# INSTATRAIN: ADAPTIVE TRAINING VIA ULTRA-FAST NATURAL ANNEALING WITHIN DYNAMICAL SYSTEMS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Time-series modeling is broadly adopted to capture underlying patterns and trends present in historical data, allowing for prediction of future values. However, one crucial aspect in such modeling is often overlooked: in highly dynamic environments, data distributions can shift drastically within a second or less. Under this circumstance, traditional predictive models, even online learning methods struggle to adapt to the ultra-fast and complex distribution shift present in highly dynamic scenarios. To address this, we propose **InstaTrain**, a novel learning paradigm that enables frequent model updates with microsecond-level intervals for real-world prediction tasks, allowing it to keep pace with rapidly evolving data distributions. In this work, (1) We transform the slow and expensive model training process into an ultra-fast natural annealing process that can be carried out on a dynamical system. (2) Leveraging a recently proposed electronic dynamical system, we augment the system with a parameter update module, extending its capabilities to encompass both rapid training and inference. Experimental results across highly dynamic datasets demonstrate that our method delivers on average, a significant $\sim$4,000$\times$ training speedup, $\sim 10^5\times$ reduction in training energy costs, and a remarkable lower test MAE over SOTA methods running on GPUs without / with the online learning mechanism.

## 1 INTRODUCTION

Time-series prediction lies at the heart of artificial intelligence, powering applications ranging from weather forecast (Karevan & Suykens, 2020; Hewage et al., 2020) to product and content recommendation (Kang & McAuley, 2018; Zhang et al., 2021). Current neural network methods have achieved remarkable success by learning the joint distribution between inputs and predictions (Lim & Zohren, 2021; Patton, 2013). However, these methods often implicitly assume that the learned joint distribution remains stable over a considerably long period, an assumption that can easily be violated when the underlying distribution undergoes severe shifts, consequently causing significant failures in pre-trained models. In response to this challenge, the community has pivoted towards more adaptive learning strategies, such as online learning and continual learning approaches (Hoi et al., 2021; Chen et al., 2021; He & Sick, 2021; Prabowo et al., 2023). These methodologies are designed to incrementally adjust model parameters, thereby maintaining alignment with current data trends. Despite their advancements, they struggle to adapt to the circumstances in which data distribution evolves rapidly due to their insufficient adaptation speed. This underscores the pressing need for even more agile and responsive learning mechanisms that can swiftly adapt to shifts in data distribution and ensure model effectiveness.

In the post-Moore's Law era, the limitations of speed improvements in digital processors (such as CPUs and GPUs) have become more pronounced, attracting growing attention in novel computing substrates that harness natural power, a promising yet largely untapped area of research. As a promising candidate, a recently emerged electronic dynamical system (Afoakwa et al., 2021; Sharma et al., 2022) stands out, demonstrating the capability to support ultra-fast computing due to its remarkable low power consumption and exceptionally fast computational speed. Rooted in statistical physics, the behavior of the system is governed by its Hamiltonian (energy function), similar to natural dynamical systems where particles naturally move toward lower energy states. In the electronic dynamical system, lower energy states are rapidly reached through **natural annealing** – the auto-

matic movement of electrons among capacitors seeking equilibrium at "speed of electrons", with minimal power consumption in milliwatt-scale.

However, despite this system having been utilized to accelerate graph learning inference in previous work (Wu et al., 2024), the model training process still relies on traditional digital processors, where the training speed falls short of keeping pace with rapidly evolving data distributions in real-world applications. Consequently, a more advanced learning paradigm is critically needed to fully exploit the potential of the dynamical system that taps into nature's computing power. Since the system specializes in performing natural annealing, we can address the stringent agility demands for ultra-rapid model learning if we can transform the sluggish offline-training process into the natural annealing process. This idea is inspired by the Forward-Forward Algorithm proposed by Hinton (2022), which advocates for conducting both training and inference on the same hardware, similar to the way brains function. This unified approach, known as "mortal computation", is expected to offer significantly lower costs compared to traditional neural networks running on digital hardware.

In response to this opportunity, we propose **InstaTrain**, which extends the extraordinary computational efficiency of the electronic dynamical system from inference to training, addressing the need for capturing rapidly evolving data distributions. The overall framework of this approach is illustrated in Fig. 1, comprising two major components. **(1) Training Algorithm:** Formulated as a dynamical system, our model is determined by the trainable parameters in the Hamiltonian, or energy function. The proposed algorithm accomplishes training through an iterative natural annealing process, which pushes the lowest energy state of the dynamical system to match the ground truth provided by training data. **(2) Hardware Augmentation:** We enhance the dynamical system with parameter update modules to realize a self-training mechanism. This allows both training and inference to be carried out on the same dynamical system, resulting in outstanding computational efficiency and essentially, achieving real-time model adaptation upon highly dynamic data distributions.

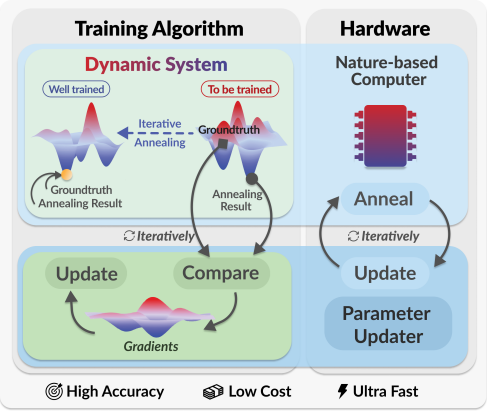


Figure 1: Overview of InstaTrain framework.

The core contributions of this paper can be summarized as follows:

- We propose InstaTrain, a novel learning paradigm that directly responds to the demands for agility and responsiveness in applications with fast evolving distributions.
- We transform the training of a nature-based processor into an iterative natural annealing process within the dynamical system, which enables ultra-fast model training and updating.
- We augment the original nature-based processor, extending its capabilities from fast inference to encompass both rapid training and inference.
- Experimental results across three highly dynamic datasets show that the proposed method with $\sim$1W power delivers a significant $\sim$3,000$\times$ inference speedup, $\sim$4,000$\times$ training speedup, $\sim 10^5\times$ energy cost reduction in training, and a remarkable lower test MAE over SOTA methods running on GPUs without /with dynamic model updating.

## 2 Background

**Dynamical System Model.** A dynamical system is a mathematical model that describes how components (spins) interact and influence each other's states over time, driving the system's evolution toward equilibrium. These systems feature an energy landscape defined by a Hamiltonian function. Dynamical systems have been utilized to accelerate graph learning inference (Wu et al., 2024), with a real-valued Hamiltonian function $\mathcal{H}(\mathbf{s}) = -\sum_{i\neq j}^{N} J_{ij}\sigma_i\sigma_j + \sum_{i}^{N} h_i\sigma_i^2$, where $\sigma_i \in \mathbb{R}$. $\mathbf{s} = \{\sigma_1, \sigma_2, ..., \sigma_N\}$ denotes the spins in the dynamical system governed by the Hamiltonian, $J_{ij}$ represents the relationship between spin $\sigma_i$ and spin $\sigma_j$, and $h_i$ refers to the self-reaction strength, and is forced positive.

