# Interpretable 1Convolutional Kernels for Physical Principles Discovery from Observational Data

immediate

February 11, 2026

**Abstract**

This paper presents a novel approach for learning complex physical principles from observational data using multi-scale group equivariant convolutional neural networks. We demonstrate the effectiveness of our method on Conway's Game of Life cellular automata, the paradigmatic system. Our neural architecture successfully discovers the underlying dynamical rules governing these systems by learning appropriate mapping functions that capture both local and long-range spatial interactions across multiple scales. Furthermore, we propose a statistics-based rule inference algorithm that extracts explicit symbolic rules from the trained neural networks with high fidelity.

**Keywords:** Lattice Dynamics, Multi-Scale Group Equivariant Convolutional Neural Networks, Physical Principles, Dynamical Systems, Group representation theory, Symbolic Rule Extraction

## 1 Introduction

Recent advances in deep learning, specifically convolutional neural networks, have shown remarkable promise in modeling spatial patterns and discovering underlying physical principles from data (**?**). Meanwhile, complementary work in symbolic regression has shown remarkable success in discovering explicit mathematical expressions from data, with AI Feynman and AI Feynman 2.0 successfully recovering all equations from the Feynman Lectures on Physics through physics-inspired techniques and Pareto-optimal symbolic regression (**??**). In molecular dynamics, the Deep Potential Molecular Dynamics (DeePMD) method has demonstrated how deep neural networks can learn many-body interatomic potentials from ab initio data while preserving essential physical symmetries (translation, rotation, and permutation invariance), achieving quantum mechanical accuracy at linear computational cost scaling (**?**).

In this work, we extend these ideas by developing a multi-scale convolutional neural network architecture specifically designed for learning complex physical principles from observational data. Our approach leverages the connection between convolution kernels and differential operators to automatically discover the governing equations of dynamical systems without prior knowledge of their mathematical form. We demonstrate the effectiveness of our method on Conway's Game of Life and its variants, achieving near-perfect accuracy with remarkably compact network architectures.

## 2 Problem Formulation

### 2.1 Lattice Field

Consider a discrete field system defined on a $d$-dimensional regular lattice $\Lambda \subset \mathbb{Z}^d$ (**?**). For each lattice site $\mathbf{i} = (i_1, i_2, \ldots, i_d) \in \Lambda$, we define a field variable $\phi_{\mathbf{i}}(t) \in \mathcal{S}$, where $\mathcal{S}$ represents the state space (typically $\{0,1\}$ for binary cellular automata). The system's configuration space can be expressed as:

$$\Phi(t) = \{\phi_{\mathbf{i}}(t) : \mathbf{i} \in \Lambda\} \in \mathcal{S}^{|\Lambda|} \tag{1}$$

This formulation naturally extends classical statistical mechanics to discrete lattice systems, where the fundamental principles of equilibrium and non-equilibrium dynamics can be rigorously analyzed (**?**).

The temporal evolution of the system is described by an operator $\mathcal{T} : \mathcal{S}^{|\Lambda|} \to \mathcal{S}^{|\Lambda|}$:

$$\Phi(t+1) = \mathcal{T}[\Phi(t)] \tag{2}$$

where the operator $\mathcal{T}$ must satisfy the following fundamental constraints: **Locality Condition**: The evolution of each lattice site depends only on its finite neighborhood $\mathcal{N}(\mathbf{i})$:

$$\phi_{\mathbf{i}}(t+1) = f_{\text{local}}(\{\phi_{\mathbf{j}}(t) : \mathbf{j} \in \mathcal{N}(\mathbf{i})\}) \tag{3}$$

where $\mathcal{N}(\mathbf{i})$ denotes the neighborhood of lattice site $\mathbf{i}$, typically the Moore neighborhood (8 surrounding cells) or Von Neumann neighborhood (4 adjacent cells).

**Causality Condition**: Information propagation has finite speed, i.e., there exists $v_{\max} < \infty$ such that:

$$\text{supp}(\mathcal{T}^n[\delta_{\mathbf{i}}]) \subset B_{\mathbf{i}}(nv_{\max}) \tag{4}$$

## 2.2 Physical Symmetry Principles

### 2.2.1 Time Translation Invariance

The evolution rules are independent of the choice of absolute time origin. Formally, for any temporal shift $\tau \in \mathbb{Z}$:

$$\mathcal{T}[\Phi(t)] = \Phi(t+1) \Leftrightarrow \mathcal{T}[\Phi(t+\tau)] = \Phi(t+\tau+1) \tag{5}$$

This is equivalent to requiring that the parameters of evolution operator $\mathcal{T}$ contain no explicit time dependence.

