Information Maximization and Contrastive Learning Generative Adversarial Network for Simulated Quantum Annealing

Pengyuan Zhai

Department of Industrial Engineering and Operations Research, Department of Civil and Environmental Engineering, University of California, Berkeley, CA, United States

Abstract: *This study aims to build a deep generative model for simulated quantum annealing. We train a Generative Adversarial Network (GAN) variant with mutual information maximization objectives to capture the distribution of spin configurations of a (2+1)-dimensional transverse field quantum Ising model. The training samples are obtained from quantum (path-integral) Monte Carlo simulations at continuous transverse field values. To stabilize training and prevent generator mode collapse, we use the 1-Lipschitz constrained discriminator (critic) to approximate the Kantorovich-Rubenstein dual form Wasserstein metric in order to minimize the optimal transport distance between the real and generated spin distributions during training. Additionally, we maximize the lower bound on mutual information (MI) between the input conditional field value and the generated spin states to guide the generator to output realistic spin configurations conditioned on the transverse field. We implement and compare three objective functions for maximizing the lower bound of MI, based on: i. variational lower bound through Difference of Entropies (DoE); ii. Noise-contrastive Estimation (NCE); iii. Wasserstein dependency measure (WDM) and Wasserstein Predictive Coding (WPC).*

To the best of our knowledge, this work is the first to investigate GAN's learning capabilities on a (2+1)-dimensional transverse field quantum Ising model, in an effort to achieve faster and more memory efficient simulated quantum annealing. We tested and compared SQA-GAN's generative capabilities based on a 32×32 *quantum Ising system and conducted experiments to compare the performance of different combinations of GAN objectives (Wasserstein vs non-Wasserstein) and MI lower bound maximization formulations. This study not only proves the capability of information-theoretic and game-theoretic deep learning methods in modeling complex quantum physical systems, but also showcases the promising potential of GAN's data generation capabilities in simulating stochastic system.*

1 Introduction

Replicating complex system behaviour has been an increasing challenge in the data explosion era, in which Deep Learning (DL) has shown great potential in learning and describing complicated physical systems in quantum physics and astronomical sciences (LeCun et al. (2015),Baldi et al. (2015),Chen et al. (2014)). Statistical physics combines powerful statistics methods to describe and simulate many-body physical systems such as in field theory or spin models. Given a Hamiltonian or energy configuration, we want to describe the system behavior by tracking the distribution of elementary degrees of freedom, which are variables (such as spins in a spin model) that interact with each other according to the Boltzmann distribution.

The Ising model is an example of statistical physics model whose behavior is governed by the Hamiltonian. The classical Ising model, when combined with annealing algorithms, has application value as it is a powerful optimization tool that can solve hard combinatorial problems (such as traveling sales problem, integer factoring, portfolio optimization, and protein modeling, etc), while preventing being trapped in local minima by probabilistic thermal fluctuations to jump over energy barriers.

Quantum annealing is the quantum analogue of classical annealing. Quantum annealing replaces thermodynamics in classical annealing by quantum dynamics through introducing additional degrees of freedom of quantum nature ("transverse field"). Instead of "thermal jumps" to escape local minima, quantum annealing is able to find lower energy configurations through quantum tunneling (Farhi et al. (2000), Farhi et al. (2001), Farhi et al. (2002)), reaching lower energy states directly through energy barriers.

Both classical annealing (CA) and quantum annealing (QA) can be simulated by Markov chain Monte Carlo (MCMC) through the Metropolis-Hastings algorithm, which guarantees asymptotic convergence to the true distribution.

This conveniently bridges CA or QA with stochastic systems. As the "temperature" or "transverse field" lowers, MCMC generates Ising spin configurations closer to the ground state, which represents the global minimum of the optimization problem.

As the Ising model size increases, MCMC simulations quickly become computationally heavy. We thus wish to take advantage of the great potential of deep learning to capture and describe the quantum annealing process. Specifically, we are interested in training a model that outputs Ising states congruent to the real distribution conditioned on the "temperature" or "transverse field".

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 predicting 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.

In this work, we build a GAN framework with mutual information (MI) lower bound maximization objectives to efficiently train the model to output realistic Ising spins congruent to the target label. We name our framework SQA-GAN and the main contributions are:

i. To the best of our knowledge, this work is the first to investigate GAN's learning capabilities on a (2+1)-dimensional transverse field quantum Ising model, in an effort to achieve faster and more memory efficient simulated quantum annealing (SQA).

ii. We incorporated three families of Mutual Information (MI) lower bound maximization methods: 1. variational lower bound through Difference of Entropies (DoE); 2. Noise-contrastive Estimation (NCE) based methods including NCE and infoNCE; 3. Wasserstein dependency measure (WDM) and Wasserstein Predictive Coding (WPC). We conducted comparison experiments to study the effect of each MI lower bound maximization formulation in combination with Wasserstein or non-Wasserstein GAN objectives.

iii. We give rigorous derivations to provide a unifying view on various contrastive learning objectives for mutual information (lower bound) maximization, by formulating them as a cross entropy maximization problem between contrastive sample distribution (from joint distribution $p(x,y)$ and the product of marginals $p(x)p(y)$) and a likelihood estimator function $f(x,y)$. We discuss further research directions involving distribution divergence measures and how to further unify them through the information-theoretic scope.

2 Related Work

Simulated Quantum Annealing 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). (add MCMC simulation here)

Generative Adversarial Network The concept of Generative Adversarial Network (GAN) was first introduced by Goodfellow et al. (2014), which is a generative, game-theoretic model that synthesizes unseen data from the learned distribution. 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).

GAN in computational physics GAN's data generation capabilities have been explored in computational and statistical physics. Liu et al. (2017) used the conditional GAN (CGAN) to simulate the classical (thermal) 2D Ising model near critical temperature and verified its effectiveness in generating realistic Ising states when trained on data from conventional Monte Carlo (MC) simulations. Additionally, Singh et al. (2020) used GAN and variational MI lower bound method for sampling and detecting phase transition of the classic XY model, and showed the model can be used as an unsupervised indicator of transitions, by constructing measures of the model's susceptibility to changes in tuning parameters.

Mutual Information Maximization Mutual Information (MI) measures the amount of information obtained about $X \sim p(x)$ by observing $Y \sim p(y)$. It is formally defined as the

Kullback–Leibler (KL) divergence between the joint and the product of the marginal distributions:

$$I(X,Y) = D_{KL}(p(x,y)\|p(x)p(y)) = \mathbb{E}_{p(x,y)}\left[log\frac{p(x,y)}{p(x)p(y)}\right]. \quad (1)$$

However, in high-dimensional data, the exact value of $I(X,Y)$ is often intractable. Nguyen et al. (2010), Belghazi et al. (2018), van den Oord et al. (2019), Poole et al. (2019), and Gutmann and Hyvärinen (2010) have proposed maximizing tractable lower bounds to the MI which are applicable in gradient-based deep learning settings. In this study, we maximize the lower bound to the MI between the (real or fake) spin configurations and corresponding input labels (transverse field, Γ) in order to help the model capture spin distributions conditioned on transverse fields. Specifically, we implement and compare three families of MI maximization formulations: i. variational lower bound through Difference of Entropies (DoE); ii. Noise-contrastive Estimation (NCE); iii. Wasserstein dependency measure (WDM).

3 Background

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

$$H_I^c(s) = -\sum_{<i,j>} J_{ij}s_is_j - \sum_j h_js_j \quad (2)$$

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 $\mathbf{H_I^c(s)}$.

