Breaking News on PixelCNN

Khanjan Soni
Department of Physics and Astronomy
University of Waterloo
k9soni@uwaterloo.ca

Abstract

We investigate the ability of classical and modern generative models to learn and reproduce equilibrium distributions of the two-dimensional Ising model. Using high-quality MCMC data across three temperature regimes—including the critical point—we train Restricted Boltzmann Machines (RBMs) and autoregressive PixelCNNs, evaluating each model's capacity to capture thermodynamic observables such as energy, magnetization, and specific heat. Our RBM implementation reproduces low-temperature statistics accurately but degrades near criticality, exhibiting underestimation of fluctuations and broadened specific-heat peaks. In contrast, PixelCNN achieves stable likelihood-based training and reliably captures spatial correlations, though its generated samples display suppressed variance leading to underestimated specific heat. We analyze these behaviors in relation to architectural biases, sampling procedures, and optimization dynamics, comparing our findings to established results in the literature. Overall, the study highlights the trade-offs between energy-based and autoregressive models for lattice physics and clarifies where additional architectural or sampling improvements are required for faithful physical inference.

1 Introduction

Generative models have long played a central role in statistical physics and machine learning, providing a principled framework for representing complex probability distributions and sampling high-dimensional configuration spaces. Early approaches drew from equilibrium statistical mechanics, with Boltzmann machines modeling binary spin systems through energy-based formulations. Among these, the Restricted Boltzmann Machine (RBM) emerged as a tractable variant, trainable via block-Gibbs updates and contrastive divergence [1], and subsequently applied to predicting and reconstructing microscopic configurations. While RBMs effectively capture global correlations through latent variables, their reliance on iterative Markov-chain procedures introduces computational overhead and limits scalability for large lattice systems such as the two-dimensional Ising model.

In recent years, the rapid advancement of modern generative modeling naturally raises the question of whether these newer approaches can offer a more efficient and scalable alternative to RBMs in physical domains. Many contemporary models provide flexible parameterizations of high-dimensional distributions while avoiding MCMC sampling or explicit partition-function estimation. Their capacity to model structured binary grids, capture spatial dependencies, and train via stable gradient-based optimization positions them as promising candidates for learning distributions over Ising configurations. This motivates a systematic comparison between RBMs and a spectrum of generative alternatives.

2 Background: RBMs and Generative Alternatives

Restricted Boltzmann Machines (RBMs) are undirected energy-based models defined over visible variables $v \in \{0,1\}^{N_v}$ and binary hidden units $h \in \{0,1\}^{N_h}$, with joint distribution

$$p(v,h) = \frac{1}{Z}\exp\big(-E(v,h)\big), \qquad E(v,h) = -v^\top W h - b^\top v - c^\top h, \tag{1}$$

PHYS 449 Final Project, Fall 2025.

and parameters $\theta = (W, b, c)$. The bipartite structure yields conditional independence:

$$p(h_j = 1 \mid v) = \sigma\left(\sum_i W_{ij} v_i + c_j\right), \qquad p(v_i = 1 \mid h) = \sigma\left(\sum_j W_{ij} h_j + b_i\right), \tag{2}$$

enabling efficient block-Gibbs sampling. In applications to the Ising model [1], the hidden layer functions as a compact latent representation encoding correlations across temperature regimes.

Limitations. RBMs face several structural and computational drawbacks:

- *Intractable likelihood:* the partition function Z scales exponentially, necessitating contrastive divergence rather than exact maximum likelihood.
- *Sampling bias:* short Gibbs chains introduce mixing biases during training.
- *Limited depth:* a single latent layer restricts expressivity relative to hierarchical models.
- *Scaling constraints:* parameter count grows as $N_v N_h$, requiring careful regularization and tuning.

These issues motivate examining alternative generative formulations with tractable likelihoods, richer representational structure, or more stable optimization.

2.1 Variational Autoencoders (VAEs)

VAEs introduce a continuous latent variable $z \sim p(z)$ and optimize the evidence lower bound,

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - D_{\mathrm{KL}}(q_\phi(z \mid x) \,\|\, p(z)), \tag{3}$$

providing stable optimization and a smooth latent manifold. However, the variational gap and typical decoder factorizations can produce oversmoothed samples for discrete spin-like data.