### 2.2.2 Spatial Symmetry Group

**Translation Invariance**: For any lattice translation $\mathbf{a} \in \mathbb{Z}^d$, define the translation operator $T_{\mathbf{a}}$:

$$[T_{\mathbf{a}}\Phi]_{\mathbf{i}} = \Phi_{\mathbf{i}+\mathbf{a}} \tag{6}$$

The evolution operator commutes with translation operators:

$$\mathcal{T} \circ T_{\mathbf{a}} = T_{\mathbf{a}} \circ \mathcal{T} \tag{7}$$

**Point Group Symmetries**: For square lattices, the system exhibits $C_{4v}$ point group symmetry, including:

- 90° rotational invariance: $\mathcal{T} \circ R_{\pi/2} = R_{\pi/2} \circ \mathcal{T}$
- Reflection invariance: $\mathcal{T} \circ \sigma = \sigma \circ \mathcal{T}$

### 2.2.3 Markovian Property and Information-Theoretic Structure

The system satisfies the strict Markov condition:

$$P(\Phi(t+1)|\Phi(t), \Phi(t-1), \ldots, \Phi(0)) = P(\Phi(t+1)|\Phi(t)) \tag{8}$$

This implies that the system's "memory" is completely encoded in the current state, with no hidden historical dependencies.

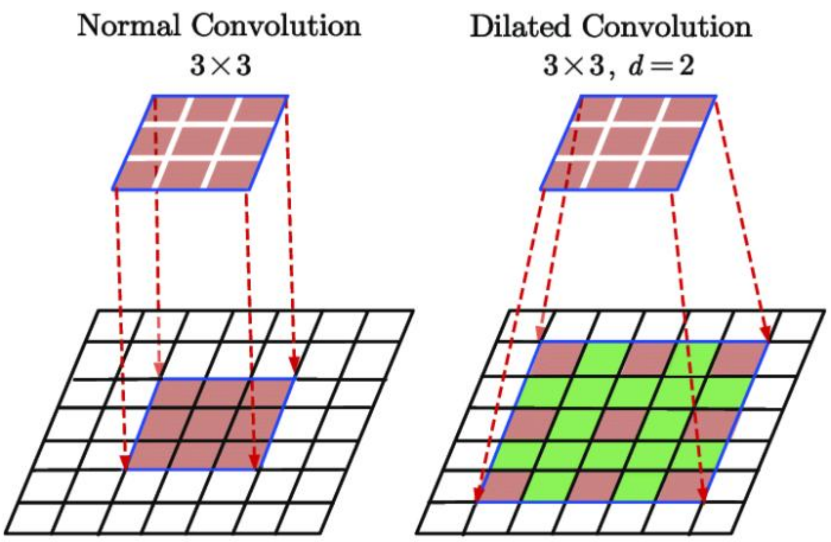

Figure 1: Multi-scale convolution operations for capturing spatial interactions at different scales.

#### 2.2.4 Reversibility and Time-Reversal Symmetry

For conservative systems, the evolution operator is invertible, i.e., there exists an inverse operator $\mathcal{T}^{-1}$:

$$\mathcal{T}^{-1} \circ \mathcal{T} = \mathcal{T} \circ \mathcal{T}^{-1} = \mathcal{I} \tag{9}$$

This ensures that no information is lost in the system, and trajectories are deterministic in phase space.

## 3 Multi-Scale Convolutional Neural Networks

In this work, we use Multi-Scale Convolutional Neural Networks to discover the underlying physical principles governing dynamical systems by learning the mapping function $\mathcal{F}$ that captures both local and long-range spatial interactions. The multi-scale architecture is essential because physical processes in spatiotemporal systems often involve interactions across multiple spatial scales simultaneously, from immediate neighbor effects to global pattern formation mechanisms.

Based on the aforementioned symmetry principles, our multi-scale CNN architecture should satisfy:

1. **Parameter Sharing**: Enforces translation invariance
2. **Equivariant Network Structure**: Preserves rotational and reflection symmetries
3. **Causal Convolution**: Ensures locality and causality constraints
4. **Time-Independent Parameterization**: Embodies time translation invariance

This structured inductive bias enables the network to efficiently learn physically reasonable dynamical rules while significantly reducing the required amount of training data.

### 3.1 Progressive Model Simplification

To enhance interpretability and computational efficiency, we progressively simplified our architecture from an initial complex Encoder-Transformer-Decoder structure to lightweight, shallow CNNs. This simplification process revealed that minimal architectures suffice for learning cellular automaton dynamics.