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

$$P_\beta(\mathbf{s}|\mathbf{T}) = \frac{e^{-\beta\mathbf{H_I^c(s)}}}{Z_\beta}, Z_\beta = \sum_{\mathbf{s}} e^{-\beta H_I^c(s)}, \beta = (k_B T)^{-1}, \quad (3)$$

where $\beta = (k_BT)^{-1}$ 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_BT 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 $|x_1x_2...x_b\rangle, x_j \in \{+1,-1\}, \forall j \in \{1,...,b\}$, 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:

$$|\psi(t)\rangle = e^{-\sqrt{-1}\mathbf{H_t}}|\psi(0)\rangle, \quad (4)$$

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

$$\sigma_j^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{5}$$

The quantum Hamiltonian of the quantum Ising model thus becomes:

$$\mathbf{H}_I^q = -\sum_{<i,j>} J_{ij}\sigma_i^z\sigma_j^z - \sum_j h_j\sigma_j^z, \tag{6}$$

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 (6) a diagonal matrix. $\mathbf{H}_I^q$ is thus also a diagonal matrix (the second term of (6) 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

$$\mathbf{H}_x = -\sum_j \sigma_j^x, \tag{7}$$

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

$$\sigma_j^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{8}$$

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)

$$\mathbf{H}_D(t) = A(t)\mathbf{H}_X + B(t)\mathbf{H}_I^q. \tag{9}$$

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

$$\mathbf{H}_{aI}^c(s) = -\sum_{l=1}^{\tau}[B(t)\sum_{<i,j>} J_{ij}s_{il}s_{jl} + J(t)\sum_j s_{jl}s_{j,l+1}], \tag{10}$$

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:

$$J(t) = -\frac{\tau T}{2}\ln[\tanh(\frac{A(t)}{\tau T})], \tag{11}$$

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)

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

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)

$$\begin{aligned}\triangle E_{2i}^{(k)} = -\sum_{l=1}^{\tau} B(t_k)[\sum_{j=1}^{i-1} J_{ij} s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)}) \\ + \sum_{j=i+1}^{b} J_{ij} s_{jl}^{(k-1)}(s_{il}^{(k)} - s_{il}^{(k-1)})].\end{aligned} \tag{13}$$

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 $\mathbf{H}_I^c(\mathbf{s}^{(k)})$.

3.4 Generative Adversarial Network

GAN The original 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 (the non-saturating GAN objective):

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

In Equation (14), 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 (14) until reaching the optima or designated number of iterations.

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 **??**). 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:

$$\min_G \max_D E_{x\sim p_r(x)}[\log(D(x,y))] + E_{z\sim p_z}[\log(1 - D(G(z,y),y))] \tag{15}$$

Wasserstein distance Unlike KL divergence, which is sensitive to small differences in data (ref Oord MI) **? ?**, Wasserstein distance is a metric-aware divergence. The Wasserstein distance measures the optimal transport between two distributions. If denoting the real and generated data distributions (densities) as p_r and p_g, their optimal transport distance (1-Wasserstein) is:

$$W(p_r, p_g) = \inf_{\gamma \in \prod(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||], \tag{16}$$

where $\prod(p_r, p_g)$ is the set of all joint distributions $\gamma(x,y)$ whose marginals evaluate to p_r and p_g respectively, i.e., $p_r(x) = \int_Y \gamma(x,y)dy$ and $p_g(y) = \int_X \gamma(x,y)dx$. $||x - y||$ denotes the distance (or cost) of transferring an infinitesimal amount of density value from $x \sim p_r$ to $y \sim p_g$.

The primal Wasserstein distance formulation in Eq.16 is intractable in a gradient-based deep learning setting. The Kantorovich-Rubinstein (Villani, 2008) dual form of the 1-Wasserstein distance is:

$$W(p_r, p_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)], \tag{17}$$

where we take the supremum over all 1-Lipschitz functions $f : X \to \mathbb{R}$. $f(\cdot)$ can be represented by a neural network with parameter θ such that $f_\theta(\cdot)$ is 1-Lipschitz, which can be updated through gradient-based optimization.

The Wasserstein GAN (WGAN) Arjovsky et al. (2017)) uses Eq.17 as the minimax objective with the advantage of preventing mode collapse and stabilizing training:

$$\min_G \max_{D:\|D\|\leq 1} \mathbb{E}_{x\sim p_r}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))], \tag{18}$$

where the 1-Lipschitz constraint of the discriminator (or the critic), $D(\cdot)$, is enforced by weight clipping (Arjovsky et al. (2017)), gradient penalty (Gulrajani et al. (2017)), or spectral normalization (Miyato et al. (2018)). The conditional WGAN objective is thus:

$$\min_G \max_{D:\|D\|\leq 1} \mathbb{E}_{x\sim p_r}[D(x,y)] - \mathbb{E}_{z\sim p_z}[D(G(z,y),y)]. \tag{19}$$

3.5 Mutual Information maximization

Intuition Besides creating "fake" Ising configurations consistent with the real Boltzmann distribution, the generator is also required to output Ising states that are congruent with the target conditional label in order to become a meaningful Ising simulator. We achieve this through Mutual Information (MI) maximization, which measures the amount of information obtained about $X \sim p(x)$ by observing $Y \sim p(y)$. Given two different target labels (transverse fields), y_{c1} and y_{c2}, we want the generator to create Ising samples $G(z_1, y_{c1})$ and $G(z_2, y_{c2})$ such that they each contain maximal information about the corresponding y_{ci}. Besides increasing the entropy between $G(z_1, y_{c1})$ and $G(z_2, y_{c2})$ (making them "different" from each other), we need to establish a correspondence between Ising samples and the correct continuous labels. We train the discriminator (encoder) on real samples so that it correctly identifies congruent (matching) data-label pairs from incongruent (unmatching) pairs. An optimally trained discriminator extracts high-level representations from the real Ising data x that best reflect information about the correct transverse field y, which is equivalent to optimizing the discriminator's feature extraction mechanism to maximize the estimated (lower bound) mutual information between congruent data and labels (and, if applicable by formulation, decrease the MI between incongruent data and labels).

Such information learnt by the discriminator is passed to the generator through gradient backpropagation, when we also train the generator to output any fake sample $G(z, y_c)$ that has the maximal estimated mutual information about y_c evaluated by the discriminator.

MI definition Mutual Information (MI) is formally defined as the Kullback–Leibler (KL) divergence between the joint and the product of the marginal distributions:

$$I(X,Y) = D_{KL}(p(x,y) \| p(x)p(y)) = \mathbb{E}_{p(x,y)}\left[log \frac{p(x,y)}{p(x)p(y)}\right]. \tag{20}$$

MI is difficult to estimate in high dimensional spaces. Nguyen et al. (2010), Belghazi et al. (2018), van den Oord et al. (2019), Poole et al. (2019), and Gutmann and Hyvärinen (2010) have proposed maximizing tractable lower bounds to the MI which are applicable in gradient-based deep learning settings. In this study, we maximize the lower bound to the MI between the (real or fake) spin configurations and corresponding input labels (transverse field, Γ) in order to help the model capture spin distributions conditioned on transverse fields. Specifically, we implement and compare three families of MI maximization formulations: i. variational lower bound through Difference of Entropies (DoE); ii. Noise-contrastive Estimation (NCE); iii. Wasserstein dependency measure (WDM).

Variational lower bound with difference of entropies The Variational Lower Bound method (Barber and Agakov (2003)) stems from an equivalent expression or of MI terms of the difference of entropies (DoE):

$$I_{DoE}(X,Y) = H(Y) - H(Y|X) = I_{KL}(X,Y), \tag{21}$$

and by Gibb's inequality,

$$H(Y) = -\sum_{Y} p(y) log\, p(y) \leq -\sum_{Y} p(y) log\, q(y), \tag{22}$$

$$H(Y|X) = -\sum_{Y,X} p(y,x) log\, p(y|x) \leq -\sum_{Y,X} p(y,x) log\, q(y|x), \tag{23}$$

where $q(y)$ and $q(y|x)$ are any valid probability distribution estimators in their respective variable domains ($q(y) \in [0,1], \forall y \in Y$ and $q(y|x) \in [0,1], \forall x \in X, \forall y \in Y$), which are minimized (ideally) to estimate the true prior $p(y)$ and posterior $p(y|x)$:

$$H(Y) = \inf_{q_Y} -\sum_{Y} p(y) log\, q(y) = \inf_{q_Y} H(p_Y, q_Y), \tag{24}$$

$$H(Y|X) = \inf_{q_{Y|X}} -\sum_{Y,X} p(y,x) log\, q(y|x) = \inf_{q_{Y|X}} H(p_{YX}, q_{Y|X}). \tag{25}$$

Although $I_{DoE}(X,Y) = H(Y) - H(Y|X) = \inf_{q_Y} H(p_Y, q_Y) - \inf_{q_{Y|X}} H(p_{YX}, q_{Y|X})$ provides no guarantee of any bound to MI, in application scenarios where $H(Y)$ is typically fixed, we arrive at the following lower bound to MI:

$$I_{DoE}(X,Y) = H(Y) - \inf_{q_{Y|X}} H(p_{YX}, q_{Y|X}) \geq H(Y) + L_{DoE}(X,Y). \tag{26}$$

$$L_{DoE}(X,Y) = -H(p_{YX}, q_{Y|X}) = \sum_{Y,X} p(y,x) log\, q(y|x) \tag{27}$$

This bound (27) is named the variational lower bound on mutual information (ref Barber) and this method is used in deriving the information-theoretic GAN (infoGAN) objective by Chen et al. An auxiliary network is used to model the estimator function $q(y|x)$. We use similar configurations as Chen et al. where the auxiliary network and the discriminator share most convolutional layers and there is one final fully connected layer to output parameters for the conditional distribution $q(y|x)$. In this study we use the factored Gaussian for $q(y|x)$, so Q outputs the corresponding mean and standard deviation vectors.

