



Backpropagation of Approximate Gradients for Stochastic Lattice Models

THESIS MSC ARTIFICIAL INTELLIGENCE

Author:
Jan D. SCHERING

Supervisor:
dr. Sander KEEMINK
dr. Johannes TEXTOR

Second reader:
dr. Inge WORTEL

Abstract

Stochastic lattice models (s-LMs) are efficient computational tools for simulating non-equilibrium and morphogenetic systems. Yet, to date, no efficient method for fitting s-LM parameters to a set of objectives has been devised. The current state-of-the-art method for fitting s-LMs is Approximate-Bayesian Computation, a *gradient-free* general-purpose Bayesian inference method for estimating posterior distributions. While powerful, it is computationally expensive and quickly becomes intractable for high-dimensional parameter spaces. *Gradient-based* methods, particularly gradient descent algorithms making use of *backpropagation* could be significantly more efficient by directly fitting the parameters without needing to estimate the posterior. However, s-LMs are non-differentiable due to stochasticity and discreteness, preventing the use of backpropagation. Recent advances in Artificial Intelligence have introduced methods to circumvent this issue. The *reparameterization trick* re-enables backpropagation for stochastic models. *Straight-through Estimation*, on the other hand, enables backpropagation of *approximate gradients* for discrete models. We investigate the application of these methods for fitting four different s-LMs with respect to four common types of objectives. The results are promising, showing that *backpropagation of approximate gradients* can successfully be applied to various s-LM tasks. However, some challenges remain that will require further investigation.

July 12, 2023

Contents

1	Introduction	2
2	Research Statement	3
3	Results	3
3.1	Trajectory fitting of a 1-parameter Susceptible-Infected s-LM via gradient-based optimization	3
3.2	Stability fitting of a morphogenetic reaction-diffusion s-LM for pattern generation	5
3.3	State fitting of initial lattice configurations from images via backpropagation of approximate gradients	7
3.4	Statistics fitting of a 1-Pixel Random Walk CPM	9
4	Relevant Works	11
5	Summary & Discussion	11
6	Conclusion	13
7	Methods	13
7.1	Backpropagating approximate gradients through s-LMs	13
7.1.1	Dealing with stochastic nodes	14
7.1.2	Dealing with discrete nodes	15
7.1.3	Dealing with stochastic categorical nodes	15
7.1.4	Implementing the approach in software	15
7.2	1-parameter SI s-LM	16
7.3	Malevanets-Kapral s-LM	16
7.3.1	Diffusion	16
7.3.2	Reaction	17
7.4	Gradient-based foreground-background segmentation	18
7.5	1-Pixel Random Walk Cellular Potts model	19

1 Introduction

Stochastic lattice models (s-LMs) provide an efficient tool for modelling non-equilibrium and morphogenetic systems (Haselwandter and Vvedensky, 2008). They have been applied successfully to several fields including Oncology (Szabó and Merks, 2013; Rubenstein and Kaufman, 2008), Hydrodynamics (Chopard and Droz, 2005; Rothman and Zaleski, 2004), Molecular Morphogenesis (Dab et al., 1991; Malevanets and Kapral, 1997) and Artificial Life (Chan, 2018). Commonly, s-LMs are used for phenomenological studies of real-world phenomena such as Cell migration in Cell Biology (Wortel et al., 2021; Scianna and Preziosi, 2021). A particularly popular choice for modelling non-equilibrium dynamics in Cell Biology is the Cellular Potts model (CPM) for cells and tissues introduced in Graner and Glazier (1992). Among other things, the CPM is considered the standard model for the simulation of tumor growth (Szabó and Merks, 2013). To conduct phenomenological studies with s-LMs such as the CPM, the parameters of the model need to be adjusted such that model behavior accurately reflects the target behavior. Yet, despite the growing popularity of s-LMs for the study of real-world phenomena, efficiently fitting the model parameters to data remains an open challenge. As a result, these parameters often are tuned by hand through educated trial and error (Wortel et al., 2021; Voss-Böhme, 2012).

S-LMs are discrete models that stochastically map the configuration of a lattice from one timepoint to the next (Fig. 1). The mapping function f_{Θ} is typically parameterized by some set of parameters Θ that guide the behavior of the model. Fitting the parameters of s-LMs to a given objective is challenging, as their discreteness and stochasticity make them non-differentiable. Non-differentiability prevents *backpropagation* Rumelhart et al. (1986) of the loss with respect to the model parameters. Backpropagation is the basis of *gradient descent* optimization algorithms, which are the standard method for efficiently fitting models with high-dimensional parameter spaces. An example of this is Deep Neural Network models with millions of parameters, which are efficiently optimized through gradient descent algorithms (Larochelle et al., 2009).

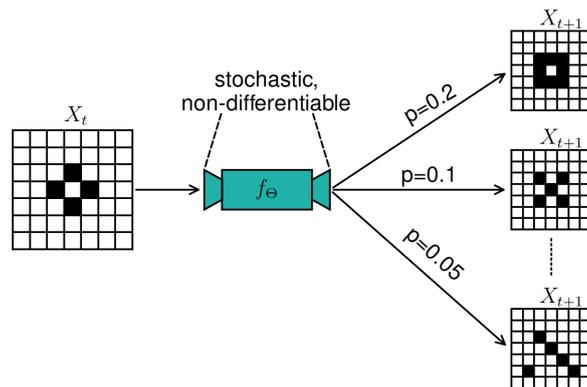


Figure 1: S-LMs evolve lattice states stochastically over time. They are computationally efficient, attractive models for several systems. However, stochasticity and discreteness make them hard to optimize.

Stochastic lattice models (blue) are stochastic mapping functions f_{Θ} , parameterized by a set of parameters Θ . They take in the current ‘state’ of the lattice X_t (i.e. the configuration of the lattice sites at time t) and sample X_{t+1} from a distribution over possible outcomes. They are non-differentiable, which makes parameter optimization challenging.

Instead, general-purpose *gradient-free* strategies are used to fit s-LMs but their generality comes with various downsides. Naïve strategies such as grid search, stochastic optimization, or genetic algorithms quickly break down outside the realm of toy problems. The current state-of-the-art fitting method for s-LMs is Approximate Bayesian Computation (ABC) (Alamoudi et al., 2023, 2022; Wang et al., 2022). ABC is a powerful general-purpose algorithm for approximate Bayesian inference (Sisson et al., 2018, 2007). While ABC has been employed successfully for several tasks (Durso-Cain et al., 2021; Carr et al., 2021; Beaumont et al., 2002), it is computationally expensive due to undirected sampling and the need to estimate the full posterior distribution of the parameters. For large numbers of parameters, ABC quickly breaks down due to the curse of dimensionality (Curtis and Lomax, 2001). Hence, enabling the use of gradient-based optimization would provide a way to efficiently fit s-LMs to data in a directed fashion without the need to estimate the full posterior distribution.

To this end, recent developments in Artificial Intelligence, especially in the area of Spiking Neural

Networks have made significant advances in applying gradient descent algorithms to non-differentiable models. The advances concern both stochasticity and discreteness. Two key advancements were made towards enabling gradient descent algorithms for non-differentiable models: The *reparameterization trick* enables backpropagation for stochastic models by making sampling operations independent of the model parameters (Jang et al., 2016; Kingma and Welling, 2013; Bengio et al., 2013). *Straight-through Estimation* (STE) enables backpropagation for discrete models by replacing the partial derivatives of non-differentiable transformations with custom functions, producing *approximate gradients* that estimate the true gradient (Hu et al., 2021; Neftci et al., 2019; Bengio et al., 2013). Combining the two methods, gradient descent algorithms can be applied to stochastic discrete models via *backpropagation of approximate gradients*. In this study, we seek to apply backpropagation of approximate gradients specifically to tackle the challenge of efficiently fitting s-LM parameters to data.

2 Research Statement

S-LMs are attractive computational tools for simulation at scale, but efficiently fitting them to data remains an open challenge that often requires hand-tuning of the parameters (Wortel et al., 2021; Voss-Böhme, 2012). Currently, gradient-free ABC is the state-of-the-art method for fitting s-LMs to data (Alamoudi et al., 2023, 2022). While it has been successfully applied to several tasks (Durso-Cain et al., 2021; Carr et al., 2021; Beaumont et al., 2002), it is inherently inefficient due to trial-and-error sampling and the need to estimate the whole posterior distribution (Curtis and Lomax, 2001). Gradient-based optimization techniques, particularly gradient descent optimization algorithms, would provide an efficient way of fitting parameters without the need to estimate the posterior distribution. Currently, however, stochasticity and discreteness of the model prevent backpropagation of the gradient.

The field of AI, faced with similar challenges, has made significant progress in recent years pertaining to the use of gradient descent for non-differentiable models. Namely, the reparameterization trick enables backpropagation through stochastic models (Kingma and Welling, 2013) while STE provides approximate gradients for discrete models (Neftci et al., 2019). In this study, we attempt to fit s-LMs to four common types of objectives via *backpropagation of approximate gradients*, combining reparameterization and STE. To this end, the following research question is put forward:

“Is it possible to fit s-LM parameters to data via backpropagation of approximate gradients?”

Assuming the viability of gradient-based optimization, we hypothesize that backpropagation of approximate gradients can be used to:

H1 : fit s-LM parameters to a dataset of observed trajectories. (*Trajectory fitting*)

H2 : fit s-LM parameters towards generating stable patterns for morphogenetic systems. (*Stability fitting*)

H3 : fit s-LM lattices to objective functions. (*State fitting*)

H4 : fit s-LM parameters to a set of summary statistics. (*Statistics fitting*)

To answer the research question, a series of experiments are performed for each of the hypotheses. That is, for each hypothesis a simple s-LM model is chosen and a parameter-fitting task devised. We report on the results of applying backpropagation via approximate gradients to each of the fitting tasks and interpret their support or opposition to the hypotheses.

3 Results

3.1 Trajectory fitting of a 1-parameter Susceptible-Infected s-LM via gradient-based optimization

Trajectory fitting is the task of fitting a model to a dataset of sample trajectories. More specifically, given a lattice state X_t , the goal is to accurately predict the next state \hat{X}_{t+1} . On average, the predicted state transitions (X_t, \hat{X}_{t+1}) should match the observed transitions (X_t, X_{t+1}) . Trajectory fitting is frequently applied to CPMs, where the goal is to model the behavior of cells over time. Currently, however, parameters need to either be tuned by hand (Voss-Böhme, 2012) or through applying ABC

(Alamoudi et al., 2023, 2022), both of which are costly. This experiment applies trajectory fitting via backpropagation of approximate gradients to a simple 1-parameter s-LM of a Susceptible-Infected (SI) system. Successful fitting of the model parameter provides evidence in favor of hypothesis ($H1$).

The SI s-LM simulates the diffusive spread of a news item through a population, adapting a Markov-chain model introduced in (Conlisk, 1976). In this spatial reformulation, every lattice site represents a member of the population. Each lattice site is binary, indicating *awareness* (=1) or *unawareness* (=0) of the news item. News items can be spread from *aware* lattice sites to *unaware* sites that they are touching (known as the *Moore neighborhood*), turning them aware. For simplicity, aware lattice sites cannot turn back to being unaware. The parameter β denotes the likelihood of successfully spreading the news item to an unaware neighbor and thus drives the rate of the news item spreading through the population (Fig. 2A). This experiment aims to fit β to a dataset of observed trajectories via backpropagation of approximate gradients.

