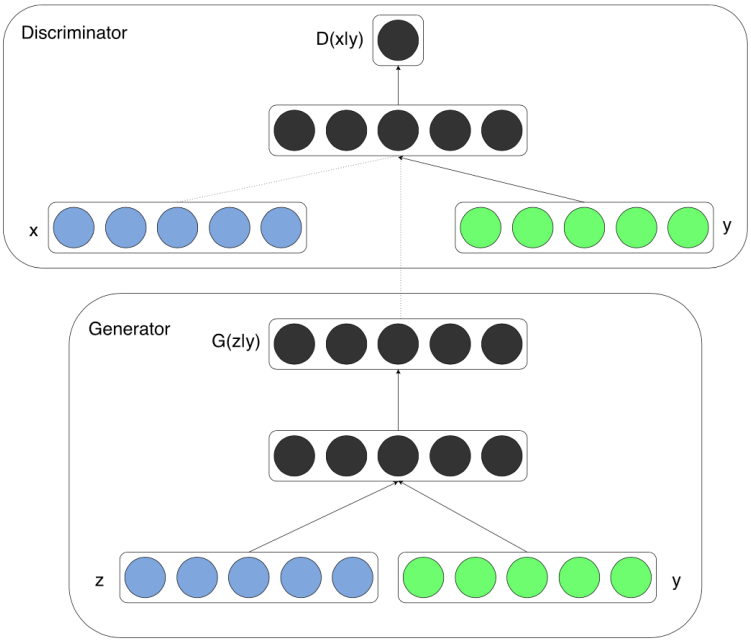

Figure 1: Conditional adversarial net

computing efficiency. In 2018, Rezagholiradeh Haidar proposed a regression-based GAN (REG-GAN), which requires the discriminator to have two outputs: one for predicting the continuous label, the other for predicting whether the given data is real or fake. The discriminator loss is naturally broken down to two parts: a supervised loss for measuring the regression error between the true and predicted label, and an unsupervised loss for measuring how well the discriminator tells apart fake data from real data:

$$\begin{aligned} L^{(D)}_{REG-GAN} &= L^{(D)}_{supervised} + L^{(D)}_{unsupervised} \\ L^{(D)}_{supervised} &= ||y - \hat{y}||_2 \\ L^{(D)}_{unsupervised} &= \mathbb{E}_{x\sim p_{data}(x)}[(1 - D(x))^2] \\ &\quad + \mathbb{E}_{z\sim p_z(z)}[D(G(z))^2], \end{aligned} \tag{17}$$

where y is a vector of the ground-truth field strengths of all real data, and $\hat{y}$ is a vector of the corresponding regression predictions. It is worth noting that in their work (ref), $L^{(D)}_{unsupervised}$ uses the least-square loss function as in Mao et al, which we will replace with the hinge loss described below.

Hinge Loss and Spectral Normalization in Conditional GAN Settings Various studies have proposed techniques to stabilize the GAN training, which cover multiple aspects from architecture designs (Radford et al., 2015; He et al., 2016), loss functions (Arjovsky et al., 2017; Mao et al., 2017a), to regularization (Arjovsky et al., 2017; Gulrajani et al., 2017; Mescheder et al., 2018; Miyato et al., 2018). Lipschitz regularization (Gulrajani et al., 2017; Miyato et al., 2018) is one of the most popular regularization methods which has also shown great potential.

Miyato et al (ref) proposed the spectral normalization (SN) method to impose Lipschitz regularity by normalizing the weight matrix W by its spectral norm $\sigma(W)$:

$$\overline{W}_{SN}(W) := W/\sigma(W) \tag{18}$$

Experiments show that SN performs better with the GAN hinge loss (ref Miyato):

$$\begin{aligned} L^{(D)}_{hinge} &= -\mathbb{E}_{x\sim p_{data}}[min(0, -1 + D(x))] \\ &\quad - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[min(0, -1 - D(G(z,y)))] \\ &= L^{(D)}_{real} + L^{(D)}_{fake} \end{aligned} \tag{19}$$

