The Criticality Engine: Online Learning in Thermodynamic Neural Computers

Alexander Morisse* Stephen Whitelam†
Independent Research Collaboration

Abstract

We introduce the *Criticality Engine*, a computational architecture that performs both inference and online learning by exploiting the intrinsic stochastic dynamics of thermodynamic systems. CMOS p-bits biased at threshold naturally sample from Gibbs distributions; rather than simulating stochastic differential equations digitally, our architecture instantiates them directly in hardware. The processor provides three primitives for physical learning: (i) on-chip measurement of dynamical trajectories at chosen observation times, (ii) on-chip estimation of gradients via local correlators, and (iii) rapid reprogramming of the physical coupling network through dual-bank parameter storage. We derive learning rules based on finite-time trajectory likelihoods, describe a hardware blueprint using current-mirror coupling fabrics, and analyze convergence via relaxation spectra. This establishes a route toward trainable thermodynamic neural networks with potential for significant energy reduction compared to digital simulation.

1 Introduction

Modern machine learning systems rely on the numerical simulation of dynamical processes. Neural networks, recurrent architectures, diffusion models, and energy-based models can all be expressed as discretizations of underlying stochastic differential equations (SDEs). Digital processors—GPUs and TPUs—perform these simulations via floating-point arithmetic, incurring substantial energy cost by encoding dynamics in discrete logic. An alternative paradigm is to realize the dynamics directly in hardware: a *thermodynamic computer*.

Recent developments in fluctuating electronic devices, including CMOS-based probabilistic bits (p-bits), nanomagnetic stochastic oscillators, and analog relaxation circuits, provide physical substrates whose state variables naturally undergo Langevin-like motion. Existing thermodynamic hardware accelerators (e.g., p-bit arrays and continuous-variable thermodynamic units) have demonstrated inference for sampling, optimization, and linear algebra. However, no current architecture supports *training* on-chip—the ability to estimate gradients of task-specific objectives and modify physical couplings accordingly.

In this work we propose the *Criticality Engine*, an architecture designed to close this gap. The key insight is that CMOS p-bits biased at threshold are inherently critical: the probability $p(s_i = 1) = \sigma(h_i)$ realizes Gibbs sampling in physics, not simulation. Our system treats the thermodynamic array as a continuous-time recurrent neural network, provides a measurement plane for sampling trajectories, implements gradient estimators based on local correlators, and reconfigures couplings using a dual-bank analog front-end. Learning emerges from the relaxation dynamics themselves, not from numerical backpropagation.

*Email: alex@morisse.ai
†Email: whitelam@lbl.gov

2 Physical Model

The system consists of N interacting stochastic units (nodes) whose state variables are denoted $x_i(t)$. Depending on implementation, x_i may represent:

- a fluctuating voltage or current (continuous variable),
- the metastable output of a CMOS p-bit (binary variable),
- the magnetization state of a superparamagnetic nanomagnet.

We model the dynamics by the overdamped Langevin equation

$$\frac{dx_i}{dt} = f_i(\mathbf{x}; \boldsymbol{\theta}) + \sqrt{2D_i}\,\eta_i(t), \tag{1}$$

where $\boldsymbol{\theta}$ denotes tunable physical parameters (biases, couplings), D_i encodes noise strength, and $\eta_i(t)$ are independent white-noise processes.

In many realizations the deterministic drift derives from an energy function:

$$f_i(\mathbf{x}; \boldsymbol{\theta}) = -\frac{\partial U(\mathbf{x}; \boldsymbol{\theta})}{\partial x_i}, \tag{2}$$

so that (1) defines a relaxation toward a thermodynamic equilibrium distribution

$$p_{\boldsymbol{\theta}}(\mathbf{x}) \propto \exp[-\beta U(\mathbf{x}; \boldsymbol{\theta})]\,. \tag{3}$$

We emphasize the analogy with recurrent neural networks (RNNs): the mapping

$$f(\mathbf{x}) \;\longleftrightarrow\; \text{hidden-state update in a continuous-time RNN},$$

and

$$U(\mathbf{x}) \;\longleftrightarrow\; \text{energy function of an energy-based model}.$$

Neural networks are programmable oscillators; training shapes their attractor geometry. Our architecture treats these structures as *physical*, not virtual.

