Department of Mathematics and Computer Science
Data Mining Group

Towards Efficient Probabilistic Modeling of Crystallization at Mesoscopic Scale

A Case Study on Snow Crystal Growth

Author

Pol Timmer

Graduation supervisor

Vlado Menkovski

Graduation co-supervisor

Björn Baumeier

July 12, 2024

Preface

In the heart of every snowflake's delicate structure lies a story of intricate patterns and emergent complexity, with its symmetries and detailed patterns reflecting the snowflake's unique journey up to the moment of observation. My journey into the mathematical and physical beauty of snow crystals began with the inspiring work of Kenneth G. Libbrecht, which was brought to my attention by a fascinating video from Derek Muller's Veritasium. This thesis, *Towards Efficient Probabilistic Modeling of Crystallization at Mesoscopic Scale: A Case Study on Snow Crystal Growth*, embodies my endeavor to delve deeper into this mesmerizing world, leveraging neural networks to unravel and simulate the fascinating process of snow crystal formation.

With this work, I hope to contribute to the fields of materials science and snow crystal growth research in several ways. Firstly, by providing a neural surrogate model, I aim to accelerate snow crystal growth simulations as well as simulations of crystallization at the mesoscopic scale in general. Second, I provide a comprehensive dataset of snow crystal growth, intended as a foundational resource for further development of neural surrogates in the fields of mesoscopic crystallization modeling and snow crystal growth modeling. Finally, I hope my work on latent variable neglect and decoder dropout will serve as valuable resources for future research in neural simulation and conditional autoregressive latent variable models more broadly.

This thesis is structured as follows: Chapter 1 introduces the problem of snow crystal simulation and motivates our approach. Chapter 2 provides the necessary theoretical background on snowflake formation and the numerical simulators used for snow crystal growth. Chapter 3 contrasts our work with the existing body of literature and discusses relevant theoretical frameworks in machine learning for simulation and autoregressive latent variable models. Chapter 4 offers an overview of the first-principles simulator implemented to generate our training data. Chapter 5 details the formal problem formulation for snow crystal simulation, and includes the approach and a discussion of the results related to the problem of convergence to the identity function. Chapter 6 explores the problem formulation, approach, and findings related to latent variable neglect. Finally, chapter 7 summarizes the main findings and contributions of the thesis and outlines future work.

I would like to express my gratitude to everyone who supported me throughout this extensive research journey. Special thanks go to my graduation supervisor, Vlado Menkovski, and my graduate mentor, Koen Minartz, for their invaluable time, advice, and academic support, which were crucial in shaping this thesis into its present form. The considerable effort and energy invested in this work have not only satisfied me with its outcome, but have also provided me with a deeper understanding into academics and machine learning research. This experience has granted me the confidence to pursue a career in this exciting field. Additionally, I am thankful to my family, friends, and those closest to me, whose support was essential in bringing this thesis to completion. Finally, I thank you, the reader, for dedicating time to engage with my work. I sincerely hope you find it as engaging as I have found the journey of creating it.

Abstract

Snowflakes, with their symmetrical beauty and intricate designs, serve as an everyday example of the complex, nonlinear patterns observed in mesoscopic-scale crystallization processes, revealing levels of complexity that simple molecular explanations cannot account for. Through emergence, simple molecular systems display highly nonlinear complex pattern formation at the mesoscopic scale, such as faceted and dendritic growth in snow crystals and multigrain formation in metals. The pattern morphologies that form through these processes significantly influence material properties at the macroscopic scale, making them of particular interest within materials science and metallurgy. Due to their nonlinear, stochastic nature and sensitivity to environmental parameters, modeling these processes is highly challenging. Traditional numerical modeling methods for simulating these processes are computationally intensive. This work aims to scale crystal growth simulation with a machine learning emulator. Specifically, autoregressive latent variable models are well suited for modeling the joint distribution over system parameters and the crystallization trajectories. However, successfully training such models is challenging due to the stochasticity and sensitivity of the system, as well as difficulties surrounding the latent variables. Existing approaches consequently fail to produce diverse and faithful crystallization trajectories.

In this thesis, we introduce the Crystal Growth Neural Emulator (CGNE), a probabilistic model for efficient crystal growth emulation at the mesoscopic scale that overcomes these challenges. We validate this model on the process of snow crystal growth, and we introduce and address two research questions concerning the difficulties of latent variable models. The first question surrounds the issue of convergence to the identity function, which we address by implementing a strategy of temporal downsampling to increase step sizes, effectively capturing significant state changes without oversimplification. The second research question surrounds Latent Variable Neglect (LVN), an issue where models overlook latent variables due to dominant decoder conditioning. To address this, we incorporate decoder dropout, enhancing the model's ability to integrate crucial latent information for diverse trajectory prediction.

To generate the snow crystal growth dataset we use for our experiments, we introduce a GPU-accelerated version of the Gravner-Griffeath 2D snow crystal growth algorithm. These experiments demonstrate that CGNE achieves a substantial 11-fold increase in inference speed over traditional numerical modeling, while able to simulate realistic and diverse trajectories of snow crystal growth. This work substantially advances the simulation of crystallization at the mesoscopic scale by introducing a scalable, efficient, and robust neural simulator—the first of its kind in this field—to emulate complex growth patterns.

Contents

Chapter 1

Introduction

The formation of complex structures during solidification has long captivated researchers due to the intricate interplay of nonequilibrium and nonlinear processes involved. These processes manifest in various systems, including the crystallization of metals [18] and the growth of snow crystals from water vapor. Particularly at the mesoscopic scale, emergence leads to complex structural formations during the solidification process: while at the atomic scale one might see atoms arrange themselves into a well-organized lattice, and at the macroscopic scale alloys solidify to an ingot, intricate pattern formations caused by the polycrystalline solidification process arise at the mesoscopic scale. Examples of this emergent complexity include the faceted and dendritic growth seen in snow crystals and the multigrain formation in metals. The crystalline morphologies that develop at the mesoscopic scale significantly influence the material properties at the macroscopic scale. Consequently, the study of these processes is not only of scientific interest but also carries substantial practical implications for fields such as materials science and metallurgy. For water specifically, despite its material simplicity as a monomolecular system, emergence causes it to arrange itself in an incredibly wide range of morphologies depending on environmental conditions, and the phenomenon of snow crystal growth remains poorly understood.

In "The Chemical Basis of Morphogenesis" [52], Alan Turing expressed his ideas about the nonequilibrium reaction-diffusion systems of locally interacting chemicals in the language of partial differential equations (PDE). Partial differential equations are a mathematical formalism that is especially useful for characterizing systems where a detailed description of the state at every point and moment is unfeasible, and where local rates of change in space and/or time provide a more practical approach to understanding the system's behavior. Snow crystal growth fits this category of systems, as their global emergent structures are very complex and nonlinear, while the local rules are seemingly simple.

The complex structures of snow crystals emerge from their diffusion-limited growth and highly anisotropic surface attachment kinetics. These properties make snow crystals exhibit their unique type of crystallization that combines both dendritic and faceted growth. These features make the study of snow crystal growth particularly interesting and relevant to a range of scientific fields, as understanding this complex process implies gaining a better understanding of related processes, like other forms of solidification [18]. By "understanding",

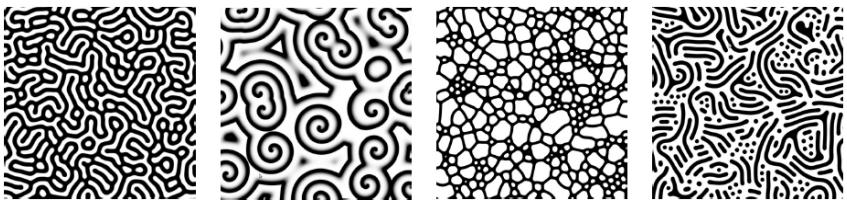


Figure 1.1: Four chemical reaction-diffusion simulations done with cellular automata. [49]

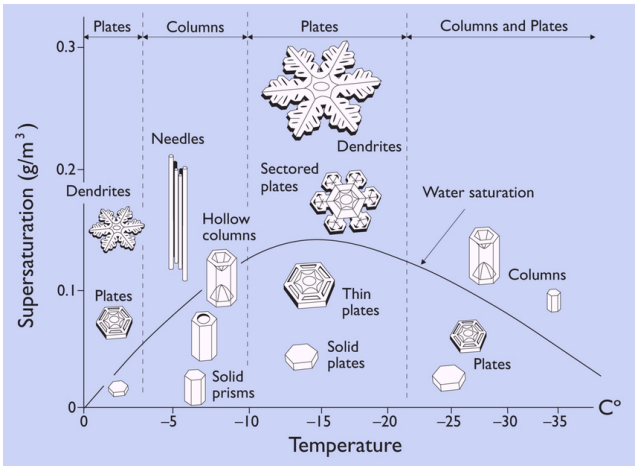


Figure 1.2: The Nakaya diagram illustrates early rigorous findings on the relations between environmental conditions and snowflake morphologies [40].

we mean the discovery of relations between environmental conditions and morphologies, shown in Figure 1.2, paired with a plausible physical explanation for these relations.

1.1 Solving a Partial Differential Equation

In situations where an analytical solution is not tractable, a common method to solve a partial differential equation is through numerical approximation, or simulation. The general approach of these methods is the discretization of the process in space and time. The underlying premise is that computational calculations are more manageable and straightforward when the problem is discretized, and when the step size is sufficiently small, the discretized approximation will closely mirror reality. Which approximation is appropriate depends on the type of PDE in question. For PDEs that involve moving boundaries, such as the boundary of a growing snow crystal, front-tracking models frequently represent a preferred approach. Various implementations of front-tracking models for snow crystal growth [56, 55, 30] have delivered structures closely resembling real-world observations. However, these implementations are often burdened by the complexity of their algorithms, which are filled with edge cases to address the numerical instabilities inherent at the sharp boundary. In the broader context of solidification problems, phase field models are generally the methodology of choice. These models have indeed provided valuable insights into the underlying mechanisms of snow crystal growth and have allowed for the prediction of various morphologies under different growth conditions [23, 24, 4, 21].

However, these models failed to capture the full range of morphologies that snow crystals can present. In particular, obtaining faceted growth in combination with dendritic branching was a challenge, due to dynamic and numerical instabilities [2, 32]. For more detail on front-tracking and phase field models we refer to section 2.2.1.

1.1.1 Cellular Automata

A notable category of numerical solvers for PDE's, particularly adept at simulating diffusion processes, is cellular automata. This class of computational models has led to significant breakthroughs in the realm of snow crystal simulation, capturing previously unseen morphological properties [35]. Cellular automata approach discretization by establishing a grid to represent spatial dimensions and employing discrete computational steps to simulate the progression of time. In this framework, each cell within the grid adheres to a uniform set of

(often local) rules. At each computational step, the future state of a cell is determined based on the state of its neighborhood, thereby integrating many simple local interactions over time to model complex global behaviors. This paradigm is further explained in section 2.2.3.

1.2 Neural Simulators and PDE Solvers

However, these discretized simulations frequently suffer from a scaling problem. Their computation time increases linearly as the temporal resolution is enhanced, and scales as $\mathcal{O}(n^d)$ with the spatial resolution, where n is the spatial resolution, and d is the number of spatial dimensions. This poses a significant challenge: a lower resolution might expedite the computation, but at the cost of diminished accuracy, potentially leading to simulations that fail to capture critical dynamics or fine details. In contrast, high-resolution simulations, though more precise, can become computationally prohibitive, especially for high-dimensional, large-scale, or complex systems.

