Proceedings of Machine Learning Research vol vvv:1–17, 2024

Learning Locally Interacting Discrete Dynamical Systems: Towards Data-Efficient and Scalable Prediction

Beomseok Kang BEOMSEOK@GATECH.EDU
Harshit Kumar HKUMAR64@GATECH.EDU
Minah Lee MINAH.LEE@GATECH.EDU
Biswadeep Chakraborty BISWADEEP@GATECH.EDU
Saibal Mukhopadhyay SMUKHOPADHYAY6@GATECH.EDU
School of Electrical and Computer Engineering, Georgia Institute of Technology, GA, USA

Abstract

Locally interacting dynamical systems, such as epidemic spread, rumor propagation through crowd, and forest fire, exhibit complex global dynamics originated from local, relatively simple, and often stochastic interactions between dynamic elements. Their temporal evolution is often driven by transitions between a finite number of discrete states. Despite significant advancements in predictive modeling through deep learning, such interactions among many elements have rarely explored as a specific domain for predictive modeling. We present **A**ttentive **R**ecurrent **N**eural **C**ellular **A**utomata (AR-NCA), to effectively discover unknown local state transition rules by associating the temporal information between neighboring cells in a permutation-invariant manner. AR-NCA exhibits the superior generalizability across various system configurations (i.e., spatial distribution of states), data efficiency and robustness in extremely data-limited scenarios even in the presence of stochastic interactions, and scalability through spatial dimension-independent prediction.

Keywords: Local Interaction, Discrete Dynamical System, Neural Cellular Automata

1. Dataset

Forest Fire model Forest Fire model is a well-known example of self-organizing complex system. We consider the model as a multi-agent system where four different types of agents (i.e., empty, tree, ember, and fire) are locally interacting. Our forest fire implementation is based on NetLogo, which is a widely used agent-based modeling platform (Tisue and Wilensky, 2004). We modify the interaction rule to be non-trivial based on the prior works (Rothermel, 1972; Chen et al., 1990).

There are few parameters that primarily control the evolution of fire in the model; q_{seed}, $q_{\text{threshold}}$, q_{die}, and q_{transfer}. The parameters are closely related to the agent's internal state referred to as a "heat" value (q). q_{seed} is the heat value of initial fire seeds. $q_{\text{threshold}}$ is a threshold value that defines the transition from tree agents to fire agents. q_{transfer} controls the efficiency of heat accumulation for tree agents from neighboring fire and ember agents (i.e., $q_{(i,j)}(t+1) = q_{(i,j)}(t) + q_{\text{transfer}} \sum_{S_{(i,j)}} q_s(t)$) where $S_{(i,j)}$ is a set of neighboring agents from location (i, j) and s is an element of the set. After the heat interaction between agents are executed, q_{die} defines the speed of cooling down of fire and ember agents (i.e., $q(t+1) = q(t) - q_{\text{die}}$). Once fire agents experience the decrease in the heat value, it darkens its color. It takes 6 frames to become ember from fire and 12 frames to no longer transfer heat to neighboring agents.

The forest fire model evolves following the sequence: **1)** Create 64×64 agent map filled with randomly distributed tree and empty agents. **2)** Define 3 fire seeds with q_{seed} at random locations.

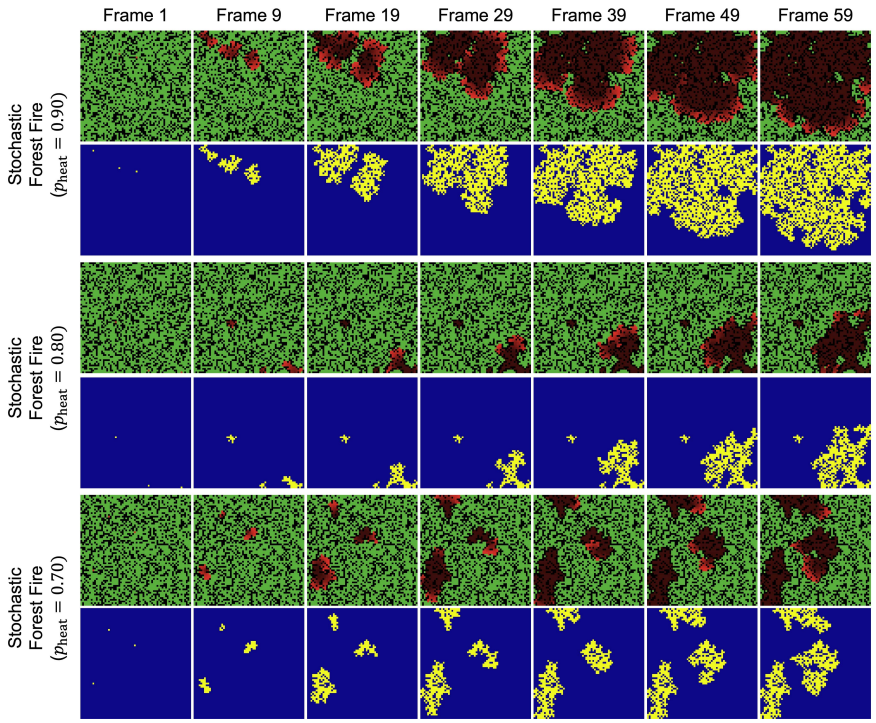

Figure 1: Stochastic forest fire with various p_{heat} values. First row is RGB-based observation, which is given to the prediction networks as input (black: empty, green: tree, brown: ember, and red: fire). The next row is the corresponding binary ground truth (fire or ember: 1, others:0)

3) Each tree agent accumulates the heat from neighboring fire and ember agents using the equation related to q_{transfer}. **4)** Check if the current heat value exceeds $q_{\text{threshold}}$. **5)** Decrease the heat values of fire and ember agents using the equation related to q_{die}. Repeat from **3)** to **5)**.

For the parameter values, we set $q_{\text{seed}} = 6$, $q_{\text{threshold}} = 3$, $q_{\text{die}} = 1$, and $q_{\text{transfer}} = 0.3$. In stochastic forest fire, we introduce another variable p_{heat}, indicating the probability of accumulating the heat from each of neighboring agents. It is applied to the equation related to q_{transfer} within a Bernoulli random variable (i.e., $q_{(i,j)}(t+1) = q_{(i,j)}(t) + q_{\text{transfer}} \sum_{S_{(i,j)}} \text{Bernoulli}(p_{heat}) q_s(t)$). Our goal is to predict fire and ember agents in the models with various p_{heat} values. Figure 1 shows the example scenes of the system with the three different p_{heat} values. In deterministic forest fire, We design the three different forest configuration; dense forest, sparse forest, and gaussian forest. The dense forest is initially filled with 65% tree agents and 35% empty agents. The sparse forest is initially filled with 60% tree agents and 40% empty agents. The spatial distribution of tree agents in dense and sparse forest follow random uniform. We observe that lower than 60% makes the fire propagation vanishes too early and higher than 65% makes the fire front too trivial (e.g., radial diffusion). For gaussian forest, we first create dense forest and then remove tree agents following two-dimensional gaussian distribution with random mean from (-1,-1) to (1,1) and random variance

