Towards Learning Self-Organized Criticality of Rydberg Atoms using Graph Neural Networks

Simon Ohler [1 2 †] **Daniel Brady** [1] **Winfried Lötzsch** [2] **Michael Fleischhauer** [1] **Johannes S. Otterbach** [2]

Abstract

Self-Organized Criticality (SOC) is a ubiquitous dynamical phenomenon believed to be responsible for the emergence of universal scale-invariant behavior in many, seemingly unrelated systems, such as forest fires, virus spreading or atomic excitation dynamics. SOC describes the buildup of large-scale and long-range spatio-temporal correlations as a result of only local interactions and dissipation. The simulation of SOC dynamics is typically based on Monte-Carlo (MC) methods, which are however numerically expensive and do not scale beyond certain system sizes. We investigate the use of Graph Neural Networks (GNNs) as an effective surrogate model to learn the dynamics operator for a paradigmatic SOC system, inspired by an experimentally accessible physics example: driven Rydberg atoms. To this end, we generalize existing GNN simulation approaches to predict dynamics for the internal state of the node. We show that we can accurately reproduce the MC dynamics as well as generalize along the two important axes of particle number and particle density. This paves the way to model much larger systems beyond the limits of traditional MC methods. While the exact system is inspired by the dynamics of Rydberg atoms, the approach is quite general and can readily be applied to other systems.

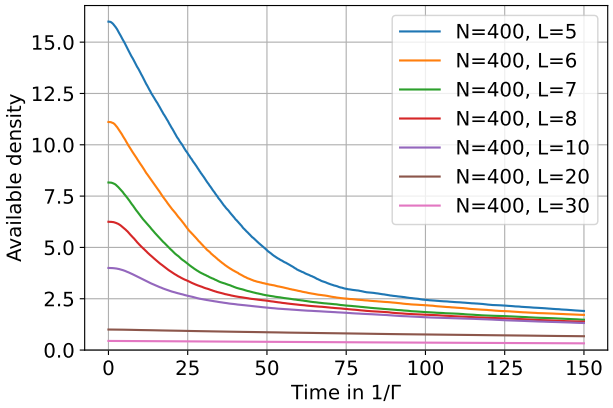

Figure 1. Results from the 60000-timestep rollout of the GNN, after being trained only on single timestep data. The available density (sum of ground and active state atom densities) shows the characteristic SOC behavior: all initial densities above the critical value of $n_c \approx 1$ converge to the same point, whereas all curves that start below n_c show only a slow decay.

1. Introduction

Deep Learning methods deliver impact for many applications ranging from Computer Vision, Natural Language Processing, Audio and Speech Processing to Optimal Control. In recent years we also see these methods drive innovation in areas of scientific interest, such as drug discovery (Jim'enez-Luna et al., 2020), protein folding (Jumper et al., 2021), molecule dynamics (Noé et al., 2020; Gilmer et al., 2017; Henderson et al., 2021) as well as the dynamics of classic multi-particle systems (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020) and many more. In the context of particle dynamics the class of Graph Neural Networks (GNNs) (Bronstein et al., 2021; Sanchez-Gonzalez et al., 2020; Shlomi et al., 2021; Lemos et al., 2022) are particularly successful. We build up on this success and investigate the application of GNNs to learn the time-evolution of a complex many-body system in the regime of so-called self-organized criticality (SOC).

SOC represents a widespread phenomenon observed in many different fields, ranging from noise in electric circuits (Bak et al., 1987), earthquakes (Sornette & Sornette, 1989), forest fires (Malamud et al., 1998; Drossel et al., 1993) over the spread of information or diseases (Gleeson et al., 2014; Rhodes & Anderson, 1996) to atomic excitation dynamics (Helmrich et al., 2020). The term refers to the property of a

[†]Work done while at Merantix Momentum [1]Department of Physics, University of Kaiserslautern, Kaiserslautern, Germany [2]Merantix Momentum, AI Campus Berlin, Germany. Correspondence to: Simon Ohler <sohler@rhrk.uni-kl.de>, Johannes Otterbach <johannes.otterbach@merantix.com>.

2nd AI4Science Workshop at the 39^{th} International Conference on Machine Learning (ICML), 2022.

complex system to drive itself towards the critical point between two dynamical phases in the relevant parameter space during its time evolution without any external control or fine-tuning. Remarkably, even though the microscopic processes in these systems may be very different, certain properties such as the scaling laws of the critical behaviour remain the same, a phenomenon termed *universality*. This allows to study the properties of SOC dynamics in one system, e.g. forest fires, through controlled simulation and experimentation of other systems, e.g. atoms. An important challenge in studying SOC-behavior is the efficient simulation of the dynamics of large systems. State-of-the-art methods employ Markov-Chain Monte-Carlo (MCMC) methods that are limited in terms of number of particles and system size (Wintermantel et al., 2021). This renders the study of large-scale systems, required for reliable predictions of key SOC properties such as critical exponents, hard. Additionally, the simulation requirements in time and memory increase with the system size and make the simulations prohibitively expensive in the absence of high-performance computation infrastructure.