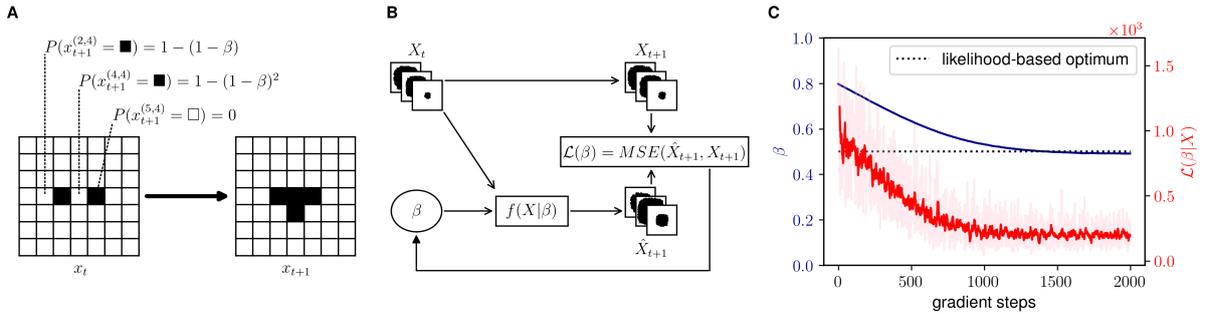


Figure 2: Gradient-based trajectory fitting of a 1-parameter S-LM for susceptible-infected systems closely matches the likelihood-based optimum.

A Aware lattice sites (black) spread to unaware neighbors (white) with probability $1 - (1 - \beta)$. The total likelihood of turning aware is a binomial experiment with one trial per aware neighbor. Once aware, lattice sites cannot turn unaware. **B** Starting from a dataset of transitions (X_t, X_{t+1}) , the model is used to predict $\hat{X}_{t+1} = f_\beta(X_t)$. Then, the pixel-wise Mean-Squared Error between prediction \hat{X}_{t+1} and observation is backpropagated and used to update β . **C** Performing minibatch Stochastic Gradient Descent (SGD) to minimize the MSE (red) w.r.t. β (blue) closely matches the results obtained by directly optimizing the likelihood function (black). Rolling mean averaging with a window size of 10 is applied to smooth the loss curve. The true loss is indicated in the background. We use a learning rate of $1e^{-7}$ and a minibatch size of 32.

The dataset is collected from 100 reference simulations of 30 steps each, using 64×64 regular lattices with a periodic torus. Each reference simulation is initialized with a single aware site in the center of the lattice. Additional noise is induced by resampling $\beta_t \sim \mathcal{N}(0.5, 0.1)$ at every timepoint t of the simulations. A sliding window algorithm transforms the simulated sequences into a dataset of transitions (X_t, X_{t+1}) , from which minibatches $\{(X_t, X_{t+1})^i\}_{i=0}^n$ of size $n = 32$ are randomly drawn. In total, the dataset consists of 2900 state transitions.

For gradient optimization, minibatch Stochastic Gradient Descent (SGD) is applied. Minibatch SGD is a variation of gradient descent suited for large datasets. Instead of optimizing over the whole dataset at once, small batches of the dataset are randomly drawn and used for optimization (Li et al., 2014). At each gradient step, a new minibatch $\{(X_t, X_{t+1})^i\}_{i=0}^n$ is sampled. The batch of inputs $X_t^{(i)}, i = 0, \dots, n$ is passed through the model to predict the next timepoint for each sample. Then, the pixel-wise Mean-Squared Error (MSE) between predictions and observations is calculated. Finally, the loss is backpropagated through the model and β is updated. A schematic of the training process is shown in Fig. 2B. For the SGD, a learning rate of $1e^{-7}$ was empirically chosen.

To judge the quality of the gradient-based β estimate, the second part of the experiment fits β to the dataset through direct usage of the likelihood function. This is achieved by applying Maximum Likelihood estimation to β , again making use of minibatch SGD. Maximum Likelihood estimation provides a way to directly estimate the mode of the posterior distribution by finding the parameters that minimize the negative log-likelihood of the dataset (Myung, 2003). The likelihood function, however, is only tractable for simple toy examples and is thus in most cases unavailable for parameter fitting (Mengersen et al., 2013).

Fig. 2C shows the results of the fitting process. After approximately 2000 steps of SGD, β converges to a value of ~ 0.47 . This closely matches the estimate obtained through the likelihood-based fitting of β , which was ~ 0.48 . Subsequently, the results provide evidence for hypothesis $H1$, showing that trajectory

fitting via backpropagation of approximate gradients can successfully be applied to s-LMs. However, the gradient-based estimate does not exactly match the likelihood-based estimate. As both were fit using minibatch gradient descent, the difference may be simply due to the noise of the optimization process. On the other hand, it could also indicate that the gradient approximation may induce a slight bias.

3.2 Stability fitting of a morphogenetic reaction-diffusion s-LM for pattern generation

Stability fitting is the task of finding parameter configurations for morphogenetic s-LMs from which stable patterns arise. The formation of stable patterns from an unstable initial configuration over time is an important aspect of biological *morphogenesis*. Starting with Turing in 1952, several reaction-diffusion models for the emergence of stable 'Turing' patterns have been proposed (Ali and Saleem, 2023; Li et al., 2015; Lengyel and Epstein, 1992). These models are usually defined as a coupled partial differential equations (PDEs) system. For simulation purposes, however, it is common to discretize and convert these PDEs into s-LMs (Wang et al., 2011; Malevanets and Kapral, 1997). Independent of the model choice, Turing patterns have been shown to only emerge for a small subset of the parameter space which is generally hard to identify (Scholes et al., 2019; Vittadello et al., 2021). In this study, stability fitting via backpropagation of approximate gradients is applied to a morphogenetic reaction-diffusion s-LM. Successful application of the method to this task would provide strong evidence for hypothesis (*H2*).

Specifically, we consider the "Malevanets-Kapral" reaction-diffusion s-LM introduced by Malevanets and Kapral (1997). The Malevanets-Kapral model is a spatial reformulation of the Fitzhugh-Nagumo equations (FitzHugh, 1961), which simulates the spatial reaction-diffusion of two chemical species over time. Section 7.3 provides an in-depth description of the model. Generally, the Malevanets-Kapral s-LM models per-chemical concentration of two chemical species A, B in space as a 2-layer square lattice $L = (A, B)$ (Fig. 9A). The two sub-lattices represent the same spatial dimension. That is, $A^{(i,j)}$ represents the concentration of species A at the lattice site (i, j) , whereas $B^{(i,j)}$ denotes the concentration of species B at the same lattice site. Every lattice site has a maximum capacity of N molecules per species. At every time step, the model simulates:

1. Independent diffusion of A and B, governed by the diffusion coefficients D_A, D_B
2. The reaction of A and B with each other, governed by the reaction rates k_1, k_2, k_3

Additionally, a hyperparameter γ is used as a timescale of the reaction process in relation to the diffusion process. By iteratively applying the two steps to the lattice, the reaction-diffusion process of the two chemicals is simulated. For a subset of parameters, this interaction generates stable labyrinthine patterns. An example of this is shown in Fig. S1. A series of three experiments is performed on the Malevanets-Kapral model to investigate the viability of stability fitting via backpropagation of approximate gradients:

- E1* : Joint optimization of the model parameters $\Theta = (D_A, D_B, k_1, k_2, k_3)$
- E2* : Optimization of the reaction rates k_1, k_2, k_3 for fixed diffusion coefficients
- E3* : Optimization of the diffusion coefficients D_A, D_B for fixed reaction rates

Performing the experiments for fixed subsets of parameters is done to isolate and understand the impact of different types of parameters on the fitting process.

To perform the experiments, a general strategy for gradient-based stability fitting was devised. For this purpose, the assumption is made that the ability to *maintain* stable patterns implies the ability to *form* stable patterns. Pattern formation performance is hard to quantify (Vittadello et al., 2021). Pattern maintenance, on the other hand, is very easy to quantify as we simply need to compare the state before and after simulation. Therefore, pattern maintenance is used as an alternative (*surrogate*) objective. To validate this assumption, pattern formation capability is tested periodically throughout the fitting process. That is, we perform *pattern formation tests* by periodically using the current parameters to run a set of simulations starting from a uniform, unstable state and observing if stable patterns are formed.

A schematic of the fitting process is provided in Fig. 3. Before performing the experiments, the model is run with a reference configuration of parameters Θ_{ref} that is known to produce stable patterns. The resulting stable pattern is denoted by X_{ref} . For each gradient step, the evolution of X_{ref} is simulated

for τ timesteps to produce $X_{ref+\tau}$. Afterward, we calculate the pixel-wise MSE of $(X_{ref}, X_{ref+\tau})$ to determine how well the pattern was maintained. The MSE is then backpropagated through the model to update the parameters.

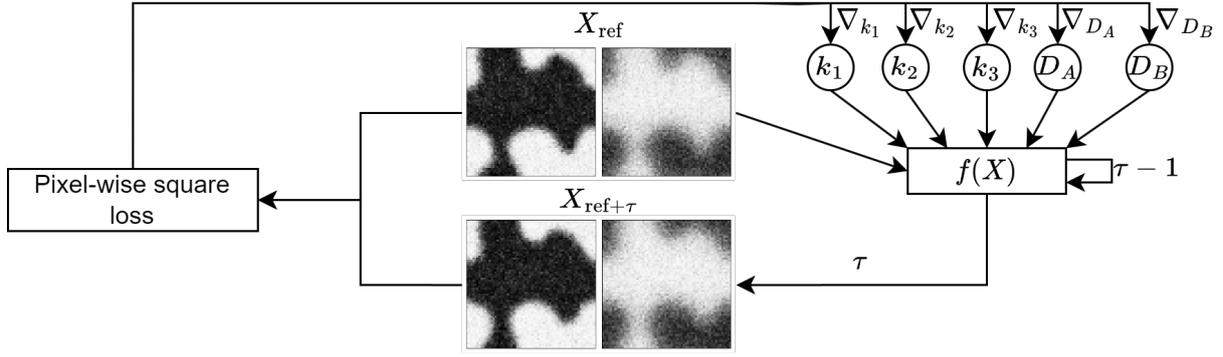


Figure 3: Stability fitting approach for the Malevanets-Kapral model using gradient-based optimization.

Maintenance of a stable pattern over time is used as a surrogate for the ability to form stable patterns. At every iteration, the evolution of a stable pattern X_{ref} is simulated for τ steps with the current parameter configuration. The pixel-wise square difference between $X_{ref+\tau}$ and X_{ref} is calculated and used to update the model parameters via backpropagation. A square difference of zero would indicate perfect maintenance of the pattern over time.

Each experiment was performed on 64×64 regular grid lattices with a periodic torus. For optimization, standard gradient descent is applied with a learning rate of 0.05, maintenance time $\tau = 500$, reaction timescale $\gamma = 0.005$, and maximum capacity $N = 50$. For experiment 2, we fixed $k_1 = 0.98$, $k_2 = 0.1$ and $k_3 = 0.2$, based on pattern forming parameters found in Malevanets and Kapral (1997). Similarly, for experiment 3 we fixed $D_A = 0.1$ and $D_B = 0.4$. The results of the experiments are illustrated in Fig. 4. Plotted above the graphs are samples taken from the pattern formation test at different points in the fitting process. The samples show the distribution of species A after fifty-thousand simulation steps, starting from a uniformly initialized lattice. For simplicity, species B is omitted. The top row of graphs shows how the parameters evolve throughout the fitting process. Below, the corresponding gradients for each parameter are shown as a rolling average with a window size of 30. Finally, the last row shows the corresponding loss curve per experiment.