Noise-contrastive estimation NCE (Gutmann and Hyvärinen (2010)) has inspired a family of MI lower bounding methods (ref infoNCE, Contrastive Distillation). This family of methods independently sample data pairs of two types: congruent (positive) pairs from the joint

distribution ($s^+ = \{(x^+,y^+)_i \sim p(x,y), i = 1,...,M\}$) as well as incongruent (negative) pairs from the product of marginals ($s^- = \{(x^-,y^-)_j \sim p(x)p(y), j = 1,...,N\}$).

The contrastive loss function is the cross entropy between the independently sampled sets' distribution $p(s^+,s^-) = p(s^+)p(s^-)$ and a tractable likelihood estimator function $q(\cdot)$:

$$L_{contrast} = \frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)} log\, q(s^+,s^-). \tag{28}$$

By Gibb's inequality, $q(s^+,s^-)$ is maximized, it is equal to the sample probability, i.e., $q^*(s^+,s^-) = p(s^+)p(s^-)$, and it can be proven that $\mathbb{E}_{p(s^+)p(s^-)} log\, q^*(s^+,s^-)$ is a lower bound to $I(X,Y)$ (See Appendix for proof):

$$\sup_q L_{contrast} \leq I(X,Y) \tag{29}$$

We prove that the following equation (28) is indeed a unifying representation of different NCE-based formulations and that it is a lower bound to the mutual information between X and Y (See Appendix).

In the original NCE formulation, $q(s^+,s^-) = \prod_{i=1}^M f((x^+,y^+)_i)\prod_{j=1}^N (1-f((x^-,y^-)_j))$ where $f(\cdot) \in [0,1]$ is the estimated Bernoulli parameter denoting the probability that a pair $(x,y)_i$ is congruent (and thus $1-f((x,y)_i)$ is the probability that the pair is incongruent):

$$\begin{aligned} L_{NCE} &= \frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)} log(\prod_{i=1}^M f((x^+,y^+)_i)\prod_{j=1}^N (1-f((x^-,y^-)_j))) \\ &= \frac{1}{M}\mathbb{E}_{p(s^+)}\sum_{i=1}^M log(f((x^+,y^+)_i) + \\ &\frac{1}{M}\mathbb{E}_{p(s^-)}\sum_{j=1}^N log(1-f((x^-,y^-)_j))) \\ &= \mathbb{E}_{p(x,y)} log(f((x,y)) + \frac{N}{M}\mathbb{E}_{p(x)p(y)} log(1-f((x,y)))) \end{aligned} \tag{30}$$

InfoNCE (van den Oord et al. (2019)), on the other hand, models $q(\cdot)$ into a softmax expression and sets M=1 (only one positive pair):

$$\begin{aligned} L_{infoNCE} &= \frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)} log\, q(s^+,s^-) \\ &= \frac{1}{M}\mathbb{E}_{p(x^+,y^+)p(s^-)} log \frac{exp(f((x^+,y^+)))}{exp(f((x^+,y^+))) + \sum_{j=1}^N exp(f((x^-,y^-)_j))} \\ &= \frac{1}{M}\mathbb{E}_{p(x^+,y^+)} f((x^+,y^+)) - \\ &\frac{1}{M}\mathbb{E}_{p(x^+,y^+)p(s^-)} log\left[exp(f((x^+,y^+))) + \sum_{j=1}^N exp(f((x^-,y^-)_j))\right]. \end{aligned} \tag{31}$$

Wasserstein dependency measure (WDM) The WDM (Ozair et al. (2019)) is the Wasserstein distance between the joint distribution $p(x,y)$ and the product of marginal distributions $p(x)p(y)$. This is an alternative measure of "mutual information" (loosely defined) between two distributions based on the optimal transport (Villani (2009)) other than the KL divergence:

$$\begin{aligned} \mathscr{I}_{WDM} &= W(p(x,y),p(x)p(y)) = \sup_{\|f\|_L \leq 1} L_{WDM} \\ L_{WDM} &= \mathbb{E}_{p(x,y)}[f(x,y)] - \mathbb{E}_{p(x)p(y)}[f(x,y)], \end{aligned} \tag{32}$$

where (32) is the Kantorovich-Rubinstein dual form as in Eq.(17). Eq.(32) can be seen as a "contrastive" (loosely defined) formulation, as the function $f(x,y)$ aims to maximize the difference of scores evaluated on congruent samples $(x,y) \sim p(x,y)$ and incongruent samples $(x,y) \sim p(x)p(y)$.

Interestingly, Ozair et al introduced the Lipchitz-1 constraint in Eq(32) to Eq(61) in order to force the discriminator to represent more aspects of the data rather than collapsing on a few easily discernible sample differences. Their method is named "Wasserstein Predictive Coding":

$$\begin{aligned} \mathscr{I}_{WPC} &= \sup_{\|f\|_L \leq 1} L_{WPC} \\ L_{WPC} &= \mathbb{E}_{p(x^+,y^+)} f((x^+,y^+)) - \\ &\mathbb{E}_{p(x^+,y^+)p(s^-)} log\left[exp(f((x^+,y^+))) + \sum_{j=1}^N exp(f((x^-,y^-)_j))\right]. \end{aligned} \tag{33}$$

We end this section by noting that the contrastive estimator functions, i.e., $f(x,y)$ in L_{NCE}, $L_{infoNCE}$, L_{WDM} and L_{WPC} are modeled directly by the discriminator with convolution layers, which we will cover in detail in Section 4.

4 Proposed Method

4.1 Ising Model of Interest

This study aims to incorporate mutual information maximization strategies covered in the previous section into the GAN framework, which implicitly learns the distribution of the quantum Ising states output at given transverse fields 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

$$\mathbf{H} = -\sum_{\langle ij \rangle} J_{ij}\sigma_i^z\sigma_j^z - \Gamma\sum_i \sigma_i^x, \tag{34}$$

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

$$\begin{aligned} \mathbf{H}^c_{aI}(s) &= -\sum_{l=1}^{\tau}[\sum_{<i,j>} J_{ij}s_{il}s_{jl} + J(t)\sum_j s_{jl}s_{j,l+1}], \\ J(t) &= -\frac{\tau T}{2}\ln[\tanh(\frac{A(t)}{\tau T})]. \end{aligned} \tag{35}$$

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?).

4.2 Architectures

Conditional generator The GAN generator uses a dense layer and multiple transposed convolution 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 transposed convolution layers transform $fm_1(z)$ into a synthetic sample $G(z)$.

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 $G(z,\Gamma)$. (Figure 1)

Discriminator for contrastive learning NCE, infoNCE, WDM, and WPC formulations share structural similarities (contrastive learning). Besides telling apart real congruent data-label pairs ($x,y \sim p_{real}(x,y)$) from synthetic congruent data-label pairs ($\tilde{x},y_c \sim p(G(z,y_c),y_c)$), the discriminator's output $D(x,y)$ is also useful for contrasting incongruent pairs from congruent pairs within the real or synthetic sample groups. Specifically, we use the discriminator output, $D(x,y)$ to directly model the estimator function $f(x,y)$ in NCE, infoNCE, WDM, and WPC formulations for contrasting congruent samples from the joint distribution ($x,y \sim p_{data}(x,y)$ for real data, and $x,y \sim p(G(z,y_c),y_c)$ for synthetic data) with incongruent samples from the product of marginals ($x,y \sim p_{data}(x)p_{data}(y)$ for real data, and $x,y \sim p(G(z,y_c))\,p(y_c)$ for synthetic data). We show that using $D(x,y)$ for both the conditional GAN and contrastive learning objectives does not compromise the training with regard to either objective.