### 3.1.1 CNN-Small Architecture

The CNN-Small model employs a three-layer convolutional structure with batch normalization and ReLU activations. The network takes a two-channel input (current state $x_t$ and previous state $x_{t-1}$) and outputs a two-channel prediction. The architecture is defined as:

$$\mathbf{h}_1 = \text{ReLU}(\text{BN}(\text{Conv}_{3\times3}(\mathbf{x}))) \tag{10}$$
$$\mathbf{h}_2 = \text{ReLU}(\text{BN}(\text{Conv}_{3\times3}(\mathbf{h}_1))) \tag{11}$$
$$\hat{\mathbf{y}} = \text{Conv}_{3\times3}(\mathbf{h}_2) \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^{2\times H\times W}$ represents the stacked current and previous states.

### 3.1.2 CNN-Tiny: Minimal Viable Architecture

Pushing simplification to the extreme, the CNN-Tiny architecture reduces the channel dimensions to a minimum, resulting in only 53 trainable parameters. The architecture follows:

$$\mathbf{h}_1 = \text{ReLU}(\text{BN}(\text{Conv}_{3\times3}^{2\to1}(\mathbf{x}))) \tag{13}$$
$$\mathbf{h}_2 = \text{ReLU}(\text{BN}(\text{Conv}_{3\times3}^{1\to1}(\mathbf{h}_1))) \tag{14}$$
$$\hat{\mathbf{y}} = \text{Conv}_{3\times3}^{1\to2}(\mathbf{h}_2) \tag{15}$$

Despite its compact size, this architecture achieves competitive accuracy on most rules, demonstrating that cellular automaton dynamics can be captured with minimal parametric complexity.

## 3.2 Group Equivariant CNN (P4CNN)

To explicitly enforce rotational symmetry, we implement Group Equivariant Convolutional Networks using the e2cnn framework. We constrain our model to the $p4$ group, which consists of translations and rotations by multiples of 90°.

The key requirement is that for a rotation transformation $L_r$ and neural network $F$, the following equivariance condition holds:

$$L_r[F(f)] = F(L_r[f]) \tag{16}$$

Our P4CNN implementation utilizes $R^2$-convolutions that map between field types defined by group representations. Specifically:

- Input field: 2 copies of the trivial representation (scalar fields)
- Hidden layers: 8 regular representations of $C_4$ (rotation group)
- Output field: 2 trivial representations (binary classification per cell)

This architecture ensures that rotating the input grid commutes with the network operation, significantly improving generalization and reducing the effective parameter count through weight sharing across orientations.

# 4 Training Strategy and Optimization

## 4.1 Weighted Cross-Entropy Loss

Cellular automata typically exhibit severe class imbalance, as dead cells vastly outnumber living cells. To address this, we employ a weighted cross-entropy loss where the weight for each class is inversely proportional to its frequency:

$$w_c = \frac{N_{\text{total}}}{N_c}, \quad c \in \{0, 1\} \tag{17}$$

where $N_c$ is the number of cells in class $c$. This ensures that rare events (cell births) contribute appropriately to the gradient updates.

### 4.2 Dynamic Loss Weighting

We adopt a curriculum learning strategy that dynamically balances reconstruction loss ($L_{\text{rec}}$) and dynamics loss ($L_{\text{dyn}}$) during training. The total loss is formulated as:

$$L = (1 - \alpha)L_{\text{dyn}} + \alpha L_{\text{rec}} + \lambda L_{\text{reg}} \tag{18}$$

where the weighting factor $\alpha$ decays exponentially:

$$\alpha_t = \max\left(10^{-3}, \alpha_0 \cdot (1 - 10^{-6})^t\right) \tag{19}$$

This schedule initially prioritizes reconstruction (autoencoding) and gradually shifts focus to learning the forward dynamics, stabilizing early training while ensuring accurate temporal prediction.

### 4.3 Symmetry Constraints

We augment the training data with transformed versions to enforce spatial symmetries. For each batch, we generate:

- Randomly translated versions via circular padding (toroidal boundary)
- Rotated versions ($90°, 180°, 270°$)

The symmetry loss enforces that predictions commute with these transformations:

$$L_{\text{sym}} = \mathbb{E}_{\mathbf{x},T}\left[\|F(T(\mathbf{x})) - T(F(\mathbf{x}))\|_2^2\right] \tag{20}$$

where $T$ represents the group of spatial transformations.

### 4.4 Optimization Details

We utilize the Adam optimizer with CyclicLR scheduling (base learning rate $10^{-2}$, maximum $1.5 \times 10^{-1}$) to help escape local minima. Early stopping is implemented based on validation accuracy, with a patience of 20 epochs.