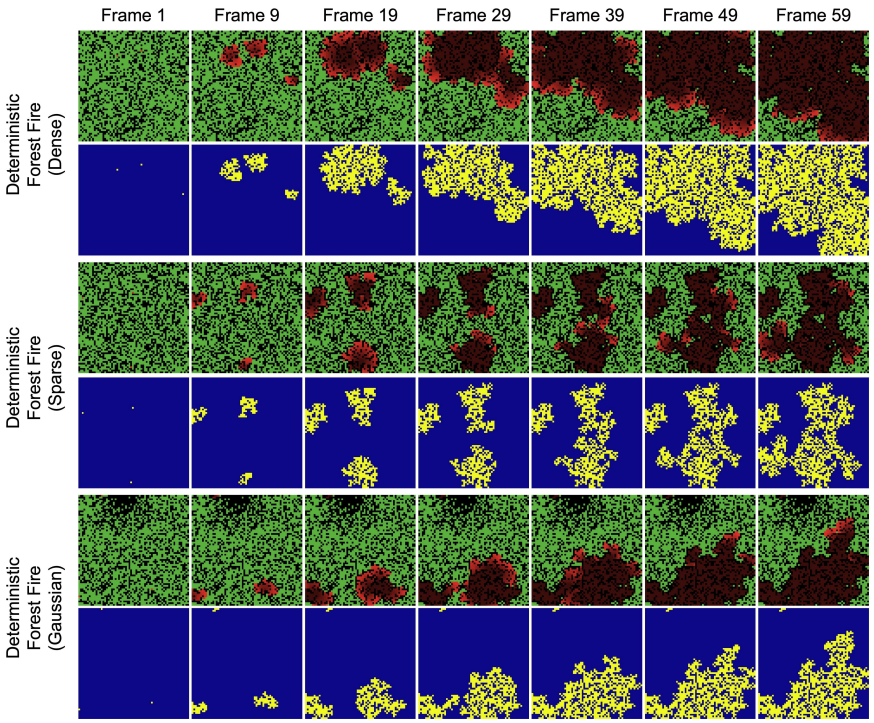

Figure 2: Deterministic forest fire with different forest configurations.

$\in [0.05, 0.1]$ assuming the forest coordinate is normalized to (-1,1). Figure 2 shows the three forest configurations. We generate 300 test chunks for the dense forest and 100 test chunks for the others.

Host-Pathogen model Host-pathogen model is originated from the theoretical understanding of the interaction between viruses and host organisms in a population level. It can be also considered a two-dimensional version of prey-predator model, where two types of agents in an adversarial relationship compete each other. We use the prior implementation developed by PyCX (Sayama, 2013), which is a python-based complex system simulator. Our host-pathogen model has four different types of agents (i.e., empty, dead, healthy, and infected) with local and stochastic interactions.

The host-pathogen model evolves following the sequence: **1)** Create 64×64 agent map filled with randomly distributed infected (1%), healthy (75%), and empty (the rest) agents. **2)** For dead agents, cured by each of neighboring healthy agents with the probability p_{cure}. **4)** For healthy agents, infected by each of neighboring infected agents with probability p_{infect}. **5)** For infected agents, transitioned to dead agents. Repeat from **2)** to **5)**.

We observe that if the agent map is full of healthy agents, the evolution of systems become trivial (radial diffusion) depending on the probability. To avoid such scenarios, we introduce additional empty agents which are not interacting with any other agents (i.e., inactive cells). We fix $p_{\text{infect}} = 0.85$, which is the default value in the implementation, and change p_{cure}. Our goal is to predict healthy agents in the models with various p_{cure} values. Figure 3 shows the example scenes of the system with the three different p_{cure} values.

Stock Market model The complexity of investor's behavior in the stock market has been studied through cellular automata in the prior work (Wei et al., 2003). They aimed to model the "imita-

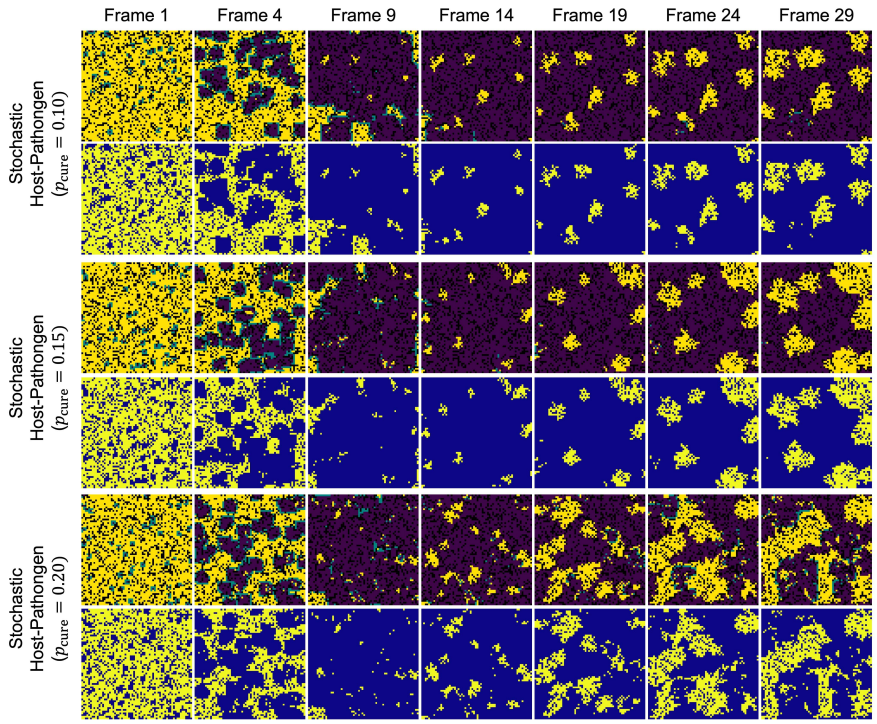

Figure 3: Stochastic host-pathogen model with various p_{cure} values. First row is RGB-based observation (green: infected, yellow: healthy, purple: dead, and black: empty). The next row is the corresponding binary ground truth (healthy: 1, others:0)

tion" behavior of investors affected by neighboring investors, which often makes the stock market unstable and complex. Reversely, we aim to predict this system assuming the unknown stochastic interaction between investors (i.e., agents). There are four different types of actions in the agent (i.e., hold, sell, buy, and inactive). The inactive action is defined by ourselves to prohibit too repeated buying actions. Table 1 shows the probabilistic transition matrix in the positive market scenario, which is already given in the paper (Wei et al., 2003).

The stock market model evolves following the sequence: **1)** Create 64×64 agent map filled with randomly chosen actions (e.g., buy, sell, hold). **2)** Each of agents makes the stochastic action based on neighboring agents. **3)** The agents made two consecutive buying actions become inactive. **4)** The inactive agents in the previous sequence do the buying action. Repeat from **2)** to **4)**.