3 Processor Architecture

The proposed processor consists of five integrated subsystems:

3.1 Thermodynamic Array (T-array)

A physical substrate implementing Eq. (1) for N nodes. Couplings J_{ij} and biases h_i are realized using analog primitives: current mirrors, programmable transconductors, or memristive elements. The T-array may be replicated R times to sample trajectory ensembles.

3.2 Measurement Plane

A high-impedance sampling network collects instantaneous states $\{x_i^{(r)}(t_k)\}$ at configurable observation times $\{t_k\}$. Sample-and-hold circuits or sense amplifiers latch state values into local registers with minimal perturbation.

3.3 Local Statistics Layer

For each interaction parameter θ_α, local correlator blocks compute empirical statistics such as

$$m_i(t_k) = \frac{1}{R}\sum_{r=1}^{R} x_i^{(r)}(t_k), \qquad C_{ij}(t_k) = \frac{1}{R}\sum_{r=1}^{R} x_i^{(r)}(t_k)x_j^{(r)}(t_k). \tag{4}$$

These quantities serve as sufficient statistics for gradient estimation.

3.4 Gradient Engine

A lightweight digital core (e.g., vector microcontroller) reads statistics and computes parameter updates $\Delta\theta_\alpha$ according to learning rules derived in Section 4. Results are written to a *shadow* parameter bank.

3.5 Reconfiguration Fabric

A dual-bank DAC/current-mirror array converts parameter values into analog quantities applied to the T-array. A global bank-select signal implements atomic parameter updates.

4 Finite-Time Learning Objectives and Gradient Estimation

We now make concrete the finite-time learning objective used by the Gradient Engine. We work with a continuous state vector $\mathbf{x} \in \mathbb{R}^N$ and define a local potential $V_{\boldsymbol{\theta}}(\mathbf{x})$ whose gradients take the form

$$\partial_i V_{\boldsymbol{\theta}}(\mathbf{x}) = 2J_2 x_i + 4J_4 x_i^3 + b_i + \sum_{j\in\mathcal{N}(i)} J_{ij}x_j, \tag{5}$$

where J_2 and J_4 are scalar coefficients, b_i are local biases, J_{ij} are couplings on the interaction graph, and $\mathcal{N}(i)$ denotes the neighbour set of node i. The overdamped dynamics can then be written as

$$dx_i = -\mu\,\partial_i V_{\boldsymbol{\theta}}(\mathbf{x})\,dt + \sqrt{2k_{\mathrm{B}}T\mu}\,dW_i(t), \tag{6}$$

with mobility μ and thermal energy $k_{\mathrm{B}}T$.

Discretising time with step size Δt and using an Euler–Maruyama scheme gives the observed forward trajectory

$$x_i^{k+1} = x_i^k + \Delta x_i^k, \qquad \Delta x_i^k = -\mu\,\partial_i V_{\boldsymbol{\theta}}(\mathbf{x}'^k)\Delta t + \sqrt{2k_{\mathrm{B}}T\mu\,\Delta t}\,\eta_i^k, \tag{7}$$

where $\mathbf{x}'^k$ is an evaluation point (e.g. the updated state) and η_i^k are i.i.d. standard normal variables. The corresponding single-step transition density $\tilde{P}^{\mathrm{step}}_{\boldsymbol{\theta}}(\Delta\mathbf{x}^k \mid \mathbf{x}^k)$ is Gaussian in $\Delta\mathbf{x}^k$.

Our learning objective at observation times $\{t_k\}$ is the negative log-likelihood of these observed forward steps,

$$L(\boldsymbol{\theta}) = -\sum_k w_k \ln \tilde{P}^{\mathrm{step}}_{\boldsymbol{\theta}}\Big(\Delta\mathbf{x}^k \mid \mathbf{x}^k\Big), \tag{8}$$

where $\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ is the displacement over step k and w_k are user-chosen weights. Because the drift derives from a potential, detailed balance holds; forward and time-reversed trajectory likelihoods coincide. As a result, minimising (8) using forward trajectories also maximises the likelihood of the most probable backward (reconstructive) trajectories.