In this study, we investigate the use of GNNs to accurately approximate the time-evolution operator of a complex system inspired by the dynamics of a classical ensemble of Rydberg atoms in the SOC regime. The GNN acts as an effective surrogate model and replaces the costly MCMC step operator. We model the atoms as nodes in a GNN with internal degrees of freedom encoded in the node attribute and the Euclidean distances between the endpoints attached as edge attributes. We learn the one-timestep update function of the node attributes from data simulated using classical MCMC methods. We demonstrate that the GNN is capable to efficiently learn this operator and reliably reproduce the characteristic properties of the system such as the facilitation dynamics, Rydberg blockade effects and SOC excitation trajectories. Moreover, we show early indications that the model generalizes over several orders of magnitudes along two important axes: particle number and system size (particle density). This property enables the application of this model to system sizes beyond what can be simulated using classical MCMC methods, while also enabling computations on moderate hardware.

In addition, we are, to the best of our knowledge, the first to extend the simulation capabilities of GNNs to systems with dynamics in internal as well as external degrees of freedom. While previous work was mostly focused on simulating the position dynamics of the nodes (Battaglia et al., 2016; Kipf et al., 2018; Sanchez-Gonzalez et al., 2020), an external degree of freedom, we also discuss the dynamics of the node attribute, i.e. the internal degree of freedom, based on the interaction of a node with its neighbors.

We make both the code for our experiments as well as the Monte Carlo code to create our datasets publicly available upon acceptance.

2. Related Work

GNNs have been used in a multitude of applications. Most relevant to our study is the work by Sanchez-Gonzalez et al. (2020) to learn the dynamics of challenging physics motions such as viscous fluids and elastic bodies simulated using third party tools. They do so by breaking the objects into a discrete set of volumes that are subsequently represented as nodes in a graph. The position update of the volume is learned by predicting the next position using an encoder-decoder system with several message passing (MP) layers.

Pfaff et al. (2021) expand on this work to simulate the dynamics of wings and flags under external forces. They do so by introducing two different meshes, one for the surface of the object under study and a second one encoding the real-world, Euclidean distance between the nodes of the surface mesh. Using MP they learn to predict the surface mesh positions in the next time-step from simulated data.

Battaglia et al. (2016) study the use of graph representations in the data to enable a multi-layer perceptron to reason about complex interactions of physical systems. By making the relationships between physical objects explicit in the graph data structure, they are able to predict the orbital motion of planets as well as the dynamics of solids under non-trivial, external constraints.

Shlomi et al. (2021) give a review of the use of GNNs in high-energy particle physics. They draw attention to the benefits of using GNNs to model the different particle interactions through clever design of the node, edge and vertex attributes when representing the data as relationship graphs.

Lemos et al. (2022) use GNN simulators in conjunction with symbolic regression to rediscover Newton's law of gravity from simulated data of multi-planetary motions. This serves as a demonstration to infer physical laws from complex dynamics as multi-planetary motion can behave chaotically under certain circumstances.

Martinkus et al. (2021) discuss the construction of a hierarchical GNN in order to scale the simulation of particle dynamics from a few hundreds to tens of thousands. By recursively partitioning the space and compute interactions between the cells at each hierarchy they are able to reduce the $O(N^2)$ interaction computation to $O(N \log N)$ complexity.

Prakash & Tucker (2021) investigate the use of multi-task learning in GNNs to simultaneously learn the dynamics and the unknown physical parameters of an ensemble of

objects with Newtonian dynamics. They show that using this approach one can circumvent the design of inductive biases in the GNN construction, making the GNN architecture more versatile across datasets.

Olsson & Noé (2019) use a graphical model to describe molecular kinetics, where each molecular subsystem changes its internal state based on interactions with the neighboring subsystems. Instead of using a single global state, the approach of using interacting subsystems is key in reducing the computational effort.

Jin & Voth (2018) study interfacial systems using an ultra-coarse grained model that is constructed by including the local particle density in the internal states of the coarse-grained sites. This construction allows to distinguish different phases and to successfully capture phase coexistence.

Zhang et al. (2022) use a GNN architecture not for physical simulations, but for sequential recommender systems in social networks. Instead of using only the previous user-item interactions of a single person to predict future interactions, the authors include dynamic collaborative signals among different users.

In contrast to these works, we are the first to discuss the dynamics of a complex atomic many-body system whose entities have an additional internal state, i.e. where the node attributes can change depending on the neighbors. This extends the use of GNNs to efficiently simulate systems that have internal as well as external degrees of freedom. On top of this we also demonstrate the ability of the GNNs to learn non-trivial spatio-temporal correlations, despite being trained on a spatially local, one-step prediction task.

3. Self-Organized Criticality in Rydberg Atoms

A typical SOC system consists of many identical entities, each taking one of three internal states. The entity's state can vary over time as a function of the system's dynamics. Since in our case the entities are Rydberg atoms, we label these three states as the *ground*, *active* and *inactive* states, respectively (see Figure 2(b)). For illustration, in disease spread models these states are commonly referred to as susceptible, infected and recovered. The key ingredient to produce the characteristic SOC dynamics lies in the transitions between the internal states of each atom, as shown in Figure 2(a): the ground state $|g\rangle$ can only become active via the neighborhood-dependent rate $p_{g\to a}(N)$, called the *facilitation* process. The active state $|a\rangle$ can undergo the reverse process and return to the ground state again with a neighborhood-dependent rate $p_{a\to g}(N)$, or transition into the inactive state $|0\rangle$, which is a terminal state in the process.