There is two variables p_{invest} and M to control the system. We set $M = 0.05$ (i.e., the market is positive) and try various p_{invest} values. We introduce the inactive state to make the system more dynamic. Note, the basic implementation without the inactive state can quickly make the system static (no more evolution) with clustered agents. Our goal is to predict agents with the buying action in the models with various p_{invest} values. Figure 4 shows the example scenes of the system with the three different p_{invest} values.

Table 1: Transition matrix in the stock market model. "Neighbor" in the first row indicates the most used action by neighboring agents.

Neighbor	Next Action		
	Buying	Holding	Selling
Buying	$p_{\text{invest}} + M$	$0.5(1 - p_{\text{invest}} - M)$	$0.5(1 - p_{\text{invest}} - M)$
Holding	$(1 - p_{\text{invest}})(0.5 + 0.5M)$	p_{invest}	$(1 - p_{\text{invest}})(0.5 - 0.5M)$
Selling	$(1 - p_{\text{invest}})(0.5 + 0.5M)$	$(1 - p_{\text{invest}})(0.5 - 0.5M)$	p_{invest}

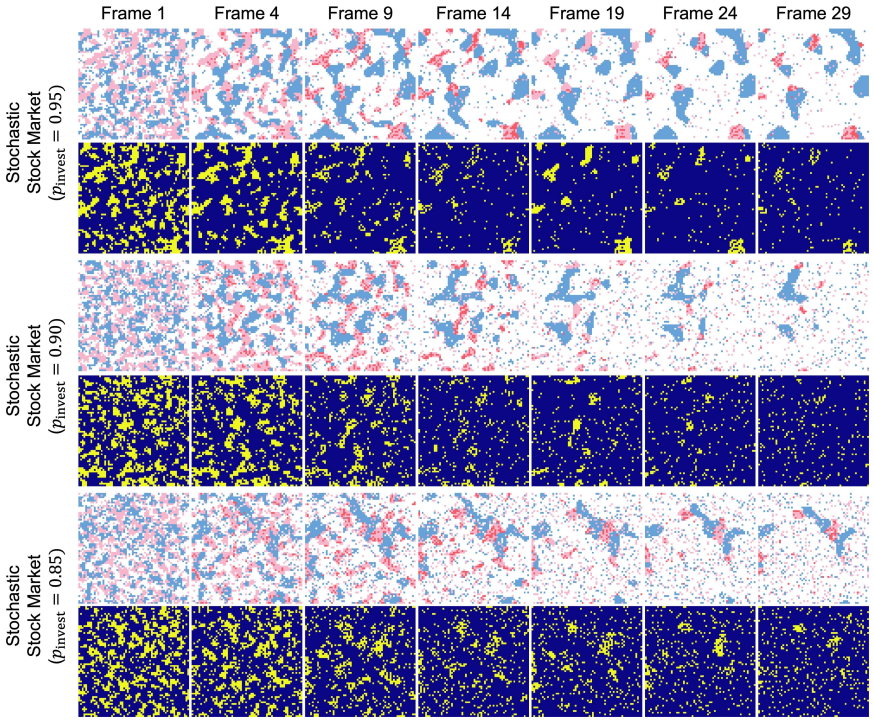

Figure 4: Stochastic stock market model with various p_{invest} values. First row is RGB-based observation (white: hold, blue: sell, pink: buy, and red: inactive). The next row is the corresponding binary ground truth (buy: 1, others:0)

Table 2: Comparison in RA-NCA networks with various encoding dimensions. Compared encoding dimensions (n) are 16 (6,185 parameters), 32 (19,065 parameters), and 48 (39,113 parameters). The networks are trained with 1% training data.

Method (Parameters)	Forest Fire ($p_{\text{heat}} = 0.90$)		Host-Pathogen ($p_{\text{cure}} = 0.15$)		Stock Market ($p_{\text{invest}} = 0.95$)	
	F1 ↑	AUC ↑	F1	AUC	F1	AUC
RA-NCA ($n = 16$)	0.8335±0.0934	0.9696±0.0292	0.6810±0.1160	0.9287±0.0479	0.5560±0.1909	0.7738±0.0852
RA-NCA ($n = 32$)	**0.8672±0.0729**	**0.9768±0.0220**	0.6957±0.1078	0.9443±0.0405	0.5762±0.1887	0.7751±0.0855
RA-NCA ($n = 48$)	0.8636±0.0729	0.9758±0.0224	**0.7001±0.1087**	**0.9485±0.0372**	**0.5848±0.1821**	**0.7756±0.0852**

2. Analysis on RA-NCA network

Various Encoding Dimensions We explore other encoding dimensions of RA-NCA. The encoding dimension (n) is associated across the feature extractor ($R^3 \rightarrow R^n$), Recurrent cellular attention module ($R^n \rightarrow R^n$), and the decoder ($R^n \rightarrow R^1$) in RA-NCA. We consider three encoding dimensions, $n = 16$, $n = 32$, and $n = 48$, in stochastic forest fire, host-pathogen, and stock market models with 1% amount of training data. The results are given in Table 2. We observe relatively higher accuracy gap between $n = 16$ and $n = 32$ models, than between $n = 32$ and $n = 48$ models. It implies that the limited capacity of n=16 model is relieved from $n >= 32$ models. From this result, we choose $n = 32$ as a baseline dimension (compact but not too small capacity). The $n = 48$ model is worse than the n=32 model in the stochastic forest fire, but better in the host-pathogen and stock market datasets. It might be because 1% training data is too less compared to the larger capacity of the $n = 48$ model, particularly in the forest fire dataset. Note, the required amount of training data to avoid the degradation will be different in the three environments since the models predict 50 frames in the forest fire, while predict 20 frames in the host-pathogen and stock market.

Different Neighborhood Size in RA-NCA We investigate another scale of 5x5 neighborhood in RA-NCA. The training and evaluation environment is the deterministic forest fire model. The interaction rules in the dataset remain the same. The full amount of training data is employed, and both the training and test environments are Dense Forest. It is important to mention that the 5×5 model does not know about the interaction scale in the forest fire dataset. Figure 5 shows the training dynamics (BCE loss) of the two models. We observe that the training speed is different in the two models, but their minimum validation loss is converged into the similar level. Table 3 compares the F1 score and AUC of the two models, and the neighborhood scale of 5×5 does not degrade the performance (lower F1 but higher AUC (ROC)). However, we would like to also mention that 3x3 Moore's neighborhood is the most common configuration in Cellular Automata, and hence neural CA architectures are generally aimed to perform on the 3x3 Moore's neighborhood (Mordvintsev et al., 2020; Hernandez et al., 2021; Gilpin, 2019).