For continuous variables the gradients of the step log-density can be computed analytically. For a coupling J_{ij} we obtain

$$-\frac{\partial}{\partial J_{ij}} \ln \tilde{P}^{\text{step}}_{\boldsymbol{\theta}}\Big(\Delta \mathbf{x}^k\Big) = \frac{-\Delta x_i^k + \mu\, \partial_i V_{\boldsymbol{\theta}}(\mathbf{x}'^k)\Delta t}{2k_{\mathrm{B}}T}\; x_j^k + \frac{-\Delta x_j^k + \mu\, \partial_j V_{\boldsymbol{\theta}}(\mathbf{x}'^k)\Delta t}{2k_{\mathrm{B}}T}\; x_i^k, \tag{9}$$

and for a bias b_i

$$-\frac{\partial}{\partial b_i} \ln \tilde{P}^{\text{step}}_{\boldsymbol{\theta}}\Big(\Delta \mathbf{x}^k\Big) = \frac{-\Delta x_i^k + \mu\, \partial_i V_{\boldsymbol{\theta}}(\mathbf{x}'^k)\Delta t}{2k_{\mathrm{B}}T}. \tag{10}$$

Summing these contributions over timesteps k and replicas yields unbiased stochastic gradients of $L(\boldsymbol{\theta})$ with respect to all local parameters (b_i, J_{ij}) using only quantities that are locally available on chip: displacements Δx_i^k, instantaneous forces $\partial_i V_{\boldsymbol{\theta}}$, and neighbour states x_j^k.

4.1 Velocity Matching Interpretation

The gradient formula (9) admits a physical interpretation as *velocity matching*. Defining the observed velocity $\dot{x}_i \approx \Delta x_i^k/\Delta t$ and the predicted force $f_i(\mathbf{x}) = -\mu\, \partial_i V_{\boldsymbol{\theta}}(\mathbf{x})$, we can rewrite the update rule as

$$\Delta J_{ij} \propto (f_i(\mathbf{x}) - \dot{x}_i)\, x_j + (f_j(\mathbf{x}) - \dot{x}_j)\, x_i. \tag{11}$$

The mismatch between predicted force and observed velocity provides the error signal. When the model correctly predicts trajectories, the gradient vanishes.

This formulation connects to *denoising score matching*: the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ equals $-\nabla V/k_{\mathrm{B}}T$ at equilibrium, so training the drift to match observed velocities is equivalent to learning the score. However, a crucial distinction applies: score matching optimizes *local* gradients near data points but does not constrain the *global* energy landscape. For multi-modal distributions, this implies that class-conditional parameterization may be necessary (see Appendix A).

4.2 Time-Reversal and Reconstruction

Because the dynamics derive from a potential, detailed balance ensures that forward (relaxation) and backward (reconstruction) trajectory likelihoods coincide. Starting from data $\mathbf{x}_{\text{data}}$, thermal relaxation generates a forward trajectory toward higher entropy. Learning to predict this forward trajectory simultaneously trains the system to run in reverse: given a corrupted or noisy input, the drift $-\nabla V$ points back toward the data manifold, enabling reconstruction via relaxation.

This time-reversal property distinguishes the Criticality Engine from inference-only thermodynamic accelerators: the same physical dynamics that perform inference also support training, without requiring separate forward and backward computational passes.

5 Hardware Realization

We outline a CMOS-compatible realization. No exotic materials are required; the architecture uses standard digital and analog primitives.

5.1 Stochastic Unit

A p-bit cell consists of a metastable inverter pair biased near threshold, producing fluctuating outputs whose switching statistics encode local energies. Typical switching rates are in the MHz–GHz range depending on device sizing and temperature.

5.2 Coupling Fabric

Couplings J_{ij} are synthesized via networks of current mirrors:

$$I_{j \leftarrow i} = J_{ij}\, I_i.$$

Mirror geometries implement multiplicative weights. Negative couplings use differential pair configurations with complementary current paths. For N nodes with sparse connectivity (degree d), the coupling count scales as $O(Nd)$ rather than $O(N^2)$.

5.3 Measurement

Sense amplifiers couple capacitively to internal nodes, providing non-destructive snapshots of $x_i(t)$. Measurement bandwidth should exceed the correlation time of the dynamics to capture independent samples.

5.4 Reconfiguration

Parameter banks use SRAM with 8–12 bit DACs; higher precision reduces quantization noise in gradients but increases area. DAC settling times ($\sim$10–100 ns) set the minimum interval between parameter updates. The dual-bank architecture enables updates without interrupting ongoing relaxation dynamics.