To address these computational constraints, the advent of neural network-based simulators offers a promising alternative. Leveraging the power of machine learning, these neural simulators can learn to approximate the behavior of a system from a set of training data. While a high spatial and temporal resolution is necessary for the numerical stability and accuracy of traditional simulators, neural simulators aren't bound by this constraint. These approximations can therefore be at resolutions significantly lower than those required from traditional simulations, while maintaining simulation stability and accuracy. Once trained, these neural simulators can predict system behavior under new conditions much more rapidly than traditional simulations, without the need for real-time computation of every fine detail. This approach significantly reduces computational load while retaining a high level of accuracy, making it an invaluable tool in scenarios where the speed of traditional PDE simulators is insufficient. Motivated by these advantages, we explore the idea of developing a neural simulator for snow crystal growth.

In the development of a neural simulator, or any computational model, the selection of an appropriate evaluation metric is crucial. In light of our aim to simulate realistic, probabilistic growth patterns across various environmental conditions, pixel-wise accuracy would be an inadequate evaluation metric. For a highly branched snowflake, a minor deviation, such as a thin branch displaced by a mere pixel from its expected position, would drastically reduce pixel-wise accuracy. However, at a macroscopic level, this snowflake, with its marginally shifted branch, would still bear close resemblance to the ground truth. Their visual similarity would be apparent on a stylistic level, rather than pixel-wise level, and a human observer would place the two snowflakes within the same 'family'. A human, style-based assessment like this focuses less on the precise positioning of branches and more on higher-level morphological properties such as degree of branching (branchedness), branch thickness, length, and total number of branches. Two intuitive morphological properties are crystal area and boundary length. As the branchedness increases, we observe a decrease in crystal area and a corresponding increase in boundary length. These high-level morphological properties, therefore, serve as reliable indicators of a snowflake's branchedness, and therefore its macroscopic appearance. In this thesis, we introduce a novel data-driven neural simulator dedicated to the simulation of snow crystal growth. To evaluate our model, we focus on validating these morphological properties—specifically, the area and boundary length of the simulated crystals.

Specifically, given a dataset of snow crystal growth trajectories paired with environmental parameters generated from a ground-truth simulation process, we aim to develop a data-driven simulator that can generate these trajectories based on the environmental conditions. The goal is for the simulated growth trajectories to closely resemble those produced by the ground-truth simulation process, both in terms of accuracy and diversity. To measure

this, we compare the joint distribution over crystal area and boundary length from the ground-truth simulator to our learned simulator.

1.2.1 First-Order Markov Autoregressive Models

Another crucial selection component in the development of a neural simulator is model architecture. For neural simulation, the paradigm of autoregressive modeling is a suitable choice. Specifically, we opt for an autoregressive latent variable model with a first-order Markov property. The model architecture resembles a conditional variational autoencoder [26], where the decoder $p_\theta(x_{t+1} \mid z, x_t)$ is informed by the latent variable z in conjunction with a conditioning signal, which in this case is the current state of the simulation x_t. The model features a learned conditional prior $p_\theta(z \mid x_t)$ inspired by the PNS framework by Minartz et al. [38]. Inference of the model is formalized as:

$$p_\theta(x_{t+1} \mid z, x_t), \quad z \sim p_\theta(z \mid x_t) \tag{1.1}$$

Where θ are the learned parameters for the model decoder and conditional prior. The encoder, or variational posterior, is only used for training, and autoencodes over the simulation state at the subsequent timestep x_{t+1}. It is formalized as:

$$q_\phi(z \mid x_t, x_{t+1}) \tag{1.2}$$

We motivate this model and provide a more comprehensive description in chapter 5, and we supply theoretical background in chapter 3.

1.2.2 Challenges in Autoregressive Latent Variable Models

However, training autoregressive latent variable models presents significant difficulties. These issues are particularly associated with low signal-to-noise ratio of the gradients flowing through the latent variables, and minimal differences between sequential data points. A key contribution of this thesis is the clear identification of these challenges, and a comprehensive overview of methods to address them.

A distinct problem emerges in autoregressive models that lack memory or positional encoding with respect to the time step t, especially when the difference Δx_t between sequential data points is marginal. In such models where a first-order Markov property holds, the proximity of x_t to x_{t+1} can lead the model to converge to a local minimum where the model learns a simplistic identity function, which presents a challenge for learning more complex patterns. In the context of neural simulators, this issue frequently arises due to the inherently small state changes in first-principles simulators, necessary to ensure accuracy. To address this, we put forward the following research question:

RQ1 How can we best model systems characterized by very small state changes using first-order Markov autoregressive models, while avoiding convergence to the simplistic identity function?

Additionally, the minimal differences between consecutive data points often make it viable to largely predict the next data point x_{t+1} based only on the preceding one x_t, without the need for any other information, such as that coming from a latent variable. A recurring issue in autoregressive latent variable models, particularly those with a decoder conditioned on a signal that substantially predicts the output, is the tendency of the model to overlook the latent variable z. This behavior emerges due to a low signal-to-noise ratio in the latent variable compared to the conditioning signal that contains the previous data point x_t. This often creates a local minimum from which the model struggles to escape. In our model, this problem manifests itself because the decoder $p_\theta(x_{t+1} \mid z, x_t)$ is conditioned on x_t, and

therefore is largely able to predict x_{t+1} without the latent variable z. This behavior is not only specific to our model, but is also observed in similar models across different data domains, such as text [6]. However the exact conditions under which this issue occurs are not fully known. In light of this, we propose the following research question:

RQ2 What are the specific conditions and model classes where the issue of latent variable neglect arises, and what strategies can effectively address this challenge for such models under these conditions?

In addressing this issue, we pinpoint the specific class of models where this problem frequently arises and suggest an effective solution to alleviate this challenge within these models.

1.3 Contributions

This work brings the following contributions to the academic community:

- We present an open-source, GPU-accelerated implementation of the predominant algorithm for simulating snowflake formation. This optimized implementation significantly improves computational efficiency over previous non-GPU versions. This software is optimized for dataset generation, thereby providing an accessible method for generating novel datasets on snow crystallization.
- We present the Crystal Growth Neural Emulator (CGNE), a data-driven neural emulator for efficient probabilistic modeling crystallization processes at the mesoscopic scale. We validate this model on our synthetic dataset of snow crystal growth. This neural surrogate significantly accelerates the simulation process.
- We identify the class of models and training conditions necessary for the problem of the model converging to the identity function.
- We present a method to mitigate the risk of convergence to the identity function.
- We formalize the problem of Latent Variable Neglect (LVN), as well as the class of models which suffer from this LVN.
- We present a method to mitigate LVN, which has significant implications for neural simulation problems beyond crystal growth.

1.4 Research Outline

The rest of this research is structured as follows. Chapter 2 contains the necessary background knowledge and literature in the context of crystallization and classical numerical modeling methods. Chapter 3 contains the essential background knowledge and literature about simulation with machine learning. Chapter 4 includes a detailed overview of the first-principles simulator. Chapter 5 contains the problem formulation, approach, and results related to RQ1. Chapter 6 contains the approach and results related to RQ2. Finally, chapter 7 presents the conclusions and future work section of the thesis.

Chapter 2

Background and Related Work on Crystallization and Classical Simulators

2.1 Snow Crystal Formation

Despite the many shapes that snow crystals exhibit, as seen in Figure 2.1, they all share distinct characteristics that unmistakably identify them as snow crystals. One key characteristic is their faceted nature, often visible as flat, plate-like formations that form the basic structure of the snowflake. Another defining feature is their branching patterns, which can range from non-existent, resulting in a simple plate-like snowflake, to highly intricate, creating a complex and branched structure. Finally, all these formations adhere to hexagonal symmetry, a defining trait of snow crystal morphology. This symmetry guides the development of both the facets and the branches, resulting in a vast variety of forms, each uniquely intricate yet unmistakably snowflake-like.

The interplay of branched growth caused by diffusion-limited solidification and faceted growth caused by highly anisotropic attachment kinetics both governed by hexagonal symmetries makes for the distinctive look of snow crystals. These three characteristics of faceting, branching, and hexagonal symmetries all have underlying physical explanations.

2.1.1 Hexagonal Symmetry

The hexagonal symmetry of snow crystals can be attributed to the hexagonal molecular crystal lattice of water molecules, as seen in Figure 2.2, which fundamentally shapes

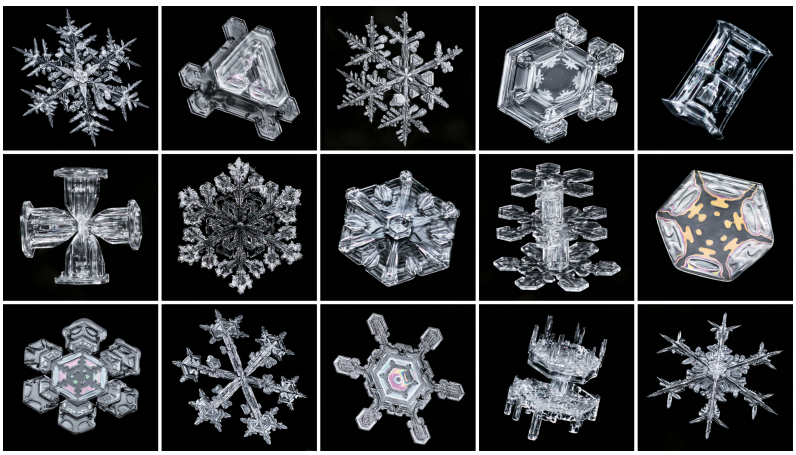


Figure 2.1: Various shapes of snow crystals.

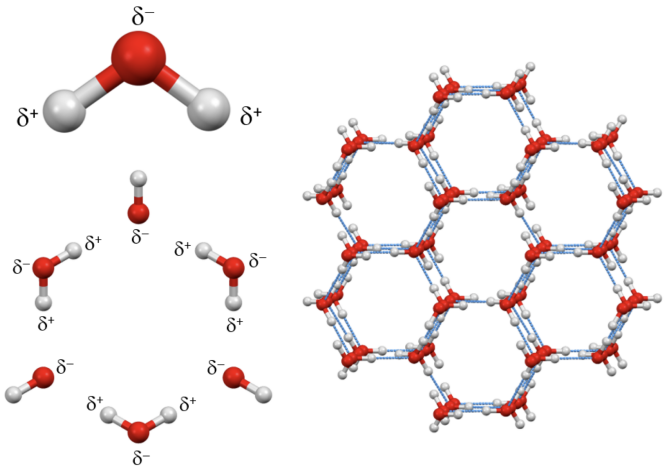


Figure 2.2: The ice crystal lattice.

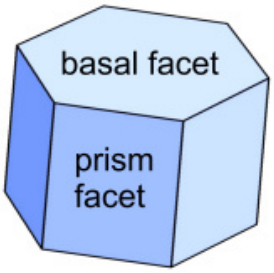


Figure 2.3: A prism and its facets.

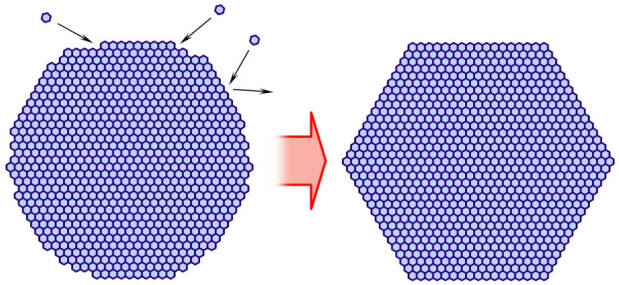


Figure 2.4: The origin of facets in nature. Rough spots have more available chemical bonds, and therefore molecules are more likely to attach to those spots than to smooth surfaces. This causes the formation of flat facets.

the symmetry observed in snow crystals. When water vapor crystallizes under specific atmospheric conditions, it naturally aligns into this hexagonal pattern. This alignment not only dictates the microstructural arrangement, but also influences the macroscopic form of each snowflake, culminating in these hexagonal symmetries appearing all over the snowflake's structure.