Herein, we take advantage of spectral normalization in combination with the hinge loss and adopt the supervised regression loss to accommodate for the continuous label (Γ). The total discriminator loss is thus:

$$\begin{aligned} L^{(D)} &= L^{(D)}_{regression\_error} + L^{(D)}_{hinge} \\ &= \mathbb{E}_{y\sim p_{data}}[(y - \hat{y})^2] - \{\mathbb{E}_{x\sim p_{data}}[min(0, -1 + D(x))] \\ &\quad + \mathbb{E}_{z\sim p_z, y\sim p_{data}}[min(0, -1 - D(G(z,y)))]\} \end{aligned} \tag{20}$$

5.2.2 Generator Loss

The major goal of the generator is to create realistic Ising cubes given a target transverse field strength Γ. Two implications arise: i. the generator should confuse the discriminator by generating data close to the real distribution. ii. the generator should produce fake data that are realistic with respective to a given conditional label, i.e., the target Γ.

Heuristic Game Loss To confuse the discriminator, the generator tries to "weaken" the discriminator's classification performance. However, the original formulation of the generator loss (Goodfellow et al., 2014a), i.e., $\min_G\ 1 - D(G(z))$, does not perform especially well. This is because the generator's gradient vanishes when the discriminator has high confidence of distinguishing generated samples from the real samples, i.e., when $D(G(z)) \to 0$. Instead, with the heuristically motivated game concept (Goodfellow, 2016), the generator instead minimizes $-D(G(z))$. The heuristic-game generator loss with conditional input y is thus

$$L^{(G)}_{heuristic} = -\mathbb{E}_{z\sim p_z, y\sim p_{data}}[D(G(z,y))], \tag{21}$$

which, combined with the discriminator hinge loss, Eq.19, follows the same hinge loss structure in (ref limYe, miyato, multi-hinge Kavalerov) but is adjusted for conditional input y.

Transverse Field Regression Error We assume that the real simulated data are the ground truth configurations adhering to the laws of quantum physics. The discriminator regression, which is trained only on the real data, provides a direct measure of the generator's capability of creating spin configurations true to the target field strength Γ. This is measured via:

$$L^{(G)}_{RE} = ||\Gamma - \hat{\Gamma}||_2, \tag{22}$$

where Γ is a vector of all target field strength values supplied to the generator (as conditional label y) during training, and $\hat{\Gamma}$ is a vector of corresponding regression outputs predicted by the discriminator.

Feature Matching Feature matching is a technique that prevents over-training the generator and increases the stability of the GAN (Salimans et al., 2016). It requires the generator to produce samples which result in similar features on an intermediate layer of the discriminator network as do the real samples. Therefore, the generator loss considering feature matching is formulated as:

$$L^{(G)}_{feature\_matching} = \left\| \mathbb{E}_{x\sim p_{data}(x)} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z)) \right\|_2^2 \tag{23}$$

where $f(x)$ is the activations of an intermediate layer of the discriminator for a given sample x. In this study, $f(x)$ is defined by the ReLU (Nair and Hinton, 2010) activation on the flattened output of the last convolutional (Conv) layer of the discriminator network.

Average Magnetization Matching Our Ising cube data are not naturally occurring, day-to-day images, but the average magnetization of an Ising cube summarizes the average number of spins that are up/down (as marked by black/white squares), which is a human observable feature. Liu and Rodrigues proposed using the magnetization per spin as an auxiliary state for the generator to match the real data at a given Γ, which was shown to be effective for learning Ising spin configurations:

$$L^{(G)}_{avg\_mag} = \mathbb{E}_{(x,y)\sim p_{data}, z\sim p_z}[(M(x) - M(G(z,y)))^2], \tag{24}$$

and we calculate the average magnetization across all Trotter slices:

$$M(\mathbf{s}) = \frac{1}{b\tau}\sum_{i=1}^{b}\sum_{l=1}^{\tau} s_{il}. \tag{25}$$

Finally, combining all parts together, the total generator loss is:

$$L^{(G)} = L^{(G)}_{heuristic} + L^{(G)}_{RE} + L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag} \tag{26}$$

5.2.3 SQA-GAN Algorithm

By introducing a conditional label input to the generator, adding a regression output to the discriminator, and incorporating spectral normalization in the discriminator layers, our SQA-GAN design takes the structure in Fig **??**.