An given input Ising state (either real $x \sim p(x|y)$, or generated $\tilde{x} = G(z,y_c)$) first goes through a series of convolution layers, and the penultimate convolution layer output is concatenated with the congruent label value y or y_c and is then input to the final convolution layer. The final convolution layer's output is reshaped and fed to a fully connected layer (Table 2) which outputs the final value $D(x,y)$. Please refer to Figure (1) for illustrations of the contrastive GAN architecture.

Discriminator-auxiliary network for DoE On top of the single-output discriminator architecture explained above (only used for conditional GAN objectives in the case of DoE), the DoE formulation (26) requires an auxiliary network that models $q(y|x)$ as in (26), which is replicated below:

$$\begin{aligned} L_{DoE}(X,Y) &= H(Y) - H(p_{YX}, q_{Y|X}) \\ &= H(Y) + \sum_{Y,X} p(y,x)q(y|x), \end{aligned} \tag{36}$$

where $H(Y)$ is treated as a constant. We follow the a similar implementation of infoGAN (Chen et al.) where the auxiliary network (D_q) shares most convolution layers (except the last one) with the discriminator, followed by a separate shallow neural network (fully connected layers) to output the mean and standard deviation vectors for the factored Gaussian distribution approximation for $q(y|x)$. The label information is not disclosed to D_q, so the fully connected layers takes input from the penultimate convolution layer which is not yet concatenated with the label information y. In this work, we treat D_q as a part of the discriminator, and refer to the combined parameters of the discriminator-auxiliary network also as D_{mega}. This is in fact a valid definition of parameter scope, as when D_q is being updated according to L_{DoE} (36), the parameters unique to the discriminator D are fed with zero-valued gradients, hence no updates to D. Please refer to Figure (2) for the discriminator-auxiliary network illustration.

4.3 GAN-MI training objective

We optimize the generator and discriminator to maximize the approximated MI lower bound between: i) the real Ising states $x \sim p_{data}(x|y)$ and the corresponding real label y, i.e., maximize $L(X,Y)$; ii) the conditional input label $y_c \sim p_{data}(y)$ and generated Ising states conditioned on y_c, i.e., maximize $L(G(Z,Y_c),Y_c)$, where $Z \sim p(z)$ is the latent noise vector with parametrized distribution such as the factored Gaussian.

For contrastive learning formulations (NCE, infoNCE, WDM, and WPC), $D(x,y)$ is a continuous score that is used in both the conditional GAN objective (15 or 19) and as the estimator function $f(x,y)$. For the DoE formulation, $D(x,y)$

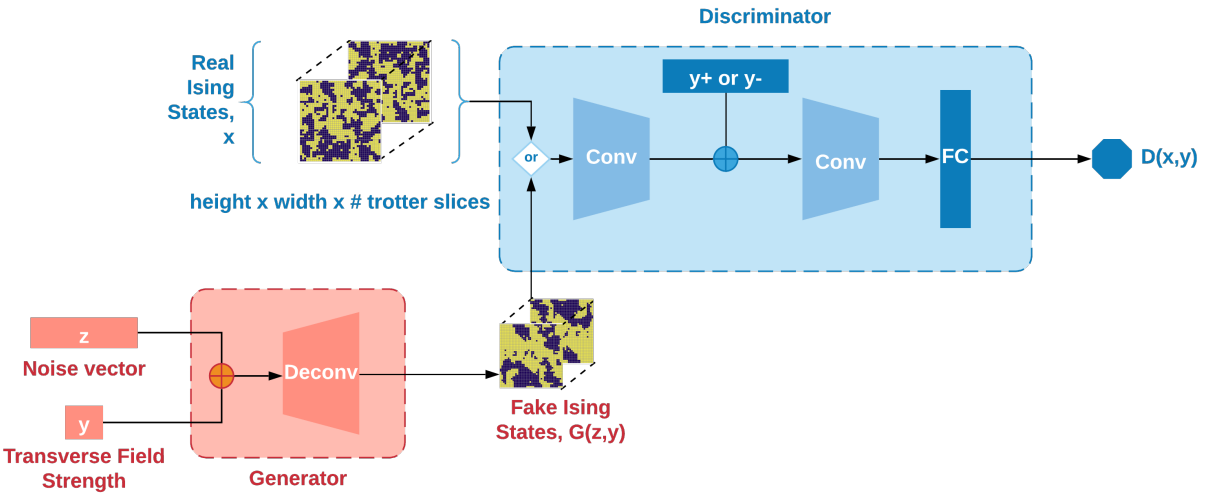

Figure 1: SQA-GAN-Contrastive with noise-contrastive estimation. Note that during training, both congruent labels y^+ and incongruent labels y^- are provided, and the discriminator is trained to correctly identify congruent data $x, y \sim p(x,y)$ from incongruent data $x, y \sim p(x)p(y)$ using NCE-based mutual information lower bound maximization methods.

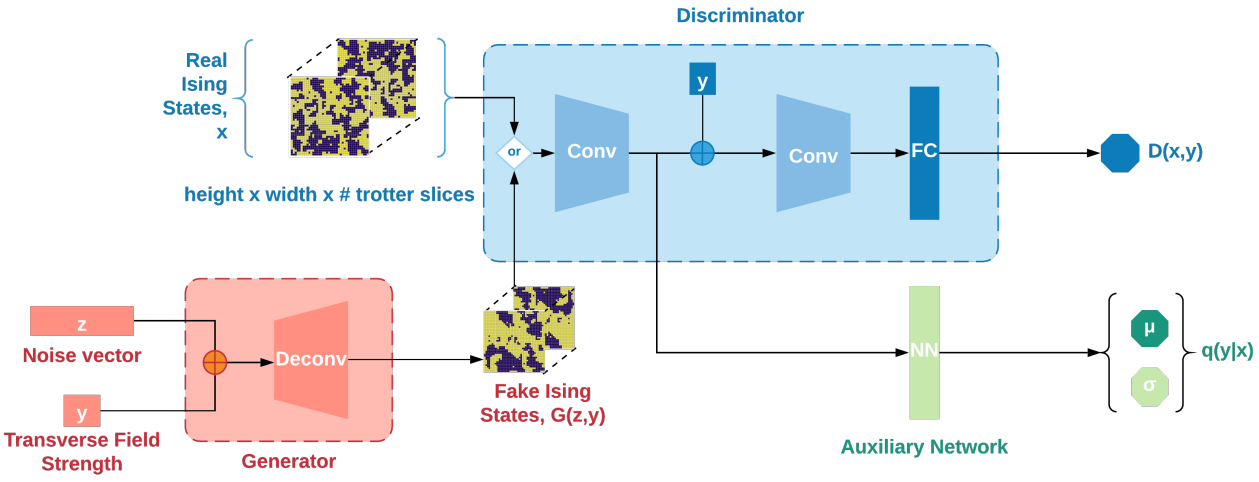

Figure 2: SQA-GAN-Auxiliary with auxiliary network for DoE (variational lower bound) mutual information maximization. The auxiliary network shares most convolution layers with the discriminator, and it outputs the mean (μ) and standard deviation (σ) for the approximated posterior distribution $q(y|x)$. The label information is not disclosed to the auxiliary network.

is only used for the GAN objective, and $D_q(y|x)$ is used as the posterior estimator function $q(y|x)$.

Training the discriminator (D) or the auxiliary network (D_q) with objective i enables D/D_q to encode the Ising samples into representations (high-level feature vectors) that contain maximal information about y and training the generator (G) with objective ii (keeping D fixed) allows such information learnt by D to flow to G through back propagation and increase the estimated MI lower bound between $G(z,y_c)$ and $y_c \sim p_{data}(y)$, guiding the generator to output samples "congruent" to the conditional label (as learnt by the discriminator). Both D and G are updated iteratively according to the GAN-MI training objective:

$$\min_G \max_D V(G,D) - L_{\hat{D}}(G(Z,Y_c),Y_c) + L_D(X,Y), \quad (37)$$

where $L_D(\cdot)$ and $L_{\hat{D}(\cdot)}$ are one of the MI lower bound formulations: $L_{DoE}, L_{NCE}, L_{infoNCE}, L_{WDM}$, or L_{WPC}, modeled by the discriminator (D) or auxiliary network (D_q). The middle term, $L_{\hat{D}}(G(z),Y)$, signifies that we keep D/D_q fixed while updating G with generated samples, and $V(G,D)$ is the conditional GAN (15) or conditional WGAN (19) objective.