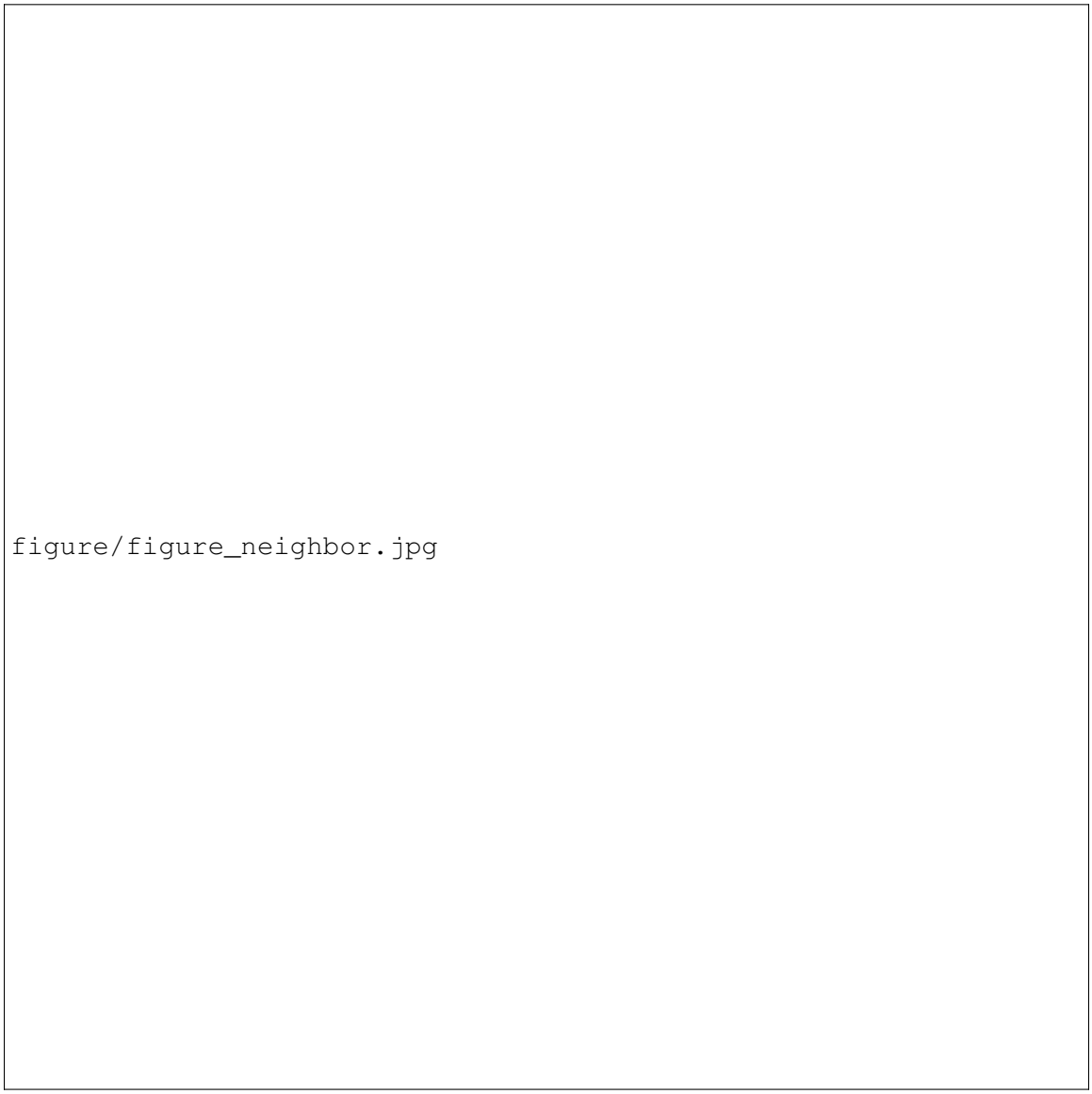

Figure 5: Training dynamics of RA-NCA with 3 ×3 operation (Moore's neighborhood) and 5×5 operation. Both the models are trained for 1000 epochs. The loss curve of RA-NCA (3×3) is omitted from 550 epochs since it starts to diverge. The training BCE loss (left) and validation BCE loss (right) are reduced to the similar level in both the models.

RNN-based RA-NCA network We investigate the Attention+RNN model. Its design is basically similar with RA-NCA, but we replace the LSTM module (Figure **??**(b) right) with the basic RNN module. The hidden state in the RNN module is defined as follows:

$$_t = \tanh(_{iht} + _{ih} + _{hht-1} + _{hh}) \quad (1)$$

Table 3: Comparison of RA-NCA networks with different neighborhood size. RA-NCA (3×3) is the model in the main text, RA-NCA (5×5) operates on 5×5 cells.

Method	Deterministic Forest Fire	
	F1 ↑	AUC ↑
RA-NCA (3×3)	0.8880±0.0610	0.9766±0.0219
RA-NCA (5×5)	0.8786±0.0592	0.9787±0.0195

Table 4: Comparison of RA-NCA networks with different recurrent neural networks. RA-NCA (LSTM) refers the proposed architecture in the main text, and RA-NCA (RNN) replaces the LSTM module to the basic RNN module. RA-NCA (LSTM) and RA-NCA (RNN) involve 19,065 and 12,729 parameters, respectively. The baseline model (Attention-CA) is also included. The networks are trained with 1% training data.

Method	Stochastic Forest Fire ($p_{\text{heat}} = 0.90$)		Stochastic Host-Pathogen ($p_{\text{cure}} = 0.15$)		Stochastic Stock Market ($p_{\text{invest}} = 0.95$)	
	F1 ↑	AUC ↑	F1	AUC	F1	AUC
Attention-CA	0.8305±0.0822	0.9705±0.0283	0.6759±0.1159	0.9433±0.0372	0.5103±0.1862	0.7692±0.0829
RA-NCA (RNN)	0.8410±0.0791	0.9725±0.0266	0.6547±0.1199	0.9373±0.0376	0.5692±0.1857	**0.7756±0.0848**
RA-NCA (LSTM)	**0.8672±0.0729**	**0.9768±0.0220**	**0.6957±0.1078**	**0.9443±0.0405**	**0.5762±0.1887**	0.7751±0.0855

The training environments are stochastic and data-limited forest fire, host-pathogen, and stock market models with 1% training data. The comparison is provided in Table 4. Their parameters are not exactly matching each other (10k for Attention-CA, 12k and 20k for RA-NCA (RNN) and RA-NCA (LSTM)) since we aim to understand the efficacy of the memory module in the same architecture. The F1 score and AUC are generally higher in both the RA-NCA models except for the host-pathogen. Our primary observation is that the accuracy gap between Attention-CA and RA-NCA (LSTM) is higher in the forest fire and stock market, indicating the memory function (LSTM) plays an important role in these environments than the host-pathogen. This might be because the forest fire model and stock market involve more complex interactions, such as dynamic heat dissipation and accumulation (forest fire) and dynamic inactive state (stock market). In this light, introducing the memory function will easily improve the accuracy in the forest fire and stock market. However, as RNN is often unstable due to the lack of cell states, this may make the prediction rather unstable in less dynamic systems. In summary, introducing the memory function is generally helpful if the systems involve the complex behaviors. Also, considering AUC of RA-NCA (RNN) in marginally higher than that of RA-NCA (LSTM), we can say that RA-NCA (LSTM) generally performs better than the two models.

