Exploring Artificial Spin Ice Through Machine Learning Analysis

by

Mahdis Hamdi

A thesis submitted to The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

Master of Science

Department of Physics and Astronomy
The University of Manitoba
Winnipeg, Manitoba, Canada
November 2023

Thesis advisor

Robert Stamps

Author

Mahdis Hamdi

Exploring Artificial Spin Ice Through Machine Learning Analysis

Abstract

Artificial spin ice (ASI) systems have emerged as a captivating research field due to their ability to exhibit intriguing properties, including topological defects, magnetic monopoles, and complex spin textures. These systems offer a versatile platform for studying geometric frustration and exploring emergent magnetic phenomena. Moreover, the controllable nature of ASIs holds promise for potential applications in data storage, microwave filtering, and spintronics. ASI also serve as valuable model systems for understanding the physics of magnetic materials and frustrated systems.

This thesis focuses on investigating the accuracy of a machine learning model on ASI materials using the framework of the restricted Boltzmann machine (RBM). RBM offers interpretability within the realm of statistical physics and can effectively analyse and interpret the vast amount of data generated in condensed matter physics experiments and simulations.

By leveraging RBM, we aim to gain deeper insights into the behaviour and characteristics of defects in ASI systems. Defects possess unique properties and dynamics that hold potential for information storage, logic operations, and magnetic texture manipulation. Understanding and engineering these defects allow for precise control

over the magnetic properties of ASI, including interactions and anisotropy, enabling tailored functionalities for specific applications.

This research explores the accuracy and effectiveness of RBM models specifically applied to ASI systems. By utilizing RBM, we aim to analyse and uncover the intricate behaviours and features of defects in ASI, contributing to a deeper understanding of these materials and their potential applications.

The findings from this study provide valuable insights into the behaviour of defects in ASI materials and offer avenues for optimizing these systems for desired functionalities. By leveraging machine learning techniques, such as RBM, we can further explore and harness the potential of ASI systems for future advancements in magnetic materials and condensed matter physics.

Contents

List of Figures

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Professor Robert Stamps, for granting me the opportunity to work under his guidance. This research would not have been completed successfully without his invaluable assistance, insightful suggestions, encouragement, consideration, and kindness. I am truly grateful for his mentorship and expertise.

I would also like to extend my heartfelt gratitude to my beloved husband, for his unwavering support throughout this journey. His constant encouragement, understanding, and belief in my abilities have been a source of strength and motivation. I am grateful for his presence by my side.

I would like to acknowledge the tremendous support my family and friends have given me. Their unwavering confidence in my capabilities and continuous support has been the driving force behind my achievements. I also want to thank my friend and colleague Rehana Poppy for all the help and advice she gave me at every step of my study project.

Susan Beshta, an office assistant, and Robyn Beaulieu, an administrative assistant, have helped and supported me a lot during this study project.

Furthermore, I would like to express my gratitude to the Natural Science and Engineering Research Council (NSERC) and Research Manitoba for their generous support and funding, which have made this research possible.

I am immensely grateful to all those who have contributed to this journey and have played a part in its success.

Research Tools

In this thesis, I benefited from some powerful tools for the simulations, modelling, and enhancing my language proficiency.

I used Python [3] and its powerful libraries such as NumPy [4], SciPy [5], Random [6], and Math [6] to run simulations using the Monte Carlo approach on both the Ising model system and the artificial spin ice system. The results were plotted using Matplotlib [7].

I also used Python [3] to build machine learning models with the help of modules like NumPy [4], SciPy [5], Random [6], Math [6], Matplotlib [7], and Scikit-learn [8].

Lastly, effective communication of findings was important, and this is where AI-driven tools came into play. Tools like QuillBot [9] and Grammarly [10] provided assistance to enhance my language proficiency.

Chapter 1

Introduction

1.1 Background: Artificial spin ice

In 1933, Bernal and Fowler [28] observed that hydrogen atoms in water ice would remain disordered even at absolute zero. This implies that water ice retains a certain randomness, even when cooled to its lowest temperature. The hexagonal crystal structure of common water ice is responsible for this, as oxygen atoms in the structure are connected to four neighboring hydrogen atoms. Two of these hydrogen atoms are near, forming the traditional H_2O molecule, while the other two are farther away, belonging to adjacent water molecules. Pauling [11] noticed that configurations adhering to this "two-near, two-far" rule grow exponentially with system size. This discovery was supported by specific heat measurements, although creating pure crystals of water ice is quite challenging. Spin ices are materials with regular corner-linked tetrahedra of magnetic ions, each possessing a magnetic moment or "spin." These spins must follow a "two-in, two-out" rule in their low-energy state within each tetrahedron of the

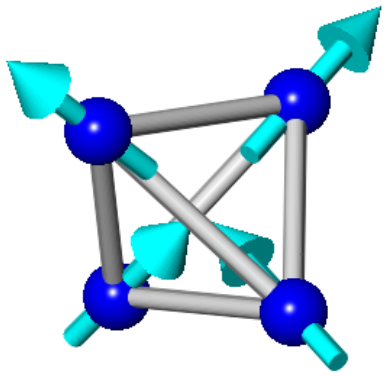


Figure 1.1: The alignment of the magnetic moments (shown by light blue arrows) is being discussed in relation to an individual tetrahedron within the spin ice state. In this context, the magnetic moments follow the rule of two-in, two-out. Image source: Wikipedia.

crystalline structure, as depicted in Figure 1.1. Similar to Pauling's demonstration that the ice rule results in significant entropy in water ice, the two-in, two-out rule in spin ice systems also generates comparable residual entropy properties resembling those of water ice [12].

Although Philip Anderson [13] recognized the connection between the frustrated Ising anti-ferromagnet issue on a lattice of corner-shared tetrahedra and Pauling's water ice dilemma in 1956, actual spin ice materials weren't discovered until forty years later. The initial spin ice materials identified were the pyrochlores $Dy_2Ti_2O_7$(dysprosium titanate) and $Ho_2Ti_2O_7$ (holmium titanate). There's also strong evidence that $Dy_2Sn_2O_7$ (dysprosium stannate) and $Ho_2Sn_2O_7$ (holmium stannate) exhibit spin ice behaviour. These compounds belong to the rare-earth pyrochlore oxide family. Additionally, $CdEr_2Se_4$, a spinel with magnetic Er^{3+} ions on corner-linked tetrahedra, displays

spin ice properties [14].

Artificial spin ices are unique materials made up of linked nano-magnets placed in patterns that can be regular or irregular. The intriguing aspect of artificial spin ice lies in its engineered nature. By fabricating arrays of nanoscale magnetic elements, often in materials like permalloy, into various lattice geometries such as square [29], honeycomb [30], or kagome [31], researchers can create and study systems where the magnetic frustration and its resulting phenomena can be observed and manipulated. Unlike their natural counterparts, these artificial systems allow for the direct observation of individual magnetic moments, as well as the ability to tailor the lattice geometry and interaction strengths between the elements.

The study of artificial spin ice has opened up new avenues for exploring fundamental questions in statistical mechanics and magnetism. For example, these systems have been instrumental in enhancing the understanding of frustration and degeneracy in magnetic systems [2], the emergence of magnetic monopoles [15], and changes between different magnetic phases [16]. These materials also have the potential to be used as reprogrammable magnonic crystals and have been investigated for their dynamics [17]. The ability to directly visualize and manipulate the magnetic states in these artificial lattices provides a powerful platform for testing theoretical models and exploring new phenomena.

1.2 Background: Machine Learning

Machine learning is a branch of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to perform specific

tasks without explicit instructions. Instead, these systems learn and make decisions based on patterns and inferences from data. It involves creating algorithms capable of recognizing patterns and making predictions based on the data. Machine learning models learn from examples and data to make informed decisions or generate outputs. This process relies on statistical inference and optimization techniques to achieve tasks that appear intelligent, such as generating samples, image recognition, text analysis, and real-world problem-solving. AI's primary goal is to make machines act smart, and it gives computers the ability to learn without being told what to do. Machine learning is a way to train a computer program by giving it information like numbers, pictures, or text. The more information that's available, the better the program can learn. Programmers choose a machine learning model, give the program the data, and let the program find patterns or make predictions. They can change the model to improve its accuracy. Some data is set aside to test the model's accuracy with new information. Machine learning algorithms that work well can explain what happened in the past, predict what will happen in the future, or suggest what to do based on the data [18]. Supervised, unsupervised, and reinforcement learning are the three primary categories of machine learning algorithms. Supervised machine learning models need labeled data. For example, a program could be taught to recognize pictures of dogs by showing pictures of dogs and other things that people have labeled. The machine would then learn how to recognize pictures of dogs on its own. Unsupervised machine learning is when a program looks for trends in data that haven't been labeled. Unsupervised machine learning can discover patterns or trends that humans do not actively seek. For instance, an unsupervised machine learning program could

look at online sales data and determine what kinds of customers are buying things. In a grocery store, people who buy cereal more often tend to buy milk, and people who buy eggs more often tend to buy bacon. The machine would learn these relationships and trends. Reinforcement learning is a type of machine learning where an agent learns to make decisions by trying different actions and seeing the results. It's like learning through trial and error. The agent, which is the learner or decision-maker, interacts with the environment, makes choices, and gets feedback in the form of rewards or penalties. The goal is to learn the best actions to take in different situations to get the most reward over time. This learning process is useful in many real-world applications. For example, it's used in robotics for tasks like picking up objects, in video games where AI learns to play and win, and even in self-driving cars to make decisions on the road. The balance between trying new actions and using what it already knows works well is important for the agent to learn effectively [19]. Image recognition, natural language processing, and recommendation systems are just a few of the real-world problems that have been solved using machine learning. With more and more data and computer power becoming available, machine learning is becoming an increasingly important tool in many areas of science, engineering, and business.

1.3 Background: Artificial Neural Network

Artificial neural networks are a popular type of machine learning method that mimics how living things learn. Artificial neural networks are based on the idea that the brain learns from experience [20]. Usually, they are made up of hundreds of simple working units that are wired together in a complicated communication network. Each

"unit" or "node" is a simplified model of a real neuron that sends out a new signal or "fires" when it gets a strong enough "Input" signal from the other nodes it's linked to. Fundamentally, the architecture of every artificial neural network follows a consistent pattern. Certain neurons connect with the external environment to acquire information within this arrangement. Meanwhile, additional neurons transmit the network's outcomes to the actual world, and some neurons utilize these outcomes as input. The remaining neurons operate behind the scenes, in a hidden layer. The connections between the neurons are weighted, and these weights are changed during training to show the strength of each input in the final results [21]. Neural networks have various uses, like classifying, guessing, and creating. They stand out, especially in tasks that deal with complex and unstructured data, such as recognizing images or voices.

There are many different kinds of neural networks, and each one has its own pros and cons. Each type of network is made to work best with a specific type of problem or data, and the type of network to use depends on the problem being solved.

1.4 Background: Restricted Boltzmann Machine (RBM)

A Boltzmann machine is a computational model that draws inspiration from both statistical mechanics and cognitive science. It's named after the Boltzmann distribution, a concept from statistical mechanics. This model is used in machine learning and science research and was promoted by researchers such as Geoffrey Hinton, Terry

Sejnowski, and David Ackley [22]. Think of a Boltzmann machine as a network of tiny interconnected units, where each unit can be in different states, on or off. The machine undergoes a process where it chooses a unit and changes its state, repeating this process many times. Over time, the machine starts to settle into patterns dictated by energy levels. This pattern reflects the relationship between different states and helps us understand how the system works. Its connections need to be set up correctly to make the machine functional. This looks like arranging puzzle pieces. Training the machine involves adjusting its connections so that the most likely states correspond to the lowest energy levels. This training process aligns the machine's behavior with an external distribution of states, enhancing its usefulness for practical problems.

A restricted Boltzmann machine (RBM) is an advanced type of artificial neural network used for learning complex patterns and relationships in data. The term "restricted" indicates that it follows certain rules to simplify its connections. RBMs are adept at understanding data structures and have gained significant attention, mainly due to their application in machine learning under the guidance of researchers like Geoffrey Hinton [22]. RBMs operate by establishing connections between different elements in a specific manner. These connections form a pattern that helps the RBM understand and process data efficiently. This pattern allows the RBM to learn from the data and make predictions. The RBM's training process can be adapted to different tasks, whether they involve learning from labeled examples or discovering patterns in data without specific labels. You can think of RBMs as a type of building block for making robust computer systems called deep belief networks. These networks are like

a bunch of connected RBMs that work together to understand even more complicated things [23]. So, RBMs are like the building blocks that help us create more intelligent computers.

1.5 Objective

The primary objective of this research is to utilize Restricted Boltzmann Machines (RBMs) as a powerful tool to unveil intricate features within Artificial Spin Ice (ASI) systems. The focus is on utilizing RBMs to uncover detailed patterns and characteristics in ASI configurations. This exploration will help with the detection of specific vertical elements within the ASI structure.

A key objective is utilizing the RBM framework's capabilities to find flaws in the ASI system. This entails quantifying their density and precisely locating them. Training the RBM on ASI data aims to make it easier and more accurate to find defects than with current methods. This could give us a better understanding of the defects and engineering defects in other scientific applications.

By using RBMs to create samples with greater accuracy and detail, this research also intends to improve simulation capabilities. RBMs have the ability to speed up the generation of samples that capture complex features in a lot less time than traditional Monte Carlo simulations. The quality of the simulation results is maintained or even improved, while the computing effort is significantly reduced as a result. For example, the study tries to find features that would typically take 600k Monte Carlo steps with only 60k steps using the RBM framework. This speeds up the simulation process.

Ultimately, the primary objective of this study is to better understand ASI sys-

tems through the utilization of Restricted Boltzmann Machines (RBMs) as a flexible method for extracting features, detecting defects, and improving simulations. The aim is to contribute to developing more efficient and accurate techniques for characterizing and simulating complex materials, with a focus on ASI structures.

1.6 Overview of this thesis

The thesis primarily focuses on applying a machine learning model to artificial spin ice materials. We utilized an energy-based model known as RBM for this study. Initially, we created a program for RBM from scratch and tested it using simple alphabetic data. Later, we applied it to reproduce Ising model data to assess our program's accuracy. Subsequently, we employed ASI data and fine-tuned the RBM, experimenting with various hyper-parameters to optimize the machine learning model's performance. We achieved promising outcomes from our program and proceeded to evaluate the RBM's effectiveness on systems containing vertical elements.