For each training batch of m real data-label pairs, i.e., $\{(x_i, y_i), i \in 1, ..., m\}$, the detailed training procedure of the SQA-GAN is as follows:

Step 0: Initialize the discriminator D and the generator G with θ_D and θ_G, respectively.

Step 1: Feed the real batch of samples, $\{x_i, i \in 1, ..., m\}$, to the discriminator D, which outputs a vector of predicative probabilities that input sample is real, $[D(x_1), ..., D(x_m)]^T$, and a vector of predicted continuous labels $[\hat{y}_1^{real}, ..., \hat{y}_m^{real}]^T$

Step 2: Random noise vectors, $\mathbf{z} = \{z_1, \ldots, z_m\}$, are sampled from the noise prior $p_g(z)$. Each z_i is paired with the real continuous label y_i and $\{(z_1, y_1), ..., (z_m, y_m)\}$ is fed to G to conditionally generate m synthetic samples, $\{G(z_1, y_1), \ldots, G(z_m, y_m)\}$.

Step 3: Feed $\{G(z_1, y_1), \ldots, G(z_m, y_m)\}$ to D. For each $G(z_i, y_m), i \in \{1, \ldots, m\}$, D outputs a predictive probability for the data being real, $D(G(z_i, y_m))$, and a predicted continuous label $\hat{y}_i^{fake}$

Step 4: Compute the discriminator loss, $L^{(D)}$:

$$L^{(D)} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i^{real})^2 - \frac{1}{m}\sum_{i=1}^{m} min(0, -1 + D(x_i)) - \frac{1}{m}\sum_{i=1}^{m} min(0, -1 - D(G(z_i, y_i))) \tag{27}$$

Step 5: Compute the generator loss, $L^{(G)}$:

$$L^{(G)} = -\frac{1}{m}\sum_{i=1}^{m} D(G(z_i, y_i)) + \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i^{fake})^2 + ||\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(G(z_i, y_i))||_2^2 + \frac{1}{m}\sum_{i=1}^{m}(M(x_i) - M(G(z_i, y_i)))^2 \quad (28)$$

Step 6: Optimize and update the network parameters θ_D and θ_G, where η is the learning rate.

$$\theta_D \leftarrow \theta_D - \eta \cdot \nabla_{\theta_D} L^{(D)} \quad (29)$$

$$\theta_G \leftarrow \theta_G - \eta \cdot \nabla_{\theta_G} L^{(G)} \quad (30)$$

Repeat steps (1) to (8) until convergence is achieved or the designated number of iterations is reached.

6 Experiments

6.1 Experimental objective

We try to create a quantum annealing simulator using GAN-based methods. Specifically, we want to implicitly learn the Ising spin configurations output by the SQA algorithm with our proposed SQA-GAN and compare its data generation performance with other GAN methods.

6.2 Dataset

We first build a quantum annealing simulator with the path-integral SQA algorithm (ref). We then simulate a 32×32 spin-glass system and generate the training data using a linear annealing schedule from $\Gamma = 3.00$ to $\Gamma = 0.01$. We simulate one sweep for each 0.001 increment of Γ and repeat the process 50 times. There are $32 \times 32 = 1024$ spins on each Trotter slice. With 20 Trotter slices, each Ising spin configuration (x_i) has shape $32 * 32 * 20$.

6.3 Evaluation metrics

There are three aspects to measuring each model's generative capability: i. regressive evaluation by the discriminator. ii. average magnetization distribution. iii. quality of visual features.

Regressive Error The generator should not only produce realistic samples, but also generate data that realistically match the target transverse field strength Γ. Because the discriminator is only trained on real samples for regression, its predicative error on fake samples' target labels indicates the degree of dissimilarity between the real and generated data distributions. Thus, the $L_{RE}^{(G)}$ loss in Equation.22 is a direct measurement of such dissimilarity between the two distributions.

Average Magnetization Distribution The average magnetization distribution is an effective overview of an Ising simulator's statistical behavior. Besides comparing the average magnetization loss $L_{avg_{mag}}^{(G)}$, we also plot the average magnetization distribution histograms for more detailed comparison.