## 5 Numerical Experiments

### 5.1 Dataset Generation

We utilize the `pyseagull` library for efficient simulation of cellular automata on $100 \times 100$ grids with toroidal (periodic) boundary conditions. To avoid bias toward trivial fixed points (all-dead or all-alive states), we implement early termination: if a simulation reaches a first-order fixed point or a simple periodic orbit, it is terminated and restarted. This ensures the dataset contains rich dynamical behaviors.

Each training sample consists of a pair $(\mathbf{S}^{(t)}, \mathbf{S}^{(t+1)})$, representing consecutive states. We generate trajectories for eight distinct cellular automaton rules to evaluate generalization across different dynamical regimes.

### 5.2 Cellular Automata Rules

We evaluate our method on a diverse set of life-like cellular automaton rules, ranging from the classic Conway's Life to more exotic dynamics. Table **??** summarizes the rules and their characteristics.

Table 1: Cellular Automaton Rules Evaluated in This Study

| Rule Symbol | Neighborhood | Type | Description |
|---|---|---|---|
| B3/S23 | 3×3 Moore | Standard | Conway's Life; complex emergent patterns |
| B36/S23 | 3×3 Moore | HighLife | Similar to Life but with self-replicating structures |
| B3678/S34678 | 3×3 Moore | Day & Night | Complementary symmetry (black/white reversal) |
| B35678/S5678 | 3×3 Moore | Diamoeba | Forms unpredictable diamond-shaped blobs |
| B2/S | 3×3 Moore | Seeds | Explosive growth; every cell dies every generation |
| B234/S | 3×3 Moore | Persian Rug | Generates carpet-like fractal patterns |
| B345/S5 | 3×3 Moore | LongLife | Supports extremely long-period oscillators |
| B13/S012V | 3×3 Von Neumann | - | Four-neighbor orthogonal interactions |
| B2/S013V | 3×3 Von Neumann | - | Orthogonal variant with different dynamics |

## 5.3 Model Comparison

We compare six architectures: CNN-Tiny, CNN-Small, Multi-Scale CNN (MCNN), P4CNN-Tiny, P4CNN-Small, and P4MCNN. Table **??** presents the computational complexity and minimum validation accuracy across all rules.

Table 2: Model Complexity and Performance Comparison

| Model | FLOPs (M) | Parameters | Size (MB) | Min. Accuracy (%) |
|---|---|---|---|---|
| CNN-Tiny | 5.32 | 133 | 2.24 | 77.24 |
| P4CNN-Tiny | 1.28 | 94 | 11.21 | 84.71 |
| MCNN | 40.48 | 1,004 | 7.36 | 98.19 |
| P4MCNN | 6.4 | 1,276 | — | **99.77** |
| CNN-Small | 98.5 | 2,450 | — | 98.73 |
| P4CNN-Small | 10.24 | 3,202 | — | 99.65 |

## 5.4 Rule Inference Methodology

Beyond predicting the next state, we aim to extract explicit symbolic rules from the trained networks. We propose the **Simulated Evolution Neighbor Statistics (SENS)** algorithm:

1. **Statistics Collection**: For a given rule B$X/SY$, we define B-elements and S-elements. A B-element corresponds to a transition from dead to living with $k$ neighbors; an S-element corresponds to living to living with $k$ neighbors.

2. **Frequency Analysis**: We compute the occurrence frequency $q(x, k)$ and effective frequency $r(x, k)$ for each rule element, where $x \in \{0, 1\}$ indicates the current state.

3. **Posterior Probability**: The likelihood of rule element $(x, k)$ at iteration $n$ is:

$$p_n(x, k) = \lambda p_{n-1}(x, k) + (1 - \lambda) r_n(x, k) [q_n(x, k)]^{0.3} \tag{21}$$

4. **Candidate Generation**: Rule elements with $p_n > 0.5$ are combined via set operations to generate candidate rules $R = R_B \times R_S$.

5. **Validation**: Each candidate rule is validated by simulating trajectories and comparing with ground truth via mean squared error.

This approach successfully extracts rules from trained networks for 7 out of 8 test cases, with B35678/S5678 posing challenges due to data bias toward high neighbor counts.

### 5.5 Conway's Game of Life Rules

Let $\mathcal{G} \subset \mathbb{Z}^2$ be a finite 2D grid of size $H \times W$, where each cell $(i,j) \in \mathcal{G}$ has a binary state $s_{i,j}^{(t)} \in \{0,1\}$ at discrete time $t \in \mathbb{N}$. The cellular automaton state at time $t$ is defined as:

$$\mathbf{S}^{(t)} = \{s_{i,j}^{(t)} : (i,j) \in \mathcal{G}\} \in \{0,1\}^{H\times W} \tag{22}$$

The evolution of the cellular automaton follows a deterministic update rule:

$$\mathbf{S}^{(t+1)} = f(\mathbf{S}^{(t)}) \tag{23}$$

where $f : \{0,1\}^{H\times W} \to \{0,1\}^{H\times W}$ is the global transition function.