3. Comparison with other NCA networks

Challenges in Employing prior NCA networks Most NCA networks, including the state-of-the-art Attention-Transformer NCA (Tesfaldet et al., 2022), aim to update cells towards a stable fixed-point. For example, the NCA model receives an input image, returns an updated image, and then utilizes it as the next input image. It repeats this process by 8∼32 steps and generate almost no updates after 32 steps. In image denoising, once a noisy image is fully denoised, the model

should not update pixels anymore (otherwise causing a divergence). Hence, ensuring a fixed-point is an important goal as it addresses stationary data. However, it poses a challenge in predicting dynamical systems, where the model should generate a sequence. In other words, after the model generate an image at time t, it should be able to continually generate the next images at $t+1$, $t+2$, ... , instead of staying at time t treating it as a fixed-point.

Further, these approaches will have high computational complexities in both training and inference. For example, Attention-Transformer NCA requires 8$\sim$32 steps to generate a single image for training and 64 steps for inference. For the forest fire model, where the model needs to generate 50 prediction images, the computational graph to calculate its gradient can be very long (*i.e.*, up to 32$\times$50 (=1600) sequences). It significantly increases the GPU memory and training time as well. Similarly, the model should repeat the update steps up to 64$\times$50 (=3200) times for inference, which still requires a very high GPU memory and takes a long inference time.

To illustrate the preceding challenges, we adopted Attention-Transformer NCA to the deterministic forest fire model with full training data. Due to the abovementioned computational difficulties, we set the prediction timesteps to 10 (observe 10 frames and predict 10 frames). For the implementation, we employ the codebase attached in the supplement of the paper (https://openreview.net/forum?id=9t24EBSlZOa). We modify the model size (23,731 parameters) to be similar with our RA-NCA by setting the embedding dimension to 64 and MLP dimension to 16. Figure 6 and 7 demonstrate the training dynamics of the model and showcases the failure case in predicting the deterministic forest fire.

In summary, our RA-NCA follows the common architectural strategy in NCA, operating on the Moore's neighborhood, but our problem cannot be effectively and efficiently solved by the Attention-Transformer NCA model. Also, we would like to mention that the above-mentioned two challenges are applied to the other existing NCA models (non-recurrent) as well, but such discussion has been missed since their applications are focused on the stationary data.

Spatial Dependency in Interaction Learning We design an experiment to measure the spatial dependency on the order of neighboring agents in ConvLSTM-CA and RA-NCA. The networks are taken from the pre-trained ones in Figure **??**(d) (stochastic forest fire with $p_{\text{heat}} = 0.7$). We consider very small 3$\times$3 forest configurations under the same stochasticity and ignite one neighboring agent. The other cells that are not ignited are randomly chosen to be empty or tree agents. In this setting, we can estimate the prediction performance as a function of the fire seed location. Figure 8 displays the examples of the small forest configurations. We generate $\sim$1000 simulations for each of the fire seed locations. The networks predict a single frame after observing the first frame so that the experiment can directly focus on the interaction between a single fire agent to neighbor agents. The graphs in the figure describe that the performance of ConvLSTM-CA is biased to the few certain locations in neighborhood while RA-NCA is well-balanced regardless of the agent locations.

Data Efficient Interaction Learning We summarize the frame-wise mean and standard deviation of F1 score and AUC depending on the amount of training data. Table 5 and 7 show the F1 score comparison with other NCA networks in the stochastic environments trained with 100% and 1% training data, respectively. Table 6 and 8 shows the AUC comparison with other NCA networks in the stochastic environments trained with 100% and 1% training data, respectively. The elements in the tables indicate the mean and standard deviation (*i.e.*, mean$\pm$std).

Table 5: Comparison of F1 score with other NCA architectures trained on 100% training data in stochastic forest fire model (top), host-pathogen model (middle), and stock market model (bottom)

Method	$p_{\text{heat}} = 0.90$ F1 ↑	$p_{\text{heat}} = 0.80$ F1	$p_{\text{heat}} = 0.70$ F1
ConvLSTM-CA	0.8879±0.0688	0.8761±0.0728	0.8424±0.0994
Attention-CA	0.8891±0.0635	0.8719±0.0747	0.8374±0.0993
RA-NCA	**0.8987±0.0598**	**0.8801±0.0737**	**0.8588±0.0928**
Method	$p_{\text{cure}} = 0.10$ F1 ↑	$p_{\text{cure}} = 0.15$ F1	$p_{\text{cure}} = 0.20$ F1
ConvLSTM-CA	0.7040±0.1277	0.7022±0.1116	0.6072±0.1556
Attention-CA	**0.7110±0.1211**	0.7048±0.1087	0.6008±0.1588
RA-NCA	0.7101±0.1232	**0.7109±0.1065**	**0.6088±0.1530**
Method	$p_{\text{invest}} = 0.95$ F1 ↑	$p_{\text{invest}} = 0.90$ F1	$p_{\text{invest}} = 0.85$ F1
ConvLSTM-CA	**0.6167±0.1747**	**0.4229±0.2031**	**0.3913±0.1812**
Attention-CA	0.5407±0.1865	0.3507±0.2041	0.3378±0.1615
RA-NCA	0.6158±0.1745	0.4195±0.2034	0.3895±0.1788

Table 6: Comparison of AUC with other NCA architectures trained on 100% training data in stochastic forest fire model (top), host-pathogen model (middle), and stock market model (bottom)

Method	$p_{\text{heat}} = 0.90$ AUC ↑	$p_{\text{heat}} = 0.80$ AUC	$p_{\text{heat}} = 0.70$ AUC
ConvLSTM-CA	0.9864±0.0160	0.9858±0.0182	0.9823±0.0252
Attention-CA	0.9860±0.0158	0.9865±0.0166	0.9867±0.0168
RA-NCA	**0.9873±0.0152**	**0.9887±0.0150**	**0.9894±0.0151**
Method	$p_{\text{cure}} = 0.10$ AUC ↑	$p_{\text{cure}} = 0.15$ AUC	$p_{\text{cure}} = 0.20$ AUC
ConvLSTM-CA	0.9775±0.0179	0.9554±0.0322	0.9052±0.0632
Attention-CA	0.9789±0.0173	0.9568±0.0312	0.9016±0.0648
RA-NCA	**0.9795±0.0167**	**0.9578±0.0308**	**0.9062±0.0643**
Method	$p_{\text{invest}} = 0.95$ AUC ↑	$p_{\text{invest}} = 0.90$ AUC	$p_{\text{invest}} = 0.85$ AUC
ConvLSTM-CA	0.7785±0.0484	**0.7000±0.0821**	**0.6959±0.0652**
Attention-CA	0.7719±0.0833	0.6956±0.0796	0.6908±0.0616
RA-NCA	**0.7789±0.0842**	0.6997±0.0820	0.6957±0.0654