2.1.2 Faceting

The faceted growth of snowflakes stems from the underlying anisotropic attachment kinetics. As shown in Figure 2.3, water tends to arrange itself in hexagonal shaped structures, with sharp points and smooth faces, rather than molecularly rough corners. In the scenario of a macroscopically round corner, the underlying molecular structure is rough, with many available chemical bonds. Approaching water vapor particles are likely to attach to these rough spots and are less likely to attach to molecularly smooth surfaces. As a consequence, round corners tend to fill out to form sharp corners connected by smooth facets, causing mirror-like facets to appear over the snowflake.

Figure 2.5: A snow crystal simulation showing diffusion-limited growth. The level of water vapor is higher far away from the crystal (grey) but diminishes near the crystal. Branch tips are naturally closer to the water vapor needed for growth, while gaps in between branches are depleted of vapor, causing branch tips to grow further while gaps do not get filled in. Image from [35].

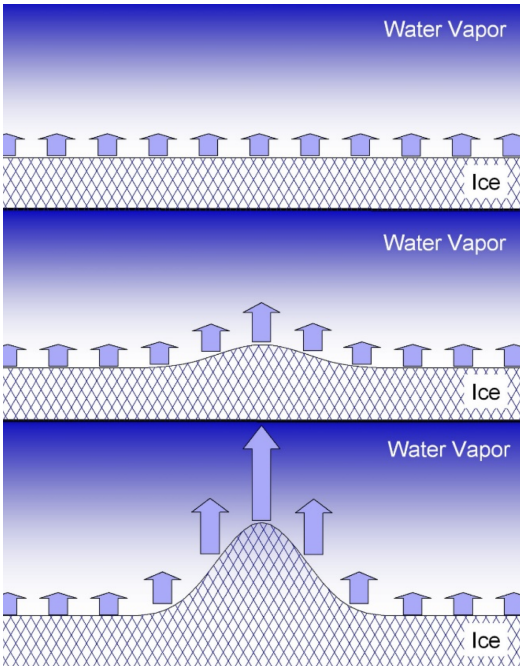


Figure 2.6: The diffusion-limited growth of an initially flat surface is succeptible to the Mullins-Sekerka instability, also known as the branching instability. This instability is caused by the positive reinforcement cycle of diffusion-limited growth. Image from [35].

2.1.3 Branching

The branching in snow crystals arises from the fact that snow crystal growth is a diffusion-limited solidification process. Diffusion-limited growth is the phenomenon in which the growth rate of a solid is controlled by the rate at which molecules can diffuse through the surrounding medium. In such a process, areas of the solid closer to regions with a high concentration of diffusive molecules experience more rapid growth. This leads to a self-reinforcing cycle where the tips of growing dendrites progressively extend towards areas with more diffusive mass, visualized in Figure 2.5, promoting the emergence of branched structures. This positive feedback loop forms an instability known as the Mullins-Sekerka instability [39], displayed in Figure 2.6. This instability implies that a small bump can trigger branch formation on a flat surface. In the context of snow crystals, their anisotropic attachment kinetics imply that branch formation tends not to start on flat surfaces, but rather on the corners of their intrinsic hexagonal structures, leading to the formation of the six primary branches characteristic of snowflakes.

This means that snow crystal formation is a specialized case of the more general problem of diffusion-limited solidification, with its special features being its six-fold symmetry, as well as faceted growth caused by its highly anisotropic surface attachment kinetics. The exact physical rules of this process have long been debated, and are still poorly understood, but have gained significant progress through the construction and analysis of numerical models.

2.2 Numerical Modeling Methods

2.2.1 Front-Tracking Models

In earlier attempts, the construction and analysis of numerical models for the research of snow crystal growth came through front-tracking models [56, 55, 30], where a sharp solidification front between solid ice and the water-vapor field surrounding it is clearly defined. As shown by Barrett et al. [2], this strategy is capable of generating 3D structures that are both faceted and branched. However, these front-tracking models are algorithmically complex, due to the challenge of maintaining the polygonal mesh that defines the surface of the crystal, and due to the numerical instabilities at the sharp boundary.

2.2.2 Phase Field Models

Another strategy for numerical modeling of diffusion-limited growth is with phase-field models. Unlike front-tracking approaches, this method introduces an artificial phase field parameter that is set to -1 for the water-vapor phase and +1 for the ice phase. This parameter smoothly transitions between these values within a spatially extended boundary region between the two phases (which is at least several pixels wide in the model). By eliminating the sharp solidification boundary in this way, phase-field models can use simpler numerical propagation algorithms [23, 24, 4, 21]. However, while previous work on phase-field snow crystal models has shown promising results in producing structures that are both faceted and branched, the method never seemed to work nicely with the highly anisotropic surface attachment kinetics of snow crystals, leading to physically unrealistic numerical models.

2.2.3 Cellular Automata

Local Cellular Automata (LCA) are a mathematical modeling paradigm for complex systems, where space and time are discrete and interactions are local. In an LCA model, a grid of cells evolves over discrete time steps according to a set of rules based on the states of neighboring cells. These simple local rules can lead to complex and often surprising behavior at a macroscopic level, making cellular automata a powerful tool for studying emergent phenomena in systems ranging from biological to computational contexts. Perhaps the most renowned example of an LCA would be Conway's Game of Life [9], which demonstrates how complexity can arise from simplicity. In the Game of Life, cells on a grid can be in one of two states: alive or dead. The state of each cell in the next generation is determined by the number of living neighbors it has, following a simple set of birth, death, and survival rules, shown in Figure 2.7. This model, despite its simplicity, can generate patterns that move, self-replicate, and interact in intriguing ways, to the point of even emulating the entire Game of Life within the Game of Life, shown in Figure 2.8. This illustrates the concept of emergence.

This aptitude to model emergent phenomena is precisely what makes them so suitable for modeling the growth of snow crystals. As highlighted by Libbrecht [35], this paradigm is increasingly recognized for its potential in rendering accurate simulations of snow crystal development. In literature, LCA have shown very impressive expressive capabilities [13], showing previously unseen morphological features, while being algorithmically far more simplistic than front-tracking and phase-field models [35].

2.3 Packard Snowflakes

One of the first works on simulating the solidification of structures with cellular automata came by Norman Packard in 1986 [43]. His so-called *Packard Snowflakes* were some of the first simulated structures resembling the shape of a snow crystal, and as such, his work

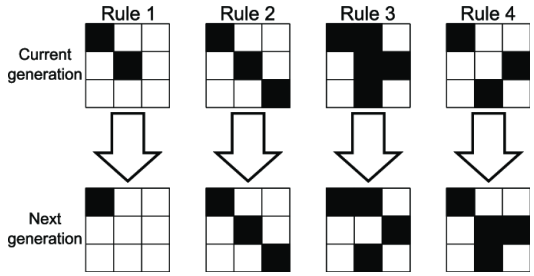


Figure 2.7: Rules of Conway's Game of Life. Even though the game has only 4 simple rules, the system is capable of displaying profoundly complex behavior. [16]

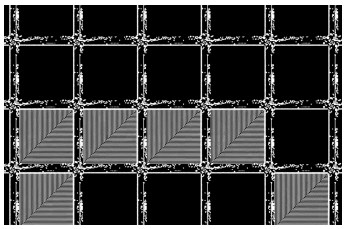


Figure 2.8: Game of Life being emulated in Game of Life

has been widely publicized to illustrate how very simple algorithms can emulate complex natural phenomena. His simplistic model was able to capture a relatively wide range of morphologies, one being displayed below in Figure 2.9.

In their rigorous 2006 study on Packard's digital snowflakes, Janko Gravner and David Griffeath present comprehensive analyses and results on Packard's snowflake models [12]. The algorithm itself is supremely simple. On a hexagonal lattice, the LCA evolves the snowflake starting from one simple "seed" cell surrounded by vapor cells, and a simple binary rule. The grid cells can only be in a frozen, or vaporous state, and can only transition from vaporous to frozen. Packard argued that snowflakes tend to grow more at the tips, therefore some of his snowflakes have the simple rule: A vapor cell freezes if and only if it has exactly one frozen neighbor. Gravner and Griffeath refer to this rule as *Hex 1*. Another, more widely publicized variant of Packard's snowflakes, is the snowflake created with the rule *Hex 135*. This rule sees a vapor cell freeze if and only if the amount of neighbors is odd (1, 3, or 5). Similarly, the rules for *Hex 134* and *Hex 14* are based on the same principle, where a vapor cell turns to ice based on a certain number of frozen neighbors, as indicated by the numbers in their names.

This work served as a cornerstone for future iterations of LCA-based snow crystal growth models. Building upon Packard's foundational approach, subsequent research in the field of LCA-based models for snow crystal growth has expanded and refined these initial concepts. These later models have introduced more complex rules and parameters to more accurately mimic the intricate and diverse patterns observed in natural snowflakes. This evolution of the field highlights the enduring impact of Packard's early work, setting the stage for a more nuanced understanding of the delicate interaction between simple rules and complex natural phenomena.

2.4 The Gravner-Griffeath 2D Algorithm for Snow Crystal Growth

Expanding upon their insightful analysis of Packard's models [12], Gravner and Griffeath unveiled a novel algorithm in 2008 for simulating snow crystal growth [13], which holds significant relevance to our research. The algorithm they present forms the foundation of the data-generating simulator used in our study, and consequently the development and analysis of our data-driven simulator. Their work aims to enhance physical realism by more accurately modeling the boundary dynamics. Their innovative approach was the first to successfully capture critical properties such as branch faceting, as exemplified in Figure 2.10, thereby bridging a gap in the modeling the interplay of branched and faceted growth.

The Gravner-Griffeath algorithm retains the Local Cellular Automaton framework operating on a two-dimensional hexagonal lattice, but whereas Packard's algorithm had a simple binary state space for the cells, the Gravner-Griffeath algorithm introduces a

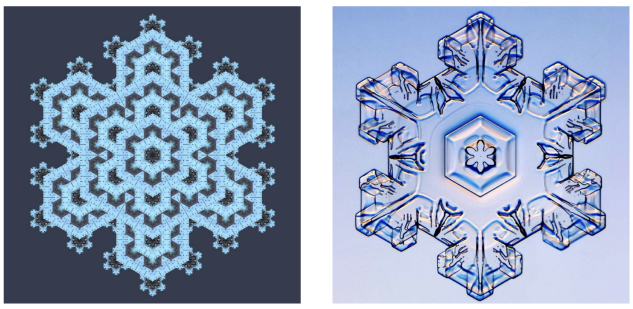


Figure 2.9: A Packard Snowflake closely resembling the shape of a real snowflake. [35]

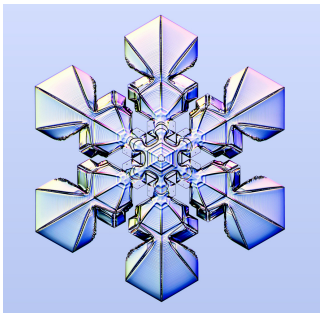


Figure 2.10: A snowflake displaying branch faceting, where its branches exhibit facets. Image from [36].

state space with three continuous dimensions and one binary dimension. Furthermore, their algorithm includes a phase not present in the Packard models [43, 12], namely the *quasi-liquid* phase. The reason behind the inclusion of this phase in their algorithm comes from the authors' stated consensus between researchers that a quasi-liquid layer makes up one of the three main effects behind the growth of snow crystals [13], along with diffusion-limited solidification and anisotropic attachment kinetics. They explain the quasi-liquid layer as a thin layer, very near to the crystal surface, where molecules are bound more tightly than in the surrounding vapor, but are not completely bound to the crystal lattice, and are therefore in a hybrid state. The authors argue that the inclusion of a separate mass-field for the quasi-liquid phase allows them to model the qualitatively different dynamics for this thin layer.