This Hamiltonian function is derived from the classic binary Ising Hamiltonian (Cipra, 1987) rooted in ferromagnetism physics, but extends its formulation to overcome the limitations of binary variables (aka spins) restricted to values of $+1$ or $-1$. Specifically, the binary limitation of the Ising model refers to the failure of naively extending its binary nodes to real values. The Hamiltonian of binary Ising model is $\mathcal{H}_b(\sigma) = -\sum_{i\neq j}^{N} J_{ij}\sigma_i\sigma_j - \sum_i^N h_i\sigma_i$. If $\sigma$ are real-valued, they evolve to $\pm\infty$ to pursue the lowest energy state, which is $-\infty$. Even if boundaries are applied to $\sigma$, $\sigma$ are only intercepted along their way to infinity, resulting in polarized nodes and essentially a binary model. In contrast, in the real-valued Hamiltonian function (i.e., $\mathcal{H}$), the quadratic term acts as an energy regulator, which prevents the energy from going down to $-\infty$, allowing spins to be localized at certain values. This extension grants variables the ability to take on real values, thus making it feasible to perform more precise modeling of real-valued systems in real life. In this work, the Hamiltonian supporting real-valued variables $\mathcal{H}$ is employed.

**Dynamical System Embodiment.** The dynamical system is physically embodied as an electronic system (Afoakwa et al., 2021), using electronic components to realize the spontaneous energy decrease (i.e., $d\mathcal{H}/dt \leq 0$). According to Lyapunov stability analysis, the spin dynamics can be designed as $d\sigma_i/dt \propto -\partial\mathcal{H}_{rv}/\partial\sigma_i$:

$$\frac{d\sigma_i}{dt} = \frac{1}{C}\left(\sum_{j\neq i}^{N}(J_{ij} + J_{ji})\sigma_j - 2h_i\sigma_i\right), \tag{1}$$

guaranteeing the system evolves towards the lowest energy state:

$$\frac{d\mathcal{H}}{dt} = \sum_i^N \frac{\partial\mathcal{H}}{\partial\sigma_i}\frac{d\sigma_i}{dt} = -\sum_i^N \frac{1}{C}\left(\frac{\partial\mathcal{H}}{\partial\sigma_i}\right)^2 \leq 0, \tag{2}$$

where the positive constant $C$ is capacitance. Additionally, variables $\sigma$ are modeled as voltages on capacitors, with $J$ and $h$ as conductance of resistors. The spin dynamics indicate that the value of a variable $\sigma_i$ is influenced by input electric currents $(J_{ij} + J_{ji})\sigma_j$ and local current $2h_i\sigma_i$. Through charging or discharging the capacitors at "speed of electrons", the variable value changes.

**Offline Hamiltonian Training.** To train the parameters $J$ and $h$ in $\mathcal{H}$, prior research (Wu et al., 2024) employed a conditional likelihood method on traditional digital processors. This approach focuses on one spin $\sigma_i$ at a time, treating other spins $\sigma_j$ as conditions. An estimated spin value is:

$$\hat{\sigma}_i = \frac{1}{2h_i}\sum_{j\neq i}^{N}\left(J_{ij} + J_{ji}\right)\sigma_j. \tag{3}$$

After estimating each spin's value, their differences from ground truths are evaluated using metrics such as MAE and MSE. By using these metrics as loss functions, the model parameters are optimized to align the ground truth with the system's lowest energy state. Consequently, during inference, the inherent process of spontaneous energy decrease drives the system toward the lowest energy state, producing the desired solution with the highest probability.

# 3 Methodology: InstaTrain

In this section, we present InstaTrain, a novel learning paradigm that leverages the natural annealing process of a dynamical system to enable ultra-fast model training, capturing rapidly evolving data distribution for prediction tasks. We first introduce our Iterative Natural Annealing based Training (INAT) algorithm, including how to formulate the prediction problem using the dynamical system and the detailed training process of the Hamiltonian parameters through iterative natural annealing. Furthermore, we redesign the original the electronic dynamical system, integrating update modules to enable the self-training feature.

## 3.1 Iterative Natural Annealing based Training (INAT)

### 3.1.1 Formulating Prediction via Natural Annealing

Our task of time-series prediction is to learn a function $f_\theta$ that maps the historical variable states $\mathbf{s}^t$ of a system to its future states $\mathbf{s}^{t+1}$, i.e., $\mathbf{s}^{t+1} = f_\theta(\mathbf{s}^t)$. The goal is to optimize parameters $\theta$ such that $f_\theta$ accurately captures the system's evolution over time.

To achieve the goal, we model the described prediction problem using the Hamiltonian of a dynamical system, with $\mathbf{s}^t$ and $\mathbf{s}^{t+1}$ representing the spin configurations of the dynamical system in consecutive time steps. Without loss of generality, we clamp the first $N/2$ spin values to the input state $(\sigma_1, ..., \sigma_{N/2}) = \mathbf{s}^t$, and aim to get the values of the remaining $N/2$ spins $(\sigma_{N/2+1}, ..., \sigma_N)$. If the model's parameters $J$ and $h$ perfectly capture the dependencies between inputs and predictions, the ground truth configuration $\mathbf{s}^* = (\mathbf{s}^t, \mathbf{s}^{t+1})$ corresponds to the lowest energy of the dynamical system. Consequently, by clamping the input spins to $\mathbf{s}^t$ and allowing the remaining spins to evolve according to the Hamiltonian $\mathcal{H}$, the natural annealing process will drive the system to chase equilibrium, resulting in the remaining spins moving towards the desired solution $\mathbf{s}^{t+1}$.

We can further interpret this annealing process using the Boltzmann distribution, which defines a mapping from energy to probability. Specifically, the lowest energy spin configuration corresponds to the maximum probability state through the following:

$$p_{\mathbf{s}^*} = \frac{1}{Z} e^{-\mathcal{H}(\mathbf{s}^*)}, \tag{4}$$

where $Z$ is the partition function defined as $\int e^{-\mathcal{H}} d\sigma$, functioning as a normalizing constant. Therefore, the system's evolution towards the lowest energy state is equivalent to finding the desired prediction $\mathbf{s}^{t+1}$ with the highest probability under the Hamiltonian $\mathcal{H}$. To elucidate more clearly, we visualize the whole process in Fig. 2. Clamping the input $\mathbf{s}^t$ confines the entire energy landscape to a subspace compatible with the given input data. The remaining unclamped spins then undergo natural annealing within this constrained landscape, spontaneously evolving towards the lowest energy state, yielding the desired solution $\mathbf{s}^{t+1}$. Notably, a physical system governed by its energy function $\mathcal{H}$ can spontaneously evolve towards its lowest energy configuration through natural annealing, leveraging the full parallelism of the underlying physical dynamics.