The physical realization of the SOC system we study here is based on a gas of Rydberg atoms subject to external laser fields. The microscopic transition rates for each atom i per time interval Δt of the model are given by

$$p_{i;g\to a} = 1 - e^{-R_i \Delta t} \tag{1}$$

$$p_{i;a\to g} = (1 - \alpha_i)\left(1 - e^{-(R_i + \gamma)\Delta t}\right) \tag{2}$$

$$p_{i;a\to 0} = \alpha_i \left(1 - e^{-(R_i + \gamma)\Delta t}\right) \tag{3}$$

$$R_i = \frac{\Omega^2}{\Delta^2 + \frac{\gamma^2}{4}} \tag{4}$$

$$\alpha_i = \frac{b}{1 + R_i/\gamma} \tag{5}$$

$$\Delta_i = \Delta_0 - \left(1 + \sum_{\ell \in N} \frac{1}{r_{i;\ell}^6}\right)^{-1} \tag{6}$$

$$\delta_\ell = \begin{cases} 1, & \text{if } s_\ell = |a\rangle \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The parameters Δ_0, Ω, b, γ and $\Delta\tau$ are details of the system and can be treated as hyper-parameters to the problem. N denotes the total number of atoms in the system. The symbol $\sum_\ell$ in Equation (6) ensures that the sum includes only the active atoms (state $s_\ell = |a\rangle$). Additionally, $r_{i;\ell}$ denotes the Euclidean distance between atom i and ℓ in natural units, where we set the length scale $R_F = 1$ (see Figure 2(b)). A more detailed introduction to this system is presented in Appendix A. Equations (1) to (6) describe a strongly interacting many-body system, for which neither trivial nor analytic solutions exist. In order to solve the system for large particle numbers and long times, approximate techniques such as MCMC (Wintermantel et al., 2021) or coarse-graining (Helmrich et al., 2020) have historically been used. It is worth pointing out that Equation (6) displays a resonance behavior dependent on the number and distance of the neighboring active atoms, giving rise to the neighborhood-dependent state transition rates: If $|j - j'|$ is large, the transition rate is suppressed and no change in the dynamics is happening. If an active atom is present at the right distance to an atom i, then $\Delta_i = 0$ and the transition is facilitated. The facilitated atom can in turn facilitate the excitation of another atom. Above a critical density this leads to an explosive spread of excitations. The system hence shows two distinct phases:

- The *active* phase is characterized by a large number of active atoms that spread through the entire system. The density of inactive atoms is small, and each active atom facilitates on average more or equal to one ground state atom before it ceases to be active.
- The *absorbing* phase is a state where the available density of ground state atoms is so low that transitions into the active state typically become inactive quickly

Figure 2. Schematics for the SOC problem construction. Figure 2(a): The model entities (atoms) have each three internal states. The ground state $|g\rangle$, the active state $|a\rangle$ and the inactive state $|0\rangle$. The atoms have a distance-dependent interaction scaling as $1/r^6$. Figure 2(b): When constructing the input graph there are two length scales that are relevant: the facilitation radius R denoting the relevant interaction distances and the cutoff radius R_c used to limit the number of edges in the graph to avoid all-to-all connectivity.

before they can spread, i.e. facilitate the transition of another ground state atom into the active state.

The critical point of the system occurs at the point where the two phases meet, i.e. where an active atom facilitates exactly one ground state atom on average before it decays. Since the inactive state is a dead-end, the available density of ground and active state atoms can only decrease over time. Consequently, the active phase automatically moves towards the critical point, which constitutes a self-organization of the critical phase termed SOC. Since SOC systems at the critical point share certain universal properties, being able to simulate a single SOC system with high accuracy and scalability allows investigations of many different physical systems.

4. Learning the Rydberg SOC Dynamics with a GNN

Training data. We model the atoms as the nodes of a GNN with the internal states of the atoms mapped to node attributes and the distance between them as the edge attributes (see Figure 2(b)). An advantage of the GNN structure over conventional discretization schemes is the independence of system density from the discretization length and thus an easier computational description of the problem at hand. Note that throughout this study we always use training data for two-dimensions, where all atoms are placed into a square box in the xy-plane.

The training data is generated using classical Monte-Carlo methods (see Appendix B) and we create a large dataset covering a broad range of particle numbers and densities to expose the model to a diverse set of scenarios during training. Each training sample consists of X, a matrix of shape (N, N_f), and the ground truth Y, a matrix of shape $(N, 3)$. Here, N_f denotes the number of node features, which are the position and internal state of each node. Note that X is not directly the input to the GNN, since we replace the absolute positions with relative distances. The ground truth matrix $Y_i = (P_i(g|s_i), P_i(a|s_i), P_i(0|s_i))$ contains for each node i with the internal state s_i the probabilities to change to or remain in each of the three states in the next timestep. Before feeding a graph into the GNN, we apply the following physically motivated processing steps to facilitate easier learning for the model:

- We mask out all nodes in the inactive state $|0\rangle$, since they do not contribute to the dynamics and can be neglected without loss of accuracy.
- The edges between nodes are created by a radius-graph scheme: For two nodes to be connected their relative (Euclidean) distance needs to be smaller than a critical radius R_c, which is a hyperparameter for our model. Increasing R_c yields higher accuracy, but also increases the computational burden due to a quadratic rise in the number of neighbors. However, as the interaction drops off as $1/r^6$ the benefit of a larger R_c is minimal. For all our calculations we use $R_c = 2$ (in natural units).
- Two nodes in the ground state $|g\rangle$ do not interact with each other, allowing us to remove edges between them. This causes no loss of accuracy and reduces the total number of edges in a graph by up to 30%, since the fraction of active atoms is typically lower than 15%.
- Each edge is given the Euclidean distance between its nodes as an edge attribute. The absolute position data is not encoded to encourage translation and rotational invariance of the model, since only relative distances are relevant for the SOC behavior.