The authors choose the centers of the hexagonal grid cells to be sites of the triangular lattice $\mathbb{T}$ so that each site $x \in \mathbb{T}$ has six nearest neighbors. Formally, the state of the system at time t at site x is $\xi_t(x) = (a_t(x), b_t(x), c_t(x), d_t(x))$, where

$$a_t(x) = \text{ the } \mathbf{a}\text{ttachment flag at } x \text{ at time } t = \begin{cases} 1 & \text{if } x \text{ is part of the crystal} \\ 0 & \text{otherwise} \end{cases}$$

$$b_t(x) = \text{ the } \mathbf{b}\text{oundary mass at } x \text{ at time } t \textit{ (quasi-liquid)}$$

$$c_t(x) = \text{ the } \mathbf{c}\text{rystal mass at } x \text{ at time } t \textit{ (ice)}$$

$$d_t(x) = \text{ the } \mathbf{d}\text{iffusive mass at } x \text{ at time } t \textit{ (vapor)}$$

The snowflake consists of entirely ice, the boundary can consist of all three mass types at once, and only vapor can exist anywhere else. The system starts off with crystal mass 1 at the origin, and diffusive mass ρ everywhere else. Formally, $a_0(\mathbf{0}) = c_0(\mathbf{0}) = 1$, $b_0(\mathbf{0}) = d_0(\mathbf{0}) = 0$, and for all $x \neq 0$ $a_0(x) = b_0(x) = c_0(x) = 0$, and $d_0(x) = \rho$. The algorithm then cycles through 4 computation steps, for thousands of iterations. These steps are:

1. Diffusion
2. Freezing
3. Attachment
4. Melting

Optionally followed by a fifth **noise** step.

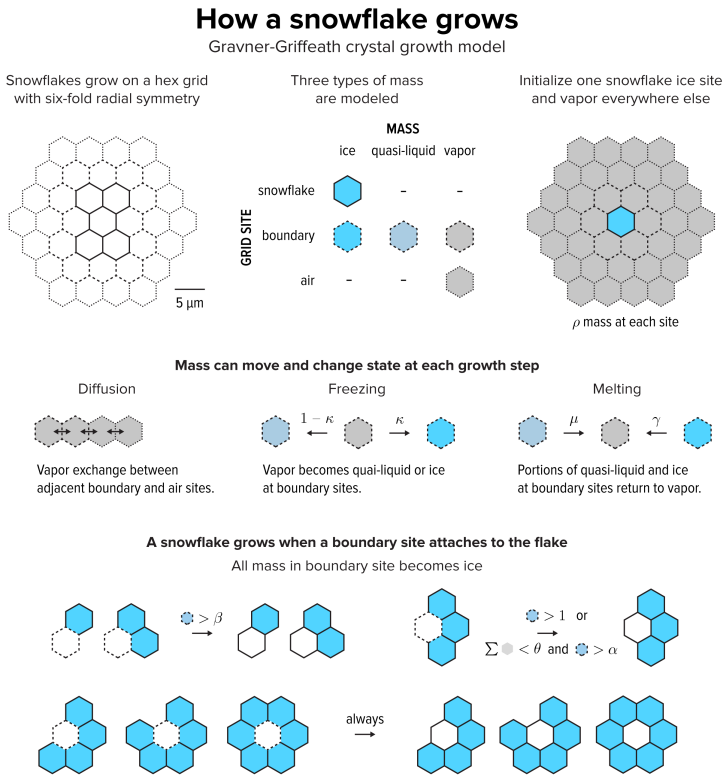


Figure 2.11: An overview of the two-dimensional Gravner-Griffeath algorithm. Image from [37].

A visual overview of the algorithm is displayed in Figure 2.11. For the remainder of this chapter, denote the neighborhood of x as $\mathcal{N}_x = \{x\} \cup \{y : y \text{ is a nearest neighbor of } x \text{ in } \mathbb{T}\}$, and set

$$\begin{aligned} A_t &= \{x : a_t(x) = 1\} && = \text{the snowflake at time } t; \\ \partial A_t &= \{x \notin A_t : a_t(y) = 1 \text{ for some } y \in \mathcal{N}_x\} && = \text{the boundary of the snowflake at time } t; \\ A_t^c &= \{x : a_t(x) = 0\} && = \text{the sites not in } A_t; \\ \bar{A}_t^c &= (A_t \cup \partial A_t)^c && = \text{the sites not in } A_t \text{ or } \partial A_t. \end{aligned}$$

Also, we use $^\circ$ (degree) and $'$ (prime) notation to denote amounts of mass before and after a substep is completed. At the end of each cycle of substeps the time t advances to $t + 1$.

2.4.1 Diffusion

The diffusion step simulates the diffusion of vapor in the system. It simulates the transport of diffusive mass over cells not in the snowflake. It is simply implemented with the discrete version of the heat equation, with uniform weight $\frac{1}{7}$ on the center site and each of its neighbors. Formally, for $x \in \bar{A}_t^c$,

$$d_t'(x) = \frac{1}{7} \sum_{y \in \mathcal{N}_x} d_t^\circ(y), \tag{2.1}$$

and for $x \in \delta A_t$ any term in the sum corresponding to $y \in A_t$ is replaced by $d_t^\circ(x)$.

2.4.2 Freezing

The freezing step transforms all vapor at boundary sites to a proportion κ of ice, and a proportion $(1-\kappa)$ of quasi-liquid. That is, for $x \in \delta A_t$,

$$\begin{aligned} b'_t(x) &= b^\circ_t(x) + (1-\kappa)d^\circ_t(x), \\ c'_t(x) &= c^\circ_t(x) + \kappa d^\circ_t(x), \\ d'_t(x) &= 0. \end{aligned} \tag{2.2}$$

2.4.3 Attachment

The attachment step determines whether a boundary site becomes part of the snow crystal. In this key step, the authors implement the anisotropic attachment kinetics at work at the crystal border δA_t. The authors implement this by having the condition for attachment of x depend on the number of attached neighbors of x at time t. Let $n^\circ_t(x) = \#\{y \in \mathcal{N}_x : a^\circ_t(y) = 1\}$ be the number of attached neighbors of x at time t. The attachment rule does a case distinction over 3 conditions:

$$\begin{aligned} &n^\circ_t(x) = 1 \text{ or } 2, \\ &n^\circ_t(x) = 3, \\ &n^\circ_t(x) \geq 4, \end{aligned}$$

Because sites with more crystallized neighbors have more available chemical bonds for the site to crystallize, attachment is naturally easier for these sites. This is reflected in the algorithm by lower required thresholds of boundary mass for sites x where $n^\circ_t(x)$ is higher. Therefore, in the first case for a boundary site with 1 or 2 attached neighbors, a high boundary mass of at least β is required for attachment, where $\beta > 1$ is a tunable parameter:

$$\begin{aligned} \text{If } x \in \partial A^\circ_t, \quad n^\circ_t(x) = 1 \text{ or } 2, \quad \text{and} \\ b^\circ_t(x) \geq \beta, \quad \text{then } a'_t(x) = 1. \end{aligned} \tag{2.3}$$

A boundary site with 3 attached neighbors joins the crystal if it either has boundary mass ≥ 1, or it has diffusive mass $< \theta$ in its neighborhood and it has boundary mass $\geq \alpha$:

$$\begin{aligned} \text{If } x \in \partial A^\circ_t, \quad n^\circ_t(x) \geq 3, \quad \text{and either} \\ b^\circ_t(x) \geq 1 \quad \text{or} \quad \left(\sum_{y \in \mathcal{N}_x} d^\circ_t(y) < \theta \quad \text{and} \quad b^\circ_t(x) \geq \alpha \right), \\ \text{then} \quad a'_t(x) = 1. \end{aligned} \tag{2.4}$$

Finally, boundary sites with 4 or more attached neighbors automatically join the snow crystal.

$$\text{If } x \in \delta A^\circ_t, \quad n^\circ_t(x) \geq 4, \quad \text{then } a'_t(x) = 1. \tag{2.5}$$

Once a site is attached, all its mass becomes crystal mass:

$$\begin{aligned} \text{If } x \in \partial A^\circ_t, \quad \text{and } a'_t(x) = 1, \quad \text{then} \\ c'_t(x) = b^\circ_t(x) + c^\circ_t(x), \quad \text{and } b^\circ_t(x) = 0. \end{aligned} \tag{2.6}$$

Attachment is permanent, and there are no further dynamics at the attached sites.

2.4.4 Melting

Proportion μ of quasi-liquid boundary mass and proportion γ of crystal mass at each boundary site becomes vapor. Formally, for $x \in \delta A_t$,

$$\begin{aligned} b_t'(x) &= (1-\mu)b_t^\circ(x), \\ c_t'(x) &= (1-\gamma)c_t^\circ(x), \\ d_t'(x) &= d_t^\circ(x) + \mu b_t^\circ(x) + \gamma c_t^\circ(x). \end{aligned} \tag{2.7}$$

This represents a reverse effect of the freezing step. Typically μ is small and γ extremely small.

2.4.5 Noise

The statistical physics of snow crystal growth are only deterministic in the limit. On the scale of a snow crystal, while nearly deterministic, the fine details are still subject to chance. Therefore, the noise step executes an independent random perturbation of proportion σ on the diffusive mass at each site:

$$d_t'(x) = (1 \pm \sigma)d_t^\circ(x) \quad \text{with probability } \frac{1}{2} \text{ each.} \tag{2.8}$$

2.4.6 Motivation of Choice

We opted for this algorithm for its balance between simplicity and expressive power, and for its comprehensive description. While Packard's snowflakes are straightforward to implement, they offer a limited range of morphological outcomes. The more physically accurate models mentioned later display a vast array of morphologies, but their algorithmic descriptions lack the clarity needed for replication. In contrast, the Gravner-Griffeath work stands out for its comprehensive description, with even the original C implementation available, though it is somewhat opaque.

An additional benefit of the Gravner-Griffeath 2D algorithm lies in its generation of 2D image data. In the context of the data-driven simulator, working with three-dimensional data presents its own difficulties. The Gravner-Griffeath 2D algorithm for snow crystal growth therefore poses a more favorable choice than its later discussed 3D counterpart.

2.5 The Gravner-Griffeath 3D Algorithm for Snow Crystal Growth

In 2009, Janko Gravner and David Griffeath improved on their earlier model with their three-dimensional approach [14]. In his book, Kenneth Libbrecht [35] calls this work "a significant breakthrough in modeling snow crystal growth, as it so clearly demonstrated the great potential of the cellular-automata method, especially for full 3D simulations". What was mainly so impressive, was the remarkable degree to which many surface structural details matched those seen in natural snow crystals. This increase in physical accuracy comes with an increase in model complexity. Changes with respect to the 2008 model [13] include the obvious step from a two-dimensional to a three-dimensional simulation. However, the model also sees an increase from 6 to a total of 22 tuneable model parameters. Furthermore, to model the dynamics of a snowflake falling from the sky, the authors have added a slight drift term in the diffusion step. Finally, the authors have removed the presence of crystal mass, and ice is now simply identified as boundary mass that is part of the crystal.