Wasserstein or non-Wasserstein GAN Objective We experiment with different combinations of conditional GAN objectives (Wasserstein or non-Wasserstein) coupled with MI lower bound maximization formulations. In the case of Wasserstein conditional GAN objective, we enforce the 1-Lipschitz condition on the discriminator through spectral normalization (SN) (ref Miyota). The following setups are studied:

i) L_{DoE} with Wasserstein conditional GAN (CGAN) objective.

ii) L_{DoE} with non-Wasserstein CGAN objective.

iii) L_{NCE} with non-Wasserstein CGAN objective (because the Bernoulli estimator function requires the probabilistic output to be between 0 and 1);

iv) $L_{infoNCE}$ with non-Wasserstein CGAN objective;

iv) L_{WDM} with Wasserstein CGAN objective due to the Kantorovich-Rubinstein dual form. 1-Lipschitz enforced on the discriminator by spectral normalization (SN).

v) L_{WPC} with Wasserstein CGAN objective with 1-Lipschitz enforce on the discriminator by SN. This is equivalent to an $L_{infoNCE}$ formulation with 1-Lipschitz discriminator.

Discriminator loss function The discriminator loss function is taken directly from the GAN-MI objective (37), where we wish to minimize:

$$\begin{aligned} J^{(D)}_{nonWasserstein} = &-[V_{nW}(G,D) + L_D(X,Y)] = \\ &-\{E_{x,y\sim p_{real}(x,y)}[\log D(x,y)] + \\ &E_{z\sim p_z,y\sim p_{real}(y)}[\log(1-D(G(z,y),y))]\} - L_D(X,Y) \end{aligned} \quad (38)$$

or, in the case of Wasserstein GAN objective:

$$\begin{aligned} J^{(D)}_{Wasserstein} = &-[V_W(G,D) + L_D(X,Y)] = \\ &-\{E_{x,y\sim p_{real}(x,y)}[D(x,y)] - \\ &E_{z\sim p_z,y\sim p_{real}(y)}[D(G(z,y),y)]\} - L_D(X,Y), \end{aligned} \quad (39)$$

where D is 1-Lipschiz.

The generator GAN loss needs special adjustment to prevent diminishing gradients (ref GoodFellow). We go over the non-saturating generator GAN loss and some additional loss terms for the generator in the following section.

4.4 Non-saturating and additional generator loss functions

We make minor adjustments to the training objective in (37) to derive the exact loss function for the generator. Additionally, we study the effects of additional loss function terms across contemporary GAN literature in hopes of increasing training efficiency and stability.

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 realistic data close to the real distribution (GAN objective). ii. the generator should produce fake data that are congruent with the given conditional label, i.e., the target Γ (Mutual Information objective).

Heuristic non-Wasserstein GAN 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., 2014), 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

$$L^{(G)}_{GAN\_heuristic} = -\mathbb{E}_{z\sim p_z,y\sim p_{data}}[log\, D(G(z,y))], \quad (40)$$

which, combined with the discriminator hinge loss, Eq.**??**, follows the same hinge loss structure in (ref limYe, miyato, multi-hinge Kavalerov) but is adjusted for conditional input y. In the case of Wasserstein GAN, the generator does not suffer from the diminishing gradient issue:

$$L^{(G)}_{Wasserstein} = -\mathbb{E}_{z\sim p_z,y\sim p_{data}}[D(G(z,y))], \quad (41)$$

where $D(\cdot,\cdot)$ is 1-Lipschitz.

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:

$$L^{(G)}_{feature\_matching} = \left\| \mathbb{E}_{x\sim p_{data}(x)} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z)) \right\|_2^2 \tag{42}$$

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:

$$L^{(G)}_{avg\_mag} = \mathbb{E}_{(x,y)\sim p_{data}, z\sim p_z}[(M(x) - M(G(z,y)))^2], \tag{43}$$

and we calculate the average magnetization across all Trotter slices:

$$M(\mathbf{s}) = \frac{1}{b\tau} \sum_{i=1}^{b} \sum_{l=1}^{\tau} s_{il}. \tag{44}$$

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

$$J^{(G)}_{nonWasserstein} = L^{(G)}_{GAN\,heuristic} - L_{\hat{D}}(G(Z,Y_c),Y_c) + L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag}, \tag{45}$$

or, in the case of Wasserstein GAN:

$$J^{(G)}_{Wasserstein} = L^{(G)}_{Wasserstein} - L_{\hat{D}}(G(Z,Y_c),Y_c) + L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag}, \tag{46}$$

where the second term, $L_{\hat{D}}(G(Z,Y_c),Y_c)$, is taken from the GAN-MI objective (37), which is any MI lower bound maximization formulation keeping the discriminator (D) or the auxiliary network (D_q) fixed.

4.5 SQA-GAN Algorithms

By introducing a conditional label input to the generator, adding mutual information lower bound maximization mechanisms, and (where applicable), incorporating spectral normalization in the discriminator layers to enforce the 1-Lipschitz condition, the SQA-GAN architecture takes the structure in Fig **??**.

4.5.1 SQA-GAN Algorithm with DoE (variational lower bound) MI formulation

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, auxiliary network D_q and the generator G with θ_D, θ_{D_q} and θ_G, respectively, where θ_D and θ_{D_q} share most convolution layers.

Step 1: Feed the real batch of samples, $\{(x_i,y_i), i \in 1,...,m\}$, to the discriminator D, which outputs $[D(x_1,y_1),...,D(x_m,y_m)]^T$. The auxiliary network D_q outputs $\mathbf{D}_{q(real)} = [D_q(y_1|x_1),...,D_q(y_m|x_m)]$, which are estimated posterior probabilities. Calculate $L_D(X,Y)$ with $\mathbf{D}_q$ using (36).

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, $\tilde{\mathbf{x}} = \{\tilde{x}_1,...,\tilde{x}_m\} = \{G(z_1,y_1),\ldots,G(z_m,y_m)\}$.

Step 3: Feed synthetic samples, $\{\tilde{x}_1,...,\tilde{x}_m\}$ to D. For each $\tilde{x}_i, i \in \{1,\ldots,m\}$, D outputs $D(\tilde{x}_i,y_i)$, and the auxiliary network D_q outputs $\mathbf{D}_{qfake} = [D_q(y_i|\tilde{x}_i), i \in \{1,\ldots,m\}]$. Calculate $L_{\hat{D}}$ with $\mathbf{D}_{qfake}$ using (36).

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

$$\begin{aligned} J^{(D)}_{nonWasserstein} = & \\ & -\{E_{x,y\sim p_{real}(x,y)}[\log D(x,y)] + \\ & E_{z\sim p_z, y\sim p_{real}(y)}[\log(1-D(G(z,y),y)]\} - L_D(X,Y) \approx \\ & -\frac{1}{m}\sum_{i=1}^{m} log\, D(x_i,y_i) - \frac{1}{m}\sum_{i=1}^{m} log\,(1-D(\tilde{x}_i,y_i)) - \\ & \frac{1}{m}\sum_{i=1}^{m} log\, D_q(y_i|x_i) \end{aligned} \tag{47}$$

or, in the case of Wasserstein GAN:

$$\begin{aligned} J^{(D)}_{Wasserstein} = & \\ & -\{E_{x,y\sim p_{real}(x,y)}[D(x,y)] - \\ & E_{z\sim p_z, y\sim p_{real}(y)}[D(G(z,y),y)]\} - L_D(X,Y) \approx \\ & -\frac{1}{m}\sum_{i=1}^{m} D(x_i,y_i) + \frac{1}{m}\sum_{i=1}^{m} D(\tilde{x}_i,y_i) - \\ & \frac{1}{m}\sum_{i=1}^{m} log\, D_q(y_i|x_i) \end{aligned} \tag{48}$$