In Chapter 2, we went into detail to explain the artificial spin ice (ASI) system, detailing the interaction model for ASI systems, the required simulation methods for capturing system responses, and the concept of vertical elements.

Chapter 3 briefly introduced machine learning and neural networks, explaining the restricted Boltzmann machine (RBM) in detail. We explained the algorithm used by this model and discussed the important learning methods. The chapter also showcased the results of testing our RBM using basic alphabetic data and reproducing data from recent works on the Ising model.

Chapter 4 delved into the process of determining the optimal RBM hyper-parameters

for our ASI system. We then utilized the RBM on artificial spin ice data, evaluating its reliability and dissecting the outcomes. Moreover, we extended our investigation by introducing vertical elements to our system. We examined how well the RBM performed in this new setup and assessed its accuracy in dealing with the observed differences within the system.

Chapter 2

Artificial Spin Ice

2.1 Geometrical Frustration

Geometrical frustration occurs when the geometry of a lattice and the magnetic interactions between its constituent elements, such as spins, prevent the system from simultaneously satisfying all of its constraints. This phenomenon can result in the system being unable to settle into a single, low-energy state and instead exhibiting a high degree of degeneracy or multiple possible states with similar energies [24]. To illustrate this concept, let's consider three spins arranged in a lattice, as shown in Figure 2.1. These spins are equally spaced from each other. Due to the anti-ferromagnetic interaction, two of the three spins can be in a low-energy state, where their magnetic moments are oriented in opposite directions. However, the third spin will be frustrated because it cannot align with both of its neighboring spins to minimize its energy. The frustrated spin can have two possible orientations: up or down. In one configuration, it will have an anti-ferromagnetic coupling with one of the neighboring

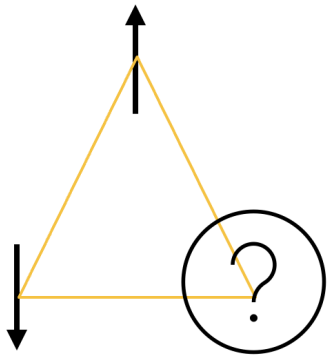


Figure 2.1: A frustrated system of three anti-ferromagnetic. Two of the three spins can be in opposite directions and be in a low-energy state. But the third spin will be frustrated and can be up or down.

spins and a ferromagnetic coupling with the other spin. In the other configuration, the couplings will be reversed.

In total, this system has $2^3 = 8$ possible spin configurations, as shown in Figure 2.2. However, due to geometrical frustration and the magnetic interactions involved, six of these configurations have low energies and the same amount of energy. The remaining two configurations have higher energies.

2.2 Water ice and Spin ice

The name "spin ice" draws an analogy between the behavior of the magnetic spins in these materials and the interactions between water molecules in ice. In water ice, the oxygen ions form a lattice structure, with each oxygen ion surrounded by four hydrogen ions. The hydrogen ions can be considered magnetic moments, with their

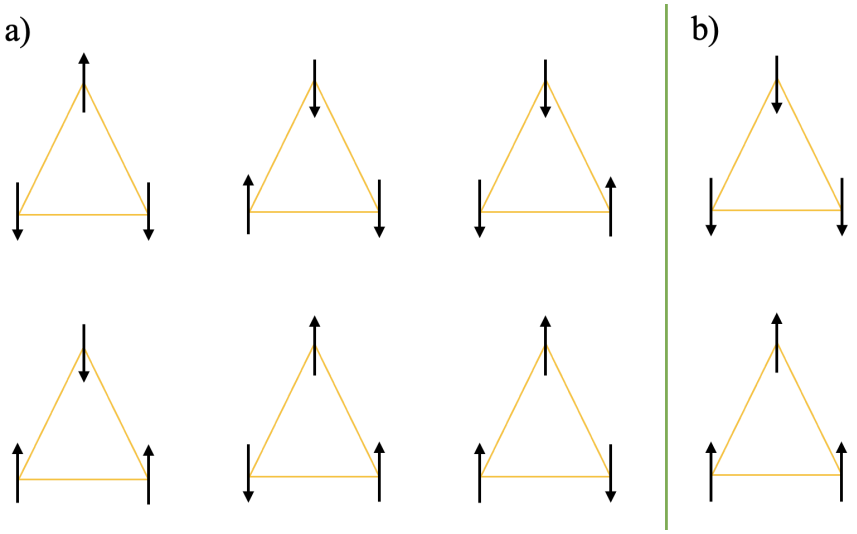


Figure 2.2: Possible spin configurations in a frustrated system of three anti-ferromagnetic. a)Six low-energy configurations. b)Two higher-energy configurations.

orientations corresponding to the direction of the hydrogen bonds.

However, water ice contradicts the third law of thermodynamics, which predicts zero entropy at absolute zero temperature. An experiment by Giauque and colleagues in the 1930s discovered that hexagonal ice possesses a residual entropy of 0.82±0.05 Cal/deg·mol, deviating significantly from zero [25]. This residual entropy is due to proton disorder in hexagonal ice [11].

Each water molecule consists of two hydrogen atoms (H^+) and one oxygen atom (O_2^-), and their arrangement within the ice structure leads to an interesting situation. In hexagonal ice, the oxygen ions (O_2^-) arrange themselves in a tetrahedral structure. A tetrahedron is a geometric shape with four triangular faces, as depicted in Figure 2.3. This arrangement allows each oxygen ion to have four neighboring oxygen ions surrounding it, forming a tetrahedron.

This structural aspect accommodates the angles between hydrogen (H) and oxygen

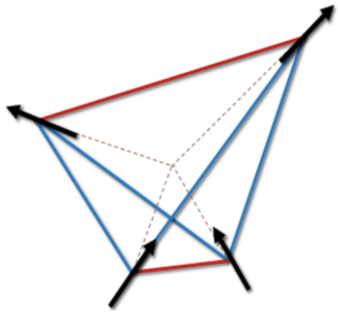


Figure 2.3: Interaction patterns in a tetrahedral arrangement of a spin ice material (adapted from ref. [1]). The prescribed ice rules entail two spins oriented inward and two outward. Satisfied interactions are denoted by blue lines, whereas unsatisfied interactions are indicated in red.

(O) atoms in a water molecule. In the hexagonal ice, these angles adjust to the typical H-O-H bond angle [26]. In the solid state, the strong chemical bonds maintain the water molecule's shape, leading to two equivalent proton positions along an O-O bond. The hexagonal wurtzite structure adheres to the Bernal-Fowler ice rules, allowing only one proton per O-O link on average. These rules dictate that each (O_2^-) ion must have two nearby and two distant H+ protons [28] [27] (see Figure 2.4).

Electrostatically, protons would prefer maximum separation, but the ice rules, preserving the integrity of the H_2O molecule, constrain proton positions, leading to low-energy frustration in proton-proton interactions.

2.3 Artificial Spin Ice

Wang et al. introduced artificial spin ice for the first time in 2006 [29]. Artificial spin ice is an exciting area of study in condensed matter physics that is making fast

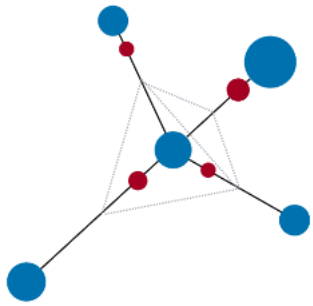


Figure 2.4: Ice rule illustration for freezing water Each oxygen ion (large blue circles) is surrounded by two nearby and two distant hydrogen ions (small red circles), resulting in a structure that complies with the ice principles. (Image's idea source [2])

progress. It involves creating artificial arrays of nano-scale magnetic islands that mimic the behavior of natural spin ice materials in a controlled and customizable manner. These magnetic islands are arranged in a lattice structure similar to the arrangement of atoms in a crystal, and their interactions are designed to look like the magnetic interactions in real spin ice systems.

In artificial spin ice systems, the macro-spins correspond to the individual nano-scale magnetic islands or connected lattice bars that make up the overall structure. These macro-spins interact with each other, and their orientations depend on how they are set up geometrically and how they interact with other dipolar spins in the system. The geometric arrangement of macro-spins frustrates the dipolar interactions between them, and ice rules describe the lowest-energy states. The islands are small enough to be single-domain, where the magnetic moment is aligned uniformly within each island. However, the islands are large enough that the transition between the

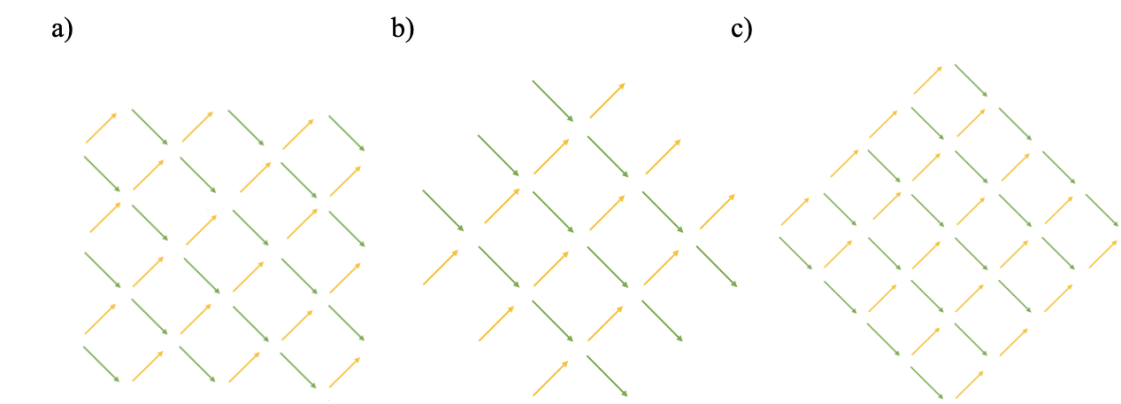


Figure 2.5: Three different edge geometries a) open edges; b) 4-island edges; c) closed edges.

two states can't just be based on the temperature.

Artificial spin-ice systems exhibit different lattice geometries and vertex configurations. So far, researchers have developed five main artificial spin ice geometries: square [29], honeycomb [30], kagome [31] [32], shakti [33], [34], and triangle [35] [36]. Among these, the focus is on the square ice lattice. In square ice, pairs of perpendicular spins interact more strongly than pairs of collinear spins. This structure is like water and spin ice but easy to work with and understand. The square ice lattice exhibits three different edge geometries, as shown in Figure 2.5, and the different edge geometries within this lattice can significantly impact the system's dynamics, especially in small arrays [37]. This work will focus on a finite-size open-edge square ice array.

In artificial spin ice, a vertex refers to a point where the spin configurations meet at the corners of each unit cell. In spin ice, there are different ways to set up the vertices. These configurations can be classified based on their energy levels in a square

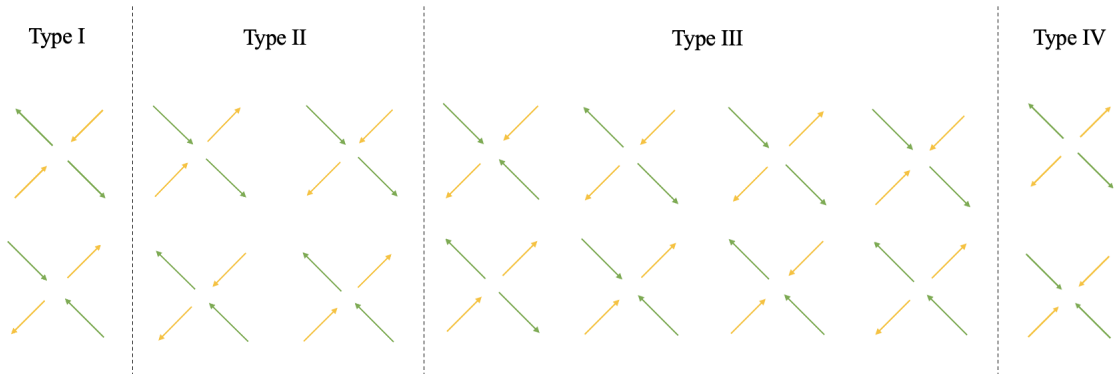


Figure 2.6: The sixteen different configurations of artificial spin ice vertices are divided into four distinct groups (designated as I, II, III, and IV). Vertices within each group possess energy levels that share degeneracy.

artificial spin ice structure with 16 vertices, as shown in Figure 2.6.

The type I configuration has the lowest energy among all configurations. The system's ground state consists of two different arrangements of type I vertices, representing a two-fold degeneracy. Type I vertices strictly follow the ice rules and have the lowest energy.

Type II vertices also follow the ice rules, but they have higher energy compared to type I vertices. They exhibit the largest net moment among all the vertex types.

Type III vertex configurations break the ice rule, with three spins pointing inward or outward and one spin pointing in the opposite direction. Similar to type II vertices, type III vertices also carry a net moment. The energies in these configurations are in the following order: $E(TypeIII) > E(TypeII) > E(TypeI)$.

Type IV vertices have the highest energy among all the configurations. In this case, all four spins are either pointing inward or outward, resulting in a zero net moment. In the vertex model, each arrow is a magnetic charge, with a positive

(+q) charge at the "head" and a negative (-q) charge at the "tail." Type III and IV vertices contain 2q and 4q pole charges, respectively, but type I and II vertices have no magnetic charge.

2.4 Model of interactions

Two models, namely the point dipole approximation and the charged dumbbell model, can be employed to analyze the interactions between islands in an artificial spin ice system. In this context, the point dipole approximation is the model chosen for analysis.

The magnetic field produced by each element in the context of magnetic interactions between elements, such as in artificial spin ice systems, can often be approximated as a point magnetic dipole located at its center. This approximation simplifies computations and allows for a straightforward examination of magnetic interactions.

Equation 2.1 can be used to describe the magnetic dipole-dipole interaction between two magnetic dipoles. Based on the magnetic dipole moments and the distance between the dipoles, this equation estimates the Hamiltonian of the system.

$$\mathrm{H}_{\mathrm{i,j}}^{\mathrm{dip}} = \mathrm{D} \sum_{\mathrm{i} \neq \mathrm{j}} \mathrm{s_i s_j} \left[\frac{\boldsymbol{\sigma}_{\mathrm{i}} . \boldsymbol{\sigma}_{\mathrm{j}}}{\mathrm{r}_{\mathrm{ij}}^3} - \frac{3(\boldsymbol{\sigma}_{\mathrm{i}} . \mathbf{r}_{\mathrm{ij}})(\boldsymbol{\sigma}_{\mathrm{j}} . \mathbf{r}_{\mathrm{ij}})}{\mathrm{r}_{\mathrm{ij}}^5} \right] \tag{2.1}$$

The constant $D = \frac{\mu_0 M^2}{4\pi a^3}$ gives an energy scale that is unique to the system and makes the system's energy unitless. M represents the magnetic moment strength, μ_0 represents the permeability constant of free space, and a is the lattice parameter. In the system, the magnetic moment of each island is shown by $\mathbf{s_i} = s_i \boldsymbol{\sigma_i}$, where $\boldsymbol{\sigma}$ is a dimensionless unit vector and shows in which direction the island is magnetized.