The algorithm retains the core structure of their previous 2D model, cycling through the same substeps of diffusion, freezing, attachment, and melting in a recurring and iterative manner. However, these substeps have been suitably modified to accommodate the transition to a three-dimensional framework. The three-dimensional lattice is simply the

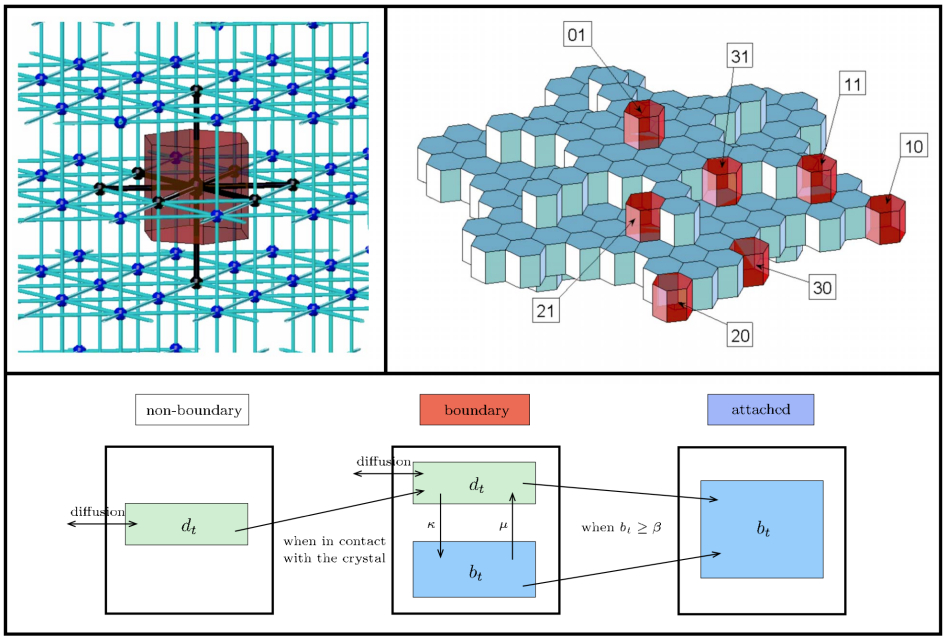


Figure 2.12: An overview of the 3D Gravner-Griffeath algorithm. The top left displays a visualization of the 3D hexagonal lattice $\mathbb{T} \times \mathbb{Z}$. The neighbors of the central site are displayed in black. The top right displays a small crystal with several boundary sites in red. The two numbers displayed on each boundary site give its boundary configuration, derived from its horizontal and vertical neighbor count, respectively. The bottom panel shows a flowchart for the algorithm, displaying the mass-flow under different conditions [14].

two-dimensional hexagonal lattice expanded by a new height dimension, as displayed in Figure 2.12. The expansion to three dimensions necessitates an increase in parameters, with the freezing parameter κ, attachment parameter β, and melting parameter μ each evolving into seven distinct parameters, determined by the count of neighboring cells that are attached.

The authors note that, with 3 spatial dimensions, this algorithm is constrained by computational efficiency. To address this, they have investigated leveraging the inherent symmetries in snow crystal structures. These include a 6-fold rotational symmetry, reflectional symmetry across each of the six branches, and a reflectional symmetry around the xy plane, along the $\mathbb{Z}$ axis. However, they note that this last symmetry is feasible only in the absence of drift. Consequently, the authors have chosen to utilize only the first two symmetries, effectively reducing the computational load to just 1/12th of the total space. The authors further reveal that their implementation is executed in C, albeit without parallelization. They acknowledge that parallelizing the simulation represents a significant opportunity for enhancing efficiency and scalability in future iterations of their work.

2.6 Physically Derived Rules

In 2008, Kenneth Libbrecht would publish his paper on simulating faceted crystal growth with cellular automata [32]. Whereas the rules applied in previous snow crystal simulation cellular automata are largely ad-hoc, Kenneth Libbrecht instead presents rules derived from physics. The goal of this is a step towards more physically accurate rules and models, thus "providing a direct connection between the numerical code and the physics of attachment kinetics and diffusion dynamics governing faceted crystal growth". He focuses on the physical

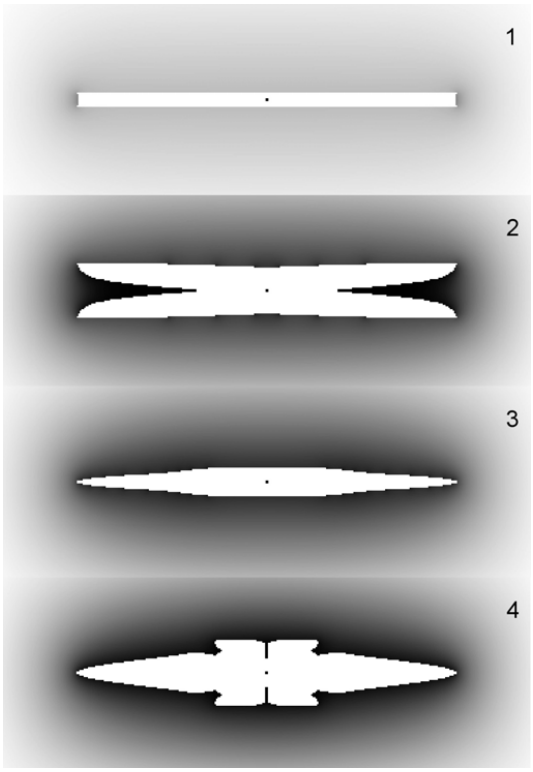


Figure 2.13: Snow crystals simulated in a 2D cylindrical coordinate system. [32]

accuracy of crystal growth velocity, particle transport, and conservation of mass. These are three areas where ad-hoc models such as the work of Janko Gravner and David Griffeath [13, 14] often strayed from physical reality.

His approach also involves assigning physical units to both the spatial and temporal dimensions of the simulation. Whereas before the simulations would be over unitless measurements of space and time, the introduction of Libbrecht's physically derived rules would allow accurate predictions on absolute physical properties such as crystal size, mass and growth speed. Kenneth Libbrecht presents derivations for these rules for the problems of 1D crystal simulation, 2D hexagonal coordinate crystal simulation (similar to the Gravner-Griffeath 2D model), 2D cylindrical coordinate crystal simulation, which simulates the side-view cross-section of the snowflake to model as shown in Figure 2.13, and finally a 3D hexagonal coordinate crystal simulation, much like the Gravner-Griffeath 3D model. For the paper's further analyses, the author implements and uses the 2D cylindrical coordinate simulator.

2.7 Physical Improvements

In their 2013 paper [25], James G. Kelly and Everett C. Boyer iterate on the Gravner-Griffeath 3D model, in an attempt to bring it closer to physical realism. Using the derivations and suggested improvements proposed in Libbrecht's 2008 paper [32], the authors address physical flaws and shortcomings they have identified in the Gravner-Griffeath model. These shortcomings include the supersaturation draining to nearly zero in boundary cells, the speed of the crystal growth not being physically motivated, the violation of conservation of mass, and the high number of hyperparameters.

The algorithm itself is an iteration of the Gravner-Griffeath work, and therefore shares

many similarities. It uses the same three-dimensional hexagonal lattice, and cycles through roughly the same substeps of diffusion, freezing, attachment, and melting, albeit with alterations. The Gravner-Griffeath model had a crystal growth rate that was too high for physical realism. Attempting to resolve this flaw introduces new challenges for computational tractability. Slowing crystal growth to a physically realistic degree implies more simulation time steps are required. Because the rate of crystal growth is very low compared to the rate of diffusion, the authors accelerate the simulation somewhat by decoupling the diffusion substep from the rest of the growth process. Their model repeatedly iterates on a diffusion rule until equilibrium is reached, before performing a single cycle of the remaining growth substeps.

An important detail of a simulation is its boundary conditions. To simulate the abundant availability of water vapor in the snow crystal's far environment, the Gravner-Griffeath work pins the supersaturation at the boundary of the simulation to a constant. However, if the snow crystal grows too close to the simulation boundary, this can lead to an artificially fast growth, leading to crystal distortion in the form of enlarged branch tips. Therefore, it is necessary for the simulation environment to be large enough such that the simulation boundary has enough distance to the crystal. However, increasing the size of the simulation grid is computationally expensive ($O(n^d)$ where n is the number of cells per spatial dimension and d is the number of spatial dimensions). A technique the authors use to create distance between the simulation boundary and the crystal without significant impact to simulation efficiency is the use of two grid resolutions. The snow crystal is grown on a full-resolution "fine" grid, which is surrounded by a large grid of low-resolution "coarse" grid cells. This means that diffusion from the simulation boundary to the fine grid can be approximated at a far lower computational cost, while the snow crystal is still grown at full resolution.

Kelly and Boyer's work represents a considerable advancement in simulating snow crystal growth with enhanced physical realism. Merging Libbrecht's physically-based rules with the Gravner-Griffeath framework, their model establishes a new benchmark for the physical accuracy of snow crystal simulations. This approach not only achieves visually realistic morphologies but also ensures the growth properties are physically authentic. Consequently, this work takes a significant leap in improving our scientific understanding and reasoning of snow crystal growth derived from these simulations.

2.8 GPU Programming

Optimizing a program through parallelization is a key strategy for improving computational speed. When computations are executed in parallel rather than sequentially, the execution time of a program generally decreases inversely and linearly with the increase in parallel compute units, assuming the task can be effectively distributed across these units. In the context of cellular automata, their inherent structure lends itself remarkably well to this form of parallel processing. Each cell in a cellular automaton operates independently based on a set of local rules, making it a prime candidate for parallel computation. As such, each cell's state can be computed simultaneously, significantly reducing the time required for each simulation step. This feature becomes particularly advantageous when simulating large cellular automata systems, where the computational load is substantial. GPU computing, with its array of parallel processing cores, is ideally suited for handling such tasks, offering a substantial improvement in performance compared to traditional CPU-based sequential processing.

In this context, our focus shifts to CUDA (Compute Unified Device Architecture), a parallel computing platform and programming model developed by NVIDIA. CUDA enables direct access to the virtual instruction set and memory of the parallel computational elements in GPUs. We choose CUDA over other parallel computing platforms for its widespread adoption in industry, which is a good sign of robustness and compatibility for a computation

framework. Furthermore, its compatibility with NVIDIA GPUs, which are widely used in scientific computing, stands out as a key feature, and CUDA's well-established development environment and comprehensive documentation provide a robust framework for implementing parallel algorithms.

2.8.1 Numba CUDA

Despite the advantages of CUDA, one significant hurdle is its steep learning curve, especially due to its programming being rooted in C syntax. To address this challenge and make our development process more efficient, we turn to Numba, an open-source JIT (Just-In-Time) compiler that translates a subset of Python and NumPy code into fast machine code. Numba is particularly advantageous for our purposes because it extends its capabilities to GPU programming by providing support for writing CUDA kernels in Python. This feature of Numba allows for the creation of CUDA kernels using familiar Python syntax, which are then JIT compiled and can be executed on NVIDIA GPUs.

This approach significantly streamlines the development process, as it enables us to leverage the powerful capabilities of CUDA while remaining within the Python ecosystem, which is widely adopted in the scientific community for its accessibility and extensive libraries. The use of Numba for CUDA programming not only simplifies the coding process but also maintains high performance, as the JIT compiled Python functions can execute at speeds comparable to traditional CUDA implementations. By utilizing Numba's CUDA functionality, we can efficiently parallelize our cellular automaton computations on GPUs, thus harnessing the full potential of parallel processing without the overhead of mastering a new programming language. This methodology greatly enhances our ability to conduct large-scale, high-speed simulations, which are crucial for the advanced analysis and modeling tasks in our study.