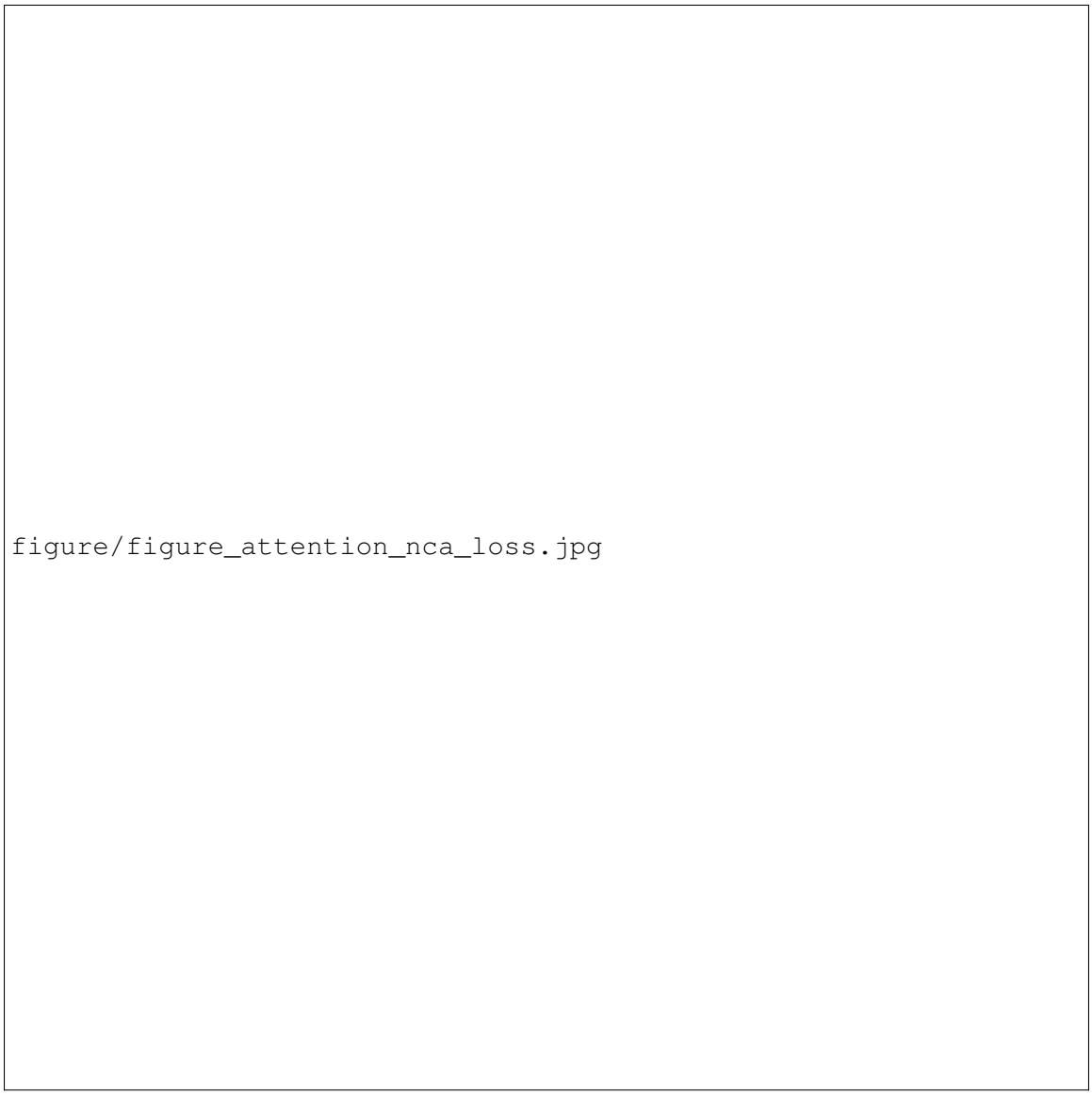

Figure 6: Training dynamics of Attention-Transformer NCA (Tesfaldet et al., 2022) in the deterministic forest fire dataset.