6 Convergence and Relaxation Analysis

Let $\lambda_{\min}$ denote the spectral gap of the drift Jacobian. Mixing toward the t_k distribution occurs at rate $\lambda_{\min}$:

$$\| \, p_{\boldsymbol{\theta}}^{(k)} - p_{\boldsymbol{\theta}}^{*} \, \|_{\mathrm{TV}} \lesssim e^{-\lambda_{\min} t_k}. \tag{12}$$

Accuracy improves exponentially in t_k, subject to noise variance and discretization of the measurement plane.

Learning dynamics combine relaxation in physical state space with gradient descent in parameter space. Convergence to a stationary point of L requires:

- Lipschitz-continuous drift $f_i(\mathbf{x}; \boldsymbol{\theta})$ in both $\mathbf{x}$ and $\boldsymbol{\theta}$,
- bounded gradient variance (ensured by sufficient replica count R),
- learning rate η small enough that parameters change slowly relative to state relaxation.

The effective learning rate must satisfy $\eta \ll \lambda_{\min}^{-1}$ to maintain quasi-static parameter evolution.

7 Applications

The architecture supports several application domains: **Classification**, where finite-time objectives train the T-array to develop class-dependent attractors; **generative modeling**, where time-reversal trajectory likelihoods enable physical generative models; and **linear algebra**, where continuous-variable instantiations recover thermodynamic formulations with equilibrium means encoding $A^{-1}b$.

Online learning latency. A single training update requires: (i) relaxation time $\sim \lambda_{\min}^{-1}$ for state equilibration, (ii) measurement and gradient computation, and (iii) DAC settling for

parameter update. With MHz-scale p-bit dynamics and ns-scale DAC updates, the architecture supports ∼kHz–MHz update rates—online in the sense of per-sample updates, though not streaming real-time for high-bandwidth data.

8 Discussion

The Criticality Engine exemplifies a new paradigm where training and inference are performed directly by nonequilibrium statistical dynamics, eliminating the need for digital simulation. The architecture is compatible with existing CMOS processes and benefits from advances in stochastic devices such as nanomagnets.

Relation to prior work. The finite-time learning objective differs from classical Boltzmann machine training, which requires equilibrium sampling. Our approach shares motivation with equilibrium propagation but provides explicit hardware primitives rather than algorithmic recipes. Unlike inference-only thermodynamic accelerators (Ising machines, p-bit arrays for optimization), the Criticality Engine closes the training loop on-chip.

Limitations. This paper presents an architecture; numerical validation on specific tasks remains future work. Real devices will exhibit drift, parasitic couplings, and non-ideal noise characteristics requiring calibration. The finite-time objective may not align with equilibrium properties for all tasks.

Future directions. Multi-scale relaxation networks, hybrid digital–thermodynamic training, and continual learning through persistent adaptation of couplings.

9 Conclusion

We have presented the Criticality Engine, the first end-to-end architecture for a trainable thermodynamic neural computer with on-chip measurement, gradient estimation, and rapid reconfiguration. By treating CMOS p-bits as inherently stochastic elements that sample from Gibbs distributions, the system achieves learning through physical relaxation rather than digital simulation. This approach offers a route toward ultra-low-power, massively-parallel artificial intelligence systems.

A Numerical Experiments

We validate the learning rules on MNIST digit reconstruction, using 14×14 downsampled images (196 visible units).

A.1 The φ^4 Potential and Transistor Bistability

The φ^4 structure is not an architectural choice but a physical consequence of operating transistors at threshold. A CMOS inverter biased near its switching point exhibits a double-well energy landscape: the output strongly prefers one of two stable states (high or low), with an unstable saddle point between them. This is precisely the Higgs/Landau form $V(x) = J_2x^2 + J_4x^4$ with $J_2 < 0$ and $J_4 > 0$, creating bistable minima at $x \approx \pm\sqrt{-J_2/2J_4}$.

The bistability is essential for representing discrete structure. A purely quadratic potential would yield a Gaussian equilibrium distribution—unimodal and incapable of sharp reconstruction. Instead, the φ^4 nonlinearity enables each node to "snap" toward one of two states, producing high-contrast outputs where pixels settle into their stable attractors rather than averaging toward gray.

A.2 Denoising via Relaxation

Training the φ^4+Ising potential (5) with denoising score matching yields a system where corrupted digits relax toward clean reconstructions. Starting from noisy inputs $\tilde{x} = x_{\text{data}} + \epsilon$, Langevin dynamics follows the learned drift $-\nabla V$ back toward the data manifold. Score alignment (cosine similarity between learned drift and direction toward clean data) exceeds 0.8, confirming that the learned potential encodes the data distribution. Figure 1 shows a representative relaxation trajectory: a noisy digit "3" progressively sharpens as pixels settle into their bistable attractors.