The vector $\mathbf{r}_{ij}$ is defined as the displacement vector from the i^{th} element to the j^{th} element. The dipolar interaction between these magnetic moments determines the magnetic field that the i^{th} element produces at the position of the j^{th} element.

In general, the strength and direction of the magnetic field between the elements depend on the relative orientation of their magnetic moments and the distance between them. These calculations approximate the magnetic field interactions within the system by treating each element as a point magnetic dipole.

2.5 Monte Carlo Method

We utilize the Monte Carlo method to determine the ground state with the lowest energy, where all vertices are Type 1. The Monte Carlo method is a powerful computational approach used to generate samples from complex probability distributions. It is a powerful method that is used a lot in many different areas, such as statistics, physics, machine learning, and Bayesian inference.

The main idea behind Monte Carlo simulation is to model the system's random temperature fluctuation from one state to the next during an experiment. Almost every Monte Carlo method uses Markov processes to generate the set of states that are used. A Markov process is a method that generates a new state v of a system given one state u. It does this in a random way, so it won't always come up with the same new state when given the starting state u. The probability of going from state u to state v is called the transition probability $\tau(u \to v)$. For a true Markov process, all transition probabilities should meet two conditions: (1) they shouldn't change over time, and (2) they should only depend on the properties of the current

states u and v, not on any other state the system has been in [38].

In a Monte Carlo simulation, a Markov process is used over and over to make a Markov chain of states. We start with a state u and use the process to make a new state v. We then use v to make another state λ, and so on. The Markov process was chosen so that if it is run long enough, starting from any state of the system, it will finally produce a series of states with probabilities given by the Boltzmann distribution. (The process of getting the Boltzmann distribution is called "coming to equilibrium.") Therefore, the main characteristic of Markov Chain Monte Carlo (MCMC) methods is their ability to converge to the Boltzmann distribution, given a sufficient number of iterations. This convergence is achieved by ensuring that the Markov chain satisfies certain properties, such as ergodicity. Ergodicity means that the Markov chain is capable of reaching any state in the target distribution with a non-zero probability. Here's a simplified explanation of the algorithm:

1. Start with an initial state in the Markov chain.

2. Generate a proposal for the next state based on a proposal distribution.

3. Compute the acceptance ratio, which compares the probability of accepting the proposed state with the current state.

4. Accept the proposed state with a probability based on the acceptance ratio. If accepted, transition to the proposed state; otherwise, stay in the current state.

5. Repeat steps 2-4 for a sufficient number of iterations to generate samples from the target distribution.

The Metropolis-Hastings algorithm is a specific type of Markov Chain Monte Carlo (MCMC) method used to generate samples from complex probability distributions. It

is particularly useful when direct sampling from the target distribution is challenging or infeasible. We use the Metropolis-Hastings algorithm in this work.

Nicholas Metropolis and the Rosenbluth and Teller families came up with the Metropolis algorithm in 1953 [39]. Hastings added to it in 1970 to cover more general cases [40]. The result is the Metropolis-Hastings algorithm, which uses a transition rate that depends on the difference in energy between the current state and the trial state to make a Markov chain. In this method, the transition rate is set to:

$$\tau(u \longrightarrow v) = \begin{cases} exp(-\Delta E/(\mathrm{k}_B T)) & if \Delta E > 0 \\ 1 & Otherwise \end{cases} \tag{2.2}$$

Where ΔE is the change in energy between the current state and the previous state, k_B is the Boltzmann constant, and T is the temperature. The algorithm works as follows:

1. Start with a random initial ASI configuration at a given temperature T and energy E.

2. Randomly select a spin within the ASI system and attempt to flip its state.

3. Calculate the change in energy of the system, ΔE, due to the trial spin flip.

4. If ΔE is less than or equal to zero ($\Delta E \leq 0$), accept the new state (flipped spin) as the current configuration.

5. If ΔE is greater than zero ($\Delta E \geq 0$), calculate the transition probability, $\tau = exp(-\Delta E/(\mathrm{k}_b T)$, where k_b is the Boltzmann constant.

6. Generate a random number i from the interval [0, 1].

7. If i is less than or equal to τ ($i \leq \tau$), accept the new configuration (flipped spin). Otherwise, retain the previous configuration.

8. Repeat steps 2–7 for a sufficient number of iterations to explore the ASI system's states and find the lowest energy state.

2.6 Vertical Elements

The concept of vertical components refers to the magnetic moments that point out of the plane of the ASI lattice. These vertical components can arise in certain types of ASI systems, such as those that have a non-uniform thickness in the out-of-plane direction or those that have a tilted orientation of the magnetic moments relative to the plane of the lattice.

The presence of vertical components can have a significant effect on the magnetic behavior of the ASI system and can lead to the emergence of new magnetic phases or transitions that are not present in purely two-dimensional ASI systems. Introducing defects in ASIs can influence ice rule violations, affect magnetic ordering, or induce emergent behavior in the system.

In the context of our study, the term "defect" is a relative concept. If our system was ideally composed and configured, anything deviating from that ideal state could be categorized as a defect. Here, we intentionally place some vertical elements in specific spots, thus introducing a three-dimensional aspect to the system. Subsequently, we evaluate the influence of these elements on the system's energy and magnetization. This method aims to create small-scale effects that mimic actual defects from non-uniform thicknesses on ASI systems.

By gaining a deeper understanding of the defects present in artificial spin ices (ASIs), researchers have the opportunity to optimize these materials according to

specific functionalities. Defect engineering involves intentionally introducing or manipulating defects in ASIs to tailor their properties and enhance their performance.

Through careful manipulation of defects, researchers can control and fine-tune the magnetic properties and interactions of ASIs. This provides the opportunity to optimize ASIs for applications such as data storage, magnetic logic devices, and magnetic devices [41] [42]. Moreover, defect engineering can also help explore new phenomena and fundamental aspects of ASIs, leading to advancements in our understanding of these complex systems. In this study, the introduction of vertical elements as defects aims to provide insights into defects in real-world systems, including aspects like density and spatial distribution.

Chapter 3

Restricted Boltzmann Machine

Geoffrey Hinton, Terry Sejnowski, and Yann LeCun introduced a Boltzmann machine in 1985 as a generative stochastic artificial neural network [22]. It is a stochastic, unsupervised learning algorithm used for dimensionality reduction [43], feature learning [44], collaborative filtering [45], and topic modeling [46].

One of the key properties of RBMs is their ability to learn distributed representations of the input data. The hidden units capture complex interactions between the visible units, allowing RBMs to extract meaningful features and discover underlying patterns in the data. During the learning or training process, RBM takes the features from the inputs and automatically decides which ones are important or relevant and how to put them together to make patterns [47].

3.0.1 Structure

The most common type of RBM has two layers of units with binary numbers. The first layer of the RBM is called the input or visible layer, and the second is called the

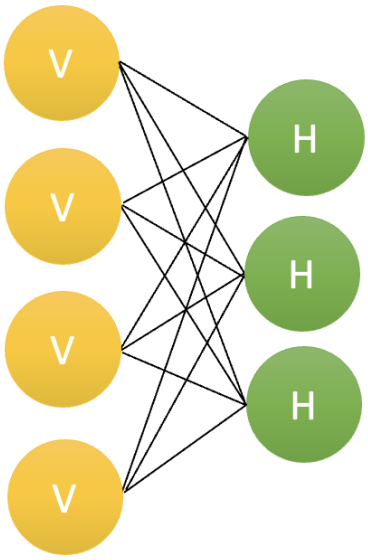


Figure 3.1: Representation of Restricted Boltzmann Machine Structure. Yellow circles are visible units in the first layer. Green circles are probabilistic hidden units in the second layer.

hidden layer. The visible layer shows the data that was given, and the hidden layer shows the features that were learned from the data. Each circle is a unit or a node. Nodes on visible and hidden layers are linked to each other, but no two nodes on the same layer are connected. (See Figure 3.1) In a Restricted Boltzmann Machine, the restriction is that there is no contact between the nodes within each layer. Instead, each node deals with the information that comes in on its own and uses probabilities to decide whether or not to pass that information on.

The neurons in the hidden layer receive input data from neurons in the visible layer. The weights are multiplied by the inputs, and the bias is then added. This

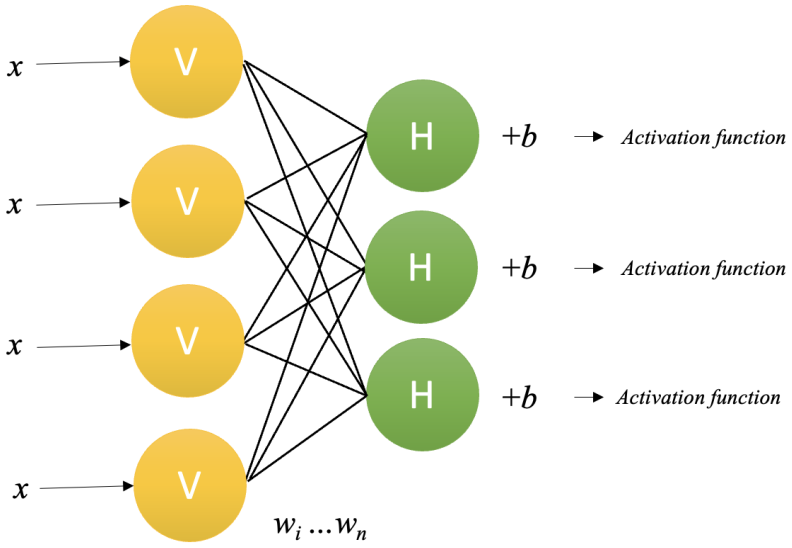


Figure 3.2: The weights are multiplied by the inputs (x), and then the bias is added. An activation function is then used on the result.

bias value serves as an additional factor in the computation, influencing the neuron's decision-making process. The resulting product goes through the activation function. The output of the activation function, which is subsequently applied to the result, determines whether or not the hidden state is activated. Continuing with the process, multiple inputs are combined and contribute to the activation of a single hidden node. Each input has its own associated weight and bias. This process is repeated for all hidden units, so as shown in figure 3.2, each hidden unit in the hidden layer can get data from every visible unit in the visible layer. The weights connecting the layers are shown in an array structure, where the rows correspond to the input nodes and the columns correspond to the output nodes.

Following the activation of the hidden layer neuron, its output value serves as a

new input. This new input is multiplied by the same weights as before and combined with the bias of the visible layer. This process is known as either reconstruction or backward pass. The original input and the newly generated input are then compared to assess whether they match or not. By comparing these inputs, the Restricted Boltzmann Machine can evaluate the quality of the reconstruction and adjust its parameters accordingly during the training process. This comparison helps the model learn to recreate the original input as accurately as possible [48].

Each weight element ($w_{i,j}$) of the matrix is linked to the relationship between the visible unit v_i and the hidden unit h_j. Also, biases a_i for v_i and b_j for h_j are used [49]. Given the weights and biases, the energy of a configuration $E(v, h)$ is described as:

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j \tag{3.1}$$

Or, in matrix notation,

$$E(v,h) = -a^T v - b^T h - v^T W h \tag{3.2}$$

The RBM model is an energy-based model where the joint probability distribution is defined by its energy function as follows:

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{3.3}$$

In this context, "Z" represents a partition function that is calculated as the sum of exponential terms ($-E(v, h)$) across all possible configurations. The probability of a visible vector can be calculated by summing the probabilities of $P(v, h)$ across all possible configurations of the hidden layer.

$$P(h) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{3.4}$$

Similarly, the probability of a hidden layer configuration can be calculated by summing the probabilities of $P(v, h)$ over all possible visible vector configurations [48].

$$P(v) = \frac{1}{Z} \sum_{v} e^{-E(v,h)} \tag{3.5}$$

The visible and hidden layer's units are totally independent [50]. Thus, if there are m visible units and n hidden units, the conditional chance of a visible unit configuration v given a hidden unit configuration h is:

$$P(v|h) = \prod_{i=1}^{m} P(v_i, h) \tag{3.6}$$

And, the conditional probability of h given v is:

$$P(h|v) = \prod_{j=1}^{n} P(h_j, v) \tag{3.7}$$

Using a probabilistic interpretation of the neuron activation function, we have:

$$P(h_{j=1}|v) = \sigma(b_j + \sum_{i=1}^{m} w_{i,j} v_i) \tag{3.8}$$

$$P(v_{j=1}|h) = \sigma(a_i + \sum_{j=1}^{n} w_{i,j} h_j) \tag{3.9}$$

Where σ denotes the logistic sigmoid as an activation function.

The hidden units in Restricted Boltzmann Machines are Bernoulli, although the visible units might either be multinomial or Bernoulli.

When modeling a single, binary event, like yes/no or success/failure, the Bernoulli distribution can be used as a discrete probability distribution. When there are just two potential outcomes to a random occurrence, this is the method of choice. The distribution is defined by a single parameter, commonly represented as "p," which reflects the likelihood of a "success" event.

The multinomial distribution is a further generalization of the binomial distribution, wherein there are more than two possible outcomes. In a scenario with multiple possible outcomes, it simulates the chances of seeing each one. There is a separate probability parameter for each possible outcome group.

In conclusion, the multinomial distribution models multiple categories or outcomes with separate probability parameters for each category, whereas the Bernoulli distribution models a single binary outcome with a single probability parameter.

A vector of real values can easily be transformed into a probability distribution using the softmax function 3.11. It converts an input vector of arbitrary real numbers into an output vector of values between 0 and 1, where the output vector's total value is equal to 1. The sigmoid function 3.10 function is used for binary classification, where the output is a probability between 0 and 1, indicating how likely the input belongs to a certain class.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.10}$$

where $\sigma(x)$ is the output of the sigmoid function and x is the input value.