Visual Quality Visual inspection is another important step for checking mode collapse and the diversity of the generated samples. In particular, we check whether the generated samples have repetitive patterns at different field strengths and whether the generated samples are just direct copies of the real data.

6.4 Network configurations

Discriminator Architecture The discriminator input takes shape $32 * 32 * 20$, we use different convolution filter sizes and batch normalization layers after each (ReLU activated) dense layer. However, no batch normalization is used for the first dense layer as suggested by (ref Radford Unsupervised Representation). Additionally, the discriminator weight matrices are spectral-normalized to impose the Lipschitz-1 condition for more stable training. The feature map output by the last convolution layer is flattened and fed into a fully connected dense layer with 2 neurons: one for the regression output and one for the real/fake classification output. The Discriminator architecture is illustrated in Table.1

Generator Architecture The noise vector **z** and the conditional label y are first processed by two separate densely-connected layers and then reshaped and concatenated for subsequent deconvolution operations. No batch normalization is used for the last deconvolution layer per (ref Radford Unsupervised Representation). The Generator architecture is illustrated in Table.2.

7 Experimental Results & Analysis

8 Conclusions & Extensions

Acknowledgements

References

Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

Azimi, M., Eslamlou, A. D., and Pekcan, G. (2020). Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review. *Sensors*, 20(10):2778.

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw,

Table 1: Configurations of the discriminator of the BSS-GAN

Layer	**Filter size (#)**	**Activation**	**Shape**	**Notes (α: $-$ive slope coef. in Leaky ReLU)**
Input	-	-	(N, 32, 32, 20)	Input RGB images of size 128×128
Conv	3×3 (32)	Leaky ReLU	(N, 16, 16, 32)	Stride = 2, $\alpha = 0.2$
Dropout	-	-	(N, 16, 16, 32)	Dropout rate = 0.25
Conv	3×3 (64)	Leaky ReLU	(N, 8, 8, 64)	Stride = 2, $\alpha = 0.2$
BatchNorm	-	-	(N, 8, 8, 64)	Momentum = 0.8
Dropout	-	-	(N, 8, 8, 64)	Dropout rate = 0.25
Conv	3×3 (64)	Leaky ReLU	(N, 8, 8, 64)	Stride = 1, $\alpha = 0.2$
Flatten	-	-	(N, 4096)	4096 = 8×8×64
Fc-layer	-	Softmax	(N, 2)	Regression and classification outputs

Table 2: Configuration of the generator of the BSS-GAN

Layer	Activation	Shape	Note
Input	-	(N, 100)	Noise vector
FC-layer	ReLU	(N, 8192)	8192=8*8*128
Reshape	-	(N, 8, 8, 128)	

Layer	Activation	Shape	Note
Input	-	(N, 1)	Input Label
FC-layer	ReLU	(N, 1024)	1024=32*32*1
Reshape	-	(N, 8, 8, 1)	

Layer	**Filter size (#)**	**Activation**	**Shape**	**Notes**
Concat	-	-	(N, 8, 8, 129)	Concatenate two feature maps above
Deconv	3×3 (128)	ReLU	(N, 16, 16, 64)	Stride = 2
BatchNorm	-	-	(N, 16, 16, 64)	Momentum = 0.8
Deconv	3×3 (64)	ReLU	(N, 32, 32, 32)	Stride = 2
BatchNorm	-	-	(N, 32, 32, 32)	Momentum = 0.8
Deconv	3×3 (3)	ReLU	(N, 32, 32, 20)	Stride = 1

J., and Rueckert, D. (2018). Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*.

Cha, Y.-J., Choi, W., and Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 32(5):361–378.

Cha, Y.-J., Choi, W., Suh, G., Mahmoudkhani, S., and Büyüköztürk, O. (2018). Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE.

Deng, J., Lu, Y., and Lee, V. C.-S. (2020). Concrete crack detection with handwriting script interferences using faster region-based convolutional neural network. *Computer-Aided Civil and Infrastructure Engineering*, 35(4):373–388.

Deng, J., Wei, D., Richard, S., Li-Jia, L., Kai, L., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255.