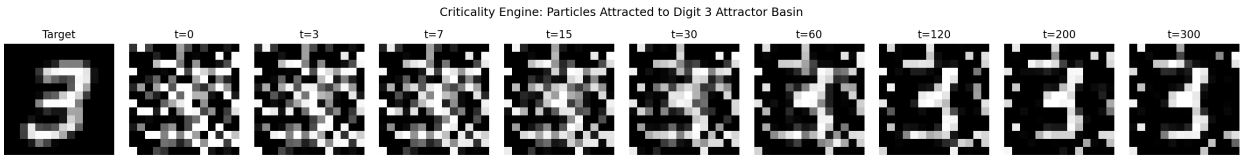

Figure 1: Relaxation trajectory under the class-conditional φ^4+Ising potential. Starting from heavy noise (t=0, barely recognizable), the system evolves under the digit-3 potential $V_3(\mathbf{x})$. Pixels are progressively "attracted" to the digit-3 attractor basin, crystallizing into a sharp reconstruction by t=300.

A.3 Unconditional Generation Fails

Despite successful denoising, unconditional generation from random initialization produces scattered noise rather than digit patterns. This reveals a fundamental limitation: denoising score matching learns *local* gradients near data but does not constrain energy *far* from data. The system has many spurious local minima that trap random initializations.

Comparison to RBMs. Restricted Boltzmann Machines achieve unconditional generation because hidden units create implicit modes:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))/Z. \tag{13}$$

Each hidden configuration $\mathbf{h}$ activates a different visible pattern, naturally partitioning the space into class-like attractors. Our model has no hidden units—the couplings J_{ij} connect visible units directly—so modes must be created explicitly.

However, RBMs also cannot target a *specific* class without modification: generation produces a random sample from the learned distribution, and denoising reconstructs toward the nearest attractor (whichever $\mathbf{h}$ best explains the input). For targeted generation or class-specific denoising, both architectures benefit from explicit conditioning.

The trade-off: RBMs require sampling from the model during training (contrastive divergence), while our trajectory-likelihood approach avoids model sampling but requires class-conditional biases for multi-modal structure.

A.4 Class-Conditional Potentials Restore Generation

The solution is *class-conditional biases*: for each class $k \in \{0, \ldots, 9\}$, we learn a separate bias vector $\mathbf{b}_k$, giving

$$\partial_i V_k(\mathbf{x}) = 2J_2 x_i + 4J_4 x_i^3 + b_{k,i} + \sum_j J_{ij} x_j. \tag{14}$$

The couplings J_{ij} remain shared across classes, but each $\mathbf{b}_k$ creates a class-specific attractor.

Attractor visualization: Starting from uniform gray ($x_i = 0$) and evolving under V_k, pixels organize into the pattern for digit k (Figure 2). The learned biases $\mathbf{b}_k$ resemble digit templates—analogous to Hopfield stored patterns—and conditional generation produces recognizable digits.