On the other hand, the softmax function is used when there are more than two classes. It outputs a probability distribution over multiple classes. Therefore, both

functions output probabilities, but the sigmoid does so for two classes (e.g., true or false), while softmax does it for multiple classes. Since we have multinomial visible units, the softmax function replaces the sigmoid function [46].

$$\text{Softmax}(Z)_i = \frac{e^{Z_i}}{\sum_{k=1}^{K} e^{Z_k}} \tag{3.11}$$

where $\text{Softmax}(Z)_i$ is the probability of the i^{th} class. Z is a vector of real numbers (the input). Z_i is the i^{th} element of vector Z and K is the total number of classes.

3.0.2 Training In RBM

RBM training is based on Gibbs Sampling and Contrastive Divergence, an iterative procedure that approximates the gradient of the log-likelihood function. This method makes RBM training computationally efficient and scalable.

Gibbs Sampling

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm used to simulate a complex probability distribution when direct sampling is not feasible. It is named after the physicist J. Willard Gibbs. In 1984, about 80 years after Gibbs' death, Stuart and Donald Geman described the algorithm [51]. The basic idea behind Gibbs sampling is to generate samples from a joint probability distribution by iterative sampling from the conditional distributions of each variable given the current values of the other variables. In other words, given a set of variables in a joint distribution, we can update the value of one variable at a time, conditioned on the current values of the other variables, until we have sampled values for all the vari-

ables. Here is a simple example to illustrate the Gibbs sampling algorithm. Suppose we have a joint distribution $P(x, y)$ that we want to sample from, where x and y are two variables. We can initialize the values of x and y to arbitrary values. Then, in each iteration of the algorithm, we alternate between updating the values of x and y based on the following steps:

Sample x from $P(x|y)$: set x to a new value drawn from the conditional distribution of x given the current value of y.

Sample y from $P(y|x)$: set y to a new value drawn from the conditional distribution of y given the current value of x. After repeating these steps many times, the resulting sequence of (x, y) pairs represents samples from the joint distribution $P(x, y)$. Gibbs sampling is widely used in Bayesian inference and machine learning applications, where it can be used to estimate the posterior distribution of a set of parameters given some observed data. It is a powerful and flexible algorithm that can be applied to a wide range of problems. However, it can be computationally intensive and may require a large number of iterations to converge to the true distribution [52].

Contrastive Divergence

Contrastive Divergence (CD) is an optimization algorithm commonly used in training Restricted Boltzmann Machines (RBMs) and other energy-based models, particularly for generative models like deep belief networks and deep Boltzmann machines. The CD was first introduced by Hinton in 2002 in his paper titled 'Training Products of Experts by Minimizing Contrastive Divergence.' [53]

CD aims to maximize the log-likelihood function. Named 'contrastive divergence,'

the algorithm minimizes the contrastive Divergence between the model distribution and the data distribution. It achieves this by running a series of Markov Chain Monte Carlo (MCMC) steps to generate a 'fantasy' distribution. It starts with a training sample and generates a 'fantasy' sample through a few steps of Gibbs sampling from the visible units of the training sample. The algorithm updates the RBM parameters based on the difference between the statistics of the training sample and the fantasy sample.

Here's an overview of the Contrastive Divergence algorithm:

1. Initialize the RBM with random weights.

2. Sample a visible input from the training data.

3. Perform a forward pass through the RBM to compute the probabilities of the hidden layer activations.

4. Sample the hidden layer activations based on the computed probabilities.

5. Perform a backward pass to reconstruct the visible layer activations based on the sampled hidden layer activations.

6. Again, perform a forward pass using the reconstructed visible layer activations to compute the probabilities of the hidden layer activations.

7. Sample the hidden layer activations based on the reconstructed visible layer activations.

8. Update the weights of the RBM using the outer product of the original visible layer and the sampled hidden layer activations, then subtract the product of the reconstructed visible layer and the sampled hidden layer activations.

9. Repeat steps 2-8 for a fixed number of iterations or until convergence.

The key idea behind Contrastive Divergence is to approximate the gradient of the log-likelihood function by taking the difference between the expected values under the data distribution and the model distribution. The CD algorithm provides a computationally efficient approximation of the gradient and is relatively easy to implement.

3.0.3 The Algorithm

In order to train an RBM, we have to maximize the product of probabilities assigned to the training set v (a matrix, where each row of it is treated as a visible vector v) [50]:

$$Max \prod_{v \in V} P(v) \tag{3.12}$$

Or, equivalently, maximize the expected log probability of v. The expected log probability is the average of the logarithm of the probabilities assigned to each training example. Taking the logarithm helps in numerical stability and simplifies the calculations:

$$MaxE \left[\sum_{v \in V} logP(v) \right] \tag{3.13}$$

We can use stochastic gradient descent to find the optimal weight and consequently minimize the objective function. But when we derive, it gives us two terms, called positive and negative gradients. The positive phase increases the probability of training data. The negative phase decreases the probability of samples generated by the model. The negative phase is hard to compute. So, training an RBM is per-

formed by the "Contrastive Divergence Learning" algorithm. Contrastive Divergence is actually a matrix of values that is computed and used to adjust the values of the W matrix. Changing W incrementally leads to the training of W values. Then, on each step (epoch), W is updated to a new value, W' through the equation below:

$$W' = W + \alpha CD \tag{3.14}$$

where α is learning rate, W is weight matrix. This is how the contrastive divergence algorithm works [47]:

1. Start with a training sample v and calculate the probabilities of the hidden units. Sample a hidden activation vector h from this probability distribution.

2. Compute the outer product of v and h, which represents the positive gradient.

3. From h, generate a reconstruction v' of the visible units. Then, resample the hidden activations h' based on this reconstruction (Gibbs sampling step).

4. Calculate the outer product of v' and h', representing the negative gradient.

5. Update the weight matrix W using the positive gradient minus the negative gradient, multiplied by a learning rate.

6. Update the biases a and b in a similar manner.

The pseudocode for the algorithm is shown in Figure 3.3

3.0.4 Examining Algorithm Effectiveness

In this study, we chose not to use pre-existing libraries but instead developed our own program to gain a comprehensive understanding of Restricted Boltzmann Machines (RBMs). To test the effectiveness of our program, we utilized simple known

Pseudocode for Restricted Boltzmann Machine (RBM)

Initialize the weight matrix W, bias vectors a and b

Set the states of visible unit V_i as the training vector

While i < number of epochs
    For j = 1, 2 .., m (All hidden units)
        Compute $P(h_{1j} = 1 | V_1)$
        Gibbs Sampling $h_{1j} \in \{0,1\}$ from $P(h_{1j} | V_1)$
    End For

    For i = 1, 2 .., n (All visible units)
        Compute $P(V_{2i} = 1 | h_1)$
        Gibbs Sampling $V_{2i} \in \{0,1)$ from $P(V_{2i} | h_1)$
    End For

    For j = 1, 2 ..., m (All hidden units)
        Compute $P(h_{2j} = 1 | V_2)$
    End For

    // Update rule:
    For j = 1, 2, ..., m (All hidden units)
        $W := W + \in (P(h_{1j} = 1 | V_1) \ V_1^T - P(h_{2j} = 1 | V_2) \ V_2^T)$
    End For

    $a := a + \in (V_1 - V_2)$
    $b := b + \in (P(h_1 = 1 | V_1) - P(h_2 = 1 | V_2))$

End While

Figure 3.3: Pseudocode for the RBM

data, specifically binary representations of the letters “T” and “H .”Although the dataset was limited in size, we evaluated the performance of the RBM by reconstructing the data and measuring the error between the original and reconstructed versions. As depicted in Figure 3.4, after 200 epochs of training, we achieved an error that approached zero, indicating a successful outcome.

Subsequently, we proceeded to assess the RBM’s performance by introducing a new set of unseen data. This data comprised the original images of the letters “T” and “H” with additional noise intentionally added, rendering them challenging to discern from the original patterns. The outcomes of this experiment, as illustrated in

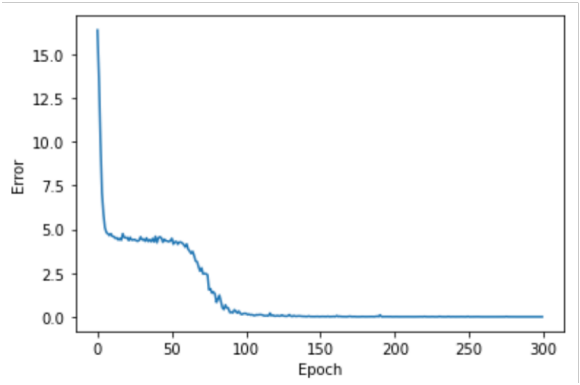


Figure 3.4: Error vs. Epoch for original and reconstructed data

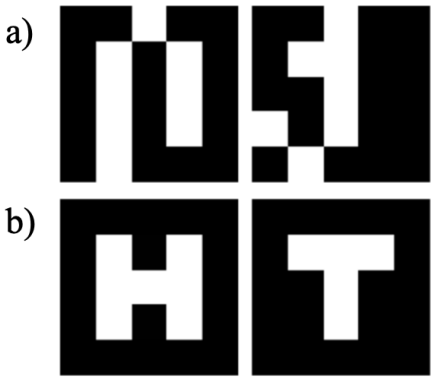


Figure 3.5: Sample dataset. a) Perturbed image of letter T and H b) Reconstructed image of letter T and H

Figure 3.5, provided us with reasonable confidence in the reliability of our code and prompted us to proceed with testing real data using the RBM.

3.0.5 Analyzing RBM Results for the Ising Model

As a subsequent phase of our research, we aimed to validate the effectiveness of our RBM model for analyzing magnetic data. To ensure the reliability and accuracy of our RBM implementation, we sought to reproduce the results of a previously published paper by David Yevick and Roger Melko named "The Accuracy of Restricted Boltzmann Machine Models of Ising Systems." [54] By replicating the findings of this paper's results, we could establish a benchmark and confirm that our RBM successfully captures and models magnetic data patterns. This rigorous validation process demonstrates the robustness of our approach and builds confidence in the reliability of our RBM for analyzing new magnetic data. The paper explores the effectiveness and accuracy of Restricted Boltzmann Machine (RBM) models in capturing the behavior of Ising systems. Ising models are mathematical representations used to study phase transitions and critical phenomena in materials science and statistical physics. The authors investigate how well RBMs, a type of generative model, can learn and reproduce the patterns and correlations present in Ising systems. They compare the RBM-generated samples with the true configurations of Ising systems to evaluate the accuracy of the RBM model. Through a lot of simulations and analysis, they quantify the discrepancy between the RBM-generated samples and the true configurations, providing insights into the limitations and capabilities of RBMs in capturing complex physical phenomena. The Ising system is a mathematical model that describes a collection of interacting spins, typically represented as discrete variables, such as up and down or +1 and -1 [55]. The Ising model is widely used to study systems with magnetic properties, phase transitions, and other phenomena. In an Ising system,

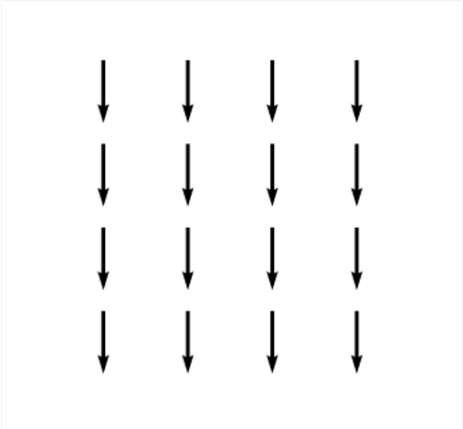


Figure 3.6: Ising system in Ferromagnetic state

the spins are arranged on a lattice, such as a square grid. Figure 3.6 shows the ferromagnetic state of an Ising system. The interaction between neighboring spins in the lattice determines the behavior of the system. The Ising model assumes that each spin interacts only with its nearest neighbors, and the strength of these interactions is typically represented by a coupling constant J_{ij}.

The spins in an Ising system can be influenced by various factors, including temperature, external magnetic fields, and the interactions between spins. In the Ising model, the energy function, denoted by E, is a measure of the system's energy at a given configuration. The energy function in the Ising model in the absence of an external magnetic field can be expressed as follows:

$$E = -J_{ij} \sum_{i,j} \mathbf{s_i s_j} \tag{3.15}$$

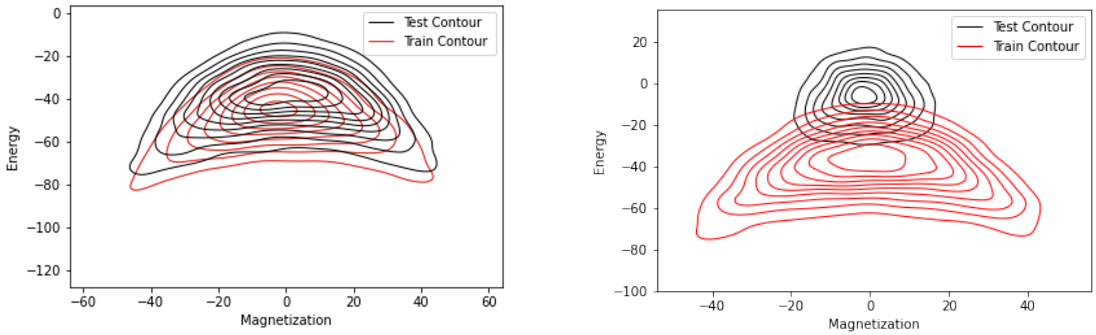
Where J_{ij} represents the strength of the interaction between neighboring spins, $\mathbf{s_i} = s_i \boldsymbol{\sigma_i}$ and $\mathbf{s_j} = s_j \boldsymbol{\sigma_j}$ are the spin values at lattice sites i and j, respectively, and the summation is taken over all pairs of neighboring lattice sites. The magnetiza-

tion, denoted by M, represents the net magnetic moment of the system. It can be calculated as the sum of the spin values over all lattice sites:

$$M = \sum_{i=1}^{N} \mathbf{s_i} \tag{3.16}$$

Where the summation is taken over all lattice sites. These mathematical formulations capture the energy interactions and magnetization behavior in the Ising model, providing a quantitative description of the system's properties. We initiated our Ising system with an 8 by 8 lattice and employed a random initial configuration. Following the protocol outlined in the paper [54], we conducted 60,000 Monte Carlo steps at a temperature of 3.53. This temperature was chosen because it's above the critical point, turning the material paramagnetic. This means more varied states for the RBM to learn from, as it leads to a wider distribution of data. At each step, we computed the system's energy and magnetization. Subsequently, we generated contour plots illustrating the evolution of energy and magnetization throughout the 60,000 Monte Carlo steps, as depicted in Figure 3.7a and Figure 3.7b with red lines.