Network architecture. To construct the network architecture we first note some characteristics of the probability distribution for the state transition probabilities. As the data is generated using Monte-Carlo simulations of a continuous process, the state transition probabilities are small in order

Figure 3. We use a residual architecture for the network structure to encourage the model to learn the background mean- eld and the uctuations on top of the mean eld separately. To this end we feed the node embeddings into a simple MLP that learns the factorized node distribution. The uctuations are learned using a Graph Convolution network that receives the node as well as edge embeddings as input. Finally, we add the mean- eld and uctuations before feeding them through the nal MLP to predict the update probabilities.

to minimize the numerical error of the simulation. As a result, the probability of each atom to remain in its current state is large in order to preserve probability. This results in a bimodal distribution of the state transition probabilities with the actual transitions orders of magnitude smaller, i.e. $O(10^{-7} - 10^{-1})$ than the probabilities to remain in the current state $O(1)$ (For more details see Appendix B). However, the dynamics of the system are encoded in the changes in the state transition probabilities and without accounting for the sizable differences in the probabilities the model will simply learn the large background probabilities and not pay much attention to the subtle uctuations in the transition probabilities that are caused by neighborhood interaction effects.

To address this challenge, we make use of a residual network architecture as depicted in Figure 3, where we give the network the opportunity to model the large background distribution, termed the *mean- eld* contribution, and the neighborhood effects of the changes in the state transition probabilities, termed *uctuations*. This terminology is in accordance with corresponding techniques in the traditional modelling of dynamic systems, where one tries to describe the system in terms of small variations, the uctuations, around a stationary point, the mean- eld. More speci cally, we embed the state of each node into a higher-dimensional embedding space. Note that we removed the inactive state $|0\rangle$ from the data. We then feed each node state individually through a simple feed forward network. Since this network does not take into account the edge attribute, it effectively learns only the factorized, non-correlated features of the nodes, which corresponds to the mean- eld part of the dynamics. To capture the correlations, we combine the embedded node states and edges and feed them through a Message Passing Graph Convolution layer (Gilmer et al., 2017). We then add the outputs of the mean- eld and uctuation modules before passing them through a nal MLP network to predict the state transition probabilities of each node. We use RELU nonlinearities as activation funtions and normalize the activations using LAYERNORM for the MLPs.

Loss function. As noted before the transition probability distribution changes over many orders of magnitude and hence we need a loss function that works over such a large interval. We experimented with mean-squared-error (MSE) and Kullback-Leibler divergence (KLD) and found the latter to be more stable. This can be motivated by the fact that the logarithmic dependency of the KLD penalizes differences in the exponents of the probabilities much more severely than the MSE loss that ignores such differences if the probabilities are small in favor of minimizing the differences for large probabilities.

5. Experiments & Results

Having established the modelling approach, we now turn to the experimental validation. We implement the GNN based on PYTORCH (Paszke et al., 2019) using the PYTORCH LIGHTNING framework (Falcon & the PyTorch Lightning team, 2019) as well as PYTORCH GEOMETRIC (Fey & Lenssen, 2019). The network consists of a embedding module for the internal node state with an output dimension of 10, a state encoder network (mean- eld component) consisting of 2 fully connected layers with hidden dimension of 50 and a nal output dimension of 32, a Graph Convolution layer with a single message passing step and a nal decoder with 2 fully connected layers and hidden dimension 50. The Graph Convolution uses the NNCONV module from (Gilmer et al., 2017) that is implemented in PYTORCH GEOMETRIC and has an output dimension of 64. The MLP inside of the NNCONV has a single hidden layer with 50 units. Inside the uctuations component we apply another MLP of 2 hidden layers with 100 units each and an output dimension of 32, before adding both components. In total the network has approximately 81000 parameters.

For all experiments we use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3}, beta values of $(0.9, 0.999)$ and batch size 15. We set the cutoff radius $R_c = 2$ (in natural units) and train the model for 25 epochs with 6000 training instances each on an Intel® Xeon® Gold

Processor 6154 with a peak memory usage of around 240 MB. The training instances are drawn from three datasets that we created that all contain $N = 150$ atoms per instance but had varying densities. In two datasets we used $dx = 0.25$ and $dx = 0.5$, respectively. Here, we define dx as $dx = L/\sqrt{N}$, where L is the length of one edge of the square box in which we place the atoms. Intuitively, dx represents the distance between the atoms if they were placed in an ideal square lattice. In the third we artificially increased the fraction of active state atoms to 50% at a density of $dx = 1.25$ to address the fact that active atoms are otherwise rare in the training data. Finally, for reasons of numerical stability we transform the probabilities using a square-root transformation. This ensures the probabilities stay in the range $[0, 1]$ while mitigating the extreme bimodal nature of the ground-truth probability. See the Appendix B for more details.