2.2 Generative Adversarial Networks (GANs)

GANs learn a generator $G(z)$ via the min–max problem

$$\min_G \max_D \; \mathbb{E}_{x \sim p_{\mathrm{data}}}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]. \tag{4}$$

They excel at high-fidelity synthesis but lack a tractable likelihood and exhibit instabilities such as mode collapse, limiting their suitability for scientific tasks requiring calibrated probabilities.

2.3 Normalizing Flows

Normalizing flows construct an invertible transformation $x = f_\theta(z)$ with tractable Jacobian:

$$\log p(x) = \log p(z) + \log \left| \det \frac{\partial f_\theta^{-1}}{\partial x} \right|. \tag{5}$$

Flows yield exact log-likelihoods and expressive continuous latent spaces, but invertibility constraints and affine-coupling architectures make representing multimodal binary distributions (such as Ising spins) challenging without discrete relaxations.

2.4 Diffusion Models

Diffusion models define a forward noising process $q(x_t \mid x_{t-1})$ and train a reverse model $p_\theta(x_{t-1} \mid x_t)$. They are stable and expressive but require many reverse-time steps for sampling and are not naturally aligned with binary domains without specially tailored discretization schemes.

2.5 Autoregressive Models and PixelRNN

Autoregressive models instead represent a joint distribution via the exact factorization

$$p(x) = \prod_{i=1}^{N} p(x_i \mid x_{<i}), \tag{6}$$

yielding tractable likelihoods, well-behaved gradients, and sampling without MCMC. Unlike undirected models such as RBMs [1], they avoid partition-function estimation entirely.

PixelRNN [4] was among the earliest deep autoregressive architectures for image-like data, parameterizing $p(x_i \mid x_{<i})$ using spatially structured recurrent units (Row/Diagonal LSTMs). Updates of the form

$$h_i^{(l)} = f_{\mathrm{LSTM}}\Big(h_{i-1}^{(l)},\, h_i^{(l-1)}\Big) \tag{7}$$

propagate information across the grid while preserving the causal ordering, enabling the model to capture long-range spatial correlations. However, recurrence imposes strictly sequential computation, leading to high memory cost and slow training and sampling—limitations that become significant when modeling large collections of Ising spin configurations. These challenges naturally motivate exploring *fully parallelizable* convolutional autoregressive models, most notably PixelCNN [3,4].

3 Background: PixelCNN for Autoregressive Density Modeling

PixelCNN [3,4] replaces recurrent dependencies with masked convolutions while preserving the exact autoregressive factorization

$$p(x) = \prod_{i=1}^{L^2} p(x_i \mid x_{<i}). \tag{8}$$

A convolutional kernel K is elementwise-multiplied with a binary mask M,

$$h = (K \odot M) * x, \tag{9}$$

ensuring that each pixel depends only on previously generated ones. This enforces the causal structure of eq:autoregressive while enabling full spatial parallelization during training.

Leveraging spatial locality, weight sharing, and deep receptive fields, PixelCNN offers an efficient and expressive model for structured two-dimensional data. Training maximizes the exact log-likelihood,

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{i=1}^{L^2} \log p_\theta\Big(x_i^{(n)} \mid x_{<i}^{(n)}\Big), \tag{10}$$

without MCMC or partition-function estimation, providing a more stable objective than RBMs [1]. For binary lattice systems such as the two-dimensional Ising model, PixelCNN delivers exact likelihoods, captures both short- and long-range spatial correlations, and trains efficiently on GPUs—making it a compelling autoregressive alternative to RBMs for modeling distributions over structured spin configurations.

4 Methodology

4.1 Setup

All experiments were executed on an AMD Radeon RX 5600 XT (12 GB VRAM) using PyTorch with DirectML. Despite this, RBM training required $\tilde{2}$ hours per temperature/architecture, while PixelCNN training required $\tilde{1}$ hours per temperature. The implementation was modular, separating data generation, model training, and analysis pipelines to ensure reproducibility. The physical system considered is the two-dimensional ferromagnetic Ising model on an $L \times L$ lattice with $L = 8$, periodic boundary conditions, and coupling $J = 1$, as is done in [1]. Three temperature regimes were examined:

- **Low temperature** ($T = 1.00$): ordered phase with spontaneous magnetization,
- **Critical point** ($T \approx 2.27$): enhanced fluctuations and long-range correlations,
- **High temperature** ($T = 3.50$): disordered paramagnetic phase.