Simultaneously, we stored the state configurations to construct our dataset for the RBM. Then, we inputted the data into our RBM model and obtained the corresponding reconstructed data. From the reconstructed data, we calculated the energy and magnetization using the same procedure as above. We generated contour plots and compared them with the original data, as depicted in Figure 3.7a and Figure 3.7b. We generated contour plots for two cases: one with a smaller learning rate, which gave us better results for specific heat Figure 3.7b, and one with a larger learning rate, which gave us better results for the joint probability of energy and magnetization of



(a) 64 hidden nodes and 10^{-3}

(b) 64 hidden nodes and 10^{-4}

Figure 3.7: The joint energy magnetization distribution of the 60,000 steps Monte Carlo samples for an 8×8 two-dimensional Ising model at a temperature T = 3.53. Black lines: RBM results, Red lines: Original results.

the system Figure 3.7a. Notably, the contour plots in this case were represented by red lines. Encouragingly, our results exhibited excellent agreement with the findings reported in Yevick and Melko's paper [54].

This study addresses the importance of model accuracy in RBMs and highlights the challenges and potential improvements for RBM-based modeling of Ising systems. The findings provide valuable insights into RBM performance and enhance our understanding of the applicability of RBMs in modeling and analyzing Ising systems. By referencing and replicating the results presented in this paper, our objective is to validate the efficacy of our RBM model in analyzing magnetic data obtained from the Artificial Spin Ice (ASI) system. Through this validation process, we aim to demonstrate that our RBM model is capable of effectively capturing and modeling patterns that resemble those observed in magnetic systems.

Chapter 4

Utilizing RBM for Artificial Spin Ice Data

This study utilizes a restricted Boltzmann machine on a square spin ice model to assess the effectiveness of this machine learning approach on our magnetic system. Our goal is to utilize the model to aid in identifying and exploring specific defects in our spin ice materials. This study's findings can potentially benefit experimentalists and the academic community in their respective research endeavours. This research involves a series of steps, which are summarized here. The primary objective is to evaluate our RBM model's accuracy by identifying certain system characteristics. These characteristics allow us to compare real data with machine-generated data. We conduct a Monte Carlo simulation in the initial step to collect information about the system's states. This data is then utilized to determine specific characteristics, such as energy and magnetization. Visualizing these results provides us with a clearer understanding. Moving on to the third step, we employ the Monte Carlo simulation

results to train our RBM. This enables the machine to learn the relationships and patterns among the system's states. Subsequently, we ask the machine to reconstruct the data based on what it has learned. Finally, in the last step, we use the data generated by the RBM to extract characteristics such as energy and magnetization. We can compare these findings with the original data and assess the machine's accuracy by visualizing them. More details will be provided about these steps.

4.1 Initialize the system

For this study, a 2-dimensional lattice of open-edge square spin ice was selected, as shown in Figure 2.5 b). The lattice size is 6 by 6 with equal spacing. Initially, random configurations of vertices were assigned to the lattice, and the position and orientation of each spin were recorded in separate arrays. Additionally, the distances between the centres of each spin were calculated, resulting in a matrix illustrating the distances between all spins in the lattice. By utilizing these parameters along with Equation 2.1 from Chapter 1, the energy of the system in its initial state was computed as below:

$$\mathrm{E}_{\mathrm{i,j}}^{\mathrm{dip}} = \mathrm{D} \sum_{\mathrm{i} \neq \mathrm{j}} \mathrm{s_i s_j} \left[\frac{\boldsymbol{\sigma}_\mathrm{i} . \boldsymbol{\sigma}_\mathrm{j}}{\mathrm{r}_{\mathrm{ij}}^3} - \frac{3(\boldsymbol{\sigma}_\mathrm{i} . \mathbf{r}_{\mathrm{ij}})(\boldsymbol{\sigma}_\mathrm{j} . \mathbf{r}_{\mathrm{ij}})}{\mathrm{r}_{\mathrm{ij}}^5} \right] \tag{4.1}$$

After computing the system's energy using equation 3.16, the next step in the Monte Carlo algorithm is randomly flipping a spin, which changes the system's configuration. Once the new system's energy is calculated and the energy difference is determined, we can evaluate the transmission rate using Equation 2.2 from Chapter 1. Based on this transmission rate, we decide on whether to accept or reject the proposed

change in the system. We run the Monte Carlo simulation described in Chapter 1 at a constant temperature for as many steps as our system requires to reach a steady state in which the system's energy no longer fluctuates considerably. The appropriate number of steps required to achieve this steady state varies depending on the system's features. In order to determine the optimal number of Monte Carlo (MC) steps, we conducted seven sample runs with varying MC step counts ranging from 3,000 to 30,000. By plotting the average trend along with the standard deviation, we aimed to identify the most suitable number of steps. The trends of these seven samples are depicted in Figure 4.1, where the black line represents the average, and the gray area corresponds to the lower bound (average - standard deviation) and upper bound (average + standard deviation) of the data. Our setup includes 36 spins functioning at a temperature of 3.5. Following analysis, we opted for 15,000 iterations to allow the system to attain equilibrium. Once equilibrium was reached, we conducted a simulation for an additional 60,000 iterations to gather samples from the stabilized system.

The next step is setting up the Restricted Boltzmann Machine (RBM) parameters. There are multiple ways to do this. Here, we set the biases to zero and use a standard normal distribution to pick small numbers for the weights to start with randomly. We chose to have 36 visible units because that's how many data points we want the RBM to learn about and model the links between. Two essential hyper-parameters significantly influence the learning process: the number of hidden nodes and the learning rate. Finding the optimal values for these parameters poses a challenge. There is no rule to determine how many hidden units are needed. While some suggestions by

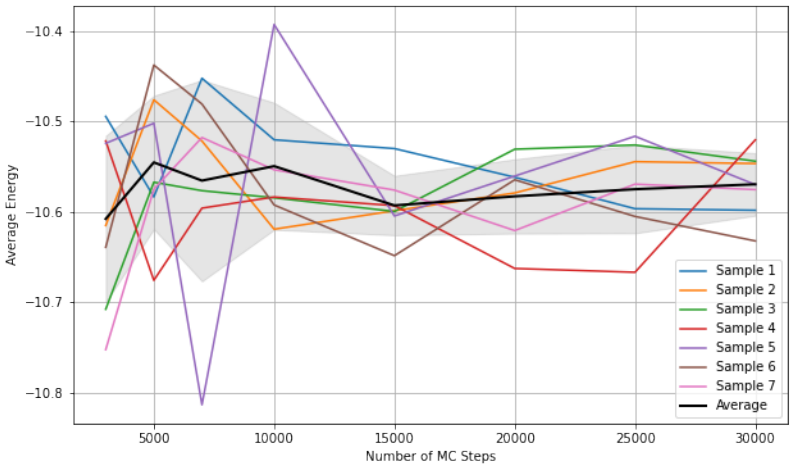


Figure 4.1: Average Energy of ASI System for Various Monte Carlo Steps

Hinton exist [48], the best approach is to experiment with different numbers of hidden units to find an optimal number. To begin, we experiment using various numbers of hidden nodes and identify the minimum error to determine the optimal number of nodes for our model. Figure 4.2 demonstrates that 36 hidden nodes are sufficient for training our model.

We expect a comparable analogy for ASI to the one presented in Yevick and Melko's paper [54]. Depending on the particular problem, choosing different parameters for the learning rate and hidden nodes might be necessary. We also experimented with 6 hidden nodes to compare the results as part of our exploration.

The next thing that needs to be set is the learning rate. During the training process, the machine's biases and weights should be updated so that it can learn. The stochastic gradient descent method is used to train most neural networks how to work. Stochastic gradient descent is an optimization method that finds the error gradient

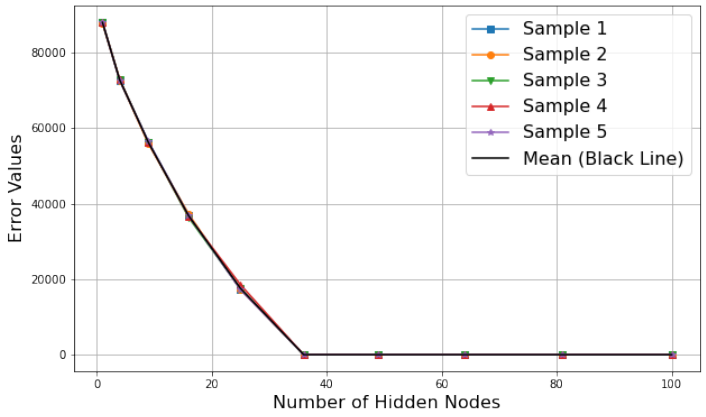


Figure 4.2: Minimum Training Error of RBM vs. Number of Hidden Nodes

for the model's current state by looking at examples from the training dataset. Then, it uses back-propagation to update the model's parameters. When using a gradient descent function, we navigate the function to find the minimum error, and the learning rate determines the size of each step we take toward this minimum.

If the learning rate is too high, the model can quickly settle on an answer that could be better, while if it's too low, the process can get stuck. Finding the correct learning rate can significantly affect how well the model works, so it's essential to spend time and effort picking a good number to get the best results. We tried different values for 5 different samples and looked at the related errors to find the most suitable learning rate for our problem. By studying the graph in Figure 4.3, we can determine the learning rate that leads to the least error. The results of these tests are shown in Figure 4.3, helping us choose 0.001 and 0.0001 for our experiments.

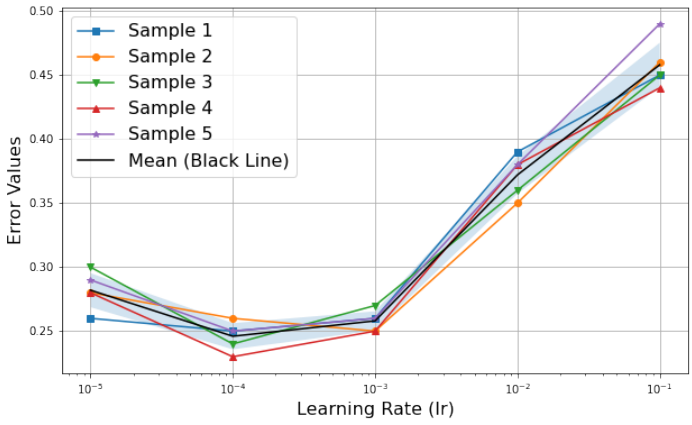


Figure 4.3: Minimum Training Error of RBM vs. Learning Rate

4.2 Characterizing the data and representing the results

We utilize the energy computed using Equation 4.1 and additionally determine the magnetization and stray field at each step. As a result, we compile lists of energy, magnetization in the x-direction, magnetization in the y-direction, and stray field values. Hence, the collection of N spin arrays can be represented as $S\{E, Mx, My, Hz\} = \{\mathbf{s}_i\}$. Here, the index i corresponds to the specific position of a spin within the array lattice. This notation indicates that the set S includes the spins within the array, where each spin possesses a distinct combination of energy and magnetization in the y and x directions (M_y and M_x, respectively), and a perpendicular stray field component (H_z). Magnetization components and stray fields are given by:

$$\mathrm{M}_x = \sum_{i=1}^{N} \mathbf{s}_i \cdot \hat{x} \tag{4.2}$$

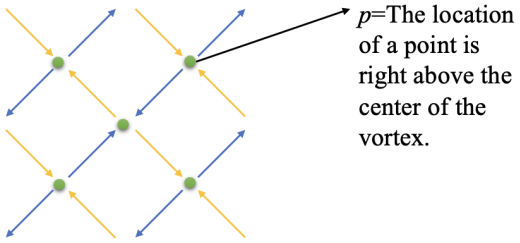


Figure 4.4: Location of the observation point for H_z calculation

$$\mathrm{M}_y = \sum_{i=1}^{N} \mathbf{s}_i \cdot \hat{y} \tag{4.3}$$

$$\mathrm{H}_z = D \sum_{p=1}^{k} \sum_{i=1}^{N} \left[\frac{3\mathbf{r}_{ip}(\mathbf{s}_i \cdot \mathbf{r}_{ip})}{r_{ip}^5} - \frac{\mathbf{s}_i}{r_{ip}^3} \right] \tag{4.4}$$

The M_x and M_y represent the magnetization of the system in the x and y directions, respectively. However, this system also has magnetization in the z-direction. To calculate M_x and M_y, we can straightforwardly sum up the x and y components. But for the z direction, the process is more complicated. To calculate H_z (stray field), we look at all the interior vortex points in the lattice. For each vortex point, we move 0.5 units above the lattice and consider it as our observation point (shown in Figure 4.4). Then, we calculate the dipolar field produced by each spin at the center of each vertex and add all these contributions from the vertices. The result gives us the H_z, as mentioned in Equation 4.4.

In Equation 4.4, p is the observation point, and the r_{ip} is the position vector from point p to the center of spin i^{th}, as shown in Figure 4.5.

After computing all these equations for the simulation and reconstructed data, we

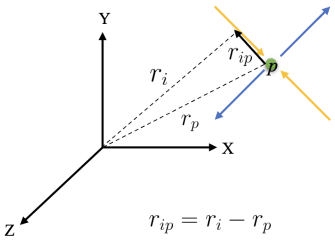


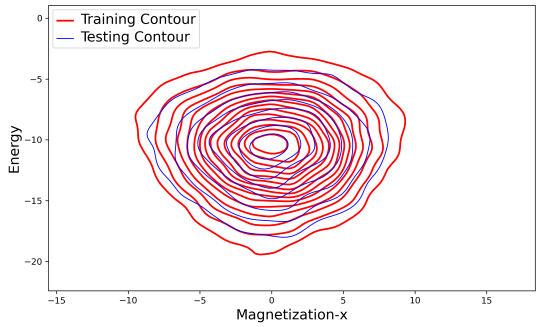
Figure 4.5: position vector from point p to the center of spin i^{th}

compare the results and determine the accuracy of RBM for different cases.