Dorafshan, S., Thomas, R. J., and Maguire, M. (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials*, 186:1031–1045.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *Biomedical Imaging (ISBI 2018)*, pages 289–293. IEEE.

Gao, Y., Kong, B., and Mosalam, K. M. (2019). Deep leaf-bootstrapping generative adversarial network for structural image data augmentation. *Computer-Aided Civil and Infrastructure Engineering*, 34(9):755–773.

Gao, Y., Li, K., Mosalam, K., and Günay, S. (2018). Deep residual network with transfer learning for image-based

structural damage recognition. In *Eleventh US National Conference on Earthquake Engineering, Integrating Science, Engineering & Policy*.

Gao, Y. and Mosalam, K. M. (2018). Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768.

Gao, Y. and Mosalam, K. M. (2020). Peer hub imagenet: A large-scale multiattribute benchmark data set of structural images. *Journal of Structural Engineering*, 146(10):04020198.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jain, N., Manikonda, L., Hernandez, A. O., Sengupta, S., and Kambhampati, S. (2018). Imagining an engineer: On gan-based data augmentation perpetuating biases. *arXiv preprint arXiv:1811.03751*.

Jiang, S. and Zhang, J. (2019). Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. *Computer-Aided Civil and Infrastructure Engineering*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Technical Report TR-2009*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liang, X. (2019). Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with bayesian optimization. *Computer-Aided Civil and Infrastructure Engineering*, 34(5):415–430.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Madani, A., Moradi, M., Karargyris, A., and Syeda-Mahmood, T. (2018a). Chest x-ray generation and data augmentation for cardiovascular abnormality classification. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105741M. International Society for Optics and Photonics.

Madani, A., Moradi, M., Karargyris, A., and Syeda-Mahmood, T. (2018b). Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1038–1042. IEEE.

Maeda, H., Kashiyama, T., Sekimoto, Y., Seto, T., and Omata, H. (2020). Generative adversarial network for road damage detection. *Computer-Aided Civil and Infrastructure Engineering*.

Maeda, H., Sekimoto, Y., and Seto, T. (2016). Lightweight road manager: smartphone-based automatic determination of road damage status by deep neural network. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 37–45.

Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., and Malossi, C. (2018). Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press, Cambridge.

Mosalam, K., Muin, S., and Gao, Y. (2019). New directions in structural health monitoring. *NED University Journal of Research*, 2:77–112.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *In Proceedings of the 27th international conference on machine learning*, pages 807–814. ICML.

Oh, B. K., Kim, K. J., Kim, Y., Park, H. S., and Adeli, H. (2017). Evolutionary learning based sustainable strain sensing model for structural health monitoring of high-rise buildings. *Applied Soft Computing*, 58:576–585.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rafiei, M. H. and Adeli, H. (2017). A novel machine learning-based algorithm to detect damage in high-rise building structures. *The Structural Design of Tall and Special Buildings*, 26(18):e1400.

Rafiei, M. H., Khushefati, W. H., Demirboga, R., and Adeli, H. (2017). Supervised deep restricted boltzmann machine for estimation of concrete. *ACI Materials Journal*, 114(2).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Villa, T. F., Gonzalez, F., Miljievic, B., Ristovski, Z. D., and Morawska, L. (2016). An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives. *Sensors*, 16(7):1072.

Xue, Y. and Li, Y. (2018). A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Computer-Aided Civil and Infrastructure Engineering*, 33(8):638–654.

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., and Yang, X. (2018). Automatic pixel-level crack detection and measurement using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1090–1109.

Yeum, C. M., Dyke, S. J., and Ramirez, J. (2018). Visual data classification in post-event building reconnaissance. *Engineering Structures*, 155:16–24.

Yi, X., Walia, E., and Babyn, P. (2019). Generative adversarial network in medical imaging: A review. *Medical image analysis*, page 101552.

Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., and Chen, C. (2017). Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering*, 32(10):805–819.

Zhang, C., Chang, C.-c., and Jamshidi, M. (2020). Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering*, 35(4):389–409.

Zhang, X., Wang, Z., Liu, D., and Ling, Q. (2019). Dada: Deep adversarial data augmentation for extremely low data regime classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2807–2811. IEEE.