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

$$\begin{aligned} J^{(G)}_{nonWasserstein} &= L^{(G)}_{GAN\,heuristic} - L_{\hat{D}}(G(Z,Y_c),Y_c) + \\ &\quad L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag} \\ &= -\frac{1}{m}\sum_{i=1}^{m} log\,\hat{D}(G(z_i,y_i),y_i) - \frac{1}{m}\sum_{i=1}^{m} log\,\hat{D}_q(y_i|G(z_i,y_i)) \\ &\quad + ||\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(G(z_i,y_i))||_2^2 \\ &\quad + \frac{1}{m}\sum_{i=1}^{m}(M(x_i) - M(G(z_i,y_i)))^2, \end{aligned} \tag{49}$$

or in the case of Wasserstein GAN:

$$\begin{aligned} J^{(G)}_{Wasserstein} &= L^{(G)}_{Wasserstein} - L_{\hat{D}}(G(Z,Y_c),Y_c) + \\ &\quad L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag} \\ &= -\frac{1}{m}\sum_{i=1}^{m} \hat{D}(G(z_i,y_i),y_i) - \frac{1}{m}\sum_{i=1}^{m} log\,\hat{D}_q(y_i|G(z_i,y_i)) \\ &\quad + ||\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(G(z_i,y_i))||_2^2 \\ &\quad + \frac{1}{m}\sum_{i=1}^{m}(M(x_i) - M(G(z_i,y_i)))^2, \end{aligned}$$

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

$$\theta_D \leftarrow \theta_D - \eta \cdot \nabla_{\theta_D} J^{(D)} \tag{50}$$

$$\theta_{D_q} \leftarrow \theta_{D_q} - \eta \cdot \nabla_{\theta_{D_q}} J^{(D)} \tag{51}$$

$$\theta_G \leftarrow \theta_G - \eta \cdot \nabla_{\theta_G} J^{(G)} \tag{52}$$

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

4.5.2 SQA-GAN Algorithm with MI Contrastive Learning

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, $\{(x_i,y_i), i \in 1,...,m\}$, to the discriminator D, which outputs $[D(x_1,y_1),...,D(x_m,y_m)]^T$. Then for each x_i, create incongruent pairs: $\{(x_i,y_j), i \neq j, \forall i,j \in 1,...,m\}$, for m samples in a batch, there are be $m(m-1)$ incongruent pairs. Use the congruent and incongruent pairs to calculate the contrastive $L_D(X,Y)$.

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, $\tilde{\mathbf{x}} = \{\tilde{x}_1,...,\tilde{x}_m\} = \{G(z_1,y_1),\ldots,G(z_m,y_m)\}$.

Step 3: Feed synthetic samples, $\{\tilde{x}_1,...,\tilde{x}_m\}$ to D. For each $\tilde{x}_i, i \in \{1,\ldots,m\}$, D outputs $D(\tilde{x}_i,y_i)$. Then for each $\tilde{x}_i$, create incongruent pairs: $\{(\tilde{x}_i,y_j), i \neq j, \forall i,j \in 1,...,m\}$, for m samples in a batch, there are be $m(m-1)$ incongruent pairs. Use the congruent and incongruent pairs to calculate the contrastive $L_{\hat{D}}(\tilde{X},Y) = L_{\hat{D}}(G(Z,Y),Y)$.

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

$$\begin{aligned} &J^{(D)}_{nonWasserstein} = \\ &\quad -\{E_{x,y\sim p_{real}(x,y)}[\log D(x,y)] + \\ &E_{z\sim p_z, y\sim p_{real}(y)}[\log(1-D(G(z,y),y)]\} - L_D(X,Y) \approx \\ &\quad -\frac{1}{m}\sum_{i=1}^{m} log\,D(x_i,y_i) - \frac{1}{m}\sum_{i=1}^{m} log\,(1-D(\tilde{x}_i,y_i)) - \\ &\quad L_D(X,Y) \end{aligned} \tag{53}$$

or, in the case of Wasserstein GAN:

$$\begin{aligned} &J^{(D)}_{Wasserstein} = \\ &\quad -\{E_{x,y\sim p_{real}(x,y)}[D(x,y)] - \\ &E_{z\sim p_z, y\sim p_{real}(y)}[D(G(z,y),y)]\} - L_D(X,Y) \approx \\ &\quad -\frac{1}{m}\sum_{i=1}^{m} D(x_i,y_i) + \frac{1}{m}\sum_{i=1}^{m} D(\tilde{x}_i,y_i) - \\ &\quad L_D(X,Y) \end{aligned} \tag{54}$$

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

$$\begin{aligned} J^{(G)}_{nonWasserstein} &= L^{(G)}_{GAN\,heuristic} - L_{\hat{D}}(G(Z,Y_c),Y_c) + \\ &\quad L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag} \\ &= -\frac{1}{m}\sum_{i=1}^{m} log\,\hat{D}(G(z_i,y_i),y_i) - L_{\hat{D}}(G(Z,Y_c),Y_c) \\ &\quad + ||\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(G(z_i,y_i))||_2^2 \\ &\quad + \frac{1}{m}\sum_{i=1}^{m}(M(x_i) - M(G(z_i,y_i)))^2, \end{aligned} \tag{55}$$

or in the case of Wasserstein GAN:

$$\begin{aligned} J^{(G)}_{Wasserstein} &= L^{(G)}_{Wasserstein} - L_{\hat{D}}(G(Z,Y_c),Y_c) + \\ &\quad L^{(G)}_{feature\_matching} + L^{(G)}_{avg\_mag} \\ &= -\frac{1}{m}\sum_{i=1}^{m} \hat{D}(G(z_i,y_i),y_i) - L_{\hat{D}}(G(Z,Y_c),Y_c) \\ &\quad + ||\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{m}\sum_{i=1}^{m} f(G(z_i,y_i))||_2^2 \\ &\quad + \frac{1}{m}\sum_{i=1}^{m}(M(x_i) - M(G(z_i,y_i)))^2, \end{aligned}$$

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

$$\theta_D \leftarrow \theta_D - \eta \cdot \nabla_{\theta_D} J^{(D)} \tag{56}$$

$$\theta_G \leftarrow \theta_G - \eta \cdot \nabla_{\theta_G} J^{(G)} \tag{57}$$

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

5 Experiments

5.1 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.

5.2 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 $32 * 32 * 20$.

5.3 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.

Average Magnetization Distribution The average magnetization distribution is an effective overview of an Ising simulator's statistical behavior. Besides comparing the average magnetization loss $L^{(G)}_{avg\_mag}$, 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 Experimental Results & Analysis

7 Conclusion

In this paper, we have demonstrated the use of GAN in the field of simulated quantum annealing. Specifically, we showed that the GAN framework combined with mutual information maximization objectives is a capable tool to capture the spin distributions at continuous transverse fields in the (2+1)-dimensional path-integral MCMC ising model. We incorporated and compared three MI maximization strategies: Variational Mutual Information Maximization through Difference of Entropies (DoE), Noise-contrastive Estimation (NCE) based formulation, and Wasserstein Dependency Measure (WDM) and conducted ablation studies to research the effect of different MI lower bound maximization formulations, as well as the Wasserstein GAN objective.

This study not only proves the capability of information-theoretic and game-theoretic deep learning methods in modeling complex quantum physics systems, but also showcases the promising potential of GAN's data generation capabilities in learning and simulating stochastic systems. Exciting future extensions include transfer learning or knowledge distillation to propagate learnt knowledge from one quantum simulator model to the other, or applying SQA-GAN to other domains with stochastic models such as queuing network, video timeseries models, language models, etc. SQA-GAN can also be used for not just generating stochastic data, but also detecting anomalies: it can detect intruders (characterized by random walks on a graph) in large-scale networks, taking advantage of its immense capability of extracting essential features from complex stochastic processes. Additionally, we would also like to combine mutual information maximization methods with other learning frameworks other than GAN, such as Graph Attention Network, etc.

8 Extension

Relationship and limitations of Wasserstein and Kullback-Leibler Why does mutual information have to be defined as the KL divergence between $p(x,y)$ and $p(x)p(y)$? What are the advantages or disadvantages between the KL MI definition and the Wasserstein Dependency Measure (WDM)? Belavkin (2018) investigated the relationship between the Wasserstein metric and the Kullback-Leibler divergence using variational and geometric principles. They have found that the Wasserstein distance is equivalent to Optimal Channel (OCP) with one additional constraint fixing the output measure, and therefore OCP with constraints on the KL divergence gives a lower bound on the Wasserstein metric. Inspired by Belavkin (2018), one can further study the convergence or tightness of bound using hybrid formulations such as the theoretic understanding of enforcing 1-Lipschitz on the KL divergence MI (as in WPC), or even

come with newer and more robust distribution divergence measures based on information and game theory.