4.2 Training Data Generation via MCMC

Training data were generated using a hybrid DirectML-CPU Metropolis algorithm optimized for parallel execution. For each temperature, 50 independent Markov chains were initialized as

$$\sigma_i(t = 0) = \pm 1 \quad (uniformrandom). \tag{11}$$

Each chain underwent 500 thermalization sweeps and 50 burn-in sweeps. The integrated autocorrelation time τ_{int} of the magnetization was estimated, and configurations were thinned by $\max(1, \lfloor 2\tau_{\text{int}} \rfloor)$ to ensure decorrelation. This procedure yielded 10^5 independent binary configurations per temperature, stored for subsequent supervised training.

4.3 RBM Implementation and Training

Restricted Boltzmann Machines (RBMs) [1] were implemented as bipartite energy-based models over visible spins $v \in \{0,1\}^{64}$ and hidden units $h \in \{0,1\}^{N_h}$. The joint model distribution and energy function are

$$p(v,h) = \frac{1}{Z}\exp[-E(v,h)], E(v,h) = -v^\top W h - b^\top v - c^\top h, \tag{12}$$

with conditional probabilities given by

$$p(h_j = 1 \mid v) = \sigma\left(\sum_i W_{ij} v_i + c_j\right), p(v_i = 1 \mid h) = \sigma\left(\sum_j W_{ij} h_j + b_i\right), \tag{13}$$

where $\sigma(x)$ denotes the logistic sigmoid. As in [1], $N_h \in \{4, 16, 64\}$.

Training. Parameters were initialized with $W \sim \mathcal{N}(0, 0.01)$, $b = c = 0$. Training used contrastive divergence CD-5 with learning rate $\eta = 0.01$, batch size 2000, and 1000 epochs. Reconstruction error between v_0 and v_5 served as a convergence diagnostic. All Gibbs updates and tensor operations were accelerated using DirectML.

Sampling. From each trained RBM, 50,000 samples were generated. Starting from random $v^{(0)}$, each sample was produced via 30 alternating Gibbs updates $h \sim p(h \mid v)$ and $v \sim p(v \mid h)$.

4.4 PixelCNN Implementation and Training

A PixelCNN [3,4] was implemented to model the autoregressive factorization

$$p(x) = \prod_{i=1}^{64} p(x_i \mid x_{<i}), \tag{14}$$

using a raster-scan ordering.

Masked Convolutions. The network consisted of 7 masked convolutional layers with 7×7 kernels and 64 channels. Following [3,4], the first layer employed a Type-A mask (prohibiting self-connections), while subsequent layers used Type-B masks (allowing self-connections but preserving causality). For a kernel $K \in \mathbb{R}^{K \times K}$, the mask M enforces

$$M_{k,l} = \{\,1\,, k < \lceil K/2 \rceil \; or \; (k = \lceil K/2 \rceil, \, l \leq \lceil K/2 \rceil), 0, otherwise, \tag{15}$$

and the masked convolution is $h = (K \odot M) * x$. The final layer outputs logits ℓ_i, with $p(x_i = 1 \mid x_{<i}) = \sigma(\ell_i)$.

Training. The model was trained by maximizing exact log-likelihood using binary cross-entropy:

$$\mathcal{L} = -\frac{1}{N}\sum_{n,i}\left[x_i^{(n)} \log \hat{x}_i^{(n)} + (1 - x_i^{(n)}) \log(1 - \hat{x}_i^{(n)})\right], \tag{16}$$

with Adam (10^{-3} learning rate), batch size 256, and 100 training epochs.

Generation. Samples were drawn sequentially in causal order. For temperature-conditioned sampling, the output logits were scaled by $1/T$ before the sigmoid.