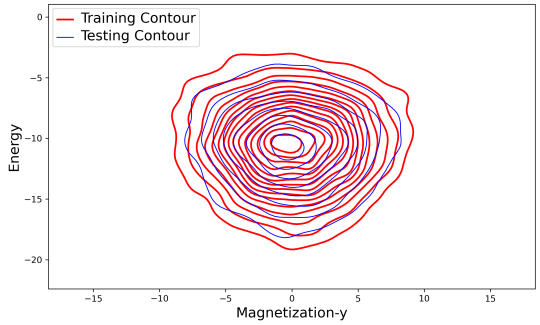
4.3 Evaluation

The widely recognized method for evaluating the performance of an RBM is to examine how well it does on data it has never seen before. To achieve this, the data is divided into two smaller sets. The dataset is shuffled, and 70 % of the data is used for training the RBM, while the remaining 30% is used for testing. The RBM is then trained on the 70%, and its performance in reconstructing the remaining 30% (which it has never seen before) is evaluated. The outcome of this evaluation is shown in Figure 4.6. By comparing the results from the testing dataset, we can see how well the RBM has learned the underlying features and how effectively it can generalize to new, unseen data.

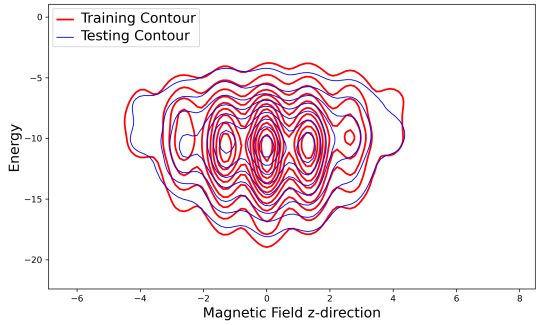
The earlier assessment involved testing the system using samples from the equilibrated distribution obtained through Monte Carlo simulation. Our objective is to employ the RBM to generate samples for us. We input the RBM with 15,000 samples from the MC simulation to achieve this and train it accordingly. After training, we provide the trained RBM with entirely random data derived from a normal random



(a) Energy vs. Magnetization in x-direction.



(b) Energy vs. Magnetization in y-direction.



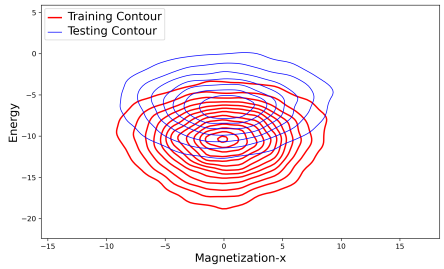
(c) Energy vs.Magnetic Filed in z-direction.

Figure 4.6: The joint energy magnetization distribution of the 15,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5. Training and testing set using RBM with 36 hidden nodes and 0.001 learning rate.

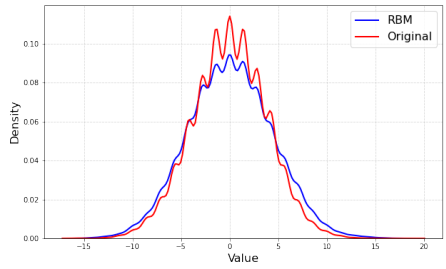
distribution, and we ask it to reconstruct these data points based on its learned knowledge. For this experiment, we utilize different RBMs with various hyper-parameters and then create plots of the energy magnetization joint distribution and heat capacities for their outputs. Finally, we compare these plots with those from the original data set.

In Figure 4.7, we utilized 36 hidden nodes with a learning rate of 0.001, while in Figure 4.8, we maintained 36 hidden nodes with a learning rate of 0.0001.To aid in the comparison of these plots, we created separate histograms for the energy distribution and magnetization. The energy distribution histograms displayed minimal variation, providing little informative differentiation, so we did not plot them. However, the magnetization histograms, situated to the right of each contour plot, revealed crucial details. They highlighted certain significant features that were occasionally not matched by the RBM's results. Notably, in the experiment with a learning rate of 0.0001, as depicted in Figure 4.8, the alignment between the reconstructed magnetization values and density appeared to be more pronounced compared to the experiment with a learning rate of 0.001, shown in Figure 4.7. Nevertheless, it's important to note that neither experiment yielded satisfactory results, particularly in terms of magnetization in the y-direction.

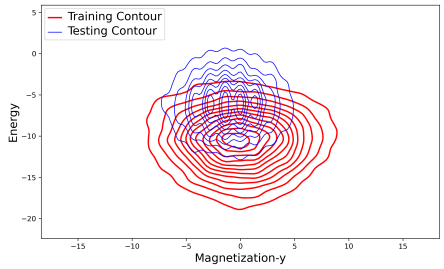
We also tested using 6 hidden nodes to check if RBM could help reduce dimensionality and potentially yield improved results. The results are shown in Supporting Data A for a learning rate of 0.001 (See Figure A.1)and 0.0001(See Figure A.2). However, none of those RBMs were able to generate accurate reconstructions of energy and magnetization contours.



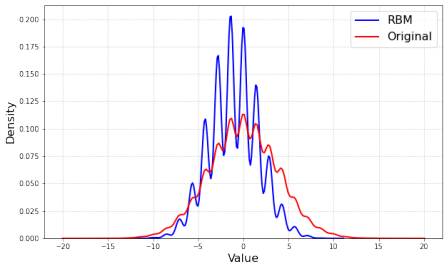
(a) Energy vs. Magnetization in x-direction.



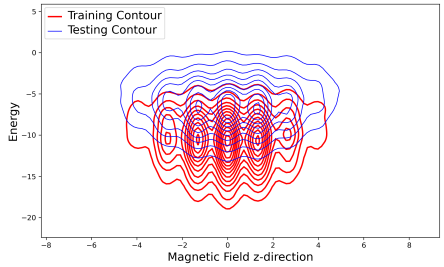
(b) Magnetization vs. Density in x-direction.



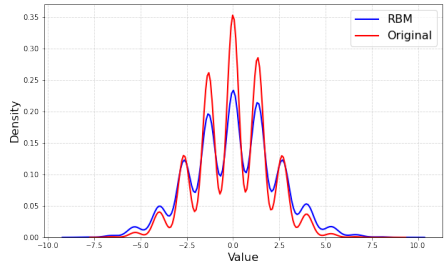
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.



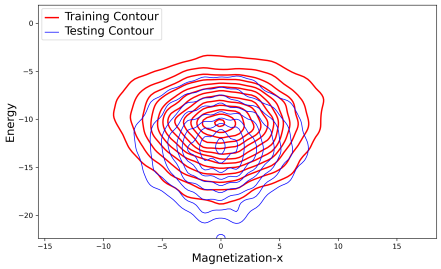
(e) Energy vs. Magnetic Field in z-direction.



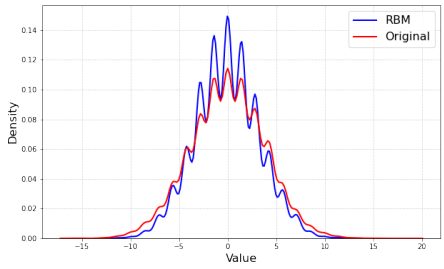
(f) Magnetic Field vs. Density in z-direction.

Figure 4.7: The joint energy magnetization distribution of the 15,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5 and their reconstruction using RBM with 36 hidden nodes and 0.001 learning rate

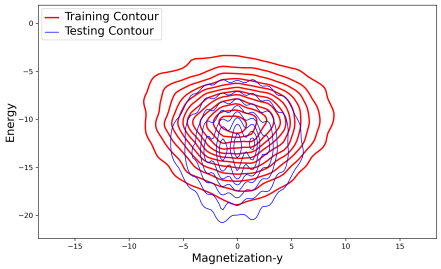
Another parameter that could be examined was the heat capacity. The RBM featuring 36 nodes and a learning rate of 0.0001 once again demonstrated the most effective performance in heat capacity. The calculated amount from the Monte Carlo



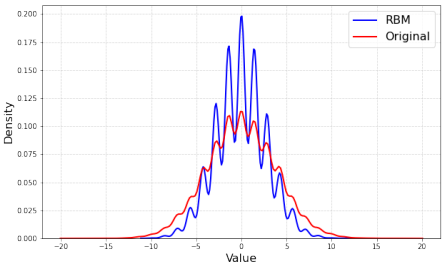
(a) Energy vs. Magnetization in x-direction.



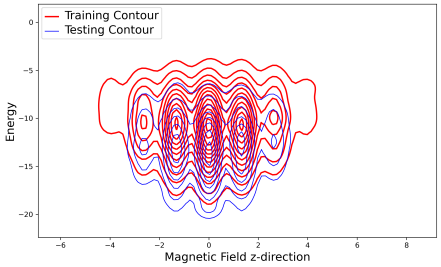
(b) Magnetization vs. Density in x-direction.



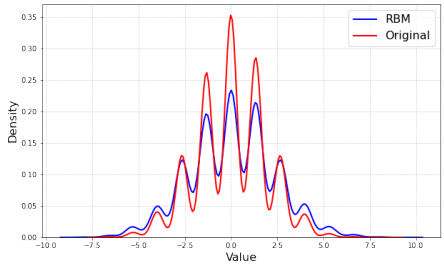
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.



(e) Energy vs. Magnetic Field in z-direction.



(f) Magnetic Field vs. Density in z-direction.

Figure 4.8: The joint energy magnetization distribution of the 15,000 steps Monte Carlo samples for a 6×6 two-dimensional ASI at a temperature $T = 3.5$ and their reconstruction using RBM with 36 hidden nodes and 0.0001 learning rate

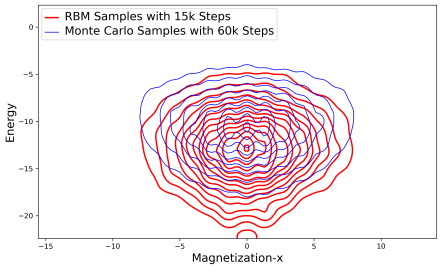
simulation for heat capacity per spin was 0.271 ± 0.001. For reconstructing energy distribution using RBM with 36 hidden nodes and a learning rate of 0.0001, the value was 0.18 ± 0.01. However, it's important to note that in a similar experiment

conducted with the Ising system, the result was remarkably close. This discrepancy can be attributed to the fact that the artificial spin ice system is significantly more complex than the Ising system.

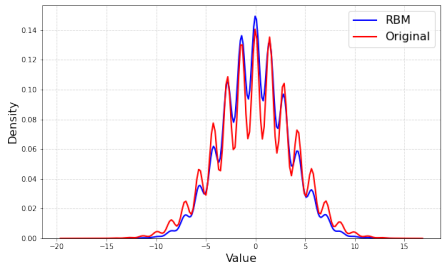
Still, we were curious about differences in M_x and M_y. The reconstructed contours and wavy lines showed certain details about sharper features and more peaked distribution around its mean that were absent in our first training set of 15,000 Monte Carlo simulation samples. These samples included non-equilibrated and equilibrated cases. To address this, we tried a new approach.

In our next experiment, we used a bigger training set of 60,000 equilibrated samples. Surprisingly, this time the generated contours for M_x and M_y showed those detail features better in the center. The RBM, trained initially on the smaller set, somehow managed to grasp these specific features. These features, hidden in the first 15,000 samples, became clearer with the more detailed 60,000 equilibrated samples, and the RBM trained on 15,000 samples managed to capture those hidden specific features. The Figure 4.9 shows the comparison between the RBM reconstructed results from 15,000 samples training and the original data set from 60,000 steps of Monte Carlo simulation.

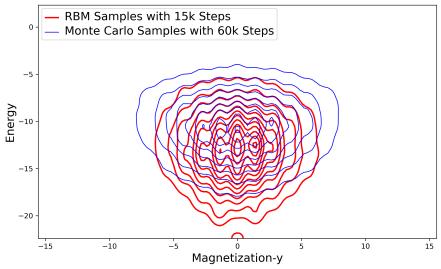
The RBM is yielding favourable results, demonstrating an advantage over Monte Carlo simulations by detecting certain features that often go unnoticed when working with smaller Monte Carlo datasets. To confirm this, we trained the RBM with 60,000 samples to see if the pattern holds the result of the training shown in supporting Data A Figure A.3. Once more, we saw differences in M_y. So, we took it a step further with a 600,000-step Monte Carlo simulation. We drew the contours in Figure 4.10 to



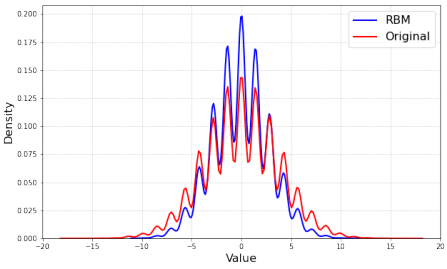
(a) Energy vs. Magnetization in x-direction.



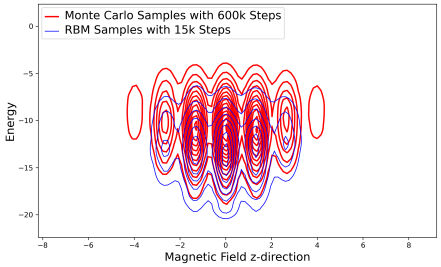
(b) Magnetization vs. Density in x-direction.



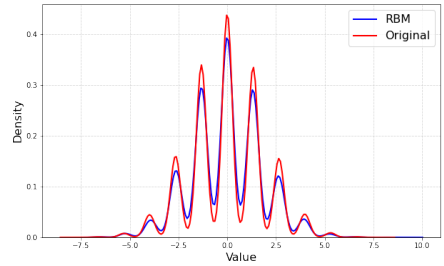
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.



(e) Energy vs. Magnetic Field in z-direction.

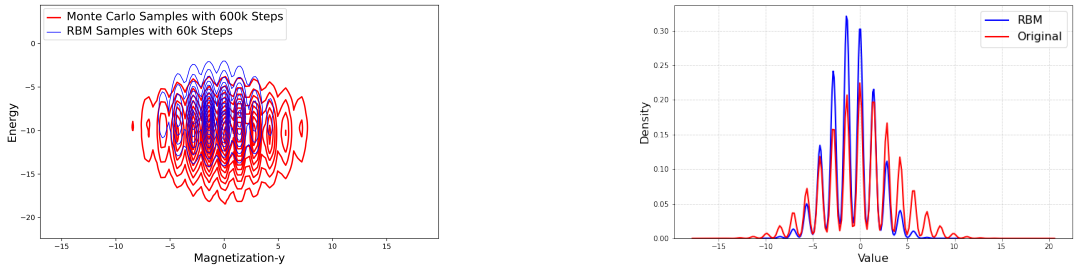


(f) Magnetic Field vs. Density in z-direction.