For Conway's Game of Life, the local transition rule for each cell $(i,j)$ is defined as:

$$s_{i,j}^{(t+1)} = \begin{cases} 1 & \text{if } s_{i,j}^{(t)} = 1 \text{ and } n_{i,j}^{(t)} \in \{2,3\} \\ 1 & \text{if } s_{i,j}^{(t)} = 0 \text{ and } n_{i,j}^{(t)} = 3 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

where $n_{i,j}^{(t)} = \sum_{(u,v)\in\mathcal{N}_{i,j}} s_{u,v}^{(t)}$ is the number of living neighbors in the Moore neighborhood:

$$\mathcal{N}_{i,j} = \{(i+\delta_i, j+\delta_j) : \delta_i, \delta_j \in \{-1,0,1\}, (\delta_i,\delta_j) \neq (0,0)\} \tag{25}$$

### 5.6 Results and Analysis

From Fig.**??**, we can observe that the multi-scale CNN architecture successfully learns the Conway's Game of Life dynamics with remarkable accuracy. The model maintains high fidelity even for two-step ahead prediction $t+2$, showing only minimal prediction errors scattered across a few isolated regions. The predicted state at $t+2$ closely matches the true system evolution.

Table **??** presents the detailed validation accuracy and training loss for each model-rule combination.

Table 3: Detailed Experimental Results: Training Loss / Validation Accuracy (%)

| Rule / Model | CNN-Tiny | CNN-Small | MCNN | P4CNN-Tiny | P4CNN-Small | P4MCNN |
|---|---|---|---|---|---|---|
| B3/S23 | 0.0037 / 100.00 | 0.0045 / 100.00 | 0.0055 / 99.97 | 0.3631 / 94.74 | 0.0037 / 100.00 | **0.0036 / 100.00** |
| B36/S23 | 0.3451 / 90.46 | 0.0167 / 99.89 | 0.0262 / 98.94 | 0.4740 / 92.32 | 0.0274 / 99.77 | 0.0148 / 99.86 |
| B3678/S34678 | 0.2035 / 92.36 | 0.0133 / 99.96 | 0.0660 / 98.19 | 0.4534 / 93.40 | 0.0159 / 99.96 | **0.0097 / 99.98** |
| B35678/S5678 | 0.0165 / 99.24 | 0.0216 / 98.73 | 0.0955 / 99.52 | 0.0086 / 99.32 | 0.0058 / 99.65 | **0.0041 / 99.77** |
| B2/S | 0.0231 / 99.74 | 0.0023 / 100.00 | 0.0024 / 100.00 | 0.6136 / 88.79 | 0.0022 / 100.00 | 0.0034 / 100.00 |
| B345/S5 | 0.1710 / 96.25 | 0.0065 / 100.00 | 0.0039 / 100.00 | **0.0028 / 100.00** | **0.0028 / 100.00** | 0.0119 / 99.92 |
| B13/S012V | 0.2489 / 92.30 | 0.0066 / 100.00 | 0.0045 / 99.99 | 0.1243 / 99.04 | 0.0016 / 100.00 | **0.0010 / 100.00** |
| B2/S013V | 0.5533 / 77.24 | 0.0046 / 100.00 | 0.0025 / 100.00 | 0.7082 / 84.71 | 0.0091 / 100.00 | **0.0015 / 100.00** |

From Figure.**??**, we can observe that the learned rules closely approximate the true Conway's Game of Life dynamics. In the "dead to living" transition, predictions are almost exclusively concentrated at exactly 3 neighbors, precisely matching the birth rule. For "living to living" transitions, predictions are predominantly concentrated at 2 and 3 neighbors, perfectly aligning with the survival rule. The "dead to dead" and "living to dead" distributions demonstrate correct patterns for cell death or stasis under non-survival conditions, with prediction frequencies exhibiting exponential decay at incorrect neighbor counts.