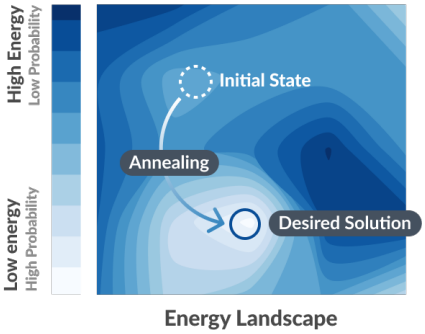

Figure 2: Prediction via annealing

### 3.1.2 Training Through Iterative Natural Annealing

Through the above description, we can perform efficient prediction on the dynamical system given the optimal Hamiltonian parameters $J$ and $h$. In terms of training, instead of undergoing costly training processes on digital processors, it is much more preferable that model training is also available on the dynamical system. To address this, we describe how to obtain the target parameters from the training data through an iterative natural annealing process, the same process used for inference.

Specifically, we seek to maximize the likelihood of the training set under the model:

$$\arg\max_{J,h} \prod_{\mathbf{s} \in T} p_{\mathbf{s}}, \tag{5}$$

where $T$ is the training set constructed as $\mathbf{s} = (\sigma_1, ..., \sigma_N) = \left(\mathbf{s}^t, \mathbf{s}^{t+1}\right)$. This is equivalent to minimizing the negative log-likelihood loss:

$$\arg\min_{J,h} \mathcal{L}\left(\mathbf{s}; J, h\right) = \frac{1}{M} \sum_{\mathbf{s} \in T} \left(ln\left(Z\right) - ln\left(e^{-\mathcal{H}}\right)\right), \tag{6}$$

where $M$ is the number of training samples. Thus, the gradients of $\mathcal{L}$ with respect to $J_{ij}$ are given by

$$\frac{\partial \mathcal{L}(\mathbf{s})}{\partial J_{ij}} = \frac{\partial \ln(Z)}{\partial J_{ij}} + \frac{1}{M} \sum_{\mathbf{s} \in T} \frac{\partial \mathcal{H}}{\partial J_{ij}}, \tag{7}$$

where the two terms are essentially expectations of spin multiplications:

$$\frac{\partial \ln(Z)}{\partial J_{ij}} = \frac{1}{Z} \frac{\partial Z}{\partial J_{ij}} = \frac{\int e^{-\mathcal{H}} \sigma_i \sigma_j \, d\sigma}{\int e^{-\mathcal{H}} \, d\sigma} = \langle \sigma_i \sigma_j \rangle_{\text{model}}, \quad \frac{1}{M} \sum_{\mathbf{s} \in T} \sigma_i \sigma_j = \langle \sigma_i \sigma_j \rangle_{\text{data}}. \tag{8}$$

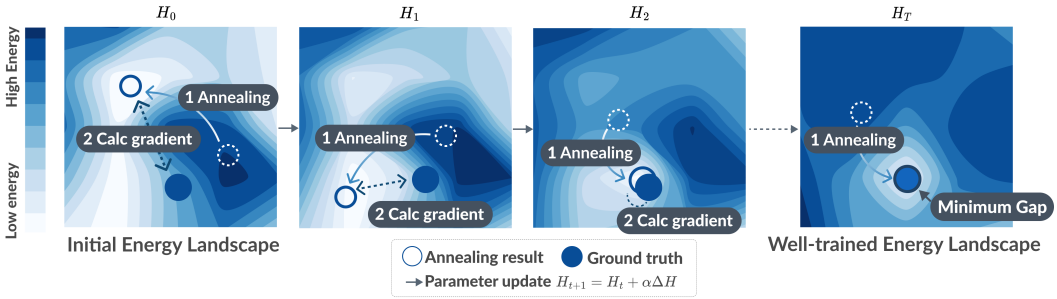


Figure 3: Model training through iterative natural annealing

Particularly, $\langle \sigma_i \sigma_j \rangle_{\text{data}}$ denotes the expectation over the training data, which is tractable, and $\langle \sigma_i \sigma_j \rangle_{\text{model}}$ corresponds to the expectation of $\sigma_i \sigma_j$ given by the current model. Consequently, the gradient for the coupling parameter $J_{ij}$ is

$$\frac{\partial \mathcal{L}(\mathbf{s})}{\partial J_{ij}} = \langle \sigma_i \sigma_j \rangle_{\text{model}} - \langle \sigma_i \sigma_j \rangle_{\text{data}} . \tag{9}$$

In the same way, the gradients of $h_i$ are given by

$$\frac{\partial \mathcal{L}(\mathbf{s})}{\partial h_i} = \frac{\partial ln(Z)}{\partial h_i} + \frac{1}{M} \sum_{\mathbf{s} \in T} \frac{\partial \mathcal{H}}{\partial h_i} = \frac{\partial ln(Z)}{\partial h_i} + \frac{1}{M} \sum_{\mathbf{s} \in T} \sigma_i^2 = -\langle \sigma_i^2 \rangle_{\text{model}} + \langle \sigma_i^2 \rangle_{\text{data}} . \tag{10}$$

Therefore, to update the parameters, we need to calculate $\langle \sigma_i \sigma_j \rangle_{\text{model}}$ and $\langle \sigma_i^2 \rangle_{\text{model}}$. They correspond to the expectation of a large number of states under the current model parameters, which requires computationally expensive sampling over the model distribution. Instead of employing expensive sampling methods to estimate $\langle \cdot \rangle_{\text{model}}$, we leverage the intrinsic dynamics of the electronic system to achieve remarkable efficiency. As described in §3.1.1, we can obtain the current model's prediction $\hat{\mathbf{s}}^{t+1}$ through clamping $\mathbf{s}^t$ to input spins and allowing the dynamical system to perform natural annealing. By measuring the spin configurations at the end of the annealing process, we can directly calculate the required model expectations $\langle \sigma_i \sigma_j \rangle_{\text{model}}$ and $\langle \sigma_i^2 \rangle_{\text{model}}$. In this way, the training process is transformed into an iterative natural annealing process, as described in Algorithm 1 and illustrated in Fig. 3. This innovative training process eliminates the need for computationally expensive sampling techniques or digital offline training. Instead, it harnesses the natural energy decrease feature to perform efficient computations, enabling ultra-fast model training.