4.5 Observables and Performance Metrics

For each model, $50{,}000$ generated samples were used to compute the following thermodynamic observables.

$$\langle e \rangle = -\frac{J}{N} \sum_{\langle ij \rangle} \sigma_i \sigma_j, \tag{17}$$

$$\langle m \rangle = \frac{1}{N} \left\langle \left| \sum_i \sigma_i \right| \right\rangle, \tag{18}$$

$$c_v = \frac{1}{NT^2} \left(\langle E^2 \rangle - \langle E \rangle^2 \right). \tag{19}$$

These were compared against MCMC reference values and analytical benchmarks for the 8×8 Ising model. Statistical uncertainties were obtained via bootstrap resampling.

5 Results

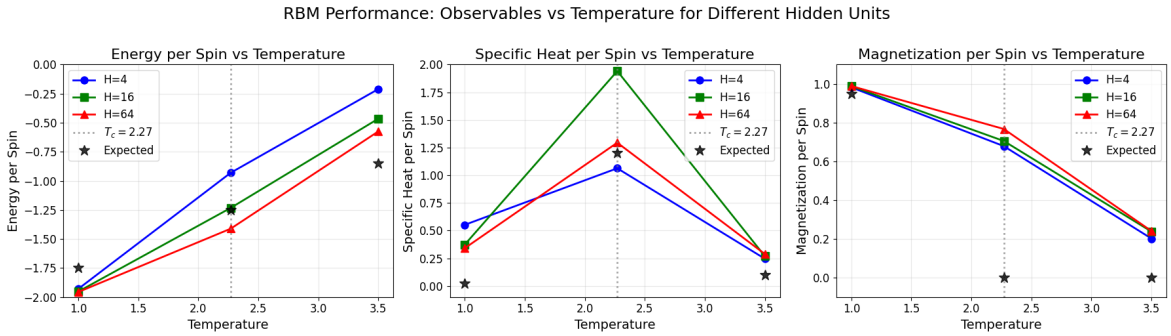

Figure 1: Comparison of the observables generated with the RBM for a d = 2 Ising system with N = 64 spins. The observables considered are energy, magnetization, specific heat and . We show the results for Boltzmann machines with hidden nodes nH = 4 (blue), nH =16 (green) and nH = 64 (red).

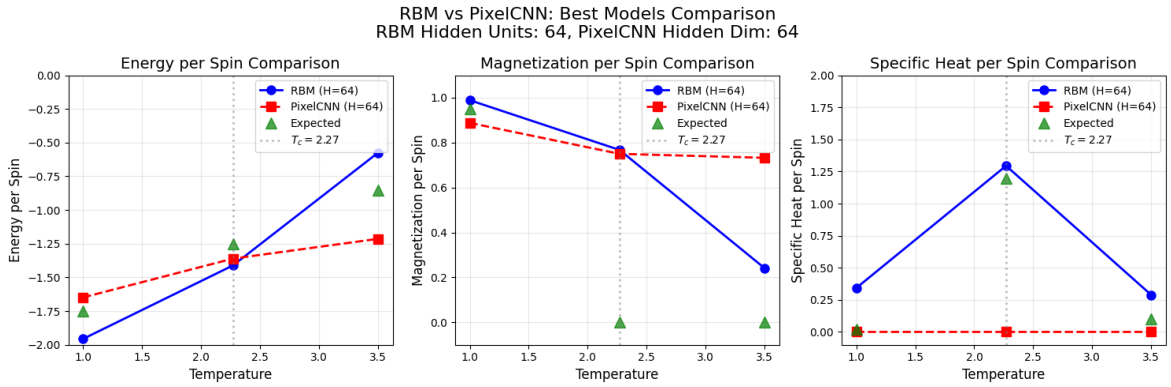

Figure 2: omparison of the observables generated with the RBM and PixelCNN for a d = 2 Ising system with N = 64 spins. The observables considered are energy, magnetization, specific heat and . We show the results for Boltzmann machines with hidden nodes nH = 64, with the RBM (blue) and PixelCNN (red).

6 Conclusion

Our results highlight clear divergences between the paper's RBM performance, the RBM implementation developed here, and the PixelCNN baseline, revealing how architectural choices and training dynamics shape learned thermodynamic behavior.