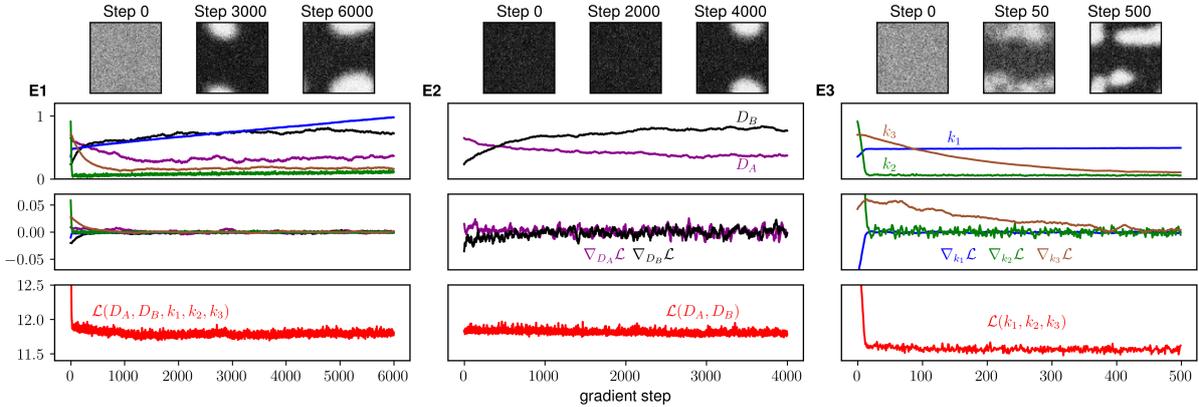


Figure 4: Morphogenetic parameter configurations of the Malevanets-Kapral model are found through gradient-based stability fitting.

E1 Joint fitting of the reaction rates and diffusion coefficients. **E2** Fitting of the diffusion coefficients for fixed reaction rates. **E3** Stability fitting applied to the reaction rates, with fixed diffusion coefficients. **Images** samples of the pattern formation test taken at different points of the fitting process. Samples show the concentration of the A species after fifty-thousand simulation steps, starting from a uniform distribution of A and B chemicals. **Top plots** Evolution of the parameters being optimized throughout the fitting process. **Middle plots** Rolling mean of the gradients with window size X . Indicated in the background of the plot are the true gradients. **Bottom plots** Pixel-wise square difference throughout the fitting process.

For experiment *E1*, 6000 steps of gradient descent are shown in Fig. 4E1. Samples of the pattern formation test are shown for the parameters before optimization, after 3000 steps, and after 6000 steps of gradient descent respectively. For the initial parameters, no patterns are formed and the concentration of species A appears to have diffused into white noise. After 3000 steps of gradient descent, the concentration of species A in the pattern formation test appears to be a lot higher and a small pattern in concentration is forming. Finally, after 6000 steps of gradient descent, the pattern formation test shows that clear patterns in the spatial distribution of species A are formed. Thus, the gradient-based fitting method has successfully identified a set of morphogenetic parameters.

The loss curve for experiment 1 shows that the loss drops significantly within the first few steps of gradient descent. Compared with the plotted gradients, this reduction is largely driven by the reaction rate k_2 . For the first few steps of gradient descent, the gradient of k_2 is significantly higher than that of the other parameters. The plot of the parameter values over time further supports this, showing that k_2 quickly drops from a high initial value of 0.82 to a value close to 0 within a few steps of gradient descent. After the initial drop, the loss curve is noisy but generally trending downward and slowly appears to converge after 5000 steps of gradient descent.

The results of experiment *E2* are summarized in Fig. 4E2. The goal of this experiment was to isolate and investigate the impact of the diffusion coefficients on the fitting process. Similar to experiment 1, Fig. 4E2 illustrates 4000 steps of gradient descent and provides samples of the pattern formation test taken before optimization, after 2000 gradient steps, and after 4000 gradient steps. When compared to the pattern formation samples of experiment 1, some interesting details stand out. Foremost, the initial parameter configuration in experiment 2 does not produce a white noise pattern. Instead, the concentration of species A appears to converge to maximum capacity for all lattice sites. This seems to be a stable state, however, no pattern is formed. After 2000 steps of gradient descent, the pattern formation test shows that this starts to change, and small stable patterns in the distribution of species A have formed. Finally, after 4000 steps of gradient optimization, the pattern formation test shows a clear stable pattern in the spatial distribution of species A.

Comparing the loss curve of experiment 2 to that of experiment 1 supports that the reaction rates have a much higher impact on the loss. With the reaction rates fixed to a morphogenetic configuration, the initial loss is significantly lower and does not show the early drop observed in experiment 1. Instead, the loss is noisy but slightly trending downwards until it converges after roughly 3000 gradient steps. The same can be observed from the gradients of the diffusion coefficients which start comparatively small and then slowly and noisily move towards zero, converging after around 3000 steps.

The diffusion coefficients initially start with $D_A > D_B$. After around 500 gradient steps, this relation is reversed and turns to $D_B > D_A$ with the ratio between the two increasing over time until D_B is roughly twice as large as D_A . Malevanets and Kapral (1997) note in their analysis of the system that the ratio D_B/D_A needs to exceed a critical value for labyrinthine stable patterns to form. This is clearly shown by the results, where patterns only start to form after about 2000 steps, as the ratio becomes large enough.

Finally, the results of experiment *E3* are shown in Fig. 4E3, similar to the previous two experiments. Standing out immediately is that fitting the reaction rates to generate stable patterns takes significantly fewer gradient steps than the diffusion coefficients. After only 50 steps of gradient descent, the pattern formation test shows that the first patterns start forming, albeit not appearing fully stable. Then after 500 gradient steps, clear stable patterns are formed. Comparing the results of experiment 2 and experiment 3, fitting the reaction rates appears to be comparatively significantly easier than fitting the diffusion coefficients. To further visualize the fitting process, experiment *E3* was repeated with custom learning rates for each reaction rate, and the full pattern formation test was animated per 50 steps of gradient descent. The resulting video visualizing the fitting process is shown in Movie S1.

For all experiments, gradient-based optimization has successfully identified parameter configurations from which stable patterns emerge. The results provide strong support for hypothesis *H2* through a successful example of stability fitting a s-LM via backpropagation of approximate gradients.

3.3 State fitting of initial lattice configurations from images via backpropagation of approximate gradients

State fitting is the challenge of initializing lattices to optimize a set of objectives. An example of this is the initialization of CPM lattices based on a reference cell image. To reproduce the reference image accurately on the CPM lattice, pixels that belong to cells need to be differentiated from those that do not belong to cells. More generally, *foreground-background segmentation* is the task of predicting

for each pixel of an image whether it belongs to the foreground or the background of the image. The result of foreground-background segmentation is a binary *mask* of predictions. This mask can then either be further post-processed (for example by clustering foreground objects in the mask) or directly used for s-LM simulations. Finding a generally robust foreground-background segmentation strategy is challenging due to e.g. significant differences in illumination statistics between different domains (Long and Keles, 2018). Hence, the development of robust foreground-background segmentation algorithms is an active area of research with recent years seeing an influx of proposed Artificial Intelligence methods (Lim and Keles, 2020; Long and Keles, 2018; Wang et al., 2017). In this study, foreground-background segmentation via backpropagation of approximate gradients is used to fit lattice states to reference images. Successful application of the method to the state fitting task would provide support for hypothesis ($H3$).

To this end, a simple strategy to fit the segmentation mask via backpropagation of approximate gradients was devised. The approach is described in more detail in Sec. 7.4. First, images are pre-processed by converting them to grayscale, followed by centering and normalization of the pixel intensities to the range $[0,1]$. To obtain a segmentation mask from the centered grayscale image, simple thresholding is applied: Pixels with an intensity above zero are classified as foreground, while the rest is classified as background. To fit the segmentation mask, the pixel intensities of the grayscale image need to be optimized such that background pixels are pushed towards zero while keeping foreground pixels larger than zero. With this in mind, a custom loss function was defined to capture this objective. Finally, gradient descent is used to minimize the loss function with respect to the pixel intensities.

To judge the performance of this method, it is applied to images from two separate domains. In experiment 1, the method is applied to a natural input image taken from the DIS5K dataset introduced in Qin et al. (2022). Then, in experiment 2 the method is applied to a sample cell image taken from the BBBC005v1 Broad Bioimage Benchmark Collection (Ljosa et al., 2012). For experiment 1 on the DIS5K seadragon image, standard gradient descent was applied to minimize $\mathcal{L}(X_{pred})$ with a learning rate of 0.01, $\lambda_a = 0.6$, $\lambda_b = 0.05$, $\lambda_c = 0.15$ and $\lambda_d = 0.5$. For experiment 2 on the BBBC005v1 cell imaging, standard gradient descent was applied with the same learning rate and $\lambda_a = 0.6$, $\lambda_b = 0.05$, $\lambda_c = 0.3$ and $\lambda_d = 0.8$. The λ parameters are scaling factors of the loss function and are explained in detail in Sec. 7.4.

The results of both experiments are illustrated in Fig. 5. We report segmentation performance via *dice score* between predicted segmentation masks and the ground-truth segmentation masks throughout the fitting process. Dice scores (also known as F_1 score) are the harmonic mean of precision (positive predicted value) and recall (true positive rate). Thus, they represent a balanced score combining the two metrics. A dice score of 1 indicates perfect prediction while a dice score of 0 denotes that there is no overlap between prediction and ground-truth. In addition to the dice score, the loss per gradient step is plotted for each image.

Fig. 5A illustrates segmentation performance for experiment 1. The fitting process starts from a very low initial dice score of ~ 0.30 indicating that no segmentation is happening. The initial foreground mask confirms this, showing that initially every pixel is categorized as foreground (white). The graph plots the loss and corresponding dice score over 200 steps of gradient descent. Within the first 50 gradient steps, the loss decreases almost linearly, while the dice score improves close to linearly to a score of ~ 0.8 . This implies a significant overlap of the predicted segmentation with the ground truth. The corresponding segmentation mask confirms this, clearly showing the shape of the seadragon. At this point of the fitting process, a few large background artifacts still get misclassified. For the next 250 steps of gradient descent, the slope of the loss curve decreases and seems to slowly converge. At the same time, the dice score plateaus at a value of 0.85. The corresponding foreground mask after 300 steps of gradient descent captures the seadragon quite well. The remaining misclassifications stem from some bright background artifacts and some dark spots in the skin pattern of the seadragon. An animation of the full fitting process is provided in Movie S2.

Fig. 5B illustrates segmentation performance for the arguably more relevant scenario of cell segmentation performed in experiment 2 for a sample from the BBBC005v1 dataset. Initial segmentation performance is a bit higher than for the seadragon but still very low with a dice score of 0.45. In the initial foreground masks, cells are visible but the mask is very noisy and the predicted cell segments are overly large. After only 20 steps of gradient descent, the noise has been successfully removed and cell sizes in the foreground mask match those of the input image. The dice score here is 0.95, indicating an almost perfect match with the ground truth. Over the next 80 steps of gradient descent, the loss does decrease further, while the dice score instead decreases slightly and plateaus at 0.94, still indicating an almost exact match with the ground truth. A possible reason for the slight decrease in dice score despite

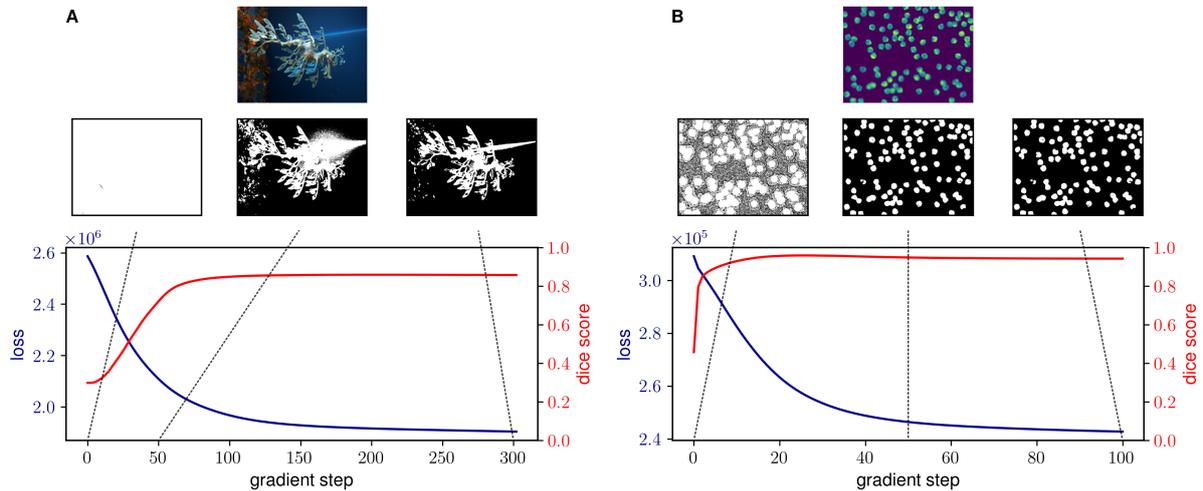


Figure 5: State fitting via gradient-based optimization achieves promising results for sample images from two separate domains.