To summarize, the outcome of the natural annealing process depends on the accuracy of the current Hamiltonian parameters in capturing the dependencies between the inputs and predictions. When

**Algorithm 1** Iterative Natural Annealing-based Training

**Input:** Training set $T = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M\}$, initial $J^0, h^0$, learning rate $\eta$, and training epochs $N_{\text{iter}}$.
**Output:** Trained Hamiltonian parameters $J, h$.

```
Initialize J ← J^0, h ← h^0.
for i = 1 to N_iter do
    for each s_j = (s_j^t, s_j^{t+1}) in T do
        Clamp the first half spins to s_j^t
        Perform natural annealing to obtain ŝ_j^{t+1}
        Get ⟨σ_i σ_j⟩_model and ⟨σ_i^2⟩_model based on s_j^t, ŝ_j^{t+1}
        Get ⟨σ_i σ_j⟩_data and ⟨σ_i^2⟩_data based on s_j^t, s_j^{t+1}
        Update J_ij ← J_ij − η · (⟨σ_i σ_j⟩_model − ⟨σ_i σ_j⟩_data)
        Update h_i ← h_i − η · (−⟨σ_i^2⟩_model + ⟨σ_i^2⟩_data)
    end for
end for
return J, h
```

the values of some spins are fixed to $\mathbf{s}^t$, two scenarios can occur: (1) If the parameters properly describe the dependencies, the annealing process will converge to the desired solution $\mathbf{s}^{t+1}$, representing the ideal case where the model has successfully learned the correct relationships between the inputs and predictions. (2) If the parameters do not accurately capture these dependencies, the annealing process will instead yield results that align with the current model's expectations, denoted by $\langle \sigma_i \sigma_j \rangle_{\text{model}}$ and $\langle \sigma_i^2 \rangle_{\text{model}}$. This outcome indicates that the model's parameters require further optimization to better represent the underlying dependencies. Regardless of the parameter accuracy, both scenarios correspond to the equilibrium state of the dynamical system.

### 3.2 Hardware Augmentation

The physical realization of this dynamical system is achieved by mapping the spin values to the voltages applied on nano-scale capacitors $C$, and modeling the Hamiltonian parameters $J$ and $h$ as the conductance of resistors. More specifically, referring to Fig. 4, the value of the spin $\sigma_i$ corresponds to the voltage $V_i$, the effective conductance of the coupling between spin $\sigma_i$ and spin $\sigma_j$ is $J_{ij}$ (yellow blocks). The effective conductance of the added variable resistor for spin $\sigma_i$ is $2h_i$, which is embedded in nodes. This mapping enables the construction of the dynamical system using a mesh of programmable resistors, which are interconnected and span across all spins. By exploiting the intrinsic dynamics (Eq. 11) of this resistor-capacitor network, the natural annealing process (energy decrease) can be physically implemented, allowing for rapid convergence towards the equilibrium state that corresponds to the desired solutions, or the model's expectations.

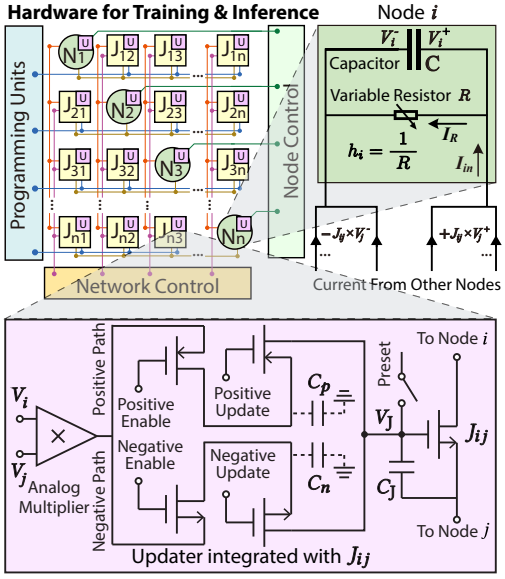


Figure 4: Redesigned *InstaTrain* hardware.

$$C\frac{dV_i}{dt} = -\frac{\partial \mathcal{H}}{\partial V_i} = \sum_{j \neq i} (J_{ij} + J_{ji})V_j - 2h_i V_i = I_{in} - I_R. \tag{11}$$

After the natural annealing process is implemented in this electronic system, we need to further make the system self-trainable. Compared to the original system, update modules need to be implemented. In particular, the update modules take the values of spins $\sigma_i$ and $\sigma_j$ as input, compute $V_i V_j$, and update the voltage $V_J$ applied to the capacitor $C_J$. A programmable parameter $J_{ij}$ is then updated according to the value of $V_J$. As depicted in Fig. 4, we embed the update modules (shown as purple blocks) in coupling units (for updating $J_{ij}$) and nodes (for updating $h_i$). The detailed steps are:

1. Initialize $J_{ij}$ through preset, giving $V_J$ an initial value.
2. Initialize $V_i$ and $V_j$. For an input node, load the ground truth, otherwise initialize it arbitrarily.
3. Start annealing to get the updated $V_i$ and $V_j$.
4. Obtain $\langle V_i V_j \rangle_{\text{model}}$ using the analog multiplier. The result is captured in capacitors as voltages.
5. Load the ground truths of $V_i$ and $V_j$.
6. Obtain $\langle V_i V_j \rangle_{\text{data}}$ as voltage using the analog multiplier.
7. Based on the voltage difference between $\langle V_i V_j \rangle_{\text{model}}$ and $\langle V_i V_j \rangle_{\text{data}}$, the positive path is enabled if the former is larger, otherwise enable the negative path.
8. The subtraction of the two voltages contributes to $V_J$, modifying the voltage to update $J_{ij}$.
9. Repeat steps (2)-(8) for the next epoch.

Through these steps, the entire training process is transformed into an iterative natural annealing process within the dynamical system, enabling ultra-fast training for highly dynamic applications.

# 4 EVALUATION

## 4.1 EXPERIMENTAL SETUP

**Datasets.** We evaluate InstaTrain on five high-frequency datasets, each providing 100 samples per second. *Carbon-Oxide* consists of time series data collected from a gas delivery platform facility, capturing readings from chemical sensors exposed to varying concentrations of carbon oxide and ethylene mixtures (Fonollosa et al., 2015b). Similarly, *Methane* includes data from chemical sensors exposed to mixtures of methane and ethylene at varying concentration levels (Fonollosa et al., 2015b). *Stock* contains sampled stock data of S&P-500 (Nasdaq). *Ammonia* includes time series recordings from a chemical detection platform, featuring data from 72 metal-oxide sensors across six different locations, all maintained under consistent wind speed and operating temperatures (Fonollosa et al., 2015a). *Toluene* comprises time series recordings from 72 sensors at one location, collected under ten varying conditions (two wind speeds and five operating temperatures) from a chemical detection platform (Fonollosa et al., 2015a). The statistics of these dataset are detailed in Table 4 in Appendix.

**Baselines.** We consider three types of baselines for comparison.

- Static Models: SOTA Graph Neural Networks (GNNs), SOTA Transformer-based time series prediction models, NPGL (Wu et al., 2024), and InstaTrain. All models are trained on the first 25% of each dataset and tested on the remaining 75% of each dataset. The GNNs include: GraphWaveNet (Wu et al., 2019), MTGNN (Wu et al., 2020), and MegaCRN (Jiang et al., 2023). The Transformer-based models include: Autoformer (Wu et al., 2021), DLinear (Zeng et al., 2023), iTransformer (Liu et al., 2023a).
- Low-Frequency Dynamic Models: Building upon the pre-trained static models above, the GNNs, Transformer-based models, NPGL, and InstaTrain are updated as new data becomes available, but with a lower update frequency. In particular, the models are updated once after observing 1,000 snapshots, equivalent to 10 seconds in the real world. After each update, the model is tested on the next 1,000 snapshots.
- High-Frequency Dynamic Models: Similar to low-frequency setup but with more frequent updates. The GNNs, Transformer-based models, NPGL, and InstaTrain are updated more frequently—once every 100 snapshots. After each update, the model is tested on the next 100 snapshots. Additionally, we include SOTA online learning models: FSNet (Pham et al., 2022), online-adapted PatchTST (Nie et al., 2022) proposed in (Wen et al., 2024), and OneNet (Wen et al., 2024). These online learning models are implemented based on their default setup for a fair comparison (updated every snapshot).