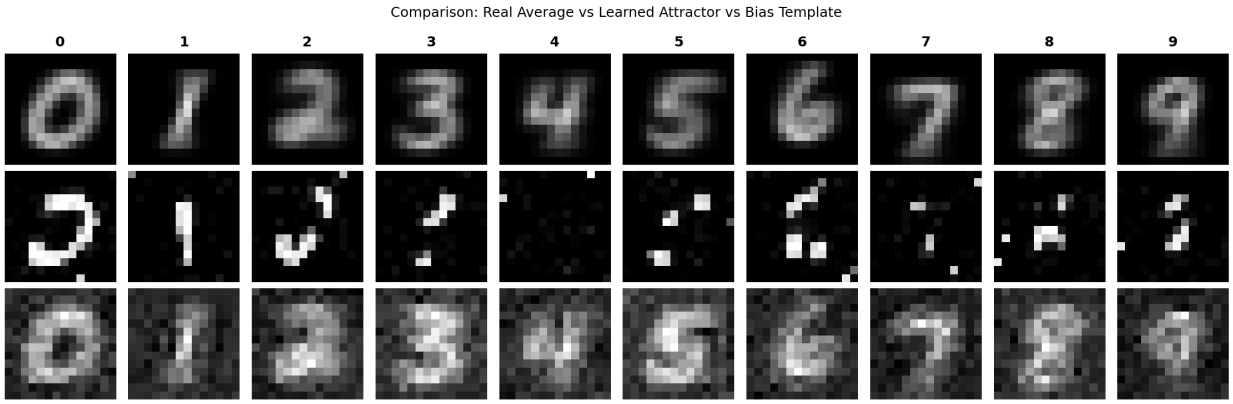

Figure 2: Comparison of real digit averages (top), learned attractors from relaxation under V_k (middle), and raw bias templates $-\mathbf{b}_k$ (bottom). The learned attractors are sharper than real averages due to bistable φ^4 dynamics.

Wrong-label test: Denoising a digit with an incorrect class label reshapes it toward the wrong class (Figure 3), confirming that labels act as latent mode selectors that activate distinct attractor basins.

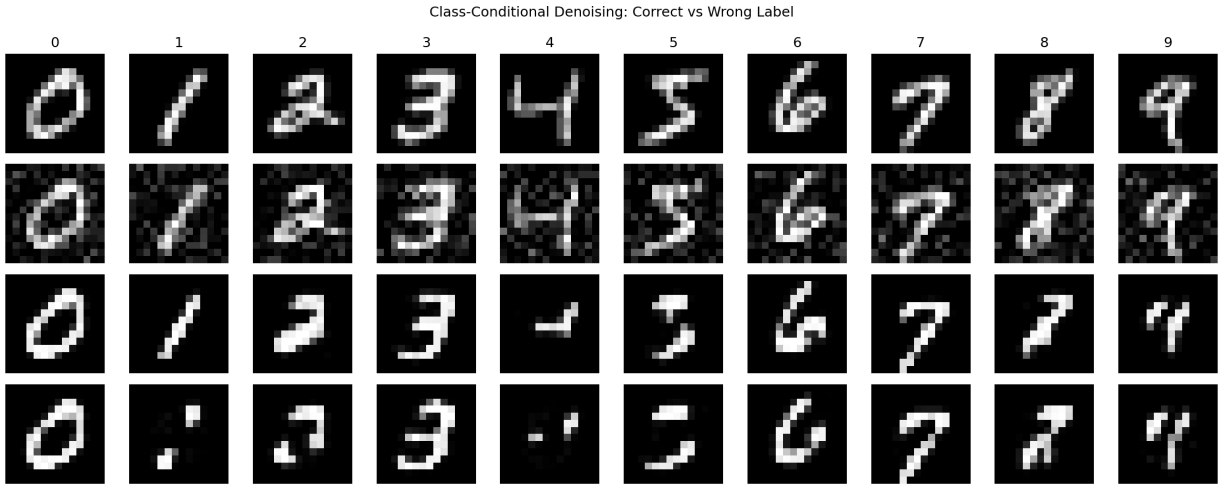

Figure 3: Class-conditional denoising. Row 1: original digits. Row 2: noisy inputs. Row 3: denoised with *correct* class label—accurate reconstruction. Row 4: denoised with *wrong* label (class 0 for all)—digits reshape toward zeros, confirming class-specific attractor selection.

A.5 Recognition and the Complete Pipeline

A key question arises: how does the system know which bias $\mathbf{b}_k$ to load for an unknown input? The answer is that the learned potentials themselves provide classification.

Energy-based recognition. Given input $\mathbf{x}$, evaluate the energy (or gradient magnitude) under each class potential:

$$k^* = \arg\min_k V_k(\mathbf{x}) \quad \text{or equivalently} \quad k^* = \arg\min_k \|\nabla V_k(\mathbf{x})\|^2. \tag{15}$$

The class with lowest energy (or smallest gradient, indicating proximity to equilibrium) is selected. This is precisely *Hopfield associative memory*: the biases $\mathbf{b}_k$ act as stored patterns, and recognition finds the nearest attractor.

Complete inference pipeline:

1. **Input**: Present noisy or partial pattern $\mathbf{x}$ to the T-array.
2. **Recognize**: Evaluate $V_k(\mathbf{x})$ for all k (parallelizable across bias banks). Select $k^* = \arg\min_k V_k(\mathbf{x})$.
3. **Load**: Reconfiguration fabric loads bias bank $\mathbf{b}_{k^*}$.
4. **Relax**: System evolves under V_{k^*}; output converges to clean reconstruction.