Chapter 3

Background and Related Work on Simulation with Machine Learning

Recently, scaling numerical simulations with neural networks has seen a surge in interest from the scientific machine learning community, with successful applications in climate science [41, 10, 5], nuclear fusion [45, 11], and fluid dynamics [15, 29]. Similarly, various approaches have been proposed in the context of crystal growth modeling. One significant branch of research focuses on Molecular Dynamics (MD) simulation of crystallization [57, 22]. While such MD methods can impressively scale to simulations of up to 1 million atoms, they are still operating at the microscopic scale, and a simulation at the mesoscopic scale would require many orders of magnitude more atoms to be simulated. Alternatively, Ren et al. [46], Nomoto et al. [42] and Xue et al. [53] accelerate phase field simulations directly at the mesoscopic scale. However, in this work we consider a stochastic crystallization process. As these methods are deterministic, they do not apply to our setting. More broadly, probabilistic neural simulation methods have been proposed, but these do not straightforwardly apply to the discrete crystal growth problem [8, 54, 3], or fail to reliably capture the sensitive and stochastic relationship between environmental parameters and crystal morphology [38]. Although these prior works provide interesting and relevant methods for crystal growth or probabilistic neural simulation in general, no method exists for efficient probabilistic crystal growth simulation at the mesoscopic scale. Furthermore, despite the considerable effort that has been dedicated to the numerical simulation of snow crystals [2, 4, 12, 13, 14, 21, 23, 24, 25, 30, 31, 32, 33, 34, 35, 40, 43, 55, 56], no attempt of any kind has been made for creating a neural surrogate in this area. In this work, we introduce the Crystal Growth Neural Emulator (CGNE), a neural surrogate model that greatly accelerates the simulation of mesoscopic scale crystallization processes. We validate this model on the stochastic process of snow crystal growth.

3.1 Background

The following section contains theoretical background on latent variable models, latent variable models in simulation, and the difficulties of training latent variable models. This material is highly relevant to the later sections of the thesis.

3.1.1 Autoregressive Models

In traditional statistics, autoregressive (AR) models are a class of statistical models used to describe certain types of sequential data. In an AR model, future observations are represented as a weighted sum of past observations, asserting that past values have a systematic effect

on current and future values. This approach is particularly suited to time series data where one naturally assumes that past events influence future events.

Formally, an autoregressive model of order p, denoted as $AR(p)$, is defined by the following equation:

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t, \tag{3.1}$$

where x_t represents the current observation, c is a constant, ϕ_i are the parameters of the model corresponding to the i-th lag, p signifies the number of lagged observations included in the model (the order of the model), and ϵ_t is white noise, representing random fluctuations or changes that are not accounted for by the lagged observations.

In the specific case where the model is only conditioned on the immediate previous observation, the model is known as a first-order Markov autoregressive model, or $AR(1)$, formalized as:

$$x_t = c + \phi x_{t-1} + \epsilon_t \tag{3.2}$$

In the context of machine learning, AR models have been extended beyond linear frameworks to include nonlinear relationships between past and present observations. These are often captured by neural networks, which can capture complex patterns, and even stochasticity in the data. The machine learning equivalent of an $AR(1)$ model can be represented as:

$$x_t = f_\theta(x_{t-1}) \tag{3.3}$$

Where f is a neural network parametrized by θ.

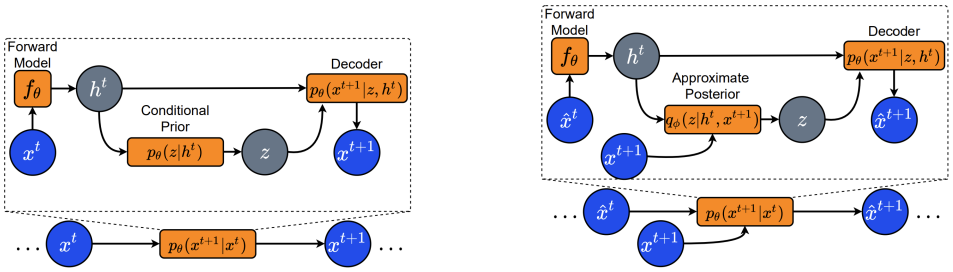
Autoregressive neural networks are widely adopted for their capabilities in modeling sequential data. They are especially useful in simulation settings. By using the current state of the system x_t as both the input and output of the model, they effectively become a surrogate for the underlying simulation dynamics. By using the same framework of iteratively updating the simulation state in discrete steps as used by classical numerical simulators, an autoregressive neural network is then able to simulate the evolution of the system over time. These neural surrogates are particularly useful in scenarios where traditional simulation methods are computationally expensive or infeasible.

In the modeling of snow crystal growth, a simulator based on an autoregressive neural network can offer an efficient approximate solution by reducing the number of temporal resolution steps needed compared to conventional simulation techniques. Traditional methods, such as the work of Janko Gravner and David Griffeath [13], might necessitate tens of thousands of steps to produce an accurate solution. However, an understanding of the growth process might be achievable with just a sequence of 100 steps. By training an autoregressive neural network $x_{t+\Delta t} = f_\theta(x_t)$ with an increased step size Δt, it is possible to significantly decrease computational time while retaining a resolution that captures the essential details of the crystal growth.

3.1.2 The Variational Autoencoder

In their pivotal 2013 work, Kingma et al. [26] significantly advanced generative modeling with the Variational Autoencoder (VAE), an evolution of the traditional autoencoder framework [1]. The VAE combines deep learning with probabilistic graphical models to efficiently learn complex data distributions.

Central to the VAE's architecture is its encoder-decoder structure. The encoder, parameterized by ϕ, transforms input data x into a probabilistic latent representation with mean μ and variance σ^2, also known as the approximate posterior $q_\phi(z|x)$. The decoder, parameterized by θ, reconstructs the input from this latent space. Unlike conventional autoencoders, the VAE's latent space is regularized to a standard Gaussian prior, not only



(a) Sampling. (b) Training.

Figure 3.1: PNS model overview [38].

ensuring continuity and structure, but also making the VAE a generative model, as it allows sampling from the latent space to generate new data instances.

This regularization is encapsulated in the model's loss function, the Evidence Lower Bound (ELBO):

$$\text{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x) \| p(z))$$

The ELBO consists of the reconstruction loss and the Kullback-Leibler (KL) divergence. The KL term serves as a regularizer, aligning the approximate posterior $q_\phi(z|x)$, being the output of the encoder, with a standard Gaussian prior $p(z)$. The adverse relation between the reconstruction term and the regularizing KL term ensures meaningful latent representations throughout all regions of the latent space within the Gaussian prior.

Maximizing the ELBO enables the VAE to learn intricate data distributions, balancing precise reconstruction with regularized latent representations to enable generative modeling. This innovation has been influential in unsupervised learning and generative model research, providing a robust framework for diverse applications.

3.1.3 PNS

In their 2023 study, Minartz et al. [38] propose an autoregressive latent variable model framework tailored for simulation with neural networks. Their framework, named Equivariant Probabilistic Neural Simulation (EPNS), is specifically designed for autoregressive probabilistic modeling of dynamical systems with equivariant distributions over trajectories. Although equivariance does not pertain to our research, their non-equivariant framework for Probabilistic Neural Simulation (PNS) is highly relevant to our work.

Central to their framework is the joint training of an approximate posterior $q_\phi(z|x_t, x_{t+1})$ alongside a conditional prior $p_\theta(z|x_t)$. This is trained within what is essentially a Conditional Variational Autoencoder (CVAE) [50] setup over x_{t+1} as follows:

$$x_{t+1} \sim p_\theta(x_{t+1}|z, x_t) \quad \text{and} \quad z \sim q_\phi(z|x_t, x_{t+1}). \tag{3.4}$$

where $q_\phi(z|x_t, x_{t+1})$ encodes x_{t+1} to z, and $p_\theta(x_{t+1}|z, x_t)$ decodes z to x_{t+1}. A KL divergence term in the loss function minimizes the distance between the prior and posterior. This setup essentially trains the prior to encode a distribution over possible future states x_{t+1} given the current state x_t. Meanwhile, the posterior, used only during training, is informed on the chosen future state to ensure accurate reconstructions for the current training trajectory. The full model setup, as taken from the paper, is visible in Figure 3.1 (albeit with an additional forward model f_θ and hidden state h^t that are not relevant for the purpose of this explanation).

Also key to their model's success is the use of pushforward training, discussed in section 3.1.4.

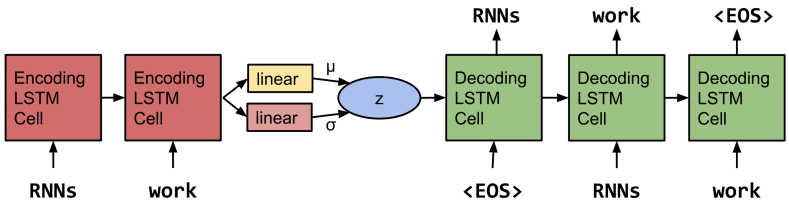


Figure 3.2: Model overview of the VAE language model from [6].

3.1.4 Pushforward Training

In their 2022 publication, Brandstetter et al. [7] present an autoregressive neural PDE solver. With a naive one-step training approach, the authors identify the main failure mode of the model to be instability, which stems from the accumulation of errors inherent to the autoregressive modeling scheme. They attribute this instability to overfitting on the ground truth during input. To mitigate this issue, Brandstetter et al. propose the *pushforward trick*. This method involves the model using its own output as input for subsequent steps during training, rather than the ground truth. By incorporating this method, the model is trained with a degree of error in its inputs, better mirroring actual deployment scenarios. Pushforward training enables the model to generate accurate outputs even when the input contains some degree of error, leading to more stable rollouts.

3.1.5 Generating Sentences from a Continuous Space

In their pioneering work, Bowman et al. [6] innovatively applied a conditional autoregressive latent variable model for the generation of text from a continuous latent space. Their model architecture is based on the variational autoencoder [26], employing a non-deterministic encoder-decoder architecture, with an additional conditioning signal on the decoder network. By harnessing the power of variational autoencoders, the authors are able to model linguistically rich and varied sentences. Both the encoder and decoder of their model are built upon Long Short-Term Memory (LSTM) networks [17], a choice driven by LSTMs' proficiency in handling sequential data such as text. The autoregressive part of the model lies in the model's decoder network. Each output step of the LSTM decoder is fed back as a conditioning input for the next step. A visual overview is provided in Figure 3.2.

The authors discuss difficulties with training this model. The difficulties of training this type of model is a recurring and key topic throughout this thesis, and the occurrence of these training challenges in works other than ours highlights their prevalence.

3.1.5.1 Posterior Collapse

One issue that the authors run into is the KL divergence loss term dominating the data reconstruction loss term in early stages of training, leading to the model getting stuck in an undesirable local minimum during optimization. In the early stages of training, the likelihood term $\log p(x|z)$ is relatively weak, such that an initially attractive state is where the posterior copies the gaussian prior $q(z|x) \approx p(z)$, bringing the KL divergence term of the loss to zero. This state is referred to as posterior collapse. In this state, encoder gradients have a relatively low signal-to-noise ratio, resulting in a stable equilibrium from which it is difficult to escape. Since the encoder fails to encode any information about the input, the model ignores the latent variable, resulting in model outputs independent of the encoder input, and defeating the purpose of a latent variable model.

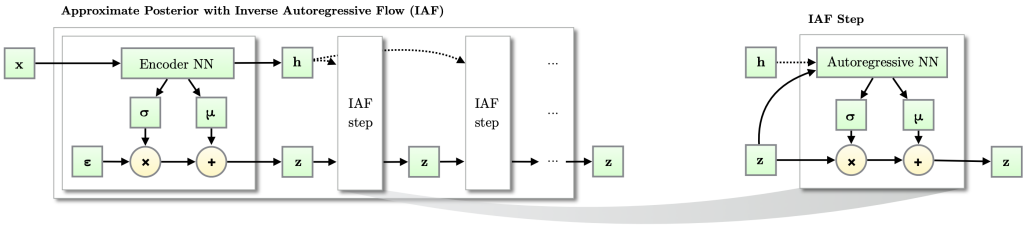


Figure 3.3: Inverse Autoregressive Flow approximate posterior architecture [27].