Task	Configuration	KLD
num. particles	N=50, dx=0.5, d=2	1.17e-4
	N=150, dx=0.5, d=2	1.50e-4
	N=500, dx=0.5, d=2	1.83e-4
	N=1000, dx=0.5, d=2	1.97e-4
	N=1500, dx=0.5, d=2	2.04e-4
density	N=150, dx=0.125, d=2	3.00e-5
	N=150, dx=0.25, d=2	8.17e-5
	N=150, dx=0.5, d=2	1.50e-4
	N=150, dx=1, d=2	7.54e-5
	N=150, dx=2.5, d=2	1.99e-5

Table 1. Test loss averaged over 2000 graphs each from different datasets, where either the particle number or density was varied. For the case of particle number generalization, we test on up to 10 times as many particles as seen during training and observe that the loss rises slowly with the particle number. For the case of varying densities we go a factor of 2 denser than the smallest density seen during training (including the engineered dataset with more active atoms), as well as a factor of two above. In both cases, the loss is reduced compared to $dx = 0.5$, which we attribute to less facilitation events and more blockade (for higher density, low dx) or more non-interacting cases (for lower density, high dx).

Transition rates of internal node dynamics. To the best of our knowledge there is no comparable publication that predicts transition probabilities in SOC systems or considers the dynamics of internal degrees of freedom of a node. Hence we developed metrics to assess the quality and validate the learning of the GNN. We base these metrics on key aspects of the Rydberg-SOC dynamics.

The fastest process in the SOC system is facilitation, the process by which an active atom promotes surrounding ground state atoms to the active state, according to the resonance feature of Equation (6). The facilitation probability of a ground state atom at a distance r to an active atom is characterized by an extremely sharp peak at $r = 1$ (in natural units), several orders of magnitudes larger than the non-interacting ($r \to \infty$) case. Additionally, as $r \to 0$ the facilitation probability decreases rapidly to zero, due to the Rydberg blockade (Lukin et al., 2001) (also see (Saffman et al., 2010) for a review). For $r \approx R_F$, the facilitation probability tends to a small, but finite value. To check this behaviour we construct a custom graph where we place an active state atom at the origin and a ground state atom at the position $(x, y) = (r, 0)$. Additionally, we randomly place other ground state atoms in the graph to keep the total particle number and density close to those seen during training. The value we record is then $P_i(a|g)$ of the ground state atom to become active at position $(x, y) = (r, 0)$. We then sweep this setting across different values of r, creating a new random graph each time. The result of this experiment is shown in Figure 4(a) and shows excellent agreement with the exact solution of the facilitation rate around the facilitation peak at $r = 1$ (in natural units) and in the $r \to \infty$ limit. In the blockade regime for $r < 1$ the GNNs prediction converges to $P_i(a|g) \approx 10^{-11}$, six orders of magnitude smaller than the limit $r \to 1$. The true convergence to zero however is not captured. This is an expected result given the fact that even making large relative errors on very small ground-truth probabilities incurs a small loss penalty for the algorithm. Nonetheless, since the predicted blockaded facilitation probability is sufficiently small compared to all SOC timescales, we hypothesize that this deviation from the ground truth does not lead to qualitative changes in the GNNs predictions.

The inverse process is called de-facilitation and is, to the best of our knowledge, a process unique to Rydberg atoms. In this case, a de-excitation (return to the ground state) of an active atom is facilitated by the presence of a second active atom. If these two active atoms are close to each other, the probability of one atom to return to the ground state increases sharply at the relative distance $r = 1$. Even although both the facilitation and the de-facilitation share the same physical origin, their ground-truth curves in Figures 4(a) and 4(b) look dissimilar on short distances. The is due to the second process that allows active atoms to return to the ground state, which is independent of the neighborhood (γ in eqn. (2)). We check this behaviour the same way as for the facilitation metric, but in this case the atom at position $(x, y) = (r, 0)$ is also active and we record $P_i(g|a)$. The results are shown in Figure 4(b). The de-facilitation is much harder to learn as the case of two active atoms at close distance is very rare in the dataset, especially in the already very narrow region around $r = 1$ where the behavior changes drastically. We find that the GNN accurately predicts the non-resonant case as well as the position of the peak at $r = 1$. However, the interval between $r \in [0.995, 1.005]$, where the ground truth values increase

(a) Model prediction of the facilitation process of a ground state atom as well as the ground-truth curve from Monte Carlo.

(b) Model prediction of the facilitated de-excitation of an active atom as well as the ground-truth curve from Monte Carlo. The inset shows a magnification of the peak around $n = 1$.

Figure 4. Benchmark and validation curves to assess correct learning of key interaction features for the SOC dynamics in Rydberg atoms.

by a factor of 10, is not reproduced, as the GNNs prediction does not exceed 0.009, an order of magnitude short of the ground truth.