To more effectively showcase the impact of high-frequency online learning, both low-frequency and high-frequency models are updated using data from the most recent 100 snapshots in the past.

**Platforms.** We evaluate the accuracy and inference latency of SOTA GNNs, Transformer-based models, and online learning models using an NVIDIA A100-40GB GPU. Training latency for NPGL, GNNs, Transformer-based models, and online learning models is also measured on the same GPU. The accuracy and inference latency of NPGL, along with the accuracy, training latency, and inference latency of InstaTrain, are assessed using a CUDA-based Finite Element Analysis (FEA) software simulator implemented based on BRIM (Afoakwa et al., 2021). Furthermore, the Cadence Mixed-Signal Design Environment is employed to evaluate the power consumption and area requirements of InstaTrain.

## 4.2 MAIN RESULTS

**Accuracy Evaluations.** In Table 1, the accuracy results are shown across five datasets with Mean Absolute Error (MAE) as the metric. The results for the dynamic models are averaged across all snapshots in test data. Here, we present the best-performing GNN and Transformer-based model (Best Trans), a complete version of this table is provided in Table A.1 in Appendix. The results demonstrate that InstaTrain outperforms the GNNs and Transformer-based moedls (Trans) in all three cases (static, low-frequency, and high-frequency) across all datasets, with comparable accuracy versus NPGL. The comparison of different update scenarios indicates that high-frequency updates are necessary to achieve better performance. However, due to the sluggish training speed of NPGL,

Table 1: Accuracy comparison across datasets. LF / HF: Low / High Frequency. Online learning methods only have HF. Gray-shaded results indicate "Not Achievable" results due to slow training.

| | Dataset | Carbon-Oxide | Methane | Stock | Ammonia | Toluene |
|---|---|---|---|---|---|---|
| Static | Best GNN | 14.40 | 19.31 | 2.85 | 13.43 | 13.26 |
| | Best Trans | 14.02 | 19.29 | 2.27 | 12.41 | 11.95 |
| | NPGL | 13.90 | 19.22 | 2.01 | 12.15 | 11.43 |
| | InstaTrain | 13.88 | 19.25 | 2.02 | 12.08 | 11.37 |
| LF Update | Best GNN | 10.28 | 11.57 | 1.70 | 4.72 | 5.19 |
| | Best Trans | 8.53 | 9.72 | 1.22 | 3.94 | 4.85 |
| | NPGL | 8.25 | 9.26 | 1.18 | 3.81 | 4.68 |
| | InstaTrain | 8.28 | 9.22 | 1.20 | 3.72 | 4.67 |
| HF Update | Best GNN | 7.16 | 7.36 | 0.80 | 1.62 | 2.11 |
| | Best Trans | 7.12 | 7.25 | 0.73 | 1.45 | 1.94 |
| | FSNet | 7.11 | 7.14 | 0.79 | 1.48 | 2.07 |
| | PatchTST | 7.05 | 7.09 | 0.80 | 1.46 | 2.02 |
| | OneNet | 6.93 | 7.11 | 0.77 | 1.42 | 1.93 |
| | NPGL | 6.81 | 7.08 | 0.68 | 1.39 | 1.90 |
| | InstaTrain | 6.79 | 7.05 | 0.68 | 1.36 | 1.86 |

GNNs, Trans, and even SOTA online-training methods, their high-frequency update results are not achievable, shown as the gray-shaded results in the table. This is because their training latency per snapshot exceeds 0.01 seconds (as shown in Fig. 5), resulting in a cumulative training time of over 1 second for 100 snapshots. Consequently, they cannot keep up with the high-frequency update schedule, making real-time adaption "Not Achievable" for them in this scenario. However, for the sake of comparison, we still calculate the accuracy of these baseline models under the high-frequency setup, assuming they could meet the update schedule. The limitation of baselines makes InstaTrain the ideal choice for achieving high accuracy, especially in the cases where data distributions change rapidly. On average, the high-frequency update result of InstaTrain achieves 75.11% MAE reduction compared to static models, and 50.53% versus low-frequency update models. Despite that the baselines are not qualified to perform high-frequency updates, InstaTrain still achieves 8.42% MAE reduction compared to them.

**Latency Evaluations.** The average training latency per snapshot and inference latency per snapshot are illustrated in Fig. 5 and Fig. 6, respectively, measured in seconds. As depicted in Fig. 5, InstaTrain achieves microsecond-level ($10^{-6}$ seconds) update time per snapshot, significantly outperforming other approaches that require tens of milliseconds. The orange dashed line marks the training latency requirement of 0.1 seconds per snapshot, which is necessary for low-frequency model updates. All models fall below this threshold, enabling them to realize the accuracy improvements associated with transitioning from static to low-frequency online learning. In contrast, the red dashed line indicates the training latency requirement for high-frequency model updates (0.01 seconds per snapshot). Under this criterion, only InstaTrain meets the requirement, while the other models fail to achieve the necessary latency for further accuracy enhancements in high-frequency online learning. In addition, the inference latency in Fig. 6 shows that InstaTrain also benefits from the exceptional speed brought by the dynamical system, resulting in a similar latency with respect to NPGL on the prediction tasks. On average, InstaTrain achieves a ∼4,000× speedup in online learning compared to all baseline models across all tasks, while achieving a ∼3,000× speedup in

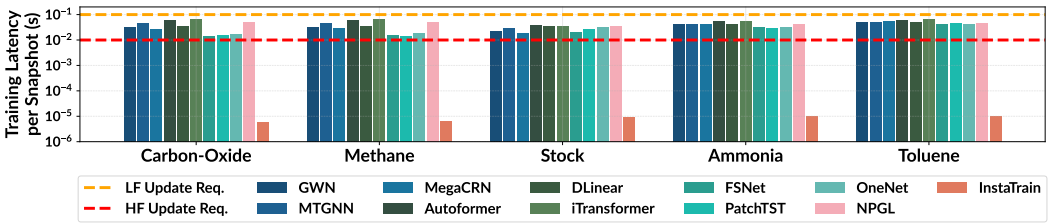


Figure 5: Average training latency per snapshot.