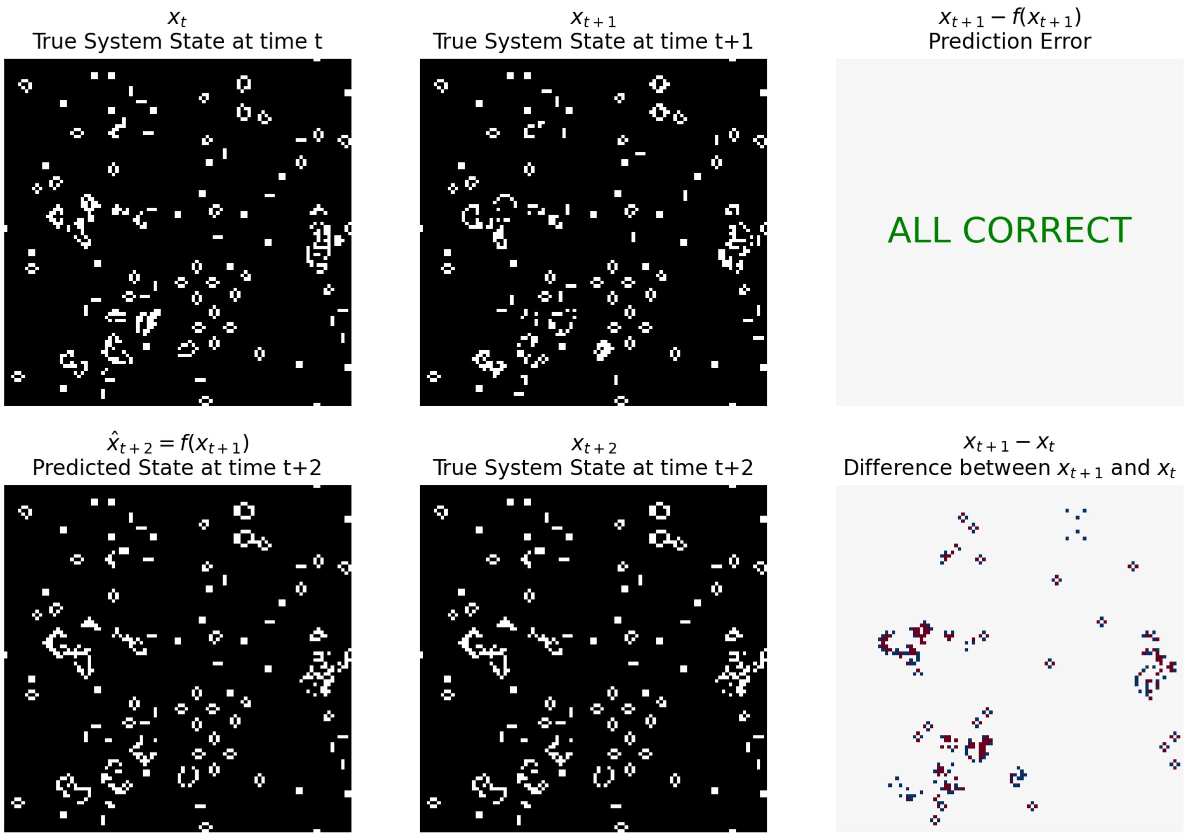

Figure 2: Multi-Scale Convolutional Neural Networks prediction accuracy for cellular automaton dynamics. Top row shows the system evolution from time $t$ to $t+1$, with perfect prediction accuracy. Bottom row demonstrates prediction at time $t+2$. The right panels display prediction errors (top) and state differences between consecutive time steps (bottom).