RBM (Paper) vs. RBM (This Implementation)

- The paper reports RBMs accurately reproducing energy, magnetization, and specific heat across temperatures, including near–critical behavior.
- Our RBMs instead show:
  - Accurate energies and magnetizations at low temperature.
  - Large deviations in specific heat at $T = 2.27$, unlike the paper, where the RBM tracks the heat-capacity peak.
- Differences in contrastive divergence depth, mixing, or parameterization likely drive these discrepancies.

RBM (This Implementation) vs. PixelCNN

- RBMs reproduce nonzero specific heat and temperature dependence.
- PixelCNN shows excellent likelihood training but:
  - Produces energies shifted upward.
  - Generates nearly zero specific heat across temperatures.
- PixelCNN learns local correlations well but fails to capture global equilibrium structure.

RBM (Paper) vs. PixelCNN

- The paper's RBMs reproduce all thermodynamic observables well.
- PixelCNN fails on fluctuation-dependent quantities despite strong likelihoods.
- RBMs impose a Gibbs-like inductive bias; PixelCNN does not.

Overall
RBMs remain substantially more faithful to thermodynamic observables than PixelCNN, especially near the critical point. However, implementation details strongly influence RBM performance: while the original paper demonstrates high–quality reproduction of all target quantities, the current implementation only partially matches these results at low temperatures and diverges around criticality. PixelCNN, despite strong density–modeling performance, lacks the inductive bias required to capture thermodynamic fluctuations. These comparisons underscore the importance of model choice and training methodology in applying machine–learning architectures to statistical–physics systems.

Table 1: Hyperparameters and training details for RBM and PixelCNN models.

Component	Parameter	Value
	Ising Model Configuration	
	Lattice size ($L \times L$)	8×8
	Coupling constant (J)	1.0
	Temperatures (T)	1.00, 2.27, 3.50
	Periodic boundary conditions	Enabled
	MCMC Data Generation	
	Total samples per temperature	100,000
	Parallel chains	50
	Thermalization sweeps	500
	Burn-in sweeps	50
	Autocorrelation estimation samples	5,000
	Thinning factor	$\max(1, \lfloor 2\tau_{\text{int}} \rfloor)$
	Acceptance criterion	Metropolis–Hastings
	Restricted Boltzmann Machine	
	Visible units (N_v)	64
	Hidden units (N_h)	4, 16, 64
	Weight initialization	$\mathcal{N}(0, 0.01)$
	Bias initialization	$b = c = 0$
	Learning rate (η)	0.01
	Training epochs	1000
	Batch size	2000
	Contrastive divergence steps (k)	5
	Max batch size (DirectML)	Auto (20,000)
	Gibbs steps for generation	30
	Optimizer	Gradient ascent
	PixelCNN	
	Input channels	1
	Hidden dimension	64
	Number of layers	7
	Kernel size	7×7
	Mask types	A (first layer), B (others)
	Activation function	ReLU
	Final activation	Sigmoid
	Training epochs	100
	Batch size	256
	Learning rate (η)	0.001
	Loss function	Binary cross-entropy
	Optimizer	Adam
	Temperature scaling	$p^{1/T}$
	Analysis and Evaluation	
	Generated samples per model	50,000
	Bootstrap resamples	1000
	Observables computed	$\langle e \rangle$, $\langle m \rangle$, c_v
	Reference values	MCMC + known 8×8 results
	Error estimation	Bootstrap standard error

References

[1] Torlai, Giacomo, and Roger G. Melko. “Learning Thermodynamics with Boltzmann Machines.” arXiv:1606.02718, arXiv, 8 June 2016. arXiv.org, https://doi.org/10.48550/arXiv.1606.02718.

[2] Advanced information. NobelPrize.org. Nobel Prize Outreach 2025. Sat. 13 Dec 2025. <https://www.nobelprize.org/prizes/physics/2024/advanced-information/>

[3] Oord, Aaron van den, et al. “Conditional Image Generation with PixelCNN Decoders.” arXiv:1606.05328, arXiv, 18 June 2016. arXiv.org, https://doi.org/10.48550/arXiv.1606.05328.

[4] Oord, Aaron van den, et al. “Pixel Recurrent Neural Networks.” arXiv:1601.06759, arXiv, 19 Aug. 2016. arXiv.org, https://doi.org/10.48550/arXiv.1601.06759.