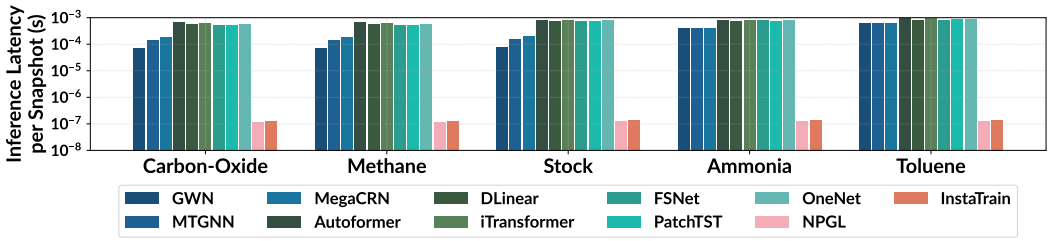


Figure 6: Average inference latency per snapshot.

Table 2: Hardware comparison with related work.

| Design | Power | Area | Real-Value | Ultra-Fast Training |
|---|---|---|---|---|
| **BRIM (Afoakwa et al., 2021)** | 250 mW | 5 $mm^2$ | No | No |
| **NPGL (Wu et al., 2024)** | 260 mW | 5.1 $mm^2$ | Yes | No |
| **InstaTrain** | 950 mW | 9.7 $mm^2$ | **Yes** | **Yes** |

inference compared to the baselines other than NPGL. These results underscore the superior computational capabilities of the dynamical system employed by InstaTrain.

**Hardware Characteristics.** The hardware characteristics of InstaTrain are compared to related work in Table 2. Despite higher power and area resources utilized, the power is still within the scale of 1 Watt, with comparable area with respect to prior work. More importantly, the proposed hardware supports ultra-fast online training that is performed on the same hardware as inference, resulting significantly lower overhead and orders of magnitude training speedup compared to NPGL upgraded from BRIM. Furthermore, the speedup and the low power nature of InstaTrain collectively contribute to a significantly lower energy cost, leading to $\sim 10^5\times$ reduction in energy consumption compared to the SOTA GNNs and online learning approaches on high-end GPUs.

**Ablation Study.** Update frequency is a critical hyperparameter that balances model accuracy and computational efficiency. To investigate its impact, we vary the model update interval from every 50 snapshots to every 1000 snapshots across all datasets. The results, presented in Table 3, demonstrate that generally, higher-frequency online learning achieves better accuracy, as evidenced by lower MAE values. This insight further underscores the significance of InstaTrain, which enables extremely high-frequency online updates.

## 5 Related Work

**Dynamical System Models.** Dynamical systems have gained attention in the machine learning community due to their unique properties and potential for solving complex problems. Drawing inspiration from statistical mechanics in physics, the evolution of a system is based on its energy function. The majority of studies showcasing the potential have been confined to relatively straightforward applications, primarily within the binary domain. For instance, the binary Ising model has been employed to formulate optimization problems (Lucas, 2014), which can be efficiently solved on specialized hardware platforms designed for the model, namely, Ising machines (Mohseni et al., 2022) (Codognet et al., 2022). Additionally, several real-world problems, such as satisfiability (SAT) problems(Sharma et al., 2023a) (Sharma et al., 2023b), traffic congestion prediction (Pan et al., 2023), uplink MIMO detection (Singh et al., 2023) and collaborative filtering (Liu et al.,

Table 3: Accuracy ablation study on update interval.

| Update Interval | Carbon-Oxide | Methane | Stock | Ammonia | Toluene |
|---|---|---|---|---|---|
| 1000 | 8.28 | 9.22 | 1.20 | 3.72 | 4.67 |
| 500 | 7.26 | 8.46 | 0.84 | 2.41 | 3.57 |
| 100 | 6.79 | 7.05 | 0.68 | 1.36 | 1.86 |
| 50 | 6.73 | 7.02 | 0.67 | 1.33 | 1.85 |

2023b), have also been formulated and addressed using the binary Ising model. While they have offered valuable insights into practical problem-solving, their methods come with binary limitations that impede further progress in real-valued applications in the real world.

Although (Wu et al., 2024) extends the original binary Ising model to a real-valued model for real-valued applications, the practical impact of their contributions is limited. Firstly, their acceleration is restricted to inference alone, leaving the primary bottleneck of training unaddressed. Secondly, the benefits of accelerated inference are diminished if the model is static and cannot be promptly updated, especially in highly dynamic applications where patterns evolve rapidly. To summarize, the slow training process hinders online learning and real-time model updates, which are crucial for adapting to fast-changing dynamics, thus limiting the real-world applicability of the proposed real-valued model in many scenarios demanding real-time adaptability and responsiveness.

**Dynamical System Embodiments.** Solving complex combinatorial optimization problems is computationally demanding for conventional von Neumann architectures, as both the required time and hardware resources grow exponentially with the size of the problem. To solve such computationally demanding problems, drawing computing power from nature has become a research direction that attracts attention. Among the various nature-based computing approaches, Ising machines have garnered significant attention due to their ability to efficiently solve optimization problems by leveraging the principles of the Ising model from statistical physics. The Ising model describes the behavior of atoms in natural magnets or spin glasses, which can adopt one of two spin states, up or down, to achieve the lowest energy configuration. This model is applicable to various combinatorial optimization problems (Bian et al., 2010), where finding the ground energy state of the Ising model equates to solving these problems.

In addition to (Afoakwa et al., 2021), various implementations have been proposed and developed, each with its unique characteristics and trade-offs. (1) Quantum-based Ising machines, such as the D-Wave system (Harris et al., 2010), leverage quantum effects, including quantum tunneling, using superconducting qubits. While they offer high computational speed, they require cryogenic environments to enable superconductivity, resulting in high energy consumption and limited practicality. (2) Optical Ising machines (Inagaki et al., 2016; Yamamoto et al., 2017; McMahon et al., 2016) employ optical parametric oscillators to simulate spins and achieve optical coupling. Although they provide significant parallelism, their scalability and stability are limited by their large size and sensitivity to temperature fluctuations. (3) Digital annealers (Yamaoka et al., 2015) are accelerators that perform simulated annealing on digital devices to emulate the continuous annealing process. While they achieve speeds significantly faster than general-purpose processors, their efficiency gains do not yet match those of analog Ising machines due to the substantial difference in hardware operating speed. Among these diverse technologies, the BRIM stands out as particularly promising, offering high-quality solutions rapidly and efficiently. Its operational efficiency, coupled with realistic power consumption and minimal chip area, positions BRIM as a leading candidate in the field of nature-based computing.

## 6 Conclusion

This paper presents InstaTrain, a novel approach to ultra-rapid model learning for prediction tasks. By transforming the training process into an iterative natural annealing process within a dynamical system, our method enables the model to self-evolve and autonomously adapt to the ever-changing correlations between inputs and predictions, addressing the pressing need for agility and responsiveness in highly dynamic applications. The developed parameter update modules augment the original dynamical system used only for inference, extending its capabilities to encompass both rapid training and inference, thereby harnessing the full potential of this innovative computing substrate. This pioneering approach transcends the limitations of conventional methods and paves the way for a new era of ultra-rapid, energy-efficient, and adaptive predictive modeling, empowering applications in domains characterized by high data volatility and stringent latency requirements. Further explorations could focus on incorporating hardware-accommodating advanced online learning strategies into the proposed method, which might yield better solutions. To highlight, InstaTrain achieves, on average, $\sim$4,000$\times$ training speedup supporting microsecond-level model update, $10^5\times$ energy cost reduction in training, as well as a remarkable lower MAE over SOTA methods without / with online learning features.