Particle number and density generalization. In addition to the SOC specific metrics we investigate the prediction accuracy on graphs consisting of more nodes, i.e. higher particle number, or denser, i.e. shorter average distance between particles, than those seen during training. The results are summarized in Table 1. We see that when trained on $N = 150$ atoms, the model suffers no significant loss of accuracy when predicting on a varying number of atoms up to $N = 1500$. The lowest test loss occurs at smaller N than during training, which might be the result of boundary effects that the model learns when trained on small system sizes. This scalability of the GNN has been observed in other particle-based regression tasks (Sanchez-Gonzalez et al., 2020) and is most likely due to the local scope of the message passing, which encodes spatial translation invariance of the dynamics operator.

In addition, we see that the test loss of the GNN predicting on datasets with different densities is also small across a range of average dx distances. During training, the model is confronted with densities $dx = 0.5$ and $dx = 0.25$ as well as artificial graphs with $dx = 1.25$. The test case of $dx = 0.5$ on an unseen dataset shows the highest test loss, which is most likely due to the fact that more facilitation events happen at this density than at lower or higher densities. In the case of lower densities, the chance of a ground state atom to be located in the small interval around $n = 1$ to an active atom is reduced, and in the case of larger densities the case of $n = 1$ occurs more frequently, where we see extremely small state-change probabilities (blockade).

Figure 5. Comparison between the 60000-timestep predictions of the GNN (solid) and the Monte-Carlo simulation (dashed) from which the training data was obtained. We do not show all initial conditions in the interest of readability. The available density is the sum of ground and active state densities.

SOC excitation trajectories. Having established that the model captures key aspects of the SOC dynamics and is able to generalize over particle numbers as well as densities, we can use the trained model to iteratively predict the time evolution of an initial ensemble of atoms. To this end we randomly create an initial 2D distribution of atoms, each in the ground state. Additionally, we assign a velocity to each atom which we sample from a Maxwell-Boltzmann distribution with the most probable velocity $v_0 = 1$. We input this initial state into the graph network and use the output probabilities to roll new internal states for each node and construct a new graph, where we record the fraction of atoms not in the inactive state. Then, we input the new graph into the GNN and repeat this procedure 60,000 steps. In each step, we update the positions of each node using the ve-

locity: $x \to x + v\,t$. We use repelling boundary conditions, where the corresponding velocity component is reflected when a particle collides with the boundary. Lastly we average the resulting excitation trajectories of 50 runs with equivalent initial conditions (N, L). The number of steps is chosen such that the total predicted time is equivalent to $T = 60{,}000\ t = 150^{-1}$ since it takes approximately $T = 150^{-1}$ for the system to reach the critical state. Here $= 1$ (the decay rate of the active state) serves as the natural unit of (inverse) time in the system. The results are shown in Figure 1. All curves that have an initial density above the critical density $_c$ 1 (in the active phase) feature three distinct regimes: (i) a fast initial decay, (ii) a convergence to the same critical density $_s$ and (iii) a slow decay beyond that. All curves that start below the critical density (absorbing phase) exhibit only feature (iii). This is a hallmark characteristic of SOC behavior. When comparing the GNN predictions to the MCMC results in Figure 5 we see that the GNN shows a slower decay in phase (i). This effect is less pronounced the lower the initial density, which suggests the deviation stems from imperfect reproduction of the facilitation peak in Figure 4(a). Indeed, since the GNN underestimates the transition probability very close to $r = 1$, less facilitation events occur here than in the Monte Carlo baseline, causing a slower initial spread of excitations.

Ablation study of the GNN architecture. We chose a residual architecture for the GNN model. The two branches can be interpreted as the mean- eld and the uctuations branch. The former is a simple MLP and only receives as input the node embeddings, while the latter contains a graph convolutional layer and is supplied with both the node and edge embeddings. Since the uctuations branch alone could learn the MCMC update function, we perform an ablation study to test the performance changes of the model when switching off the mean- eld branch of the architecture.

Using the identical training procedure as for the full model, we nd that the test loss of the non-residual model is increased by a factor of two. The performance on the facilitation metric shown in Figure 4(a) remains unaffected, however learning the de-facilitation (4(b)) is signi cantly worse. The de-facilitation peak at $= 1$ is predicted at $r = 0{:}975$ (full model: $r = 1{:}005$) at a transition probability of $P(\text{g}|\text{a}) = 0{:}0026$, which is 36 smaller than the correct result (full model 10). Additionally, the training of the non-residual model appears significantly less stable as indicated by noisier training curves, including frequent large-magnitude jumps, which are absent in the training of the full model. In this sense, the residual architecture in form of a mean- eld branch, although not strictly necessary, represents an important ingredient to obtain high-quality predictions.

Discussion of the results. There are a few key implications we want to discuss based on the experimental results. We showed that the GNN learns the one-step time-evolution operator successfully and can display the hallmark characteristics of SOC dynamics. However, by construction, the GNN convolution operators are spatially translation invariant and local in nature. This implies that the long-range correlation of the SOC phase requires a time-dimension and is truly spatio-temporal as the spatial locality of the Graph Convolution does not allow to learn long-range spatial correlations.

A second aspect is that the spatial translation invariance of the Graph Convolution operator allows us to scale the time-evolution of the system to very large particle numbers, as all that matters is the local connectivity of the graph, not the overall particle number. Hence, the traditional $\mathcal{O}(N^2)$ scaling for MCMC does not apply.