Figure 4.9: Comparison between RBM results with 15,000 samples and Monte Carlo results with 60,000 samples

compare. Surprisingly, the RBM's results closely matched the extensive simulation.

With 600,000 samples now available, we tested the RBM by training it with this larger set and checking the results. This bigger sample size adds more complexity and variety to the training data. Having more samples increases the RBM's chances



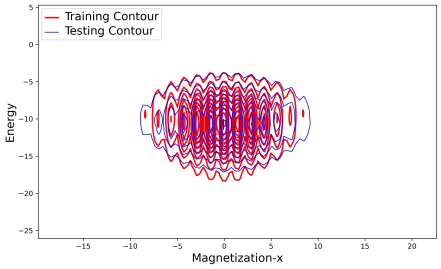
(a) Energy vs. Magnetization in y-direction.

(b) Magnetization vs. Density in y-direction.

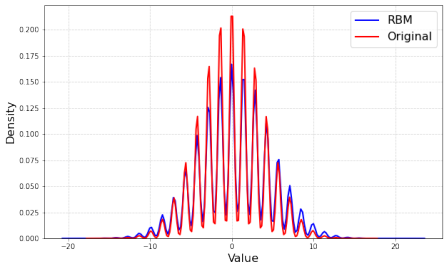
Figure 4.10: Comparison between RBM results with 60,000 samples and Monte Carlo results with 600,000 samples

of learning the underlying patterns and structures in the data.

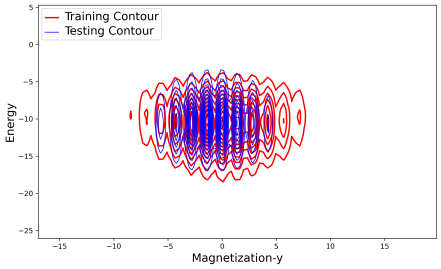
Initially, the 15,000 samples might have yet to fully capture certain features or patterns, which is why they were absent in the initial results. However, as we increased the sample size to 600,000 equilibrated samples, the RBM had a better opportunity to uncover and display these previously hidden features. This led to more accurate and detailed reconstructions of the Mx and My contours.The results are shown in Figure 4.11.



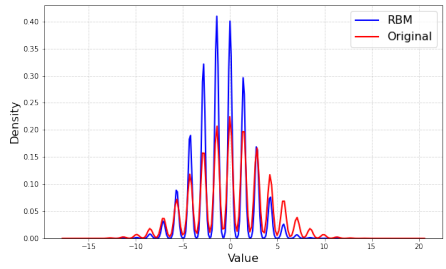
(a) Energy vs. Magnetization in x-direction.



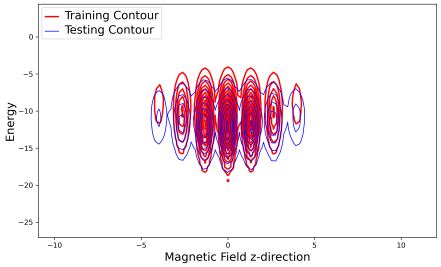
(b) Magnetization vs. Density in x-direction.



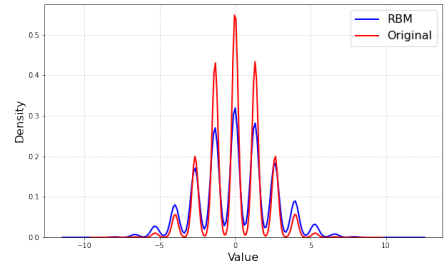
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.



(e) Energy vs. Magnetic Field in z-direction.



(f) Magnetic Field vs. Density in z-direction.

Figure 4.11: The joint energy magnetization distribution of the 600,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5 and their reconstruction using RBM with 36 hidden nodes and 0.0001 learning rate

4.4 Adding Vertical Elements

In order to introduce vertical elements into the system, a 3-dimensional lattice configuration is considered, featuring spins distributed throughout the lattice. To produce the vertical element, it is positioned at the center of a vortex and has zero x and y components, while its z component is not zero, as shown in Figure 4.12. The z-component of this vertical element can be adjusted to modify its strength. Notably, the strength and orientation of the vertical element play crucial roles in influencing the behavior of the lattice spins. When the vertical element points in the positive z direction (out of the plane), it exerts its effect on the spins around the vortex, causing it to pin toward the center of the vortex. Conversely, when the vertical element points in the negative z direction (in the plane), it pins the spins outward from the center of the vortex. Consequently, by manipulating the strength and orientation of this vertical element, it becomes possible to control and manipulate the overall configuration of the lattice. The methods for estimating the system's energy would remain unchanged; all that would change is the addition of these extra vertical elements to the system's spins and the computation of their impact, like other spins, on the entire lattice.

Based on our earlier findings in Figure 4.8, it's evident that the RBM performs better using 36 hidden elements and a learning rate of 0.0001. Thus, we used these parameters for the experiment involving additional vertical elements. Furthermore, since our focus shifts to working with vertical elements, which favour magnetization in the z-direction, we found the results from 15,000 Monte Carlo steps for magnetization in this direction to be satisfactory. Considering our previous demonstration of

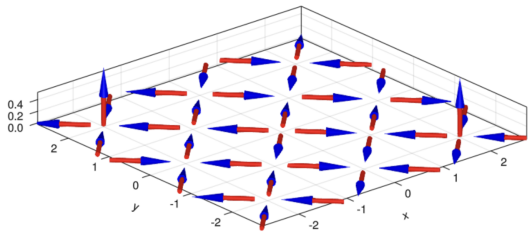
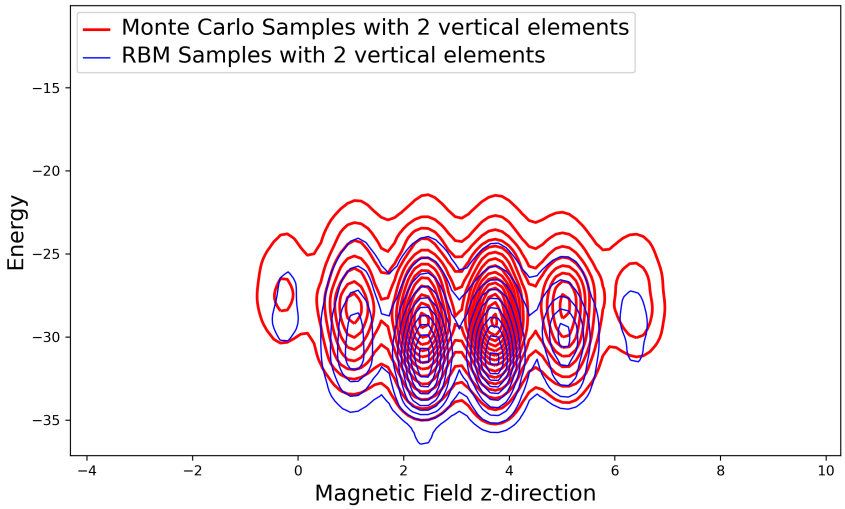
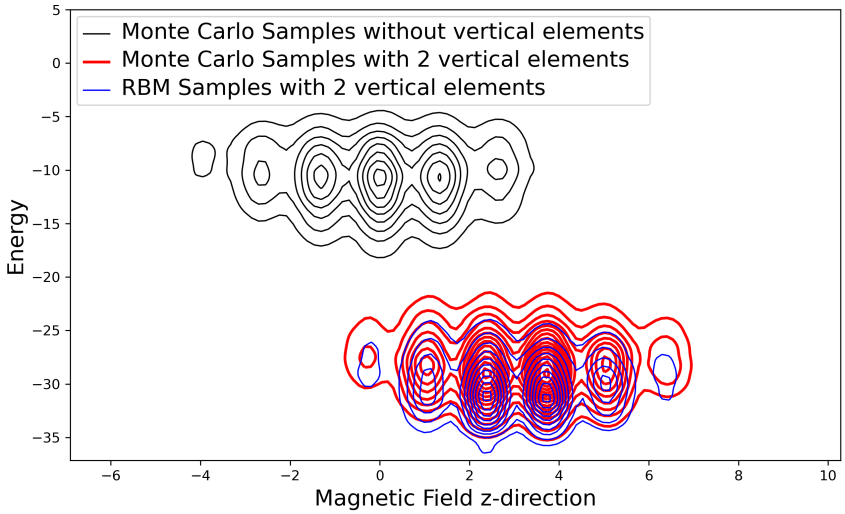


Figure 4.12: ASI system with vertical element. The vertical element is generated by positioning it at the center of a vortex. It possesses zero x and y components, while the z component is non-zero, as depicted in the figure.

the RBM's efficiency with ASI systems, we selected this sample size to save computational time. The outcomes are presented in Figure 4.13, revealing that the RBM effectively reconstructed the data for the system with the vertical elements. Figure 4.13b provides a comparison between the system with and without defects. This illustration helps us grasp the differences. It's important to note that the energy and magnetization of the system with the added vertical elements have undergone a shift compared to the system without these extra components.



(a)



(b)

Figure 4.13: Joint distribution of Energy and magnetic field in z-direction. a) Monte Carlo samples with 15,000 steps, including two vertical elements, plotted alongside their RBM reconstructions using 36 hidden nodes and a 0.0001 learning rate. b) Comparison with the system without vertical elements to illustrate differences.

Chapter 5

Conclusion

In this study, our aim was to explore various machine learning models suitable for the analysis of artificial spin ice (ASI). Our initial goal was to develop Machine Learning tools that can be used to investigate Artificial Spin Ice. Along the way, we came across the work conducted by Roger Melko and David Yevdik [54], who employed a restricted Boltzmann machine (RBM) on an Ising model. This discovery sparked our interest as ASI follows a similar concept to the Ising model.

Given this similarity, we decided to apply RBM to ASI in order to ascertain whether it could replicate or enhance the outcomes of Monte Carlo simulations. Fortunately, our efforts yielded promising results.

5.1 Summery and Results

In Chapter 2, we delved into the concept of artificial spin ice (ASI), its origins, how the elements interact in these setups, and how we used a method called Monte

Carlo simulations to understand these systems better. We demonstrated our findings from Monte Carlo simulations.

Chapter 3 provided a brief introduction to machine learning and neural networks. We then explored a particular kind of neural network called a Restricted Boltzmann Machine (RBM) in detail. We created an RBM from scratch using what we learned and tested it on simple data. We also applied it to magnetic data, reproducing results from earlier work on the Ising model [54]. Toward the end of this chapter, we introduced specific defects called vertical elements and explained why they are intriguing to study.

Moving to Chapter 4, we started using the data we collected from the Monte Carlo simulations of artificial spin ice in Chapter 2. We fed this data into the RBM we built in Chapter 3. It took us some time to find the best settings or hyper-parameters for the RBM using ASI data. This allowed us to generate accurate ASI samples from random distributions. Interestingly, through various experiments, we found that the RBM outperformed the Monte Carlo simulation in generating samples. It could capture significant detailed features that the Monte Carlo simulation couldn't achieve in the same number of steps. Lastly, we applied the RBM to ASI systems containing vertical elements. The RBM successfully reconstructed these data, indicating its ability to identify systems with vertical elements and extract important features from them.

5.2 Future Work

In our ongoing study, we've noticed that the RBM is quite effective at understanding data from systems with or without vertical elements. It's also good at spotting

changes in energy and magnetism between these systems when it gets new data that it hasn't seen before. Now, we want to learn more from these vertical elements using the RBM.

For our subsequent work, we want to learn more about how many of these up-and-down parts are and where they are in the system. To do this, we might need to use another additional neural network for classification purposes. Using features derived from the RBM, we intend to identify the density and positioning of the vertical elements.

To get this information, we plan to put more of these vertical elements within our lattice. We'll generate a dataset with different numbers of vertical elements. We can explore defect densities such as 1/13, 2/13, 3/13, and so on based on our 13-vertex lattice. These different states will be fed into the RBM, enabling it to learn and find essential features and relationships.

Assuming the RBM demonstrates a good result in learning these situations, If the RBM does a good job at learning from these situations, we use that when presented with data from a lattice containing "n" vertical elements; it could potentially identify the best-fitting match across differing defect densities. However, realizing this needs an additional layer of the neural network, wherein the RBM's reconstructions and extracted features aid in pattern classification, thereby providing us with labeled predictions.

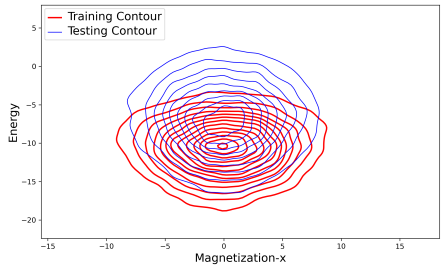
Another research path we can explore involves dividing our lattice into four or more sections and introducing vertical elements into specific sections. By comparing these different sections containing vertical elements with unaffected sections, we can

gather valuable insights. If we input this data into the RBM and utilize an additional neural network to label the positions of these sections, there's potential to uncover the location of the vertical elements on a new lattice that hasn't been shown to the RBM previously.

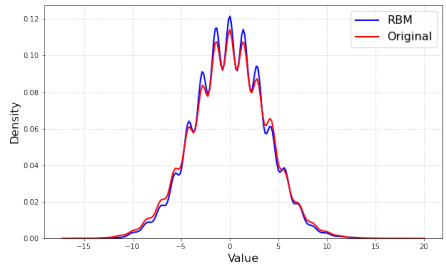
Ultimately, applying machine learning to magnetic systems represents a relatively new and exciting research field. This approach has the potential to address diverse challenges and provide valuable insights that might surpass those obtained through conventional methods. By leveraging the outcomes of this research, the opportunity arises to extend the application of acquired knowledge to other spin systems, presenting a promising avenue for further exploration.

Appendix A

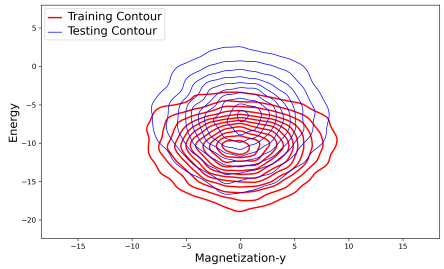
Supporting Data



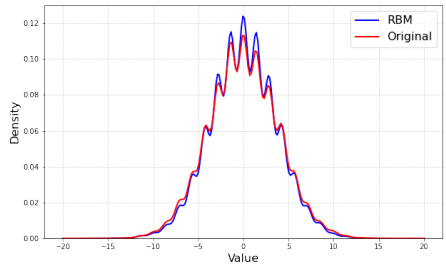
(a) Energy vs. Magnetization in x-direction.