3.1.5.2 KL Annealing

Toward this issue, the authors propose the concept of KL annealing. This simple technique adds weight to the KL term of the loss function that is varied over time. At the start of training, this weight is set to zero to allow the model to encode as much information in z as it can. Then, over time, this weight is gradually increased to 1, reintroducing the KL term of the loss function, which forces the model to fit the encodings to z into the prior. At the time the annealing weight reaches 1, the signal-to-noise ratio of the encoder gradients is higher, which makes it less likely for the model to not fall into posterior collapse again. Sønderby et al. encounter a similar problem in their 2016 work [51], and propose an equivalent solution under the name of *warm-up*.

3.1.5.3 Word Dropout

Another issue the authors encountered during training was the decoder (partly) ignoring the latent variable in favor of the conditioning signal. The issue lies in the fact that the output can be predicted to a reasonable accuracy by use of the conditioning signal alone. This issue fits perfectly in **RQ2**, as described in section 1.2.2.

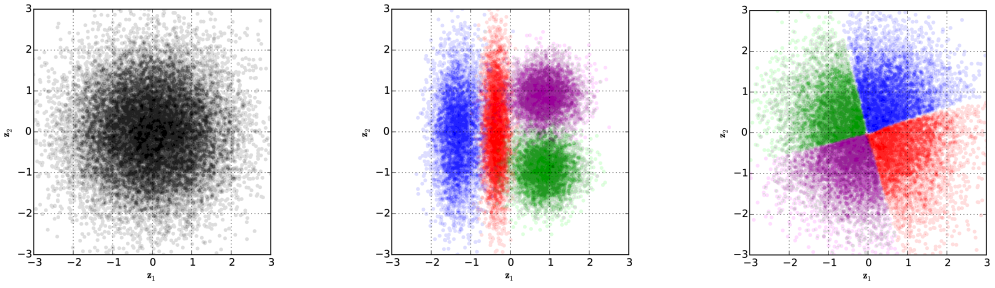
To combat this issue, the authors employ a dropout term on the conditoning signal, randomly replacing some fraction of the conditioned-on word tokens with a generic 'unknown' token. This technique forces the model to rely on the latent variable to make good predictions. In practice, a 0.0 dropout rate causes the model to fall into posterior collapse, with a KL divergence of zero, a relatively high reconstruction loss, and outputs that qualitatively lack diversity. A 0.25 dropout rate gives the authors optimal results with qualitatively diverse and realistic sentences, a non-zero KL term, and a significant improvement in the reconstruction loss term.

3.1.6 Improved Variational Inference with Inverse Autoregressive Flow

In 2016, Kingma et al. would propose Inverse Autoregressive Flow (IAF), a novel way of modeling the posterior distribution in the variational autoencoder framework. Building off their earlier work on the VAE [26], and Rezende and Mohamed's work on Normalizing Flows [47], their method involves a posterior that encodes the latent variable z by first sampling an initial z, followed by a chain of nonlinear invertible transformations of z, sampled from an autoregressive model. The process, displayed in Figure 3.3, enhances the alignment of the approximate posterior with respect to the Gaussian prior, a concept further exemplified in Figure 3.4.

3.1.6.1 Free Bits

Much like in the work of Bowman et al. [6], the authors run into the issue of posterior collapse as discussed in section 3.1.5.1. However, instead of using KL annealing to overcome this issue, they propose a new method that involves a modified optimization with *free bits*.



(a) Prior distribution. (b) Posteriors in standard VAE. (c) Posteriors in VAE with IAF.

Figure 3.4: Posterior distributions of standard VAE and VAE with IAF visualised [27].

The method prevents the KL term from going to zero during training by setting a lower bound λ in the objective for every dimension $d \in D_z$ of the latent space z. This ensures that using less than λ nats of information per dimension d (on average per minibatch $\mathcal{M}$) is not advantageous. Formally, the modified objective is:

$$\tilde{\mathcal{L}}_\lambda = \mathbb{E}_{x\sim\mathcal{M}}\left[\mathbb{E}_{q(z|x)}[\log p(x|z)]\right] - \sum_{d=1}^{D_z} \text{maximum}(\lambda, \mathbb{E}_{x\sim\mathcal{M}}[\text{KL}(q(z_d|x)||p(z_d))]) \tag{3.5}$$

Since increasing the latent information is generally advantageous for the (unaffected) reconstruction term of the objective this results in $\mathbb{E}_{x\sim\mathcal{M}}[\text{KL}(q(z_d|x)||p(z_d))] \geq \lambda$ for all d, in practice.

Chapter 4

First Principles Simulator

A significant contribution of this thesis is the development of a first principles simulator for snow crystal growth, designed specifically to create training data in the form of image sequences describing the growth process. The absence of real-world sequential image datasets on snow crystal growth necessitates the use of synthetic data for training models in this domain. By providing this simulator as open-source software, we facilitate the generation of synthetic datasets on snow crystal growth, thus enabling advances in machine learning research focused on simulating snow crystal growth.

4.1 Computation on Hexagonal Grids

We implement the 2D Gravner-Griffeath algorithm from section 2.4, a cellular automaton model that operates on a hexagonal grid. When working with hexagonal grids, one decision point of the implementation is choosing how to represent and store the hexagonal grid. An n-dimensional array is an efficient data structure for its efficient read/write operations and the non-sparse nature of the simulation state. To accurately map and reference locations on the hexagonal grid to locations in the storage array, a coordinate system is essential. Various approaches to a hexagonal grid coordinate system exist, each with their own advantages and drawbacks. The main two candidates for storing a hexagonal grid in a two-dimensional array are the offset, and the axial coordinate systems.

4.1.1 Axial Coordinate System

Also known as the "oblique" or "skewed" coordinate system, the axial system maps the hexagonal grid to a square grid through a skewing transformation. Visually and intuitively explained in Figure 4.1, this coordinate system has two axes, and is therefore easily stored in a two-dimensional array. One drawback of the axial coordinate system is the space efficiency when attempting to store computation spaces stored as rectangles. As displayed in Figure 4.2, skewing the rectangle to a paralellogram results in a loss of space when this paralellogram is stored in a rectangular array. However, for storing hexagonally shaped grids, displayed in Figure 4.3, the space efficiency is equal between the axial and offset approaches.

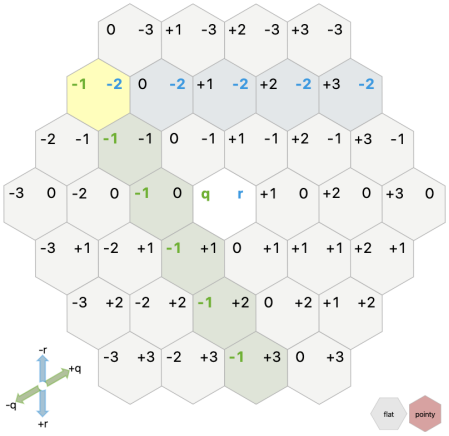


Figure 4.1: The axial coordinate system for hexagonal grids. The two coordinate axes are highlighted in blue and green.

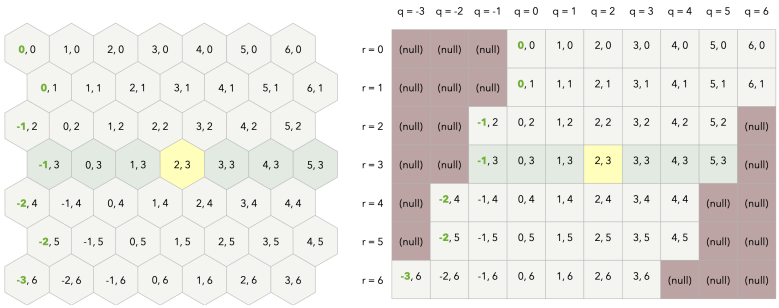


Figure 4.2: A rectangular simulation space stored in a 2D array using an axial coordinate system.

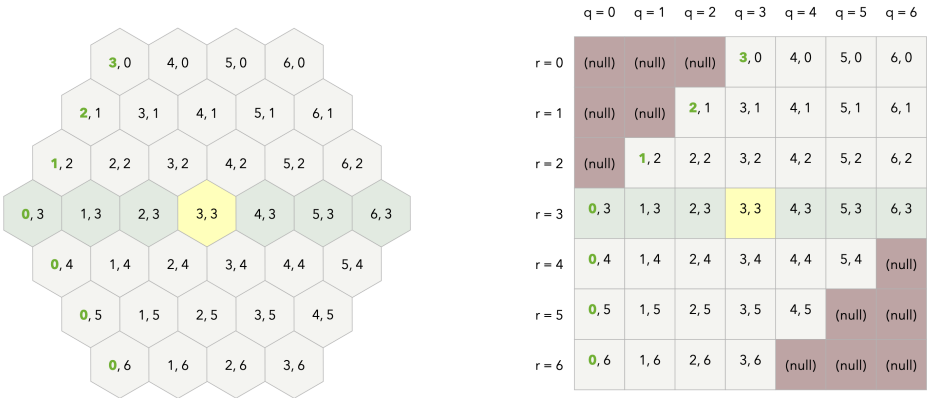


Figure 4.3: A hexagonal simulation space stored in a 2D array using an axial coordinate system.

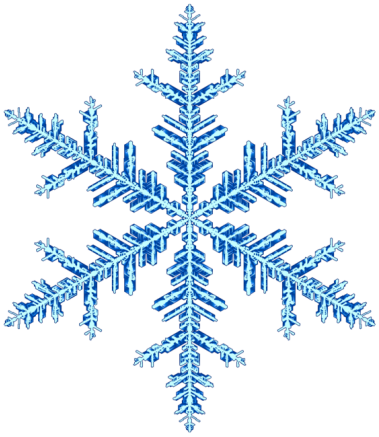


Figure 4.4: A simulated snowflake on a hexagonal grid.

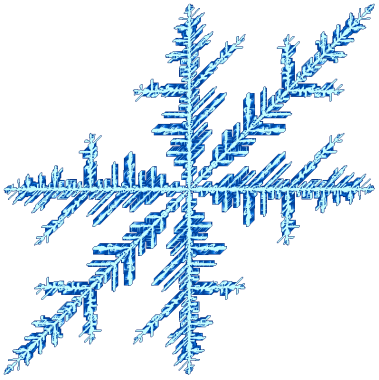


Figure 4.5: A snowflake skewed to a Cartesian grid with the axial coordinate system.

4.1.2 Offset Coordinate System

The offset coordinate system works by offsetting every other row or column. This approach means that much less space is wasted when storing a rectangular computation space. The drawback of this approach is the added complexity in mathematical operations. While linear algebra is possible on the axial coordinate system, as it is simply a linear transformation of the Cartesian coordinate system, this is not possible on the offset coordinate system. This complexity is evident in operations like calculating a cell's neighborhood, an important operation in the context of cellular automata. Whereas the neighborhood kernel of an axial coordinate system is consistent, an offset coordinate system requires two different neighborhood kernels for even and odd rows or columns, as shown Figure 4.9. Consequently, the axial system is generally simpler to work with than the offset system. Because of this, we opt for an axial coordinate system to implement the simulator.