## REPRODUCIBILITY

The CUDA-based FEA simulator, adapted from (Afoakwa et al., 2021), is a SOTA circuit-level simulation, accurately simulating voltage and current evolution with 100-picosecond time steps. To ensure the reproducibility of this work, this simulator will be open-sourced, facilitating easy verification and future research in the field.

## REFERENCES

Richard Afoakwa, Yiqiao Zhang, Uday Kumar Reddy Vengalam, Zeljko Ignjatovic, and Michael Huang. Brim: Bistable resistively-coupled ising machine. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 749–760. IEEE, 2021.

Zhengbing Bian, Fabian Chudak, William G Macready, and Geordie Rose. The ising model: teaching an old problem new tricks. *D-wave systems*, 2:1–32, 2010.

Xu Chen, Junshan Wang, and Kunqing Xie. Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning. *arXiv preprint arXiv:2106.06273*, 2021.

Barry A Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94(10): 937–959, 1987.

Philippe Codognet, Daniel Diaz, and Salvador Abreu. Quantum and digital annealing for the quadratic assignment problem. In *2022 IEEE International Conference on Quantum Software (QSW)*, pp. 1–8. IEEE, 2022.

Jordi Fonollosa, Irene Rodríguez-Luján, Marco Trincavelli, and Ramón Huerta. Dataset from chemical gas sensor array in turbulent wind tunnel. *Data in Brief*, 3:169–174, 2015a.

Jordi Fonollosa, Sadique Sheik, Ramón Huerta, and Santiago Marco. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical*, 215:618–629, 2015b.

R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S. Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose. Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. *Phys. Rev. B*, 82:024511, Jul 2010. doi: 10.1103/PhysRevB.82.024511.

Yujiang He and Bernhard Sick. Clear: An adaptive continual learning framework for regression tasks. *AI Perspectives*, 3(1):2, 2021.

Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24:16453–16482, 2020.

Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L. McMahon, Takeshi Umeki, Koji Enbutsu, Osamu Tadanaga, Hirokazu Takenouchi, Kazuyuki Aihara, Ken-ichi Kawarabayashi, Kyo Inoue, Shoko Utsunomiya, and Hiroki Takesue. A coherent ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016. ISSN 0036-8075. doi: 10.1126/science.aah4243.

Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 8078–8086, 2023.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, 2018.

Zahra Karevan and Johan AK Suykens. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023a.

Zhuo Liu, Yunan Yang, Zhenyu Pan, Anshujit Sharma, Amit Hasan, Caiwen Ding, Ang Li, Michael Huang, and Tong Geng. Ising-cf: A pathbreaking collaborative filtering method through efficient ising machine learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023b.

Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, 2:5, 2014.

Peter L McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, Robert L Byer, M M Fejer, Hideo Mabuchi, and Yoshihisa Yamamoto. A fully programmable 100-spin coherent ising machine with all-to-all connections. *Science*, 354(6312):614–617, 2016.

Naeimeh Mohseni, Peter L McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.

Nasdaq. Nasdaq data link. `https://data.nasdaq.com`.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Zhenyu Pan, Anshujit Sharma, Jerry Yao-Chieh Hu, Zhuo Liu, Ang Li, Han Liu, Michael Huang, and Tony Geng. Ising-traffic: Using ising machine learning to predict traffic congestion under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 9354–9363, 2023.

Andrew Patton. Copula methods for forecasting multivariate time series. *Handbook of economic forecasting*, 2:899–960, 2013.

Quang Pham, Chenghao Liu, Doyen Sahoo, and Steven CH Hoi. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

Arian Prabowo, Kaixuan Chen, Hao Xue, Subbu Sethuvenkatraman, and Flora D Salim. Continually learning out-of-distribution spatiotemporal data for robust energy forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 3–19. Springer, 2023.

Anshujit Sharma, Richard Afoakwa, Zeljko Ignjatovic, and Michael Huang. Increasing ising machine capacity with multi-chip architectures. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, pp. 508–521, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450386104.

Anshujit Sharma, Matthew Burns, Andrew Hahn, and Michael Huang. Augmenting an electronic ising machine to effectively solve boolean satisfiability. *Scientific Reports*, 13(1):22858, 2023a.

Anshujit Sharma, Matthew Burns, and Michael C Huang. Combining cubic dynamical solvers with make/break heuristics to solve sat. In *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023b.

Abhishek Kumar Singh, Ari Kapelyan, Davide Venturelli, and Kyle Jamieson. Uplink mimo detection using ising machines: A multi-stage ising approach. *arXiv preprint arXiv:2304.12830*, 2023.

Qingsong Wen, Weiqi Chen, Liang Sun, Zhang Zhang, Liang Wang, Rong Jin, Tieniu Tan, et al. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *Advances in Neural Information Processing Systems*, 36, 2024.

Chunshu Wu, Ruibing Song, Chuan Liu, Yunan Yang, Ang Li, Michael Huang, and Tong Geng. Extending power of nature from binary to real-valued graph learning in real world. In *The Twelfth International Conference on Learning Representations*, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling, 2019.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks, 2020.

Yoshihisa Yamamoto, Kazuyuki Aihara, Timothee Leleu, Ken-ichi Kawarabayashi, Satoshi Kako, Martin Fejer, Kyo Inoue, and Hiroki Takesue. Coherent ising machines—optical neural networks operating at the quantum limit. *npj Quantum Information*, 3(1):49, 2017.

Masanao Yamaoka, Chihiro Yoshimura, Masato Hayashi, Takuya Okuyama, Hidetaka Aoki, and Hiroyuki Mizuno. A 20k-spin ising chip to solve combinatorial optimization problems with cmos annealing. *IEEE Journal of Solid-State Circuits*, 51(1):303–309, 2015.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, pp. 11121–11128, 2023.

Qian Zhang, Jie Lu, and Yaochu Jin. Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, 7(1):439–457, 2021.

# A APPENDIX

This appendix provides additional experimental results and discussions related to our work.

## A.1 DATASET STATISTICS AND EXPERIMENTAL RESULTS

Table 4 summarizes the statistics of the datasets used in our experiments. The datasets exhibit a wide range of dimensionalities, spanning from low-dimensional (16) to high-dimensional (720). Each dataset comprises a substantial number of samples, which facilitates a robust evaluation of changes in data distributions. This diversity in dimensionality and sample size ensures that our evaluation effectively addresses varying scales of data complexity and distributional shifts.

Table 4: Dataset statistics.

| Dataset | Carbon-Oxide | Methane | Stock | Ammonia | Toluene |
|---|---|---|---|---|---|
| Number of Samples | 60000 | 60000 | 40060 | 12755 | 12854 |
| Number of Nodes | 16 | 16 | 116 | 432 | 720 |

Additionally, the complete version of Table 1 is provided below for reference.