A State fitting applied to a natural image of a seadragon (top). Foreground-background (white-black) masks after 0, 50, and 300 steps of gradient descent respectively are visualized. Below, the dice score and loss throughout the fitting process are shown. **B** State fitting applied to cell imaging (top). Foreground-background masks after 0, 50, and 100 steps of gradient descent respectively are visualized. Below, the dice score and loss over gradient steps are plotted.

the loss going down, is that the loss function does not fully capture the segmentation objective. Thus, overfitting to the loss function comes at a loss of segmentation performance. An animation of the full fitting process is provided in Movie S3.

Overall, the results of this experiment provide evidence for hypothesis $H3$ by successfully applying backpropagation of approximate gradients to fit the configuration of s-LM lattices. Segmentation performance for both samples is quite high, with a final score of 0.85 for the seadragon and 0.94 for the cell image, indicating that the method can potentially be used for accurate foreground-background segmentation. The loss function can also further be expanded to incorporate more domain knowledge, possibly increasing the performance further.

3.4 Statistics fitting of a 1-Pixel Random Walk CPM

Statistics fitting is the challenge of fitting the parameters of s-LMs, such that the *summary statistics* of the model match a set of target statistics. Being able to perform statistics fitting is crucial for studying real-world phenomena of non-equilibrium systems, where both trajectory fitting and stability fitting do not adequately capture the desired behavior. Consider, for example, a cell that migrates via simple Brownian motion over time. The cell is equally likely to move in any direction. Trajectory fitting is not a good choice as the trajectories are fully random and on average the cell remains at its initial position. Stability fitting does not make sense either, as we care about the out-of-equilibrium behavior of the cell (as opposed to a stable state). For such tasks, being able to efficiently perform stability fitting is required. In this study, stability fitting via backpropagation of approximate gradients is applied to a 1-pixel random walk CPM. Successful parameter estimation for this task would provide strong evidence for hypothesis ($H4$).

The 1-pixel random walk CPM is laid out in detail in Sec. 7.5 and a visualization of the update rules is provided in Movie S4. To model the Brownian motion of a cell over time, a CPM with hard volume (i.e. number of lattice sites belonging to the cell) constraints $1 \leq v_{\text{cell}} \leq 2$ and target volume of 1 is defined. The CPM models the migratory behavior of biological cells over time as a process of lattice sites trying to copy their identity into local neighbors. At every *sampling step*, one ‘source’ site is picked at random. Then, a ‘target’ site is picked from the Moore neighborhood of the source. Finally, the source site tries to copy itself into the target site. If the copy attempt violates the volume constraints, it is automatically rejected. Otherwise, the probability of accepting the copy attempt depends on the change in Hamiltonian energy ΔH that it would cause.

For simplicity, here the Hamiltonian simply is the square distance between cell volume and target

cell volume. In more complex CPMs, the Hamiltonian gets expanded with additional components like the cell perimeter. Copy attempts with negative ΔH (shrinking the cell from two lattice sites back to one) are always accepted. On the other hand, copy attempts with positive ΔH (growing the cell from one lattice site to two) are accepted with an *acceptance rate* of $p = e^{-1/T}$. T is the model *temperature* and controls the speed of the brownian motion. That is, it controls the MSD within a given timeframe of t sampling steps. A visualization of the Brownian motion exhibited by the cells is provided in Movie S5. The goal of this experiment is to fit T to four target Mean-Square Displacements over t sampling steps.

To this end, backpropagation of approximate gradients is applied in combination with SGD. For each iteration, a batch of $N = 4000$ regular lattices of size 128×128 with periodic torus is initialized with one cell in the center. Then, the model is applied to each sample for $t = 800$ sampling steps. Afterward, the average MSD of the batch is calculated, and the square difference to the target MSD is backpropagated to update T . The learning rate used for SGD is $1e^{-5}$. In a series of three experiments, T is fit to a target MSD of 2,3 and 4 spatial units respectively.

The results of the experiment are illustrated in Fig. 6. We report on the acceptance rate, average MSD of the simulated cells, the loss between target MSD and average MSD, and the gradient of the loss with respect to T .

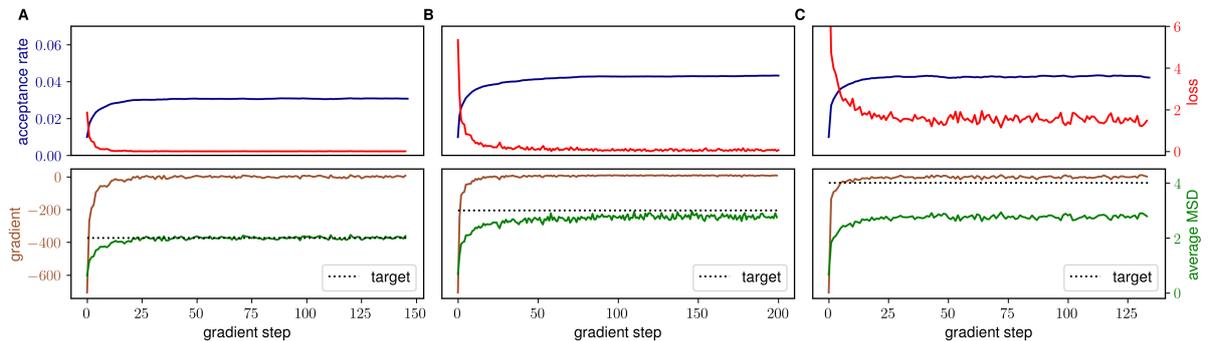


Figure 6: Fitting a 1-pixel random walk CPM to a target MSD works for low acceptance rates but exhibits vanishing gradient issues.

A Statistics fitting is applied to estimate T for a target MSD of 2 spatial units over 800 sampling steps. **B** The target MSD is increased to 3 and the experiment is repeated. **C** The experiment is repeated once more, this time with an MSD of 4. Top row: Acceptance rate (blue) $e^{-1/T}$ and loss (red) throughout the fitting process for each experiment. Bottom row: Gradient of the loss with respect to T (brown) and Mean-square displacement (green) throughout the fitting process of each experiment.

In the first experiment, T is fit to a target MSD of 2 spatial units over 800 sampling steps. Fig. 6A illustrates the results of the experiment. The top graph shows the acceptance rate and the loss over the performed gradient steps. The bottom graph additionally shows the average MSD and the gradient throughout the optimization process. The acceptance rate starts close to zero and converges to 0.03 after approximately 75 gradient steps, successfully producing average MSDs of ~ 2 spatial units. At the same time, the loss function converges to approximately zero, as does the gradient of T . The result looks promising, indicating that statistics fitting via gradient-based optimization can be a viable option.

The result of the second experiment, however, showcases that there are some issues with the current approach. For experiment 2, the target MSD is 3 spatial units over 800 sampling steps. Gradient optimization did not successfully reach that target. Instead, T converges to an acceptance rate of ~ 0.04 with an MSD of approximately 2.8 spatial units after 100 gradient steps. That is, the model estimate undershoots the target MSD. We see that despite the converged loss here being higher than in the first experiment, the gradient has converged to 0, thus indicating that there is an issue of vanishing gradients.

The observations of the second experiment are confirmed in the third experiment. For experiment 3, the target MSD was further increased to 4 spatial units over 800 sampling steps. As before, gradient optimization does not successfully find an acceptance rate to match this target. Instead, the target MSD is undershot again. Standing out is that despite the target MSD being higher than in the previous experiment, the acceptance rate still converges to the same value of ~ 0.04 . This indicates that there is an issue with the gradient vanishing as this threshold is approached. Most likely, the vanishing gradient is caused by a combination of higher acceptance rates making model behavior more noisy, and an accumulation of small approximation biases during the simulation process.

Considering that the model experiences vanishing gradients when undershooting the MSD, another experiment was performed to test what happens when initializing T to a value that is too high. Fig. S2 shows the results of this experiment. We observe that the opposite issue of exploding gradients emerges. The gradient exponentially increases in magnitude and T diverges exponentially from the goal.

Overall, the series of experiments provides mixed results for hypothesis H_4 . On one hand, the method was successful in fitting the model temperature for small MSDs, confirming at least that the approach can be successful for statistics fitting. On the other hand, experiments with increased target MSDs showed a significant issue with vanishing gradients. While this does not outright reject the hypothesis due to at least one of the experiments being successful, it is a significant issue that needs to be resolved, for the method to be generally viable.

4 Relevant Works

To the best knowledge of the author, no previous works are published that apply gradient-based optimization to s-LMs via direct backpropagation of approximate gradients through the model, without relaxation of the discreteness constraint. For example, Sumata et al. (2013) fit the parameters of a sea ice s-LM via approximate gradient descent. However, to find the gradients of the parameters, a derivative-free finite-differencing algorithm is used. While this works for the model they investigate, they acknowledge that it is computationally costly and only tractable for models with a moderate number of parameters.

In recent years, the connection between Deep Learning architectures and s-LMs has been investigated. In particular, theoretical efforts have focused on *Cellular Automata* (CA), a locally bound subclass of lattice models. While classically CAs have deterministic transition rules (Wolfram et al., 2002), stochastic “Probabilistic Cellular Automata” (PCA) have been formulated and applied to several natural phenomena (Louis and Nardi, 2018). Recently, Gilpin (2019) have shown that any cellular automaton can equivalently be represented as a Convolutional Neural Network (CNN). Neural networks are parameterized universal function approximators (Csáji et al., 2001) that can be fit using gradient descent algorithms to maximize a given objective function (Du et al., 2019). CNNs are a sub-class of neural networks particularly well suited for spatial data (O’Shea and Nash, 2015). In their work, Gilpin (2019) establish that CNNs can be trained via gradient descent algorithms on a dataset of CA state transitions to learn the update rules of any CA.

Inspired by the findings of Gilpin (2019), Mordvintsev et al. (2020) introduce the “Neural Cellular Automata” (NCA) framework to learn the update rules of s-LMs towards some objective function. To achieve this, Mordvintsev et al. (2020) relax the discreteness constraint of s-LMs, and instead use continuous-valued lattice sites. The update rule of the s-LM is then modelled as a locally-constrained CNN model. As these systems are continuous and differentiable, they can be efficiently fit via gradient descent algorithms. Mordvintsev et al. (2020) define and fit the parameters of an NCA for morphogenetic pattern growth. Then, Mordvintsev et al. (2021) use the NCA framework to model and optimize reaction-diffusion systems for generating Turing-like patterns. Finally, Sudhakaran et al. (2021) apply NCAs to grow functional 3D machines in Minecraft.