Mutual Information connection to deep metric learning Moreover, Tschannen et al. (2020) showed that the success of mutual information methods might have connections to deep metric learning. They argue that studying MI estimator through the lens of deep metric learning sheds more light on the importance of appropriately choosing the negative samples, which is "indeed a critical component in deep metric learning based on triplet losses".

New paradigm? According to (66) in the Appendix, which is replicated here:

$$\frac{N}{M}\underbrace{\mathbb{E}_{p(x^-)p(y^-)}log\,p(x^-)p(y^-)}_{\leq 0}+\underbrace{\mathbb{E}_{p(x^+,y^+)}log\,p(x^+)p(y^+)}_{\leq 0}. \tag{58}$$

Lowering the ratio $\frac{N}{M}$ actually decreases that gap between $\sup L_{contrast}$ and $I(X,Y)$, meaning that one should have fewer incongruent (negative) samples (N) and more congruent (positive) samples (M). Will, then, overwhelming the congruent samples with incongruent ones achieve a better performance? The problem for the estimator function $q(x,y)$ then becomes "correctly detecting a few negative samples from lots of positive samples".

9 Appendix

9.1 Proof of unifying NCE formulation is lower bound to Mutual Information

We show that the following formulation is a unifying expression of contemporary Noise-contrastive Estimation (NCE) based Mutual Information (MI) lower bound maximization objectives, and that it is the lower bound to the MI between $X\sim p(x)$ and $Y\sim p(y)$:

$$L_{contrast}=\frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)}log\,q(s^+,s^-), \tag{59}$$

where $s^+=\{(x^+,y^+)_i\sim p(x,y),i=1,...,M\}$ is a set of congruent data-label pairs sampled from the joint distribution and $s^-=\{(x^-,y^-)_j\sim p(x)p(y),j=1,...,N\}$ is a set of incongruent data-label pairs sampled from the product of marginals.

Lemma 1 *The two NCE-based objectives (original NCE and infoNCE, which is also called Contrastive Predictive Encoding) model $q(s^+,s^-)$ as the likelihood of a set of Bernoulli observations and the likelihood of a set of observations parameterized by the softmax function respectively.*

Proof. For the case of original NCE, we let $f(\cdot)$ denote any tractable function (such as modeled by a neural network) that approximates the Bernoulli parameter (the probability estimator that pair $(x,y)_i$ is from the congruent joint distribution $p(x,y)$, so $1-f((x,y))$ is the probability that $(x,y)_i$ is from the product of marginals $p(x)p(y)$):

$$\begin{aligned}
L_{NCE}&=\frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)}log\,q(s^+,s^-)\\
&=\frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)}log(\prod_{i=1}^{M}f((x^+,y^+)_i)\prod_{j=1}^{N}(1-f((x^-,y^-)_j)))\\
&=\frac{1}{M}\mathbb{E}_{p(s^+)}\sum_{i=1}^{M}log(f((x^+,y^+)_i)+\\
&\quad\frac{1}{M}\mathbb{E}_{p(s^-)}\sum_{j=1}^{N}log(1-f((x^-,y^-)_j)))\\
&=\mathbb{E}_{p(x,y)}log(f((x,y))+\frac{N}{M}\mathbb{E}_{p(x)p(y)}log(1-f((x,y)))),
\end{aligned} \tag{60}$$

where the last line is the original NCE objective (Gutmann and Hyvärinen (2010), Tian et al. (2020)).

For the case of infoNCE, there is only one congruent par for every N incongruent pairs, i.e., $|s^+|=1,|s^-|=N$ we let $f(\cdot)$ denote any tractable function (such as modeled by a neural network) that approximates the logit function in softmax expression $\frac{exp(f((x^+,y^+)))}{exp(f((x^+,y^+)))+\sum_{j=1}^{N}exp(f((x^-,y^-)_j))}\in[0,1]$, which can be thought as the likelihood ($\in[0,1]$) of observing s^+ and s^-:

$$\begin{aligned}
L_{infoNCE}&=\frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)}log\,q(s^+,s^-)\\
&=\frac{1}{M}\mathbb{E}_{p(x^+,y^+)p(s^-)}log\,\frac{exp(f((x^+,y^+)))}{exp(f((x^+,y^+)))+\sum_{j=1}^{N}exp(f((x^-,y^-)_j))}\\
&=\frac{1}{M}\mathbb{E}_{p(x^+,y^+)}f((x^+,y^+))-\\
&\frac{1}{M}\mathbb{E}_{p(x^+,y^+)p(s^-)}log\left[exp(f((x^+,y^+)))+\sum_{j=1}^{N}exp(f((x^-,y^-)_j))\right].
\end{aligned} \tag{61}$$

■

Lemma 2 *The NCE expression, when maximized with respect to the likelihood estimator function $q(\cdot)\in[0,1]$, is a lower bound to $I(X,Y)$:*

$$\sup_{q(\cdot)\in[0,1]}L_{contrast}=\sup_{q(\cdot)\in[0,1]}\frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)}logq(s^+,s^-)\leq I(X,Y) \tag{62}$$

Proof. Data-label pairs $(x,y)_i$ in congruent set $s^+=\{(x_i^+,y_i^+),i\in\{1,...,M\}\}$ and incongruent set $s^-=\{(x_i^-,y_i^-),i\in\{1,...,N\}\}$ are independently obtained, which means that the probability of observing s^+ is $p(s^+)=\prod_{i=1}^{M}p(x_i^+,y_i^+)$ and similarly, $p(s^-)=\prod_{i=1}^{N}p(x_i^-)p(y_i^-)$. By

Gibb's inequality, we have:

$$\begin{aligned} L_{contrast} &= \frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)} log\, q(s^+,s^-) \\ &\leq \frac{1}{M}\mathbb{E}_{p(s^+)p(s^-)} log\, p(s^+)p(s^-), \end{aligned} \tag{63}$$

where the equality is achieved when $q(s^+,s^-) = p(s^+)p(s^-)$. We can then write:

$$\begin{aligned} &\sup_{q(\cdot)\in[0,1]} M \cdot L_{contrast} = \mathbb{E}_{p(s^+)p(s^-)} log\, p(s^+)p(s^-) \\ &= \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)\prod_{i=1}^N p(x_i^-)p(y_i^-)} log\left[\prod_{i=1}^M p(x_i^+,y_i^+)\prod_{i=1}^N p(x_i^-)p(y_i^-)\right] \\ &= \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)\prod_{i=1}^N p(x_i^-)p(y_i^-)} \\ &\quad log\left[\prod_{i=1}^M \frac{p(x_i^+,y_i^+)}{p(x_i^+)p(y_i^+)}\prod_{i=1}^N p(x_i^-)p(y_i^-)\prod_{i=1}^M p(x_i^+)p(y_i^+)\right] \\ &= \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)\prod_{i=1}^N p(x_i^-)p(y_i^-)}\left[\sum_{i=1}^M log\,\frac{p(x_i^+,y_i^+)}{p(x_i^+)p(y_i^+)}\right. \\ &\quad \left. + \sum_{i=1}^N log\left[p(x_i^-)p(y_i^-)\right] + \sum_{i=1}^M log[p(x_i^+)p(y_i^+)]\right] \\ &= \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)} \sum_{i=1}^M log\,\frac{p(x_i^+,y_i^+)}{p(x_i^+)p(y_i^+)} + \\ &\quad \mathbb{E}_{\prod_{i=1}^N p(x_i^-)p(y_i^-)} \sum_{i=1}^N log[p(x_i^-)p(y_i^-)] + \\ &\quad \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)} \sum_{i=1}^M log[p(x_i^+)p(y_i^+)] \\ &= \sum_{i=1}^M \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)} log\,\frac{p(x_i^+,y_i^+)}{p(x_i^+)p(y_i^+)} + \\ &\quad \sum_{i=1}^N \mathbb{E}_{\prod_{i=1}^N p(x_i^-)p(y_i^-)} log[p(x_i^-)p(y_i^-)] + \\ &\quad \sum_{i=1}^M \mathbb{E}_{\prod_{i=1}^M p(x_i^+,y_i^+)} log[p(x_i^+)p(y_i^+)] \\ &= M\mathbb{E}_{p(x^+,y^+)} log\,\frac{p(x^+,y^+)}{p(x_i^+)p(y_i^+)} + N\mathbb{E}_{p(x^-)p(y^-)} log[p(x^-)p(y^-)] + \\ &\quad M\mathbb{E}_{p(x^+,y^+)} log[p(x^+)p(y^+)] \end{aligned} \tag{64}$$