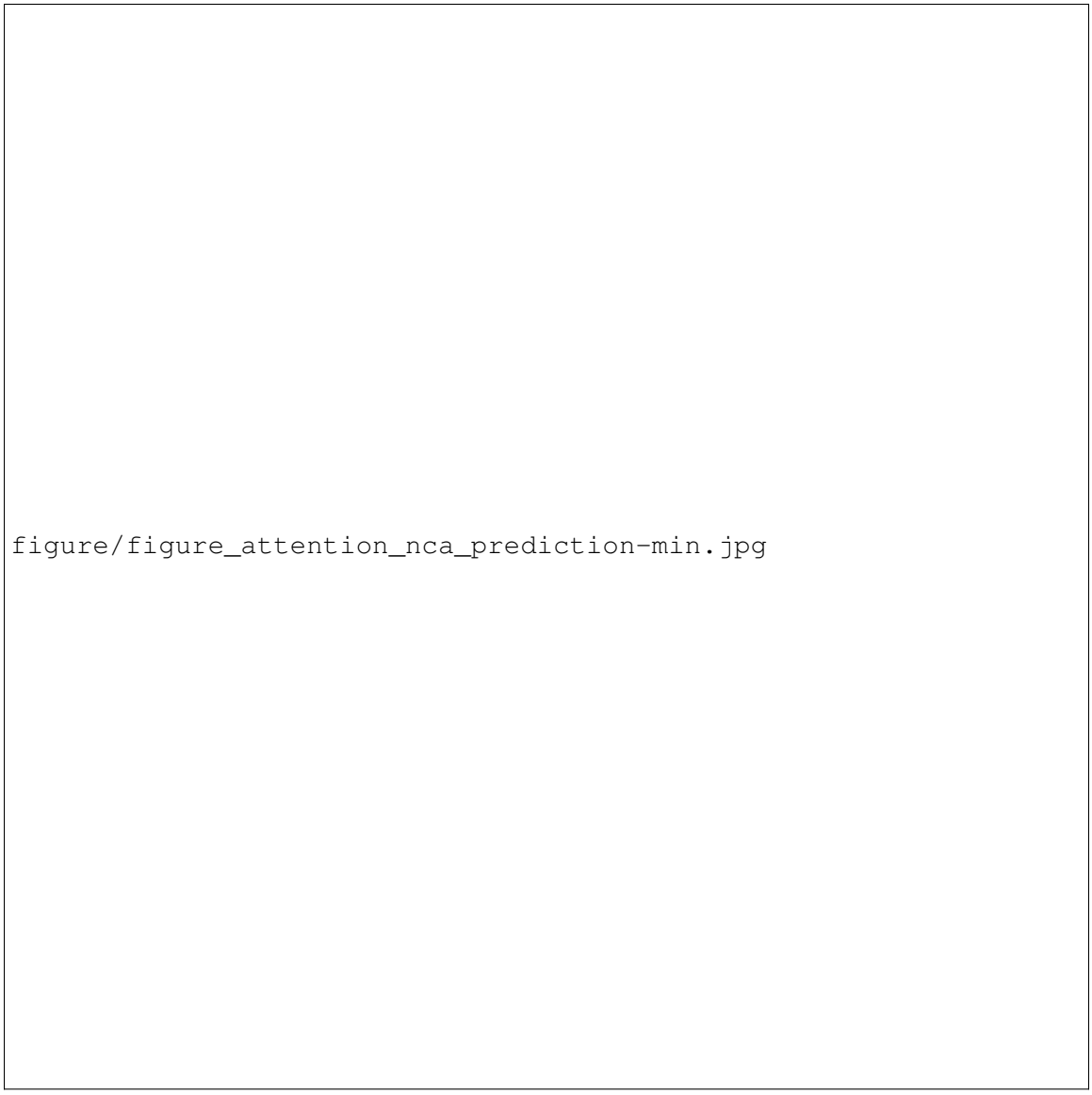

Figure 7: Prediction results of Attention-Transformer NCA (Tesfaldet et al., 2022) on training data (top) and validation data (bottom).

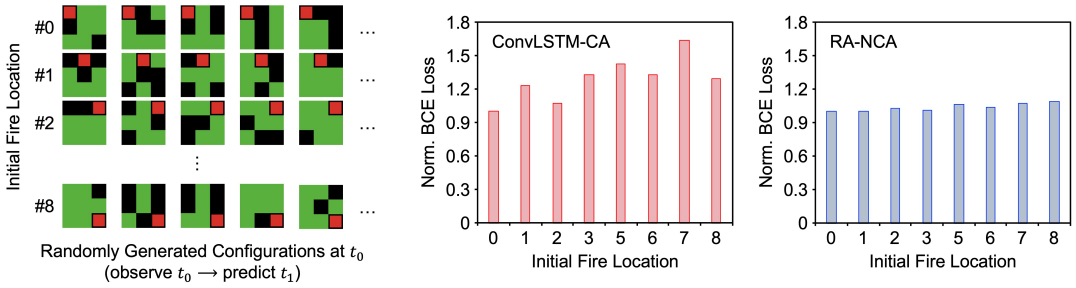

Figure 8: Spatial dependency in RA-NCA and ConvLSTM-CA. The right graphs show the average loss of predicted burning probabilities, which is normalized to make the minimium loss value as 1.

Table 7: Comparison of F1 score with other NCA architectures trained on 1% training data in stochastic forest fire model (top), host-pathogen model (middle), and stock market model (bottom)

Method	$p_{\text{heat}} = 0.90$ F1 ↑	$p_{\text{heat}} = 0.80$ F1	$p_{\text{heat}} = 0.70$ F1
ConvLSTM-CA	0.8293±0.0802	0.7546±0.1052	0.6569±0.0935
Attention-CA	0.8305±0.0822	0.7837±0.1014	0.7379±0.1210
RA-NCA	**0.8672±0.0729**	**0.8280±0.0825**	**0.8003±0.1067**
Method	$p_{\text{cure}} = 0.10$ F1 ↑	$p_{\text{cure}} = 0.15$ F1	$p_{\text{cure}} = 0.20$ F1
ConvLSTM-CA	0.5834±0.1565	0.5793±0.1317	0.3916±0.1870
Attention-CA	0.6855±0.1245	0.6759±0.1159	0.5440±0.1630
RA-NCA	**0.6883±0.1339**	**0.6957±0.1078**	**0.5870±0.1539**
Method	$p_{\text{invest}} = 0.95$ F1 ↑	$p_{\text{invest}} = 0.90$ F1	$p_{\text{invest}} = 0.85$ F1
ConvLSTM-CA	0.4687±0.1991	0.3963±0.1946	0.3681±0.1764
Attention-CA	0.5103±0.1862	0.3502±0.1895	0.3329±0.1595
RA-NCA	**0.5762±0.1887**	**0.4025±0.2036**	**0.3782±0.1826**

Table 8: Comparison of AUC with other NCA architectures trained on 1% training data in stochastic forest fire model (top), host-pathogen model (middle), and stock market model (bottom)

Method	$p_{\text{heat}} = 0.90$ AUC ↑	$p_{\text{heat}} = 0.80$ AUC	$p_{\text{heat}} = 0.70$ AUC
ConvLSTM-CA	0.9666±0.0290	0.9462±0.0453	0.9473±0.0429
Attention-CA	0.9705±0.0283	0.9679±0.0323	0.9703±0.0298
RA-NCA	**0.9768±0.0220**	**0.9732±0.0252**	**0.9763±0.0263**
Method	$p_{\text{cure}} = 0.10$ AUC ↑	$p_{\text{cure}} = 0.15$ AUC	$p_{\text{cure}} = 0.20$ AUC
ConvLSTM-CA	0.9201±0.0681	0.8934±0.0649	0.8252±0.0958
Attention-CA	0.9656±0.0235	0.9433±0.0372	0.8767±0.0755
RA-NCA	**0.9698±0.0254**	**0.9443±0.0405**	**0.8878±0.0743**
Method	$p_{\text{invest}} = 0.95$ AUC ↑	$p_{\text{invest}} = 0.90$ AUC	$p_{\text{invest}} = 0.85$ AUC
ConvLSTM-CA	0.7582±0.0874	0.6902±0.0825	0.6900±0.0647
Attention-CA	0.7692±0.0829	0.6946±0.0794	0.6892±0.0615
RA-NCA	**0.7751±0.0855**	**0.6971±0.0821**	**0.6932±0.0655**