The NCA framework has achieved promising results within the relatively short time that it has been published. However, as it requires relaxing the discreteness constraint, applying it to existing s-LMs would require defining smooth approximations for each model to optimize instead. Abandoning established s-LMs in favor of creating continuous-valued approximations that are NCA-compatible from scratch would mean losing out on a large body of existing research. Further, high-dimensional CNN models are hard to interpret and obfuscate what factors are driving its outputs, which is an issue with current Neural Network models at large (Angelov and Soares, 2020; Xu et al., 2019). Hence, they are impractical for phenomenological studies.

5 Summary & Discussion

s-LMs are attractive computational tools for the simulation of non-equilibrium and morphogenetic systems. However to date no efficient method for fitting the parameters of s-LMs to a given objective function has been devised. The main reason for this is the non-differentiability of the model, which prevents backpropagating the gradients of the model parameters. Derivative-free methods for gradient optimization circumvent this issue, but are very costly and only tractable for moderate parameter spaces. The current state of the art ABC algorithm is gradient-free and instead uses undirected sampling to estimate the posterior distribution of the parameters. This has worked for a number of applications, but

is inefficient and breaks down for high-dimensional parameter spaces due to the curse of dimensionality. In this study, we investigated an alternative approach for efficient gradient-based optimization, using backpropagation of approximate gradients.

The basis for the project is the assumption that while s-LMs are not fully differentiable, the gradient can be approximated at the non-differentiable nodes of the computational graph to obtain useful approximate gradient information. To this end, four hypotheses are put forward and tested in a series of experiments.

Hypothesis *H1* states that backpropagation of approximate gradients can be used for trajectory fitting of s-LM parameters. To test this, the method was successfully applied to the spread coefficient of a Susceptible-infected s-LM. The second hypothesis *H2* states that backpropagation of approximate gradients can be used for stability fitting of morphogenetic s-LMs. In the corresponding experiments, the method was applied to the reaction rates and diffusion coefficients of the Malevanets-Kapral reaction-diffusion s-LM, successfully identifying parameters from which stable patterns arise. The result strongly supports *H2*, though optimization of diffusion coefficients appeared to be more challenging than the optimization of the reaction rates. Hypothesis *H3* states that backpropagation of approximate gradients can be used for state fitting of s-LM lattices to a given objective function. To test this, foreground-background segmentation via backpropagation of approximate gradients was performed for sample images from two separate image domains (natural images and cell imaging) where a high segmentation performance was achieved. Finally, hypothesis *H4* states that backpropagation of approximate gradients can be used for statistics fitting of s-LM parameters. To test the final hypothesis, we fit the temperature parameter of a 1-pixel Random Walk CPM such that the average Mean-square displacement (MSD) of the simulated cells matches a given target MSD. The results of this experiment were mixed and indicate that further investigation is necessary.

On a higher level, it stands out from the results that fitting diffusion coefficients appear to be generally more challenging than fitting other types of parameters. The most likely reason for this is that each simulation only provides an empirical sample trajectory of the current generating noise process. As the rate of diffusion increases, the variance of these samples gets higher and more samples are necessary to get an accurate estimate. For backpropagation, however, we need to calculate the backward pass through the model for each step taken per sample trajectory that was generated. Because we do not have the exact gradient, every backward pass through the model introduces some slight approximation bias. The accumulation of small approximation biases coupled with the high variance of the trajectories makes diffusion coefficients more challenging targets for gradient-based optimization.

To combat the bias induced through the approximation of the gradient, some methods can be considered. Foremost, the usage of more sophisticated choices for straight-through estimators can help reduce approximation biases. For example, Shekhovtsov and Yanush (2021) recently explored the impact of different standard STE choices on optimization performance for Stochastic Binary network models. Further, Fan et al. (2022) have proposed a new variant of STEs dubbed “Gapped Straight-through estimators”, to combat high variance issues without increasing computational cost in the context of stochastic Deep Generative neural network models. Early results of this method seem promising and may improve approximation quality for backpropagation of s-LMs.

Another problem arising from backpropagation through a large number of sampling steps is vanishing gradients. A known issue in the field of Recurrent Neural Network models Pascanu et al. (2013), vanishing gradients arise when the partial derivative of a function is very flat for some regions of the input space (Hochreiter, 1998). During backpropagation, we need to apply the chain rule of derivatives iteratively to each transformation in the computational graph. The chain rule means that the gradient coming in from the next higher node in the computational graph gets multiplied by the partial derivative of the node itself. That is, each node applies some scaling to the gradient before it gets further backpropagated. If the partial derivative of a node is very flat, it scales down the gradient significantly. As we need to pass the gradient back through the model for each forward pass that was applied, nodes with flat derivatives keep scaling down the gradient. If this happens enough times, the gradient becomes vanishingly small, approaching zero.

The opposite problem of exploding gradients can also occur. In Fig. S2, we see that when starting from a high initial temperature for the 1-pixel random walk CPM, the gradients do not vanish. On the contrary, they increase exponentially over time. The intuition for this is the same. Nodes with very steep partial derivatives keep scaling up the gradient for each pass through the model, causing them to explode in magnitude. Various methods have been proposed to mitigate the effects of vanishing/exploding gradients (Pascanu et al., 2013). More recently, (Hu et al., 2021) suggest that vanishing gradient issues can also be mitigated via a careful choice of Straight-through Estimators (there referred to as “Artificial

Gradients”).

Taking a step back, more parallels between the approach shown here and some classes of Neural Networks can be found. Specifically, the optimization process of the Malevanets-Kapral model is similar to that of generative Deep Diffusion models. These models are trained by iteratively injecting noise (i.e. diffusing) an input pattern (for example an image) until the original pattern degenerates to white noise. Then, a parameterized network tries to revert the noise process and recover the original input pattern (Cao et al., 2022). The challenge here is that the noise-generating process is unknown and needs to be estimated by the network. To achieve that, gradients need to be retained and backpropagated from the white noise patterns. For the Malevanets-Kapral model, we have a known reaction-diffusion process that destroys stable patterns over time. Instead of learning how to revert the noise process, our goal is to adjust the parameters of the process itself, such that the target image becomes a stable attractor of the system.

When applying gradient-based optimization methods, it is important to be aware of the limitations and challenges that they entail. While it is an efficient method for fitting model parameters, it does introduce new hyperparameters which need adjusting themselves, such as the learning rate. The choice of learning rate, for example, can have a significant impact on the convergence rate of gradient descent algorithms (Jacobs, 1988). More sophisticated gradient descent algorithms introduce several hyperparameters to the optimization process (Kingma and Ba, 2014). For large neural networks with millions of parameters, introducing several hyperparameters that need adjusting is a comparatively small price to pay for efficient optimization. S-LMs, however, usually operate in a small parameter space. If not careful in choosing suitable gradient optimization methods according to the task, the number of hyperparameters could feasibly be larger than the number of original parameters that need to be estimated. In this, albeit slightly exaggerated scenario, we have simply replaced the original parameter estimation problem with another one of possibly the same complexity. Further challenges of gradient-based optimization methods are convergence to local optima and the problem of defining the right cost function.

Even with the limitations and challenges of gradient-based optimization in mind, the results of the method introduced here seem promising. For each of the models and objectives considered in this study, backpropagation of approximate gradients was successful in fitting the model parameters to a set of targets, with the caveat of vanishing/exploding gradient issues for the 1-pixel random walk CPM which can be mitigated in various ways. However, the models used here were simple toy examples. Following up on the research, it would for one be interesting to see the method applied to more complex models with real-world applications. Additionally, for these more complex models, the method should be compared to parameter estimation via ABC. While these methods are not directly comparable, as they do not perform the same task, we could use for example the wall clock time it takes to establish a good estimate of the model parameters as a point of comparison.

6 Conclusion

In this study, the viability of fitting s-LM parameters to objective functions via backpropagation of approximate gradients was investigated by applying the method to four different models, each being fit to a different type of objective. Overall, the experimental results show strong support for the general viability of s-LM fitting via backpropagation of approximate gradients. However, some issues of the current approach, specifically regarding vanishing/exploding gradients were identified. Despite that, we think that the results are promising enough to warrant further investigation of the method. In a follow-up project, the vanishing gradient issues can be addressed by applying more sophisticated gradient estimators. Additionally, inspiration can be taken from RNN literature where vanishing gradients have been a common issue for which several solutions have been proposed.

7 Methods

7.1 Backpropagating approximate gradients through s-LMs

Gradient descent optimization methods cannot directly be applied to s-LMs. On a high level, this is due to s-LMs being non-differentiable as they are stochastic and discrete. However, this perspective is challenged when looking at s-LMs as *computational graphs*. Computational graphs illustrate models as directed graphs of mathematical expressions. Each edge denotes the inflow/outflow of information. Nodes, on the other hand, express how the incoming information is transformed. Differentiability of a

model, from this perspective, refers to information being able to flow backward from the end of the chain (i.e. the loss calculation) to the parameters of the model in the form of partial derivatives (gradients). Non-differentiability arises from certain types of nodes blocking the backward flow of information. s-LMs generally consist of four types of nodes:

- Continuous Nodes: Continuous transformation of the inputs.
- Discrete Nodes: Discrete transformation of the inputs.
- Stochastic Nodes: Random variable sampled from a probability distribution.
- Stochastic Categorical Nodes: Random one-hot vector sampled from a categorical probability distribution.

Continuous nodes are generally differentiable, while the other three nodes block the gradient flow and cause non-differentiability (Fig. 7A1, B1). Most s-LMs, however, consist overwhelmingly of continuous nodes. That is, we assume s-LMs to define a probability distribution over lattice states that is *almost differentiable* everywhere with respect to Θ . The challenge becomes to find a way to propagate an estimate of the gradient through the blocking nodes.

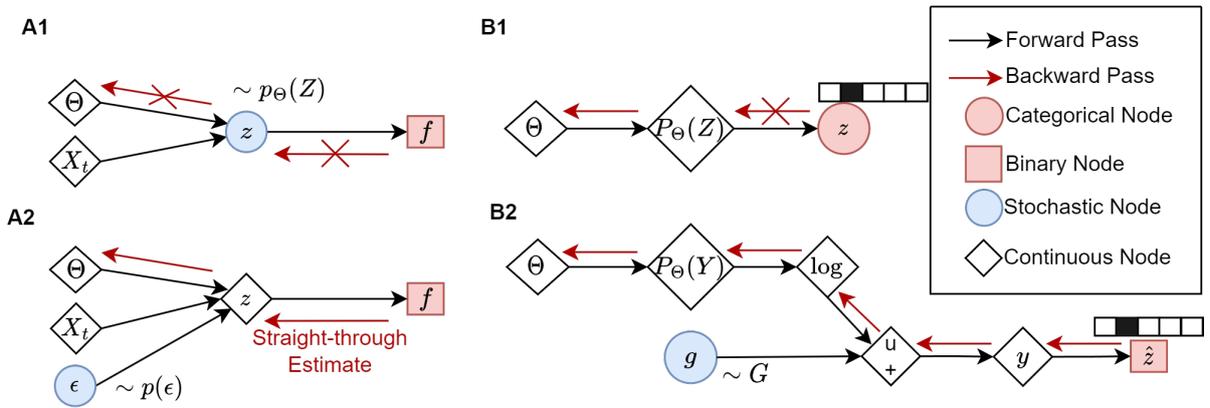


Figure 7: Gradient flow is enabled through a combination of reparameterization and straight-through estimation.