Finally, the use of a graph data structure enables us to also scale over many densities, if the model has learned the facilitation metrics correctly. This is in contrast to standard simulations based on the discretization of the real-space coordinate. Due to the necessity of having a ne-grained grid, the simulated densities are typically small in that case in order to approximate continous systems. However, this limitation does not exist for GNNs as the relevant scale is the distance encoded in the edge attributes as a fraction of the natural distance R_F. Taken together with the particle number scaling, it might allow the study of dense systems with large particle numbers.

6. Summary

We present a new method to model the time dynamics of SOC systems using machine learning on graphs. We use a physically motivated, lightweight GNN architecture to learn the one-step time-evolution operator of each atom of the SOC system. In contrast to previous studies, we extend the GNN simulation toolkit to learn the dynamics of internal degrees of freedom, i.e. the node-state, in addition to the external degrees of freedom, i.e. the distance between atoms. We demonstrate high accuracy of the predictions via excellent agreement of the learned facilitation rate compared to exact ground-truth rates. Additionally, we show that the model generalizes to at least one order of magnitude more particles than seen during training without signi cant loss of accuracy, as well as higher and lower densities than it was trained on. Moreover, we showed that the model can successfully reproduce the decay to a common critical density and the separation of time-scales in the excitation trajectories, a hallmark characteristic of SOC, despite having been trained on single-timestep data only.

We leave the extraction of critical exponents and other SOC

elements as well as scaling to even larger particle numbers to future work. Additionally, our model could most likely benefit from hierarchical graph methods as presented in (Martinkus et al., 2021), which could reduce the computational complexity of larger systems. Although we see that the graph creation and distance calculation scales linearly for our GNN compared to the quadratic scaling of all-to-all Monte-Carlo algorithms, a quantitative analysis of the computational cost and runtime of our model compared to the currently used MCMC approaches is left to future work. Lastly, the Monte Carlo training data that we used could be replaced by real observations of SOC systems. In that case, the GNN model could learn directly the interaction laws and circumvent the time discretization of differential equations, thereby providing an independent check of the underlying theory.

Contributions. SO designed, implemented and ran the experiments using GNNs and drove the project development. DB implemented the classic MCMC methods to generate the data. WL supported the GNN network design and technical implementation as well as the training deployment. MF designed and supervised the project. JSO designed and supervised the project and experiments and supported the implementation. SO, DB and MF thank the DFG for their generous support under SFB TR 185, Project Number 277625399. The simulations were executed on the high performance cluster "Elwetritsch" at the TU Kaiserslautern which is part of the "Alliance of High Performance Computing Rheinland-Pfalz" (AHRP). We kindly acknowledge the support of the RHRK.

References

Bak, P., Tang, C., and Wiesenfeld, K. Self-organized criticality: An explanation of the 1/f noise. Phys. Rev. Lett., 59:381–384, Jul 1987. doi: 10.1103/PhysRevLett.59.381. URL https://link.aps.org/doi/10.1103/PhysRevLett.59.381.

Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K. Interaction networks for learning about objects, relations and physics. ArXiv, abs/1612.00222, 2016.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021. doi: 10.48550/ARXIV.2104.13478. URL https://arxiv.org/abs/2104.13478.

Drossel, B., Clar, S., and Schwabl, F. Exact results for the one-dimensional self-organized critical forest-fire model. Phys. Rev. Lett., 71:3739–3742, Dec 1993. doi: 10.1103/PhysRevLett.71.3739. URL https://link.aps.org/doi/10.1103/PhysRevLett.71.3739.

Falcon, W. and the PyTorch Lightning team. Pytorch lightning, 2019. URL https://www.pytorchlightning.ai.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. ArXiv, abs/1704.01212, 2017.

Gleeson, J. P., Ward, J. A., O'Sullivan, K. P., and Lee, W. T. Competition-induced criticality in a model of meme popularity. Phys. Rev. Lett., 112:048701, Jan 2014. doi: 10.1103/PhysRevLett.112.048701. URL https://link.aps.org/doi/10.1103/PhysRevLett.112.048701.

Helmrich, S., Arias, A., Lochead, G., Wintermantel, T. M., Buchhold, M., Diehl, S., and Whitlock, S. Signatures of self-organized criticality in an ultracold atomic gas. Nature, 577:481–486, 2020.

Henderson, R., Clevert, D.-A., and Montanari, F. Improving molecular graph neural network explainability with orthonormalization and induced sparsity. In ICML, 2021.

Jim'enez-Luna, J., Grisoni, F., and Schneider, G. Drug discovery with explainable artificial intelligence. Nat. Mach. Intell., 2:573–584, 2020.

Jin, J. and Voth, G. A. Ultra-coarse-grained models allow for an accurate and transferable treatment of interfacial systems. Journal of chemical theory and computation, 14 (4):2180–2197, 2018.

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. Nature, 596:583 – 589, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2015.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In International Conference on Machine Learning, pp. 2688–2697. PMLR, 2018.

Lemos, P., Jeffrey, N., Cranmer, M., Ho, S.-L., and Battaglia, P. W. Rediscovering orbital mechanics with machine learning. ArXiv, abs/2202.02306, 2022.

Lukin, M. D., Fleischhauer, M., Cote, R., Duan, L., Jaksch, D., Cirac, J. I., and Zoller, P. Dipole blockade and quantum information processing in mesoscopic atomic ensembles. Physical review letters, 87(3):037901, 2001.