4. Comparison with Video Prediction networks

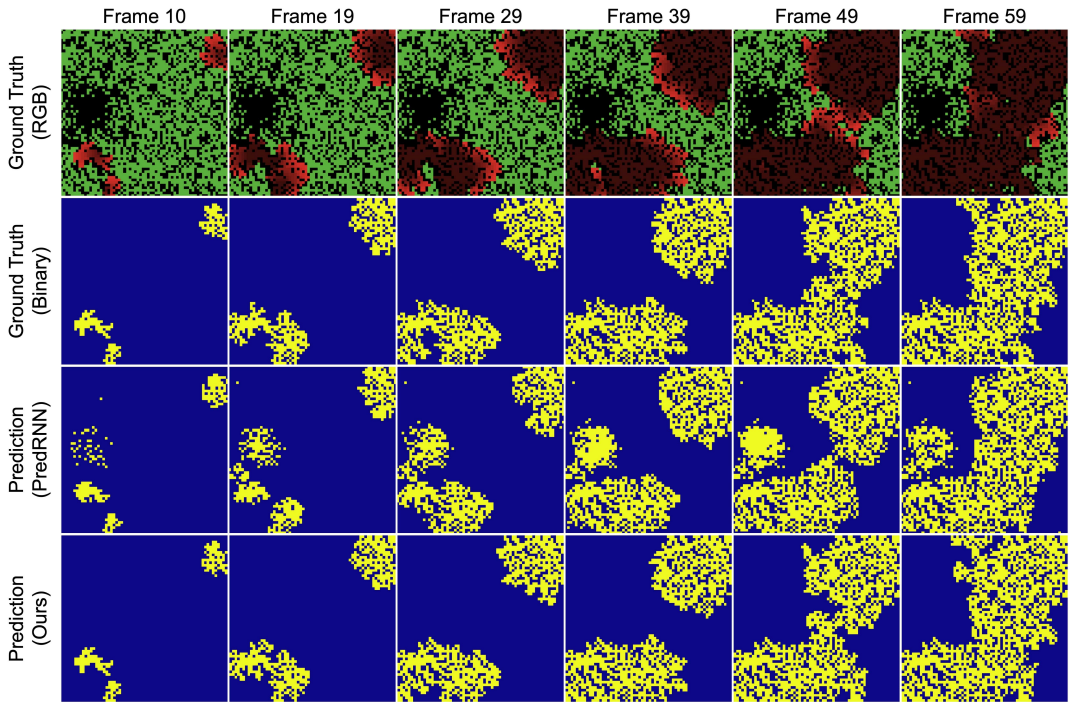

Figure 9: Unnatural prediction results from video prediction model and corresponding results from RA-NCA. Both models observe the first 10 frames (0∼9) and predict the next 50 frames (10∼59).

Unphysical Prediction in Video Prediction Networks We observe that video prediction networks often predict unnatural behavior of agents, which should not be happened if they actually learn the interaction. For example, Figure 9 shows the predicted burning states (e.g., fire for yellow and no fire for blue) of agents in forest fire model and corresponding ground truth. However, PredRNN in the figure Wang et al. (2017) predicts the sparse areas will be burning in later frames, which is not physically possible since there is no tree existing in the forest. Moreover, complex fire front is not appropriately predicted.

A possible reason contributing to the failure will be related to the spatial structures in the latent space, which makes the NCAs clearly different to the video prediction networks. NCA architectures are generally focused on performing on the individual cell, by calculating the update for each of cells, using the 3x3 neighboring information. For example, if input dimension is $^{H\times W\times C}$, (H, W: height and width, C: channel), then latent space is $^{H\times W\times C'}$, and final output is $^{H\times W\times C}$ or $^{H\times W\times 1}$ (in our case). In other words, the models perform agent-wise processing, which only changes the channel dimension ($C \rightarrow C' \rightarrow C$ or 1) preserving the spatial dimension ($H \times W -> H \times W \rightarrow H \times W$).

Table 9: Comparison with video prediction networks in deterministic forest fire with various configurations. 100% training data is used.

Method	Dense Forest (F1↑)	Sparse Forest (F1)	Gaussian Forest (F1)
PredRNN (Wang et al., 2017)	0.8473±0.0710	0.7369±0.1102	0.7968±0.1448
PredRNN++ (Wang et al., 2018)	0.8584±0.0692	0.7373±0.1357	0.8228±0.1071
ImaGINator (Wang et al., 2020)	0.5436±0.2507	0.3608±0.2289	0.4937±0.2622
SimVP (Gao et al., 2022)	0.8823±0.0528	0.7807±0.1173	0.8777±0.0673
RA-NCA (Ours)	**0.9232±0.0468**	**0.8648±0.0882**	**0.9011±0.0998**

Table 10: Comparison with video prediction networks in deterministic forest fire with various configurations (top: 100% training data, bottom: 1% training data)

Method	Stochastic Forest Fire ($p_{\text{heat}} = 0.90$)	Stochastic Host-Pathogen ($p_{\text{cure}} = 0.15$)	Stochastic Stock Market ($p_{\text{invest}} = 0.95$)
PredRNN (Wang et al., 2017)	0.7984 ± 0.0875	0.5844 ± 0.1488	0.3991 ± 0.1682
PredRNN++ (Wang et al., 2018)	0.8081 ± 0.0936	0.5338 ± 0.1608	0.3919 ± 0.1585
ImaGINator (Wang et al., 2020)	0.4865 ± 0.2000	0.1774 ± 0.1459	0.2632 ± 0.1183
SimVP (Gao et al., 2022)	0.8441 ± 0.0658	0.6777 ± 0.0974	0.4050 ± 0.1491
RA-NCA (Ours)	**0.8987 ± 0.0598**	**0.7109 ± 0.1065**	**0.6158 ± 0.1745**

Method	Stochastic Forest Fire ($p_{\text{heat}} = 0.90$)	Stochastic Host-Pathogen ($p_{\text{cure}} = 0.15$)	Stochastic Stock Market ($p_{\text{invest}} = 0.95$)
PredRNN (Wang et al., 2017)	0.4564 ± 0.2384	0.0375 ± 0.1132	0.1274 ± 0.0753
PredRNN++ (Wang et al., 2018)	0.4459 ± 0.2069	0.0256 ± 0.0627	0.1104 ± 0.0488
ImaGINator (Wang et al., 2020)	0.0005 ± 0.0019	0.2404 ± 0.1415	0.1611 ± 0.0586
SimVP (Gao et al., 2022)	0.7387 ± 0.0992	0.1576 ± 0.1177	0.2192 ± 0.1105
RA-NCA (Ours)	**0.8672 ± 0.0729**	**0.6957 ± 0.1078**	**0.5762 ± 0.1887**

References

Kan Chen, Per Bak, and Mogens H Jensen. A deterministic critical forest fire model. *Physics Letters A*, 149(4):207–210, 1990.

Zhangyang Gao, Cheng Tan, Lirong Wu, and Stan Z Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3170–3180, 2022.

William Gilpin. Cellular automata as convolutional neural networks. *Physical Review E*, 100(3): 032402, 2019.

Alejandro Hernandez, Armand Vilalta, and Francesc Moreno-Noguer. Neural cellular automata manifold. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10020–10028, 2021.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 5(2):e23, 2020.

Richard C Rothermel. *A mathematical model for predicting fire spread in wildland fuels*, volume 115. Intermountain Forest & Range Experiment Station, Forest Service, US . . . , 1972.

Hiroki Sayama. Pycx: a python-based simulation code repository for complex systems education. *Complex Adaptive Systems Modeling*, 1(1):1–10, 2013.

Mattie Tesfaldet, Derek Nowrouzezahrai, and Chris Pal. Attention-based neural cellular automata. *Advances in Neural Information Processing Systems*, 35:8174–8186, 2022.

Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Citeseer, 2004.

Yaohui Wang, Piotr Bilinski, Francois Bremond, and Antitza Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1160–1169, 2020.

Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems*, 30, 2017.

Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and S Yu Philip. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*, pages 5123–5132. PMLR, 2018.

Yi-ming Wei, Shang-jun Ying, Ying Fan, and Bing-Hong Wang. The cellular automaton model of investment behavior in the stock market. *Physica A: Statistical Mechanics and its Applications*, 325(3-4):507–516, 2003.