A1 A simple computational graph where gradient flow is blocked through a stochastic node and a discrete node. **A2** A straight-through estimate is propagated from the discrete node backward. Reparameterization removes the dependency on Θ from the sampling step, allowing the gradient to flow further. **B1** In this example, gradient flow is blocked by a categorical stochastic node. **B2** Adding Gumbel noise to the log probabilities is a way to reparameterize categorical distributions. The result gets discretized in the forward pass. STE is applied during backpropagation.

7.1.1 Dealing with stochastic nodes

Stochastic nodes block gradient flow as sampling operations do not have a well-defined derivative. To solve this, Kingma and Welling (2013) introduces the *reparameterization trick*. Fig. 7A1 illustrates a simple example. In the computational graph, the parameters Θ and the current lattice state X_t of a given s-LM are used to parameterize a probability distribution $p_{\Theta}(X_t)$. Then, a random variable $z \sim p_{\Theta}(X_t)$ is sampled. The sampling step depends on Θ and is thus ‘inside’ of the model flow. To determine the gradient of Θ , information needs to flow back through the sampling step. As there is no well-defined derivative, however, this is not possible and gradient flow is blocked.

The reparameterization trick solves this issue by moving the sampling operation ‘out’ of the model. That is, we introduce an auxiliary random variable ϵ to the model, distributed by some distribution $p(\epsilon)$ that does not depend on Θ . Then, we rewrite $z = g(\Theta, X_t, \epsilon)$ into a deterministic transformation g of the parameters, the lattice state, and the auxiliary variable. The result will be equivalent to drawing a sample from the original distribution. The random variable, however, now is simply a constant in a deterministic transformation, the sampling happens ‘outside’ of the model. Rewritten in this way, gradient information can flow independent of the random sampling (Fig. 7A2).

7.1.2 Dealing with discrete nodes

One of the most common applications of reparameterization in s-LMs is sampling from binomial distributions $z \sim p_{\Theta}(Z)$. Even when the sampling operation can be rewritten to a deterministic function, in this case that is not enough to re-establish gradient flow. As the distribution was binomial, the rewritten transformation will be discrete. Specifically, it involves the Heaviside step function h . The derivative of h is called the Dirac delta and is 0 everywhere except for the step threshold, where it is infinite. As a result, gradient flow remains blocked (Fig. 7A1). More generally, discrete transformations do not provide useful gradient information and generally block gradient flow.

Bengio et al. (2013) describe a strategy of approximating the gradient for discrete transformations. Namely, gradients are approximated through “Straight-through Estimation” (STE). The general strategy is to compute the forward pass without any changes but replace the derivative of the discrete node during the backward pass with an approximate derivative function (the straight-through estimator).

The simplest approximate derivative function is the identity function. That is, when the gradient flows to a discrete node, a simple identity mapping is applied instead of multiplying with the derivative. For binary nodes as in the binomial example described above, the viability of identity STEs has been shown in the context of spiking neuron models (Tang et al., 2022; Neftci et al., 2019). Thus, we employ identity STEs to approximate the gradient for discrete nodes (Fig. 7A2).

7.1.3 Dealing with stochastic categorical nodes

A final common cause for gradient blockage in s-LMs are stochastic categorical nodes. Stochastic categorical nodes represent the sampling of one-hot vectors from a categorical distribution parameterized by the model parameters Θ . As this operation is both discrete and involves stochastic sampling with a dependency on Θ , stochastic categorical nodes can be considered a special combination of the two problems addressed prior. Subsequently, the most common strategy to approximate gradients is a combination of reparameterization and STE.

Jang et al. (2016) first introduce a solution to backpropagation of gradients through stochastic categorical nodes. Consider a computational graph with a stochastic categorical node $z \sim P_{\Theta}(Z)$ (Fig. 7B1). The goal is to draw samples z from the categorical distribution $P_{\Theta}(Z)$ without blocking the flow of the gradients to Θ .

In the first step, a special form of the reparameterization trick is applied to make the sampling operation independent of the model parameters Θ . The special form being used is commonly called the *Gumbel-softmax trick*. Instead of sampling from the categorical probability distribution $P(Z)$, we calculate the log-odds (also called the *logits*) of each category. Then, an auxiliary random variable g_i drawn from a standard Gumbel distribution is added to the logits of each category i . The result of this operation is stored in a vector u . Finally, to receive an approximate sample from $P(Z)$, softmax with a temperature τ is applied to each element in u .

The resulting vector y is equivalent to sampling from a continuous probability distribution $P_{\Theta}(Y)$ that approximates the original categorical distribution $P_{\Theta}(Z)$. As τ approaches 0, the softmax function approaches a discrete argmax function. As a result, y becomes categorical and $P_{\Theta}(Y)$ becomes equivalent to $P_{\Theta}(Z)$ (Jang et al., 2016; Maddison et al., 2014). Fig. 7B2 visualizes the approach for an example computational graph.

The Gumbel-softmax trick provides a differentiable way to draw samples y from a continuous distribution $P_{\Theta}(Y)$ that approximates the original categorical distribution $P_{\Theta}(Z)$. As we want to uphold the discreteness of the model, we further calculate $\hat{z} = \arg \max_i(y_i)$. However, the arg max function is not differentiable. To enable gradient flow during the backward pass, a straight-through estimate is used to replace the derivative of the arg max function. Here, we simply choose the identity function. As τ of the softmax approaches 0, the straight-through estimate becomes an unbiased estimator of the gradient.

7.1.4 Implementing the approach in software

For this project, each model was implemented using the PyTorch library (Stevens et al., 2020) version 2.0.1. PyTorch is a modular Python framework, that provides tools for constructing and optimizing Deep Neural Networks. More generally, it can be applied to optimize algebraic models by providing automatic differentiation tools (Paszke et al., 2017). Automatic differentiation allows us to automatically find and accumulate the partial derivatives of each transformation in the computational graph of a model. This way, gradient descent algorithms can be efficiently applied. PyTorch also provides easy access to the

partial derivatives of each function included in the backward pass and allows us to override them with a custom function. This way, we can easily replace the derivatives of discrete nodes with identity STEs.

The code for all experiments together with the data is available at <https://shorturl.at/fl026>.

7.2 1-parameter SI s-LM

We consider a 2D square lattice $L \in \{0, 1\}^{n \times n}$ with a periodic torus, with n denoting the number of cells per row and column. Each lattice site $L^{(i,j)}$ is binary, indicating awareness (state 1) or unawareness (state 0) of the news item. The state L_t of the system at a given timepoint t is fully described through the configuration of each lattice site $L_t^{(i,j)}$.

For a given lattice state L_t , the probability of transitioning to a new state L_{t+1} is defined as:

$$P(L_{t+1}|L_t, \beta) = \prod_{L^{(i,j)} \in L} p(L_{t+1}^{(i,j)}|L_t^{(i,j)}, N_t^{(i,j)}, \beta)$$

$N_t^{(i,j)}$ denotes the number of informed neighbors within the *Moore neighborhood* of $L^{(i,j)}$ at time t . The parameter β is the diffusion coefficient of the model. It represents the conditional likelihood of an informed neighbor sharing the news item. The per-cell transition probability $p(L_{t+1}^{(i,j)}|L_t^{(i,j)}, N_t^{(i,j)}, \beta)$ is described by a probability matrix:

$$p(L_{t+1}^{(i,j)}|L_t^{(i,j)}, N_t^{(i,j)}, \beta) = \begin{array}{cc|c} & \begin{array}{c} 1 \\ 0 \end{array} & \begin{array}{c} 0 \\ (1-\beta)^{N_t^{(i,j)}} \end{array} \\ \hline \begin{array}{c} 1 \\ 0 \end{array} & \begin{array}{c} 1 - (1-\beta)^{N_t^{(i,j)}} \\ (1-\beta)^{N_t^{(i,j)}} \end{array} & \begin{array}{c} L_t^{(i,j)}/L_{t+1}^{(i,j)} \\ 1 \\ 0 \end{array} \end{array}$$

The first column of p encodes that news items cannot be forgotten once learned. The second column models the likelihood of getting informed or staying uninformed. To this end, each informed neighbor is modelled as an independent trial. For each trial, the probability of staying uninformed is $(1-\beta)$. Making use of the product rule, the total likelihood of staying unaware is thus $p_{2,2} = (1-\beta)^{N_t^{(i,j)}}$. Conversely, the likelihood of being informed is then $p_{1,2} = 1 - (1-\beta)^{N_t^{(i,j)}}$.

7.3 Malevanets-Kapral s-LM

Malevanets and Kapral (1997) provides a more in-depth description of the model. We consider a 2-layer square 2D lattice $L \in \mathbb{M}^{2 \times n \times n}$, with $\mathbb{M} = \{0, 1, 2, \dots, N\}$. That is, each lattice cell can hold up to N molecules, making N a hyperparameter of the model. By n , we denote the number of lattice sites per row and column. The first sub-lattice $\mathcal{A} \in \mathbb{M}^{n \times n}$ models the concentration of chemical species A in space, the second sub-lattice $\mathcal{B} \in \mathbb{M}^{n \times n}$ models the concentration of species B (Fig. 9A).

Transitions of the lattice from state $L_t = (A_t, B_t)$ at time t to the next timepoint $L_{t+1} = (A_{t+1}, B_{t+1})$ are defined as a two stage process. First, each chemical species performs one step of diffusion independent of each other. Then, the two species react with each other according to the following reaction scheme:



7.3.1 Diffusion

Malevanets and Kapral (1997) state, that exact solutions to the underlying Markov process of the reaction scheme allow us to assume independent diffusion processes per chemical species. The diffusion process of a species is modelled in three stages: *Excitation*, *translation*, and *settling* of the particles. An auxiliary lattice $\mathcal{E} \in \mathbb{Z}^{n \times n}$ is defined to keep track of the excited particles in the process. As an example, we consider the diffusion of the sub-lattice for chemical species A.

During excitation, at most one particle of each lattice cell becomes "excited" and is thus transferred to the auxiliary grid. For each pair of cell coordinates (i, j) , this process is defined by:

$$(A^{(i,j)}, E^{(i,j)}) = (A^{(i,j)} - h(A^{(i,j)} - N\xi), h(A^{(i,j)} - N\xi))$$

Here, h refers to the Heaviside function while $\xi \sim U(0,1)$ is a random variable sampled from the uniform distribution U . Additionally, we define that the probability of excitation for a cell that reached maximum occupancy is 1, and for a cell with no molecules it becomes 0.

Once excitation has been applied, a random direction out of 9 options (including staying in place) is chosen, and the whole lattice E is translated in that direction. The probability of not staying in place for species A is defined by the diffusion coefficient D_A , governing the rate of diffusion. Similarly, the rate of diffusion for species B is governed by D_B . To ensure uniform Brownian motion, translation in the diagonal directions has a lower probability proportional to the larger distance covered.

Finally, after E has been translated, the excited particles settle into their new positions. Thus, the particles are removed from E and merged back into A at their new positions:

$$(A, E) = (A + E, E - E)$$

An example of the diffusion process is illustrated in Fig. 9B.