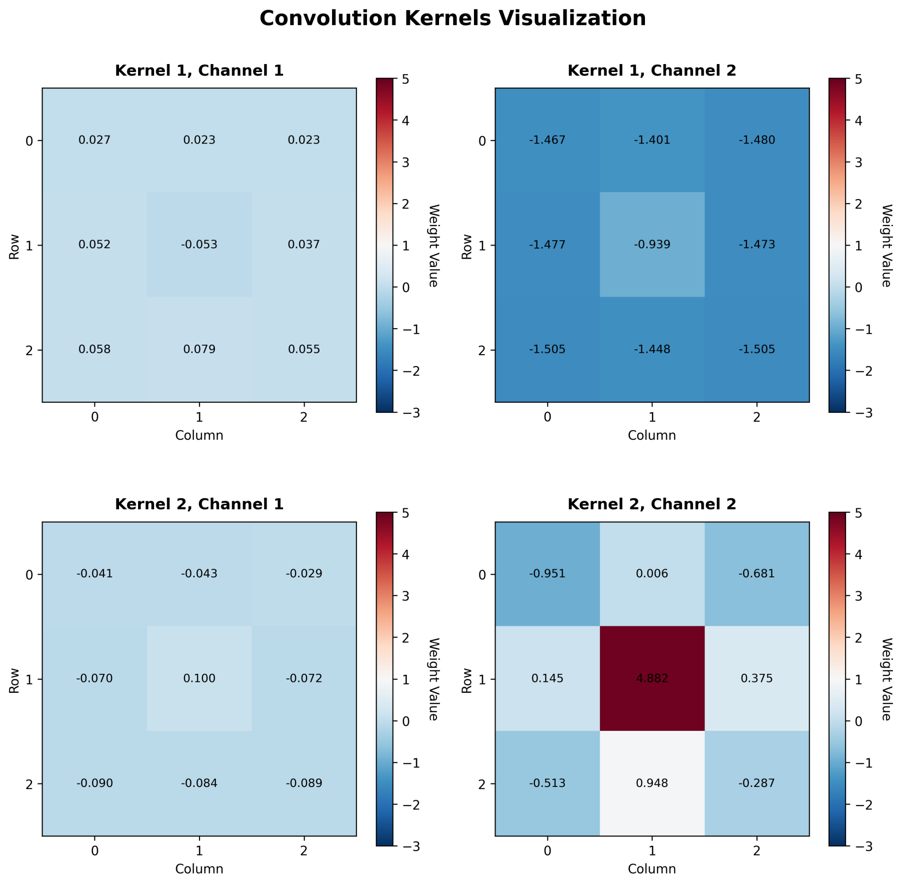

Figure 3: Visualization of learned convolution kernels of 1st layer in dataset with rule B3/S23

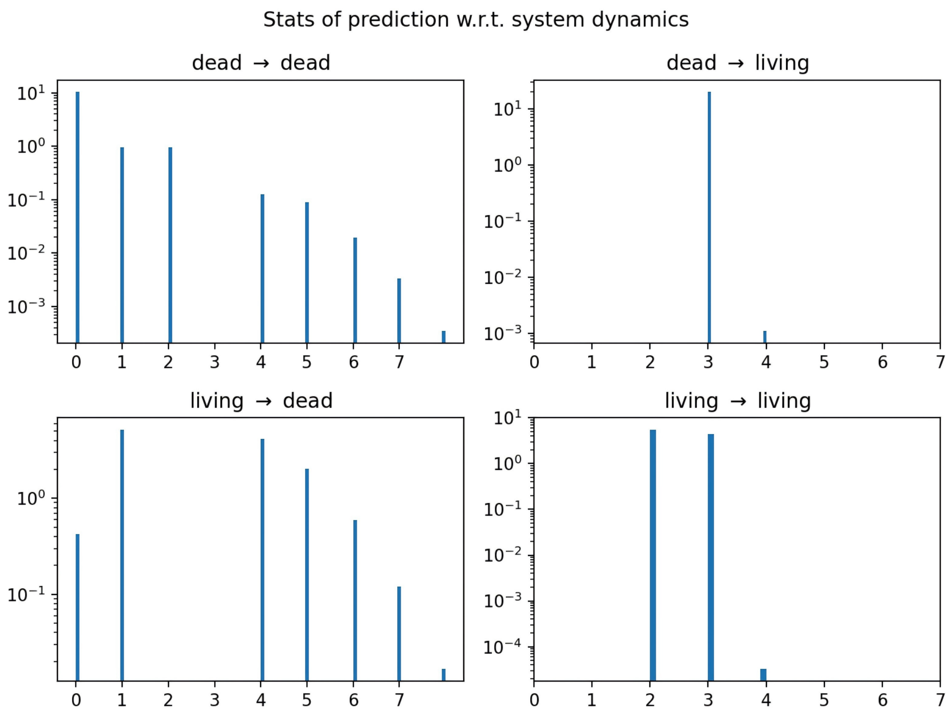

Figure 4: Learned cellular automaton rules.