Then divide both sides by M, we have

$$\begin{aligned} \sup_{q(\cdot)\in[0,1]} L_{contrast} &= \underbrace{\mathbb{E}_{p(x^+,y^+)} log\,\frac{p(x^+,y^+)}{p(x_i^+)p(y_i^+)}}_{D_{KL}(p(x,y)\|p(x)p(y))=I(X,Y)} + \\ &\quad \frac{N}{M}\underbrace{\mathbb{E}_{p(x^-)p(y^-)} log[p(x^-)p(y^-)]}_{\leq 0} + \\ &\quad \underbrace{\mathbb{E}_{p(x^+,y^+)} log[p(x^+)p(y^+)]}_{\leq 0} \leq I(X,Y). \end{aligned} \tag{65}$$

■

This decomposition provides a great way to investigate the MI lower bound tightness by investigating the last two terms:

$$\frac{N}{M}\underbrace{\mathbb{E}_{p(x^-)p(y^-)} log\, p(x^-)p(y^-)}_{\leq 0} + \underbrace{\mathbb{E}_{p(x^+,y^+)} log\, p(x^+)p(y^+)}_{\leq 0} \tag{66}$$

which, when maximized, gives the tightest lower bound of $L_{contrast}$ on $I(X,Y)$. Upon further inspection, one finds that the first expectation of (66) is already at its max (due to Gibb's inequality), and that decreasing the ratio $\frac{N}{M}$ will tighten the bound. The second term is at its max when $p(x^+,y^+) = p(x^+)p(y^+)$, which means that X and Y are independent.

9.2 Network configurations

Discriminator Architecture for Auxiliary DoE The discriminator input takes shape $32*32*20$ (height, width, number of trotter slices), 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). The penultimate convolution layer output is concatenated with the label value y before being fed to the last convolution layer. The feature map output by the last convolution layer is flattened and fed into a fully connected dense layer with 1 output neuron for $D(x,y)$. Meanwhile, the penultimate convolution layer (before concatenated with label information y is fed into a shallow NN (the auxiliary network) to output two scalars: the mean and standard deviation for the approximated posterior distribution $q(y|x)$. In the case of Wasserstein GAN objective, the discriminator is 1-Lipschitz enforced my spectral normalization.

The Discriminator-auxiliary network architecture is illustrated in Table.3.

Discriminator Architecture for Contrastive Learning The discriminator for MI contrastive learning formulations does not require an auxiliary network. it is identical to the Discriminator-auxiliary network with auxiliary network removed. The penultimate layer output can be concatenated with congruent label y^+ or incongruent label y^- as per the contrastive learning formulations. In the case of Wasserstein GAN or Wasserstein Dependency Measure objectives, the discriminator is 1-Lipschitz enforced my spectral normalization.

The Discriminator network architecture for contrastive learning is illustrated in Table.2.

Generator Architecture The noise vector $\mathbf{z}$ and the conditional label y are first processed by two separate densely-connected layers and then reshaped and concatenated for subsequent upconvolution operations. No batch normalization is used for the last upconvolution layer per (Radford

Table 1: Configuration of the SQA-GAN generator

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

Layer	Activation	Shape	Note
Input	-	(N, 1)	Condition Label, y_c
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
UpConv	3×3 (128)	ReLU	(N, 16, 16, 64)	Stride = 2
BatchNorm	-	-	(N, 16, 16, 64)	Momentum = 0.8
UpConv	3×3 (64)	ReLU	(N, 32, 32, 32)	Stride = 2
BatchNorm	-	-	(N, 32, 32, 32)	Momentum = 0.8
UpConv	3×3 (3)	ReLU	(N, 32, 32, 20)	Stride = 1, fake sample $G(z, y_c)$

Table 2: Configurations of the SQA-GAN-Contrastive discriminator

Layer	**Filter size (#)**	**Activation**	**Data Shape**	**Notes (α: $-$ive slope coef. in Leaky ReLU)**
Input	-	-	(N, 32, 32, 20)	Input Ising state of size 32×32×20
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, $\alpha = 0.2$
Concat	-	-	(N, 8, 8, 65)	concatenated with $8 \times 8 \times 1$ tensor of label value y
Flatten	-	-	(N, 4160)	4160 = 8×8×65
Fc-layer	-	Softmax	(N, 1)	D(x,y)

Table 3: Configurations of the SQA-GAN-Auxiliary discriminator

Layer	**Filter size (#)**	**Activation**	**Data Shape**	**Notes (α: $-$ive slope coef. in Leaky ReLU)**
Input	-	-	(N, 32, 32, 20)	Input Ising state of size 32×32×20
Conv 1	3×3 (32)	Leaky ReLU	(N, 16, 16, 32)	Stride = 2, $\alpha = 0.2$
Dropout	-	-	(N, 16, 16, 32)	Dropout rate = 0.25
Conv 2	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	3×3 (64)	Leaky ReLU	(N, 8, 8, 64)	Stride = 1, $\alpha = 0.2$
FC-batchnorm-lReLU-FC	-	-	(N, 2)	Auxiliary network's output (mean, std) for $q(y\|x)$
Concat	-	-	(N, 8, 8, 65)	Conv 3 concatenated with $8 \times 8 \times 1$ tensor of label y
Flatten	-	-	(N, 4160)	4160 = 8×8×65
Fc-layer 2	-	Softmax	(N, 1)	D(x,y)

et al. (2015)). The Generator architecture is illustrated in Table.1.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *ArXiv*, abs/1701.07875.

Baldi, P., Sadowski, P., and Whiteson, D. (2015). Enhanced higgs boson to $\tau^+\tau^-$ search with deep learning. *Phys. Rev. Lett.*, 114:111801.

Barber, D. and Agakov, F. (2003). The im algorithm: a variational approach to information maximization. In *NIPS 2003*.

Belavkin, R. V. (2018). Relation between the kantorovich–wasserstein metric and the kullback–leibler divergence. *Springer Proceedings in Mathematics Statistics*, page 363–373.

Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. (2018). Mine: Mutual information neural estimation.

Chen, Y., Lin, Z., Zhao, X., Wang, G., and Gu, Y. (2014). Deep learning-based classification of hyperspectral data. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 7:2094–2107.

Farhi, E., Goldstone, J., and Gutmann, S. (2002). Quantum adiabatic evolution algorithms versus simulated annealing. *arXiv: Quantum Physics*.

Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., and Preda, D. (2001). A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292:472 – 475.

Farhi, E., Goldstone, J., Gutmann, S., and Sipser, M. (2000). Quantum computation by adiabatic evolution. *arXiv: Quantum Physics*.

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

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

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans.

Gutmann, M. and Hyvärinen, A. (2010). Onoise-contrastive estimation: A new estimation principle for unnormalized statistical models.

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.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–44.

Liu, Z., Rodrigues, S. P., and Cai, W. (2017). Simulating the ising model with a deep convolutional generative adversarial network.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks.

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.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861.

Ozair, S., Lynch, C., Bengio, Y., Oord, A., Levine, S., and Sermanet, P. (2019). Wasserstein dependency measure for representation learning. In *NeurIPS*.

Poole, B., Ozair, S., van den Oord, A., Alemi, A. A., and Tucker, G. (2019). On variational bounds of mutual information.

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

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.

Singh, J., Arora, V., Gupta, V., and Scheurer, M. S. (2020). Generative models for sampling and phase transition indication in spin systems.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive representation distillation.

Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M. (2020). On mutual information maximization for representation learning.

van den Oord, A., Li, Y., and Vinyals, O. (2019). Representation learning with contrastive predictive coding.

Villani, C. (2009). *Optimal Transport, Old and New*.

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