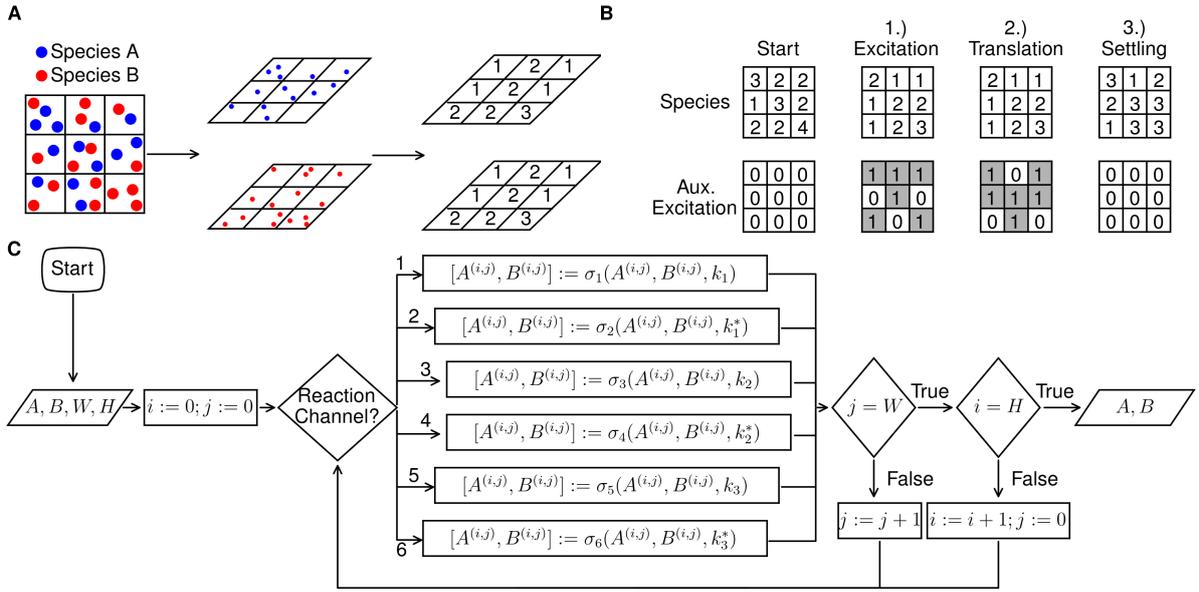


Figure 9: The Malevanets-Kapral model uses a two-stage update scheme to model the reaction-diffusion process between two chemical species.

A Chemical concentration in space is modelled with one lattice per chemical species $L = (A, B)$. Cells $A^{(i,j)}$ and $B^{(i,j)}$ correspond to the same lattice site, describing the respective current concentration. **B** Each chemical undergoes an independent diffusion process. For every lattice cell, at most one molecule becomes ‘excited’ and is moved to another lattice. The excited particles get translated in space and settle into their new positions. **C** Chemical interaction is simulated by randomly choosing and applying a reaction channel σ to each lattice site.

7.3.2 Reaction

Malevanets and Kapral (1997) translate the reaction scheme into lattice-gas equations for microscopic particle collision dynamics. Each reaction channel $r = 1, \dots, 6$ in the scheme is turned into a cell-wise operator σ_r , the combination of which simulates the reactive dynamics of the Fitzhugh-Nagumo equation. At each timestep, the reactive dynamics are simulated by iterating over each cell (i, j) on the lattice and:

1. Choosing a reaction channel r
2. Applying the corresponding cell-wise operator $[A^{(i,j)}, B^{(i,j)}] = \sigma_r(A(i, j), B^{(i,j)})$

The lattice operators σ are given by:

$$\begin{aligned}
\sigma_1(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)} + h(p_1 - \xi), B^{(i,j)}], & \sigma_2(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)} - h(p_2 - \xi), B^{(i,j)}], \\
\sigma_3(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)} + h(p_3 - \xi), B^{(i,j)}], & \sigma_4(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)} - h(p_4 - \xi), B^{(i,j)}], \\
\sigma_5(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)}, B^{(i,j)} + h(p_5 - \xi)], & \sigma_6(A^{(i,j)}, B^{(i,j)}) &= [A^{(i,j)}, B^{(i,j)} - h(p_6 - \xi)],
\end{aligned}$$

Here, $\xi \sim U(0, 1)$ is again a random variable sampled from a uniform distribution U while h refers to the Heaviside step function. Further, p_i with $i = 1, \dots, 6$ represents the reaction probability of each channel and is defined by:

$$\begin{aligned}
p_1(A^{(i,j)}, B^{(i,j)}) &= \gamma k_1 (N - A^{(i,j)}) \frac{A^{(i,j)} A^{(i,j)}}{(N-1)(N-2)}, \\
p_2(A^{(i,j)}, B^{(i,j)}) &= \gamma k_1^* A^{(i,j)} \frac{(N - A^{(i,j)})(N - A^{(i,j)})}{(N-1)(N-2)}, \\
p_3(A^{(i,j)}, B^{(i,j)}) &= \gamma k_2 \frac{(N - A^{(i,j)}) B^{(i,j)}}{N}, \\
p_4(A^{(i,j)}, B^{(i,j)}) &= \gamma k_2^* \frac{A^{(i,j)} (N - B^{(i,j)})}{N}, \\
p_5(A^{(i,j)}, B^{(i,j)}) &= \gamma k_3 \frac{(N - A^{(i,j)})(N - B^{(i,j)})}{N}, \\
p_6(A^{(i,j)}, B^{(i,j)}) &= \gamma k_3^* \frac{A^{(i,j)} B^{(i,j)}}{N},
\end{aligned}$$

The hyperparameter γ acts as the timescale of the reaction process by scaling the reaction probabilities. The parameters k_i, k_i^* for $i = 1, 2, 3$ are the reaction rates of the respective channels. Based on the results of Malevanets and Kapral (1997), we make the simplifying assumption that $k_i = k_i^*$. A scheme of the reaction process is illustrated in Fig. 9C.

7.4 Gradient-based foreground-background segmentation

Consider a $n \times m$ sized RGBA image $X \in \mathbb{R}^{|0 \leq X \leq 255|^{n \times m \times 4}}$. The goal of foreground-background segmentation is to predict a corresponding *mask* $\hat{Y} \in \{0, 1\}^{n \times m}$, where each entry $\hat{Y}^{(i,j)}$ predicts if the corresponding pixel $X^{(i,j)}$ belongs to the background ($=0$) or the foreground ($=1$) of the image.

As a first step X gets pre-processed to grayscale, then the grayscale pixel intensities are normalized and zero-centered. We denote the pre-processed grayscale image as $X_g \in \mathbb{R}^{|x| \leq \mathbb{X}_8 \leq \mathbb{K}^{n \times m}}$. Additionally, we need to define a discretization function, that maps from X_g to the predicted segmentation mask \hat{Y} . For simplicity, a simple thresholding function was defined as:

$$\forall x^{(i,j)} \in X_g : y^{(i,j)} = \begin{cases} 1, & \text{if } x^{(i,j)} > 0, \\ 0, & \text{otherwise} \end{cases}$$

Optimizing the predicted segmentation mask from this perspective is a task of moving background pixels in X_g close to zero and foreground pixels above zero. Here, a copy of X_g is defined that we denote X_{pred} . A simple loss function was devised that tries to capture this objective. It consists of the following components:

- Inverse pixel-intensity penalty: $l_a = \sum_{x \in X_{pred}} 1 - x$, penalizing pixels that approach 0.
- Background contact penalty: $l_b = \sum_{x \in X_{pred}} \sum_{u \in N(x)} x(1 - u)$, penalizing high-intensity pixels surrounded by low intensity pixels. Assumes that foreground pixels usually have contact with multiple other foreground pixels, e.g. forming objects.
- Foreground contact penalty: $l_c = \sum_{x \in X_{pred}} \sum_{u \in N(x)} (1 - x)u$, penalizing low-intensity pixels surrounded by high-intensity pixels, similar to l_b
- Square reconstruction loss: $l_d = \sum_{(i,j) \in \text{Coords}(X_{pred})} (X_g - X_{pred})^2$, ensuring that the most important information in the image is retained.

In this context, $N(x)$ is the local Moore neighborhood of pixel x while $Coords(X)$ is a function that returns a list of all pixel coordinates (i, j) for the image. The total loss function can then be written as $\mathcal{L}(X_{pred}) = \lambda_a l_a + \lambda_b l_b + \lambda_c l_c + \lambda_d l_d$. Where the λ terms are scaling factors of the different components. With the loss function in place, X_{pred} is optimized by applying standard gradient descent to minimize the loss.

Optimizing X_{pred} with this procedure can be seen as a special case of straight-through estimation. Each pixel of X_{pred} can be seen as a parameter that needs to be fitted to the loss function. However, the discretization function prevents gradient flow from the discrete mask back to the continuous pixel values. To circumvent this, we differentiate directly with respect to the continuous input values, disregarding the discretization step. The resulting gradient is an approximation of the gradient with respect to the discrete output mask.

7.5 1-Pixel Random Walk Cellular Potts model

Within the context of the CPM, we consider lattices to be a spatial arrangement of *pixels*. A *cell*, then refers to an assortment of pixels with the same assigned value. For simplicity, we consider the case of a single-cell simulation. That is, we specifically consider a square lattice $L \in \{0, 1\}^{m \times m}$. We define pixel values of zero to be the *background* and the pixels that have an assigned value of one to belong to the cell.

Simulation of the model over time is done by iteratively performing *sampling steps*. Each sampling step follows the following procedure:

1. Randomly choose a *source* pixel $L^{(i,j)}$ from the lattice
2. Randomly choose a *target* pixel $L^{(k,l)}$ from the Moore neighborhood of the source
3. Calculate the probability $p(L^{(i,j)}, L^{(k,l)})$ of the source pixel copying its value to the target pixel
4. Update $L^{(k,l)} := L^{(k,l)} + h(p - \xi)\Delta v$

In the procedure, h denotes the Heaviside step function. Further, $\xi \sim U(0, 1)$ is a random, uniformly distributed variable in the range $[0, 1]$. With Δv , we denote the change in cell volume through a successful copy attempt. The probability of a successful copy attempt p is given by:

$$p(L^{(i,j)}, L^{(k,l)}) = \begin{cases} 0, & \text{if } \Delta v + \sum_{l \in L} l > 2 \\ 0, & \text{if } \Delta v + \sum_{l \in L} l < 1 \\ 1, & \text{if } \Delta v = -1 \wedge \sum_{l \in L} l = 2 \\ e^{-1/\tau}, & \text{otherwise} \end{cases}$$

References

- Alamoudi, E., Schälte, Y., Müller, R., Starruss, J., Bundgaard, N., Graw, F., Bruschi, L., and Hasenauer, J. (2023). Fitmulticell: Simulating and parameterizing computational models of multi-scale and multi-cellular processes. *bioRxiv*, pages 2023–02.
- Alamoudi, E., Starruß, J., Bundgaard, N., Muller, R., Reck, F., Graw, F., Bruschi, L., Hasenauer, J., and Schalte, Y. (2022). Massively parallel likelihood-free parameter inference for biological multi-scale systems.
- Ali, I. and Saleem, M. T. (2023). Spatiotemporal dynamics of reaction–diffusion system and its application to turing pattern formation in a gray–scott model. *Mathematics*, 11(6):1459.
- Angelov, P. and Soares, E. (2020). Towards explainable deep neural networks (xdnn). *Neural Networks*, 130:185–194.
- Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.
- Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.
- Cao, H., Tan, C., Gao, Z., Chen, G., Heng, P.-A., and Li, S. Z. (2022). A survey on generative diffusion model. *arXiv preprint arXiv:2209.02646*.
- Carr, M. J., Simpson, M. J., and Drovandi, C. (2021). Estimating parameters of a stochastic cell invasion model with fluorescent cell cycle labelling using approximate bayesian computation. *Journal of the Royal Society Interface*, 18(182):20210362.
- Chan, B. W.-C. (2018). Lenia-biology of artificial life. *arXiv preprint arXiv:1812.05433*.
- Chopard, B. and Droz, M. (2005). “cellular automata modeling of physical systems”, 2005.
- Conlisk, J. (1976). Interactive markov chains. *Journal of Mathematical Sociology*, 4(2):157–185.
- Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*, 24(48):7.
- Curtis, A. and Lomax, A. (2001). Prior information, sampling distributions, and the curse of dimensionality. *Geophysics*, 66(2):372–378.
- Dab, D., Boon, J.-P., and Li, Y.-X. (1991). Lattice-gas automata for coupled reaction-diffusion equations. *Physical review letters*, 66(19):2535.
- Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.
- Durso-Cain, K., Kumberger, P., Schälte, Y., Fink, T., Dahari, H., Hasenauer, J., Uprichard, S. L., and Graw, F. (2021). Hcv spread kinetics reveal varying contributions of transmission modes to infection dynamics. *Viruses*, 13(7):1308.
- Fan, T.-H., Chi, T.-C., Rudnicky, A. I., and Ramadge, P. J. (2022). Training discrete deep generative models via gapped straight-through estimator. In *International Conference on Machine Learning*, pages 6059–6073. PMLR.
- FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466.
- Gilpin, W. (2019). Cellular automata as convolutional neural networks. *Physical Review E*, 100(3):032402.
- Graner, F. and Glazier, J. A. (1992). Simulation of biological cell sorting using a two-dimensional extended potts model. *Physical review letters*, 69(13):2031.
- Haselwandter, C. A. and Vvedensky, D. D. (2008). Renormalization of stochastic lattice models: Epitaxial surfaces. *Physical Review E*, 77(6):061129.

- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.
- Hu, Z., Zhang, J., and Ge, Y. (2021). Handling vanishing gradient problem using artificial derivative. *IEEE Access*, 9:22371–22377.
- Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307.
- Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1).
- Lengyel, I. and Epstein, I. R. (1992). A chemical approach to designing turing patterns in reaction-diffusion systems. *Proceedings of the National Academy of Sciences*, 89(9):3977–3979.
- Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670.
- Li, S., Wu, J., and Dong, Y. (2015). Turing patterns in a reaction–diffusion model with the degn–harrison reaction scheme. *Journal of Differential Equations*, 259(5):1990–2029.
- Lim, L. A. and Keles, H. Y. (2020). Learning multi-scale features for foreground segmentation. *Pattern Analysis and Applications*, 23(3):1369–1380.
- Ljosa, V., Sokolnicki, K. L., and Carpenter, A. E. (2012). Annotated high-throughput microscopy image sets for validation. *Nature methods*, 9(7):637–637.
- Long, A. L. and Keles, H. (2018). Foreground segmentation using a triplet convolutional neural network for multiscale feature encoding.
- Louis, P.-Y. and Nardi, F. R. (2018). Probabilistic cellular automata. *Emergence, Complexity, Computation*, 27.
- Maddison, C. J., Tarlow, D., and Minka, T. (2014). A* sampling. *Advances in neural information processing systems*, 27.
- Malevanets, A. and Kapral, R. (1997). Microscopic model for fitzhugh-nagumo dynamics. *Physical review E*, 55(5):5657.
- Mengersen, K. L., Pudlo, P., and Robert, C. P. (2013). Bayesian computation via empirical likelihood. *Proceedings of the National Academy of Sciences*, 110(4):1321–1326.
- Mordvintsev, A., Niklasson, E., and Randazzo, E. (2021). Texture generation with neural cellular automata. *arXiv preprint arXiv:2105.07299*.
- Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*, 5(2):e23.
- Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology*, 47(1):90–100.
- Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63.

- O’Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.
- Qin, X., Dai, H., Hu, X., Fan, D.-P., Shao, L., and Van Gool, L. (2022). Highly accurate dichotomous image segmentation. In *European Conference on Computer Vision*, pages 38–56. Springer.
- Rothman, D. H. and Zaleski, S. (2004). *Lattice-gas cellular automata*.
- Rubenstein, B. M. and Kaufman, L. J. (2008). The role of extracellular matrix in glioma invasion: a cellular potts model approach. *Biophysical journal*, 95(12):5661–5680.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Scholes, N. S., Schnoerr, D., Isalan, M., and Stumpf, M. P. (2019). A comprehensive network atlas reveals that turing patterns are common but not robust. *Cell systems*, 9(3):243–257.
- Scianna, M. and Preziosi, L. (2021). A cellular potts model for analyzing cell migration across constraining pillar arrays. *Axioms*, 10(1):32.
- Shekhovtsov, A. and Yanush, V. (2021). Reintroducing straight-through estimators as principled methods for stochastic binary networks. In *DAGM German Conference on Pattern Recognition*, pages 111–126. Springer.
- Sisson, S. A., Fan, Y., and Beaumont, M. (2018). *Handbook of approximate Bayesian computation*. CRC Press.
- Sisson, S. A., Fan, Y., and Tanaka, M. M. (2007). Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765.
- Stevens, E., Antiga, L., and Viehmann, T. (2020). *Deep learning with PyTorch*. Manning Publications.
- Sudhakaran, S., Grbic, D., Li, S., Katona, A., Najarro, E., Glanois, C., and Risi, S. (2021). Growing 3d artefacts and functional machines with neural cellular automata. In *Artificial Life Conference Proceedings 33*, volume 2021, page 108. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info
- Sumata, H., Kauker, F., Gerdes, R., Koeberle, C., and Karcher, M. (2013). A comparison between gradient descent and stochastic approaches for parameter optimization of a sea ice model. *Ocean Science*, 9(4):609–630.
- Szabó, A. and Merks, R. M. (2013). Cellular potts modeling of tumor growth, tumor invasion, and tumor evolution. *Frontiers in oncology*, 3:87.
- Tang, J., Lai, J.-H., Zheng, W.-S., Yang, L., and Xie, X. (2022). Relaxation lif: A gradient-based spiking neuron for direct training deep spiking neural networks. *Neurocomputing*, 501:499–513.
- Turing, A. M. (1990). The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52:153–197.
- Vittadello, S. T., Leyshon, T., Schnoerr, D., and Stumpf, M. P. (2021). Turing pattern design principles and their robustness. *Philosophical Transactions of the Royal Society A*, 379(2213):20200272.
- Voss-Böhme, A. (2012). Multi-scale modeling in morphogenesis: a critical analysis of the cellular potts model.
- Wang, W., Lin, Y., Yang, F., Zhang, L., and Tan, Y. (2011). Numerical study of pattern formation in an extended gray–scott model. *Communications in Nonlinear Science and Numerical Simulation*, 16(4):2016–2026.

- Wang, X., Jenner, A. L., Salomone, R., Warne, D. J., and Drovandi, C. (2022). Calibration of agent based models for monophasic and biphasic tumour growth using approximate bayesian computation. *bioRxiv*, pages 2022–09.
- Wang, Y., Luo, Z., and Jodoin, P.-M. (2017). Interactive deep learning method for segmenting moving objects. *Pattern Recognition Letters*, 96:66–75.
- Wolfram, S. et al. (2002). *A new kind of science*, volume 5. Wolfram media Champaign, IL.
- Wortel, I. M., Niculescu, I., Koliijn, P. M., Gov, N. S., de Boer, R. J., and Textor, J. (2021). Local actin dynamics couple speed and persistence in a cellular potts model of cell migration. *Biophysical journal*, 120(13):2609–2622.
- Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., and Zhu, J. (2019). Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8*, pages 563–574. Springer.

Supplemental Material

Supplemental Videos

The supplemental videos can be accessed at <https://shorturl.at/rIN37>

Movie S1: Simulated pattern formation tests during stability fitting show that stable patterns arise as the parameters get adjusted. We apply stability fitting to the reaction rates of the Malevanets-Kapral s-LM and show pattern formation tests per 50 steps of gradient descent. The visualization shows that patterns start forming after about 400 steps of gradient descent and become stable after 700 steps. Initial reaction rates are drawn from a uniform distribution in the range $[0, 1]$. The diffusion coefficients were fixed to $D_A = 0.1$, D_B . Further, we set $\gamma = 0.005$, $N = 50$. In this experiment, custom learning rates per reaction rates were empirically chosen as $lr_{k_1} = 0.1$, $lr_{k_2} = 0.001$, $lr_{K_3} = 0.01$.

Movie S2: State fitting applied to a natural image of a seadragon successfully extracts the shape of the seadragon. In this movie, state fitting is visualized for a natural image of a seadragon from the DIS5K dataset. By using backpropagation of approximate gradients, foreground-background segmentation is performed and the shape of the seadragon is successfully extracted.

Movie S3: State fitting applied to cell imaging successfully extracts the cell shapes from the image. In this movie, state fitting is visualized for a cell image from the BBC005v1 dataset. By using backpropagation of approximate gradients, foreground-background segmentation is performed and the cell shapes are successfully extracted.

Movie S4: Visualization of the update rules for the 1-pixel random walk CPM.

Movie S5: Tracking the cell centroids during simulation of the 1-pixel random walk CPM over time shows that the system exhibits Brownian motion. The trajectory of the centroids was tracked while simulating the movement of five cells with the 1-pixel random walk CPM. All cells display Brownian motion on the lattice over time.

Supplemental Figures



Figure S1: For a subset of parameters, the spatial distribution of the chemicals in the Malevanets-Kapral model forms stable labyrinthine patterns.

The lattice is initialized with a uniform distribution of species A chemicals and species B chemicals (For simplicity, only species A is shown). Additionally, a small stripe in the center of the species A lattice was initialized with a higher concentration. After fifty-thousand timesteps, the spatial distribution of species A forms a stable labyrinthine pattern. The simulation was run on a 512×512 lattice with $\gamma = 0.005$, $D_A = 0.1$, $D_B = 0.4$, $k_1 = 0.98$, $k_2 = 0.1$, $k_3 = 0.2$ and a maximum capacity per lattice site of $N = 50$.

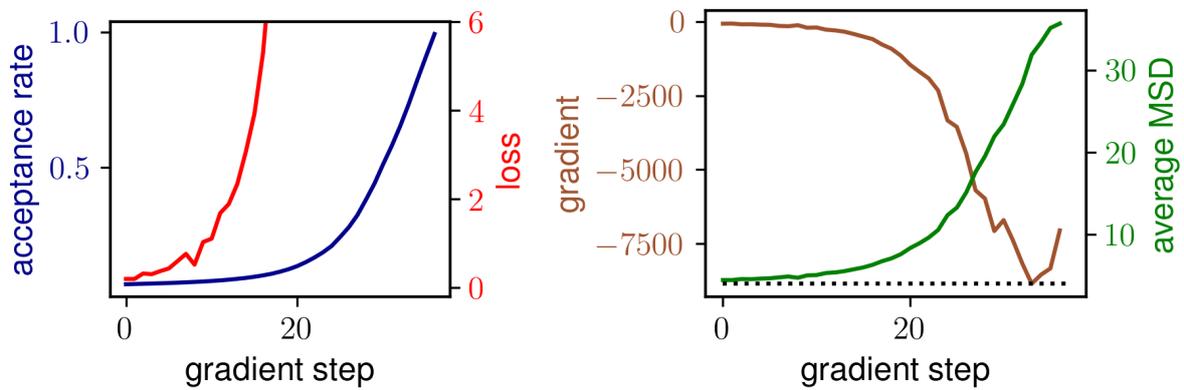


Figure S2: When the temperature is initialized too high, applying backpropagation via approximate gradients incurs exploding gradient issues.

The results of applying statistics fitting to the 1-pixel random walk CPM are shown for a target MSD of 4 spatial units over 800 steps. The temperature is initialized at a temperature that is too high. Standard gradient descent was applied with a learning rate of $1e^{-5}$. On the left, the temperature change and loss curve are shown for 40 steps of gradient descent. Over time, the loss and temperature values increase exponentially. In the right plot, the gradients and average MSDs are shown for 40 steps of gradient descent. Similar to the loss and temperature, the average MSD diverges exponentially from the target, while the negative gradient becomes exponentially larger.