In principle, this unifies classification and generation in a single energy-based framework. However, our experiments reveal a limitation: biases trained via denoising score matching achieve only $\sim$37% classification accuracy (vs. 10% random baseline). The potentials overlap significantly—some classes (e.g., "1") have broadly attractive basins that capture inputs from other classes.

This occurs because score matching optimizes *within-class* reconstruction, not *between-class* discrimination. For robust classification, the training objective should include a contrastive term that explicitly separates class energies:

$$L_{\text{contrastive}} = \sum_k \left[V_k(\mathbf{x}_k) - \frac{1}{K-1} \sum_{j \neq k} V_j(\mathbf{x}_k) \right], \tag{16}$$

encouraging the correct class to have lower energy than alternatives. This remains future work; for now, the system excels at *conditional* tasks (generation and denoising given the class label) rather than *recognition*. Figure 5 illustrates the full pipeline, showing that denoising with the true class label (row 4) consistently outperforms denoising with the energy-predicted class (row 3).

A.6 Hardware Interpretation

The class-conditional structure maps directly to hardware: the reconfiguration fabric stores K bias banks $\{\mathbf{b}_0, \ldots, \mathbf{b}_{K-1}\}$. Recognition requires evaluating K energies, which can be done in parallel using K copies of the gradient computation circuit, followed by a winner-take-all selection. Once k^* is determined, the appropriate bank is loaded ($\sim$ns DAC settling), and the T-array relaxes to produce the output. This provides a physical implementation of content-addressable memory with learned attractors.

A.7 Quantum-Classical Correspondence

The Langevin dynamics in d dimensions corresponds to a $(d{+}1)$-dimensional classical field theory, where time t plays the role of an extra spatial dimension. The trajectory probability $P[x(t)] \propto \exp(-S[x])$ with Onsager-Machlup action $S = \int dt\, (\dot{x}{+}\nabla V)^2/4\mu k_{\mathrm{B}}T$ defines this higher-dimensional system. Operating p-bits at threshold (criticality) corresponds to the critical point of this $(d{+}1)$-dimensional theory, where temporal correlations become long-range and information propagates efficiently across the time dimension.

B Derivations

Detailed derivations of gradient estimators, time-reversal likelihoods, and relaxation bounds will be provided in an expanded version.