Table 5: Accuracy comparison across datasets. LF / HF: Low / High Frequency. Online learning methods only have HF. Gray-shaded results indicate "Not Achievable" results due to slow training.

| | Dataset | Carbon-Oxide | Methane | Stock | Ammonia | Toluene |
|---|---|---|---|---|---|---|
| Static | GWN | 14.40 | 19.34 | 3.34 | 19.35 | 13.26 |
| | MTGNN | 24.47 | 19.31 | 2.85 | 13.43 | 18.74 |
| | MegaCRN | 25.94 | 23.65 | 3.45 | 18.12 | 20.15 |
| | Informer | 14.16 | 19.37 | 2.76 | 13.59 | 13.07 |
| | DLinear | 14.08 | 19.32 | 2.31 | 12.74 | 12.82 |
| | iTransformer | 14.02 | 19.29 | 2.27 | 12.41 | 11.95 |
| | NPGL | 13.90 | 19.22 | 2.01 | 12.15 | 11.43 |
| | InstaTrain | 13.88 | 19.25 | 2.02 | 12.08 | 11.37 |
| LF Update | GWN | 10.28 | 11.84 | 1.85 | 4.72 | 5.82 |
| | MTGNN | 12.51 | 11.57 | 1.70 | 4.95 | 5.19 |
| | MegaCRN | 12.34 | 13.49 | 1.87 | 5.41 | 5.93 |
| | Informer | 9.21 | 10.41 | 1.64 | 4.39 | 5.25 |
| | DLinear | 8.82 | 10.25 | 1.39 | 4.16 | 4.96 |
| | iTransformer | 8.53 | 9.72 | 1.22 | 3.94 | 4.85 |
| | NPGL | 8.25 | 9.26 | 1.18 | 3.81 | 4.68 |
| | InstaTrain | 8.28 | 9.22 | 1.20 | 3.72 | 4.67 |
| HF Update | GWN | 7.35 | 7.40 | 0.82 | 1.64 | 2.26 |
| | MTGNN | 7.41 | 7.36 | 0.80 | 1.71 | 2.11 |
| | MegaCRN | 7.16 | 7.45 | 0.86 | 1.62 | 2.18 |
| | Informer | 7.24 | 7.37 | 0.85 | 1.53 | 2.09 |
| | DLinear | 7.12 | 7.25 | 0.81 | 1.50 | 2.15 |
| | iTransformer | 7.16 | 7.28 | 0.73 | 1.45 | 1.94 |
| | FSNet | 7.11 | 7.14 | 0.79 | 1.48 | 2.07 |
| | PatchTST | 7.05 | 7.09 | 0.80 | 1.46 | 2.02 |
| | OneNet | 6.93 | 7.11 | 0.77 | 1.42 | 1.93 |
| | NPGL | 6.81 | 7.08 | 0.68 | 1.39 | 1.90 |
| | InstaTrain | 6.79 | 7.05 | 0.68 | 1.36 | 1.86 |

## A.2 ENERGY EVOLUTION OF THE DYNAMICAL SYSTEM

We present a visualization of the system's energy with respect to annealing time for a sample from the Carbon-Oxide dataset during inference in Fig. 7. The energy curve clearly demonstrates a convergence pattern, with the system's energy rapidly decreasing and stabilizing over time as it approaches equilibrium. Furthermore, we introduce perturbations at an annealing time of $0.6e-7s$

by adding Gaussian noise to the nodes at levels of 10% (blue curve), 20% (green curve), and 30% (red curve). These perturbations further confirm that the system can achieve equilibrium, as it returns to a stable state even when subjected to varying degrees of disturbance.

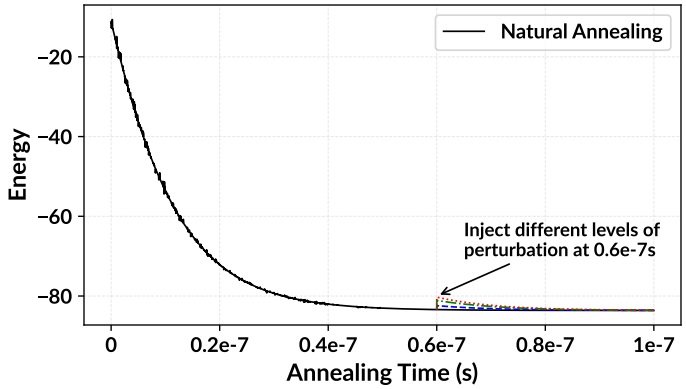


Figure 7: System energy with respect to annealing time.

### A.3 Results on other time series datasets

To provide a more comprehensive evaluation, we include several widely used time series datasets: ETTm1, Electricity, Traffic, and Weather (Wu et al., 2021). We compare the proposed InstaTrain with several SOTA models, including MegaCRN (Jiang et al., 2023) (a SOTA GNN), iTransformer (Liu et al., 2023a) (a SOTA Transformer-based model), NPGL (Wu et al., 2024), three SOTA online learning baselines (FSNet (Pham et al., 2022), the online learning version of PatchTST (Wen et al., 2024), OneNet (Wen et al., 2024)). Given that the highest frequency of these datasets is 10 minutes per sample, all baselines meet the high-frequency update schedule. Therefore, we evaluate these models in two scenarios: the static scenario and the high-frequency (HF) update scenario. The results are shown in the table below.

Table 6: Accuracy comparison (in MAE) across datasets. HF: High Frequency. Online learning methods only have HF.

| | Model | ETTm1 | Electricity | Traffic | Weather |
|---|---|---|---|---|---|
| Static | MegaCRN | 0.416 | 0.420 | 0.397 | 0.301 |
| | iTransformer | 0.372 | 0.379 | 0.329 | 0.246 |
| | NPGL | 0.367 | 0.375 | 0.323 | 0.235 |
| | InstaTrain | 0.369 | 0.375 | 0.320 | 0.231 |
| HF Update | MegaCRN | 0.257 | 0.309 | 0.285 | 0.224 |
| | iTransformer | 0.213 | 0.290 | 0.268 | 0.221 |
| | FSNet | 0.191 | 0.472 | 0.253 | 0.216 |
| | PatchTST | 0.186 | 0.224 | 0.241 | 0.200 |
| | OneNet | 0.187 | 0.254 | 0.243 | 0.201 |
| | NPGL | 0.174 | 0.216 | 0.235 | 0.194 |
| | InstaTrain | 0.172 | 0.214 | 0.231 | 0.196 |

In the static scenario, iTransformer, NPGL, and InstaTrain show competitive performance. Specifically, InstaTrain outperforms NPGL on the Traffic and Weather datasets while maintaining comparable performance on ETTm1 and Electricity datasets. In the high-frequency update scenario, all models show improved performance compared to the static scenario. InstaTrain achieves the lowest MAE across all datasets, outperforming baselines. Notably, FSNet exhibits a higher MAE on the Electricity dataset in the HF scenario, probably due to its lack of robustness with this dataset. The consistent superior performance of InstaTrain in the HF scenario underscores its effectiveness in leveraging high-frequency updates to enhance performance.