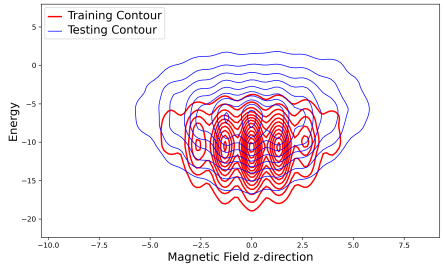
(b) Magnetization vs. Density in x-direction.



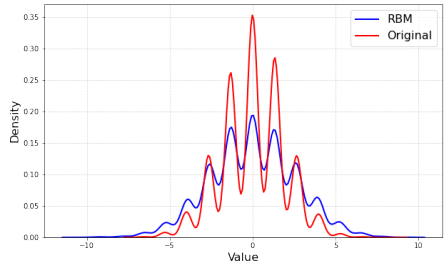
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.

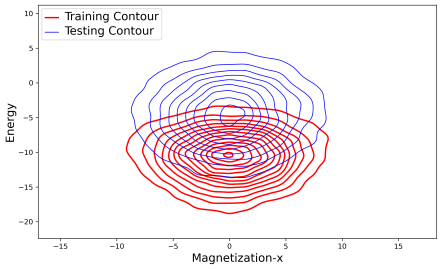


(e) Energy vs. Magnetic Field in z-direction.

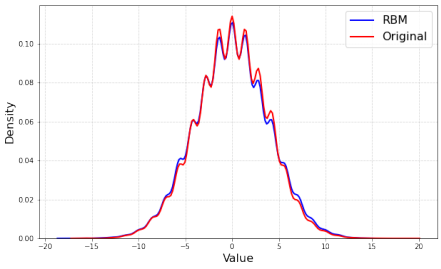


(f) Magnetic Field vs. Density in z-direction.

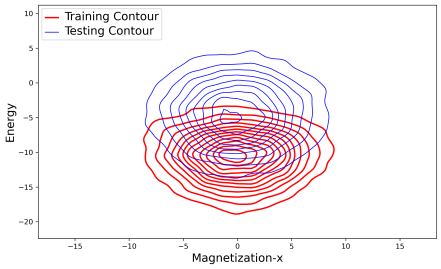
Figure A.1: The joint energy magnetization distribution of the 15,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5 and their reconstruction using RBM with 6 hidden nodes and 0.001 learning rate



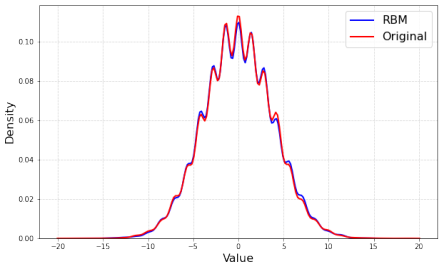
(a) Energy vs. Magnetization in x-direction.



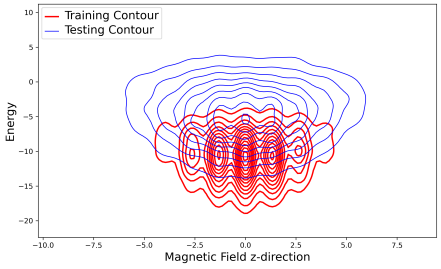
(b) Magnetization vs. Density in x-direction.



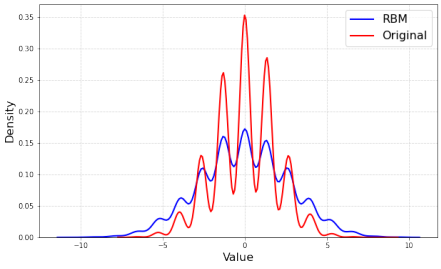
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.

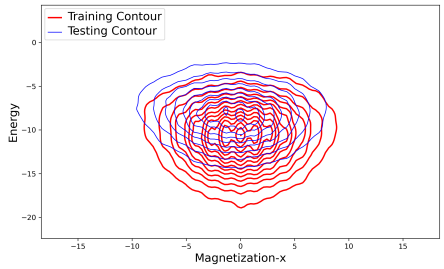


(e) Energy vs. Magnetic Field in z-direction.

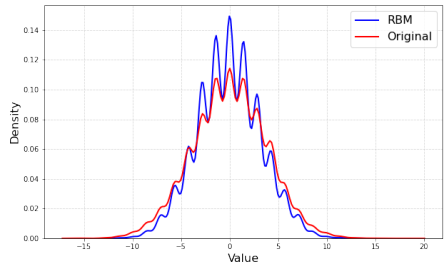


(f) Magnetic Field vs. Density in z-direction.

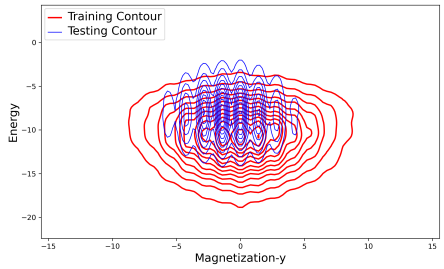
Figure A.2: The joint energy magnetization distribution of the 15,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5 and their reconstruction using RBM with 6 hidden nodes and 0.0001 learning rate



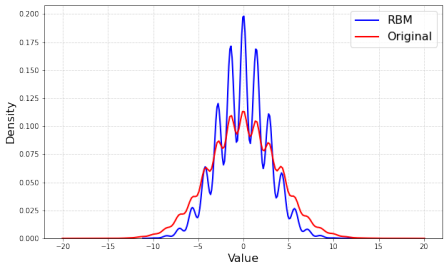
(a) Energy vs. Magnetization in x-direction.



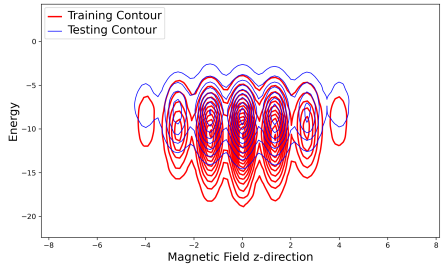
(b) Magnetization vs. Density in x-direction.



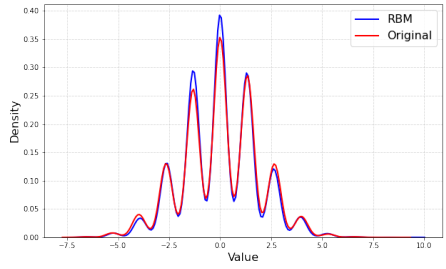
(c) Energy vs. Magnetization in y-direction.



(d) Magnetization vs. Density in y-direction.



(e) Energy vs. Magnetic Field in z-direction.



(f) MAgnetic Field vs. Density in z-direction.

Figure A.3: The joint energy magnetization distribution of the 60,000 steps Monte Carlo samples for a 6 × 6 two-dimensional ASI at a temperature T = 3.5 and their reconstruction using RBM with 36 hidden nodes and 0.0001 learning rate

Bibliography

[1] H. T. Diep. Frustrated spin systems. WorldScientific, 2013.

[2] Roderich Moessner Cristiano Nisoli, , and Peter Schiffer. Colloquium: Artificial spin ice: Designing and imaging magnetic frustration. reviews of modern. *Rev. Mod. Phys.*, 85(4):1473–1490, 2013.

[3] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser,

Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[6] Guido Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[7] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[9] Paraphrasing tool - quillbot ai, 2023. `https://quillbot.com/`.

[10] Grammerly, 2023. `https://grammerly.com/`.

[11] L Pauling. The structure and entropy of ice and of other crystals with some randomness of atomic arrangement. *Journal of the American Chemical Society*, 57(12):2680–2684, 1935.

[12] P.W. Anderson. Resonating valence bonds: A new kind of insulator? *Materials Research Bulletin*, 8:153–160, 1973.

[13] P.W Anderson. Ordering and antiferromagnetism in ferrites. *Physical Review*, 102(4):1008–1013, 1956.

[14] Michel J.P. Gingras Steven T. Bramwell. Spin ice state in frustrated magnetic pyrochlore materials. *Science*, 294:1495–1501, 2001.

[15] L.A.S. Mól J.H. Rodrigues. Towards magnetic monopole interaction measurement in artificial spin ice systems. *Journal of Magnetism and Magnetic Materials*, 458:327–334, 2018.

[16] Cristiano Nisoli Francesco Caravelli, Gia-Wei Chern. Artificial spin ice phase-change memory resistors. *New Journal of Physics*, 24, 2022.

[17] Olle G. Heinonen Sebastian Gliga, Ezio Iacocca. Artificial spin ice phase-change memory resistors. *APL Materials*, 8, 2020.

[18] What is machine learning? — ibm. `www.ibm.com/topics/machine-learning`. Accessed: 2023-06-27.

[19] CHRISTOPHER M. BISHOP. *Pattern recognition and machine learning*. SPRINGER-VERLAG NEW YORK, 2016.

[20] J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[21] Geoffrey Hinton Yann LeCun, Yoshua Bengio. Deep learning. *Nature*, 521:436–444, 2015.

[22] David H Ackley, Hinton Geoffrey E, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

[23] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455. PMLR, 2009.

[24] Roderich Moessner and Arthur P. Ramirez. Geometrical frustration. *Physics Today*, 59(2):24–29, 2006.

[25] W. F. Giauque and Muriel F. Ashley. Molecular rotation in ice at 10°k. free energy of formation and entropy of water. *Physics Today*, 43(1):81–82, 1933.

[26] V.F. Petrenko and R.W Whitworth. Physics of ice. Oxford, UK, 1999. Oxford University Press.

[27] M. Gingras. Spin ice. 2009.

[28] J. D. Bernal and R. H. Fowler. A theory of water and ionic solution, with particular reference to hydrogen and hydroxyl ions. *The Journal of Chemical Physics*, 1(8):515–548, 1933.

[29] R. F. Wang, C. Nisoli, R. S. Freitas, J. Li, W. McConville, B. J. Cooley, M. S. Lund, N. Samarth, C. Leighton, V. H. Crespi, and P. Schiffer. Artificial "spin ice" in a geometrically frustrated lattice of nanoscale ferromagnetic islands. *Nature*, 439(7074):303–306, 2006.

[30] M. Tanaka, E. Saitoh, H. Miyajima, T. Yamaoka, and Y. Iye. Magnetic interactions in a ferromagnetic honeycomb nanoscale network. *Phys. Rev. B*, 73(5):052411, 2016.

[31] Yi Qi, T. Brintlinger, and John Cumings. Direct observation of the ice rule in an artificial kagome spin ice. *Phys. Rev. B*, 77(9):094418, 2008.

[32] E. Mengotti, L. J. Heyderman, A. Fraile Rodríguez, A. Bisig, L. Le Guyader, F. Nolting, and H. B. Braun. Building blocks of an artificial kagome spin ice: Photoemission electron microscopy of arrays of ferromagnetic islands. *Phys. Rev. B*, 78(14):144402, 2008.

[33] Ian Gilbert, Gia-Wei Chern, Sheng Zhang, L. O'Brien, Bryce Fore, Cristiano Nisoli, and Peter Schiffer. Emergent ice rule and magnetic charge screening from vertex frustration in artificial spin ice. *Nat Phys*, 10(9):670–675, 2014.

[34] Gia-Wei Chern, Muir J. Morrison, and Cristiano Nisoli. Degeneracy and criticality from emergent frustration in artificial spin ice. *Phys. Rev. Lett*, 111(17):177201, 2013.

[35] L. A. S. Mól, A. R. Pereira, and W. A. Moura-Melo. Extending spin ice concepts to another geometry: The artificial triangular spin ice. *Phys. Rev. Lett*, 85(18):184410, 2022.

[36] J. Rodrigues, Lucas Mól, Winder Moura-Melo, and Afranio Pereira. Efficient demagnetization protocol for the artificial triangular spin ice. *Applied Physics Letters*, 103(9):092403, 2013.

[37] R. Macêdo, G. M. Macauley, F. S. Nascimento, and R. L. Stamps. Apparent ferromagnetism in the pinwheel artificial spin ice. *Phys. Rev. B*, 98(1):014437, 2018.

[38] M.E.J. Newman and G.T. Barkema. Monte carlo methods in statistical physics. Oxford, UK, 2011. Clarendon Press.

[39] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[40] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97—-109., 1970.

[41] Yong-Lei Wang, Zhi-Li Xiao, Alexey Snezhko, Jing Xu, Leonidas E. Ocola, John E. Divan, Ralu ; Pearson, George W. Crabtree, and Wai-Kwong Kwok. Rewritable artificial magnetic charge ice. *Science (New York, N.Y.)*, 352(6288):962–966, 2016.

[42] Sungjung Joo, Taeyueb Kim, Sang Hoon Shin, Ju Young Lim, Jinki Hong, Jin Dong Song, Joonyeon Chang, Hyun-Woo Lee, Kungwon Rhie, Suk Hee Han, Kyung-Ho Shin, and Mark Johnson. Magnetic-field-controlled reconfigurable semiconductor logic. *Nature*, 494(7435):72–76, 2013.

[43] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[44] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer

networks in unsupervised feature learning. *International Conference on Artificial Intelligence and Statistics*, 2011.

[45] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[46] Ruslan Salakhutdinov and Geoffrey E. Hinton. Replicated softmax: an undirected topic model. In *NIPS*, 2009.

[47] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36, 2012.

[48] Geoffrey Hinton. A practical guide to training restricted boltzmann machines (version 1). *Technical Report UTML TR 2010-003, University of Toronto*, 9, 2010.

[49] Restricted boltzmann machine. `https://www.educba.com/restricted-boltzmann-machine/`. Accessed: 2023-07-01.

[50] Miguel Á. Carreira-Perpiñán and Geoffrey Hinton. On contrastive divergence learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 33–40. PMLR, 2005. Reissued by PMLR on 30 March 2021.

[51] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the

bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–724, 1984.

[52] G. Casella and E. I. George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[53] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[54] David Yevick and Roger G. Melko. The accuracy of restricted boltzmann machine models of ising systems. *Computer Physics Communications*, 258(8):107518, 2020.

[55] G. Gallavotti. On the convergence properties of contrastive divergence. In *Statistical mechanics*, 1999.