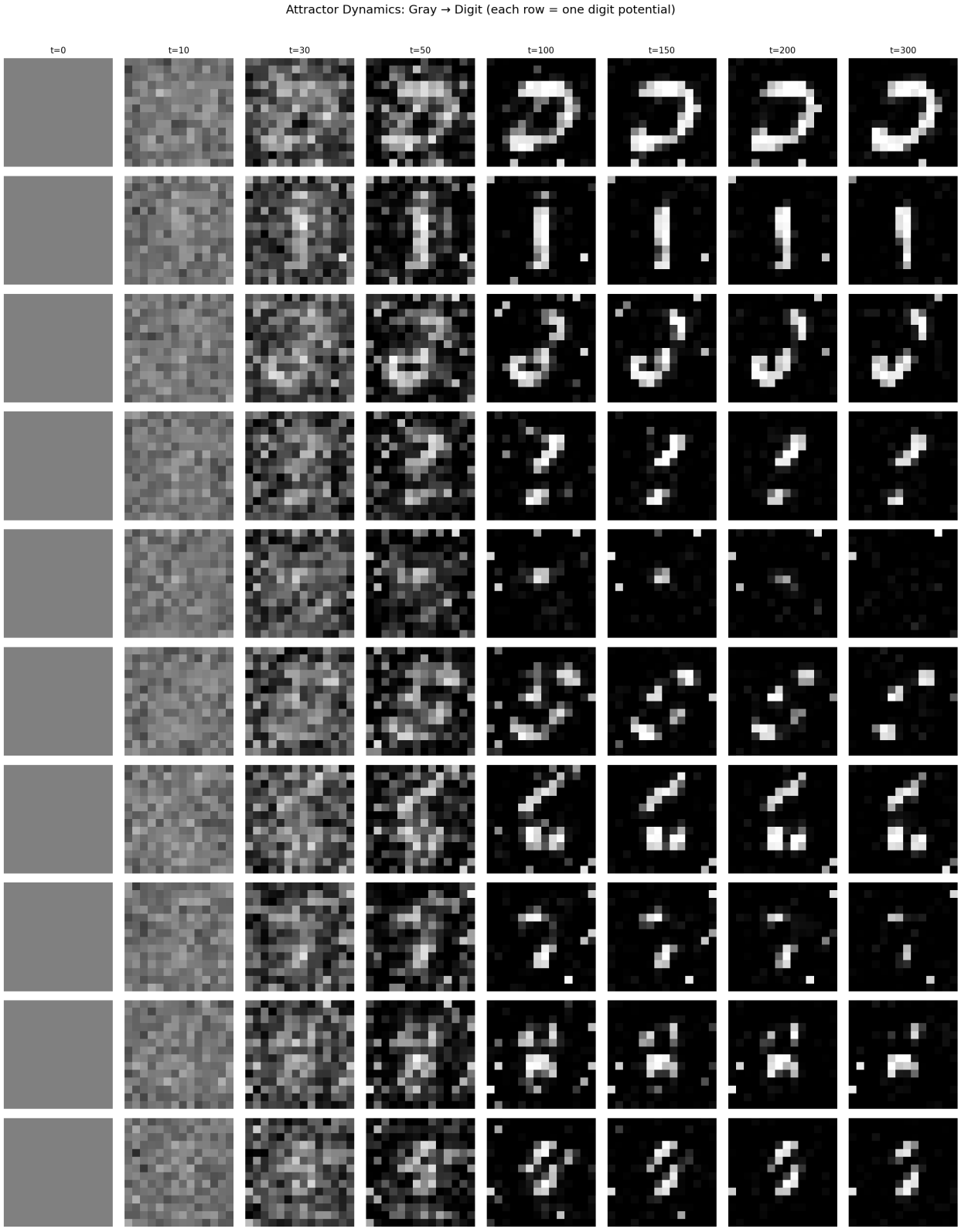

Figure 4: Attractor dynamics: each row shows evolution under a different class potential V_k. Starting from uniform gray (t=0), pixels self-organize into the corresponding digit pattern. This demonstrates that the learned biases create digit-specific energy basins.

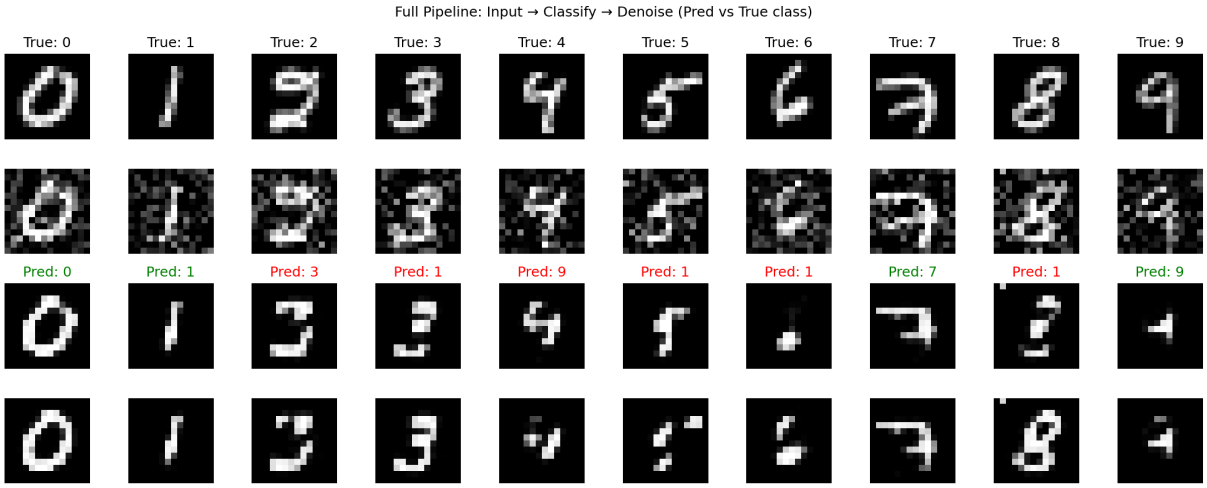

Figure 5: Full inference pipeline. Row 1: clean digits. Row 2: noisy inputs. Row 3: denoised using energy-predicted class k^* (green = correct, red = incorrect prediction). Row 4: denoised using true class label. Classification accuracy is limited, but conditional denoising (row 4) is reliable.