Malamud, Morein, and Turcotte. Forest fires: An example of self-organized critical behavior. Science, 281 5384: 1840–2, 1998.

Martinkus, K., Lucchi, A., and Perraudin, N. Scalable graph networks for particle simulations. In AAAI, 2021.

Noé, F., Tkatchenko, A., Müller, K.-R., and Clementi, C. Machine learning for molecular simulation. Annual review of physical chemistry, 2020.

Olsson, S. and Noé, F. Dynamic graphical models of molecular kinetics. Proceedings of the National Academy of Sciences, 116(30):15001–15006, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. ArXiv, abs/2010.03409, 2021.

Prakash, S. K. A. and Tucker, C. S. Graph network for learning bi-directional physics. 2021.

Rhodes, C. J. and Anderson, R. M. Power laws governing epidemics in isolated populations. Nature, 381:600–602, 1996.

Saffman, M., Walker, T. G., and Mølmer, K. Quantum information with rydberg atoms. Reviews of modern physics, 82(3):2313, 2010.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks. ArXiv, abs/2002.09405, 2020.

Scully, M. O. and Zubairy, M. S. Quantum Optics. Cambridge University Press, 1 edition, September 1997. ISBN 9780521435956 9780521434584 9780511813993. doi: 10.1017/CBO9780511813993. URL https://www.cambridge.org/core/product/identifier/9780511813993/type/book.

Shlomi, J., Battaglia, P. W., and Vlimant, J. R. Graph neural networks in particle physics. Mach. Learn. Sci. Technol., 2:21001, 2021.

Sornette, A. and Sornette, D. Self-organized criticality and earthquakes. EPL, 9:197–202, 1989.

Wintermantel, T., Buchhold, M., Shevate, S., Morgado, M., Wang, Y., Lochead, G., Diehl, S., and Whitlock, S. Epidemic growth and griffiths effects on an emergent network of excited atoms. Nature Communications, 12 (1):1–6, 2021.

Zhang, M., Wu, S., Yu, X., Liu, Q., and Wang, L. Dynamic graph neural networks for sequential recommendation. IEEE Transactions on Knowledge and Data Engineering, 2022.

A. Derivation of Rate Equations from Optical Bloch Equations / State-transition description of a Driven Rydberg Gas

In this section, we shortly summarize the derivation of the rate equations discussed in the main text. To help readers draw parallels with the physics literature, we will use the subscript r for the active (rather: Rydberg) state and the subscript g for the ground state. The derivation considers the density matrix for a two-level system consisting of the ground and Rydberg states that is coupled to an external light field which we assume to be classical (not in the single-photon limit). Since we take the full quantum mechanics of the two-level atom into account, this is called the *semi-classical* approach. For a two level system with atom-light interaction in the semi-classical limit, the optical Bloch equations are then given as (Scully & Zubairy, 1997)

$$\dot{}_{rr} = \quad i \quad (\quad _{rg} \quad \quad _{gr}) \quad \quad _{rr}; \tag{8}$$

$$\dot{}_{rg} = \quad \quad _{rg} + i \quad _{rg} \quad i \quad (\quad _{rr} \quad \quad _{gg}); \tag{9}$$

where $_{ij}$ represents the entries of the density matrix. For large dephasing , $\dot{}_{rg}$ can be adiabatically eliminated. This gives

$$_{rg} = \frac{i \quad (\quad _{rr} \quad \quad _{gg})}{\quad + i} \tag{10}$$

$$\dot{}_{rr} = \quad \frac{2 \quad ^2}{^2 + \quad ^2}(\quad _{rr} \quad \quad _{gg}) \quad \quad _{rr} \tag{11}$$

$$\quad \quad \uparrow \quad _{rr} + \quad \downarrow \quad _{gg}: \tag{12}$$

This gives the transition rates $\uparrow$ from the ground to excited state and $\downarrow$ from the excited to the ground state. To simulate these rates, we choose the microscopic parameters in such a way as to achieve a sufficient separation of timescales, see table 2. Using these values, we find the following separation of timescales:

parameter	value
=	2000
=	20
	1 (Unit of time)
=	20
t	0:0025
b	0:5

Table 2. Microscopic parameters used in the creation of Monte-Carlo data. We use units where $\hbar = 1$.

$$_{\text{facil}} = \frac{2 \quad ^2}{\quad} = 40 \tag{13}$$

$$b \quad = 0{:}5 \quad ; \tag{14}$$

$$_{\text{non-int}} = \frac{2 \quad ^2}{^2 + \quad ^2_0} \quad 0{:}004 \quad ; \tag{15}$$

$$_{\text{facil}} = 80\, b \tag{16}$$

$$b \quad \quad 125 \quad _{\text{non-int}}: \tag{17}$$

To observe SOC behavior, a separation of approximately a factor of 100 between each rate is recommended.

B. Monte-Carlo simulation of the Training Data

B.1. Creation of Training Data

As outlined in the main text, the training data for the GNN is generated using classical Monte-Carlo methods where each training instance consists of X, a matrix of shape $(N; N_f)$, and the ground truth Y, a matrix of shape $(N; 3)$. $Y_{i;s} = (P_i(gjs_i); P_i(ajs_i); P_i(0js_i))$ contains for each node i with the internal state s_i the probabilities to change to or remain in each of the three states in the next timestep.