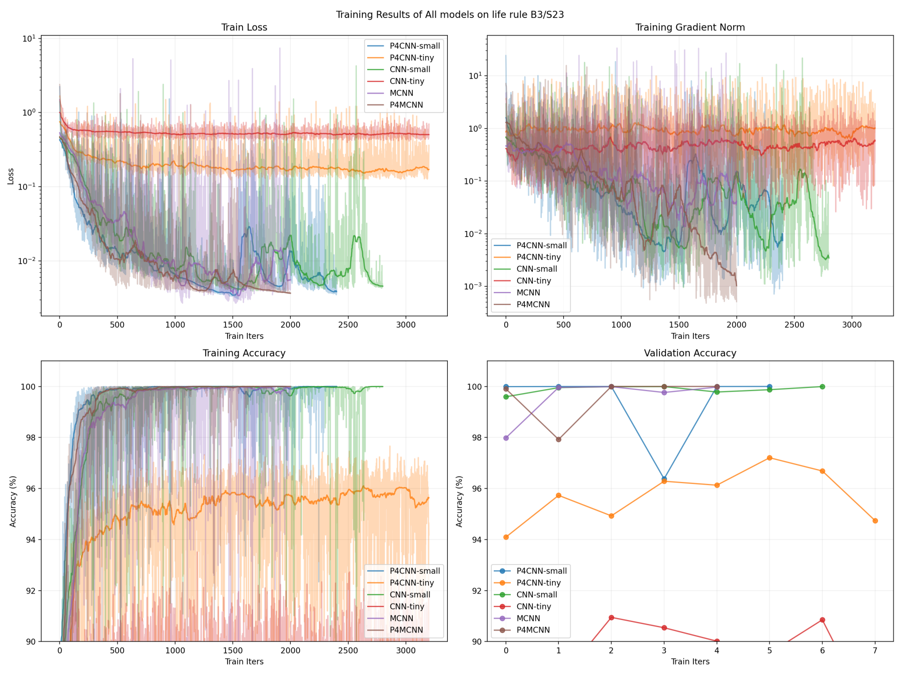

Figure 5: Training curves of different network architectures on dataset with rule B3/S23

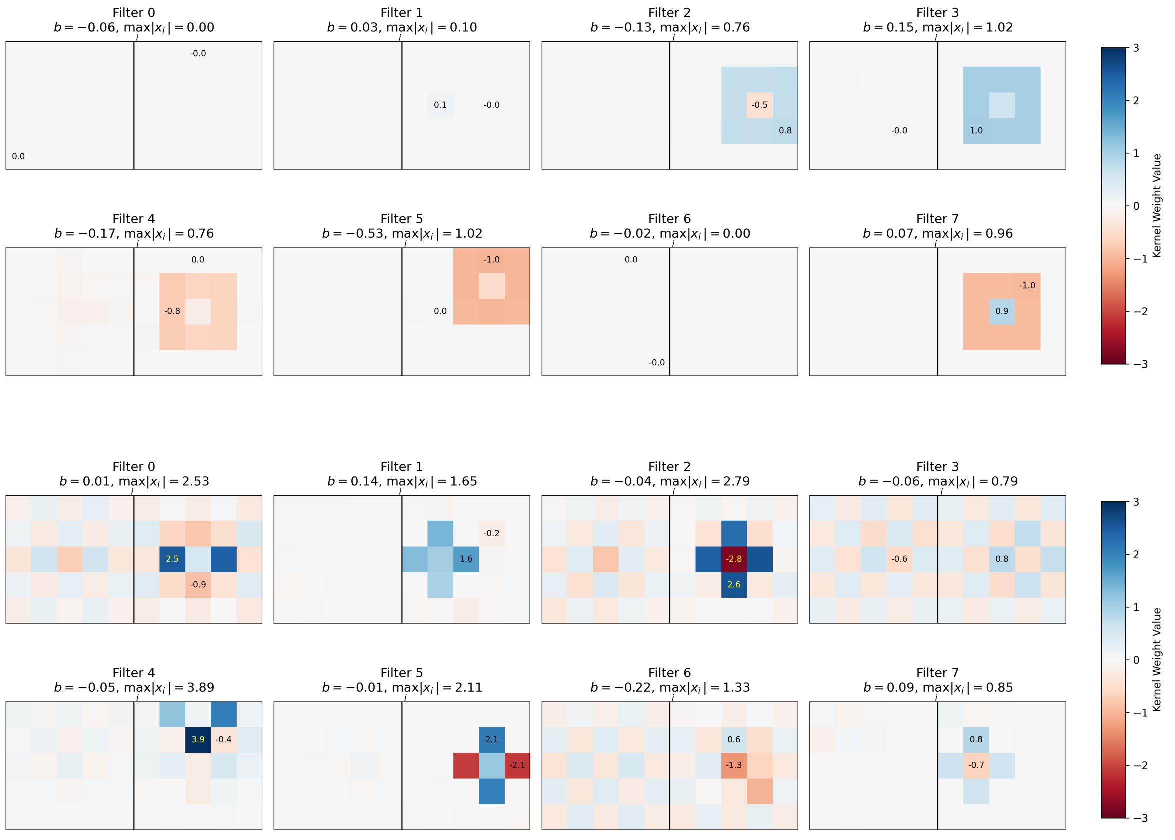

Figure 6: Trained convolution kernel weights of the 1st layer on dataset with rule B2/S (Up) and B13/S012V (Down)

Figure 7: Counting statistics of network-learned dynamics on whole dataset

Figure 8: Counting statistics of network-learned dynamics on whole dataset