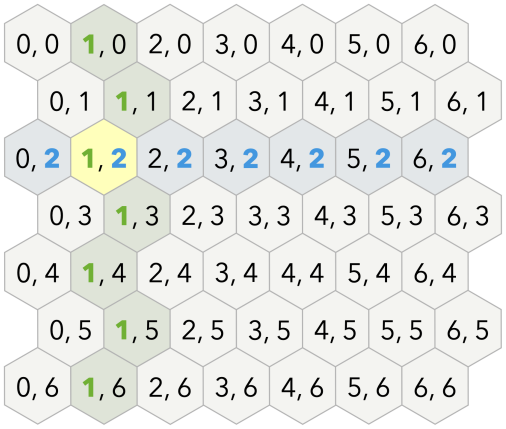


Figure 4.6: An offset coordinate system with the "odd-r" horizontal layout, which offsets the odd rows. Different offset coordinate systems are possible with choices on whether to shove even or odd rows or columns.

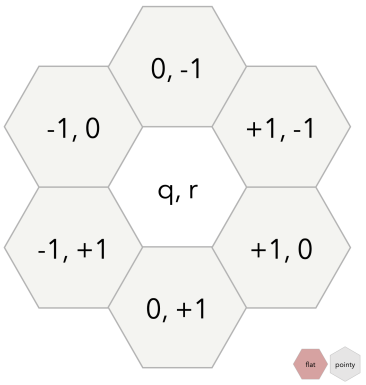


Figure 4.7: Neighborhood kernel for an axial coordinate system on a hexagonal grid.

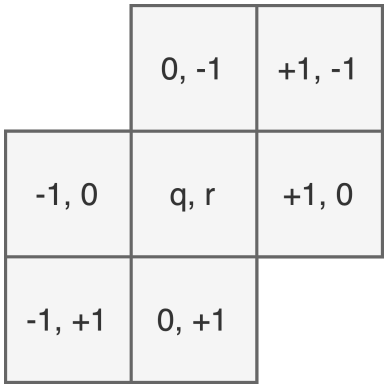


Figure 4.8: Neighborhood kernel for an axial coordinate system represented on a Cartesian grid.

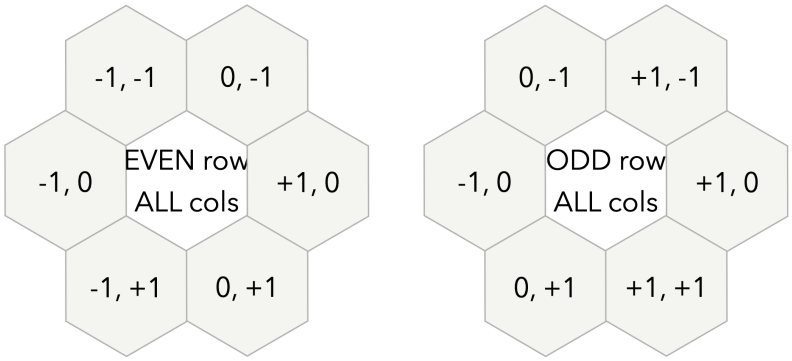


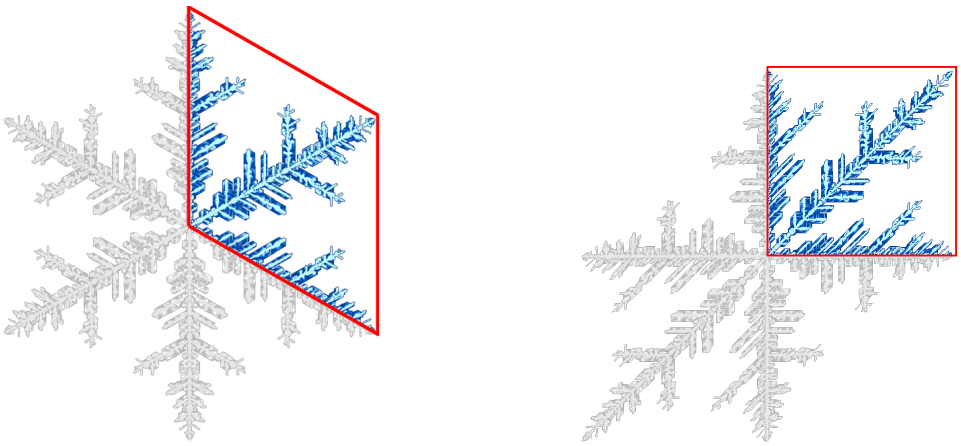
Figure 4.9: Relative coordinates of neighbor cells in an offset coordinate system vary between even and odd rows.

4.2 Computational Efficiency and Scalability

An important factor of this implementation of the Gravner-Griffeath algorithm over other existing implementations is its computational efficiency and scalability. The most prominent of these speed efficiency improvements comes from GPU acceleration. By parallelizing the cellular automaton model across the numerous cores of the GPU, we achieve a 350000× speed improvement over a simple Python CPU implementation, and 1000× over a JIT compiled version. This acceleration is facilitated by Numba CUDA, a library that seamlessly integrates CUDA computation into the Python language, as further explained in section 2.8.1.

To further improve computational efficiency we optionally utilize the reflective and six-fold rotational symmetry of snowflakes by simulating only a fraction of the crystal. Specifically, we simulate only two whole branches, or a third of the snowflake, which corresponds to a single quadrant when projected onto a Cartesian grid, as displayed in Figure 4.10b. This method quadruples simulation speed, while fully preserving the integrity of the simulated branches. Furthermore, the generated data still captures a complete, standalone branch in a square image format, ideal for convolutional neural networks.

On the implementation level, we add a reflectional symmetry operation consistent with the hexagonal grid along the two simulation edges where the crystal meets the boundary



(a) Hexagonal representation.

(b) Cartesian representation.

Figure 4.10: For simulation speed improvements, we optionally only simulate a third of the snowflake, which amounts to a quarter of the Cartesian representation.

of the simulation space. Figure 4.11 illustrates the mechanics of reflection on a hexagonal grid for the relevant axes. Moreover, when this is translated onto a Cartesian grid, as shown in Figure 4.12, it becomes apparent that hexagonal reflections work differently from regular Cartesian reflections. This figure demonstrates how the hexagonal reflection kernel is implemented on the square grid.

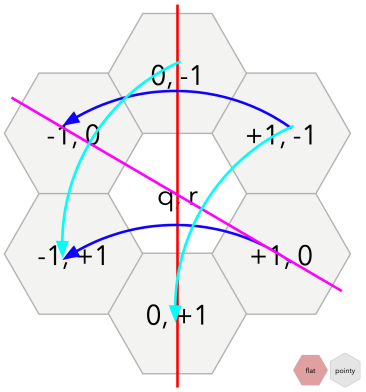


Figure 4.11: The two axes of reflection on a hexagonal grid.

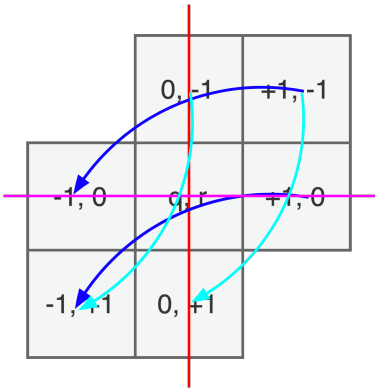


Figure 4.12: The two axes of reflection on a Cartesian grid.

4.3 Simulation Conditions

To effectively tailor this simulator toward the generation of datasets, we need a method to configure thousands of different environmental parameter combinations for the simulator to generate its snowflakes. Manually setting each parameter for every snowflake isn't practical. Instead, we use a random uniform distribution to sample parameters within the ranges provided by Janko Gravner and David Griffeath in their source code [13], as shown in Table 4.1.

	min	max
α	0	0.3
β	1.05	2.2
θ	0.02	0.57
κ	0	0.05
γ	0.0000515	0.0001
μ	0	0.01
ρ	0.3	0.9

Table 4.1: Suggested simulation parameter ranges for the Gravner-Griffeath 2D algorithm for snow crystal growth.

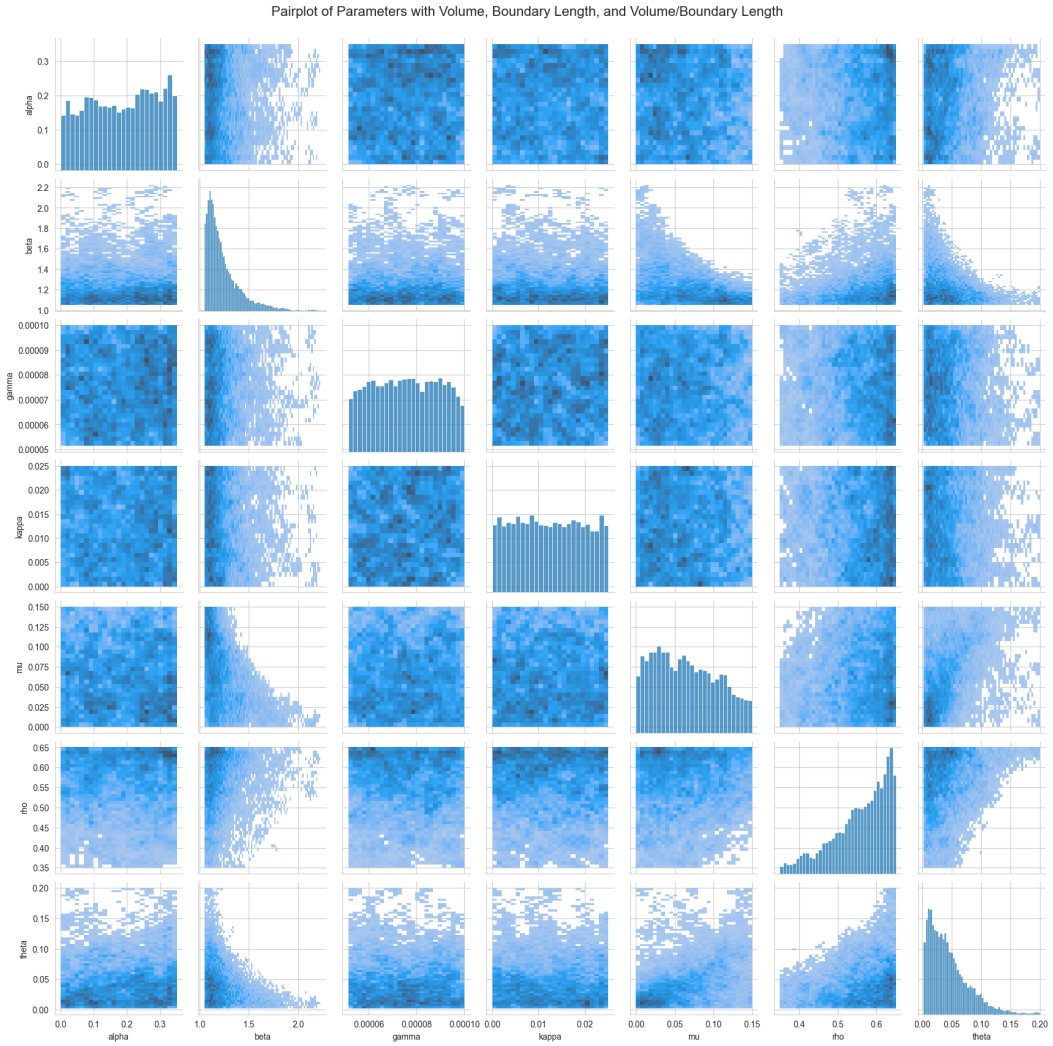


Figure 4.13: A histogram pairplot displaying how the parameters of Krzywinski's snowflakes are distributed.

However, as discovered by Martin Krzywinski and Jake Lever in their 2017 article [37], a uniform distribution in these ranges might not be ideal for achieving an even distribution of diverse and intricate snowflake morphologies. Some parameter combinations result in no growth, while others might result in excessive growth that simply forms a uniform block

of ice. Furthermore, some parameters are very sensitive in a particular range, and then much less sensitive in other ranges. These discoveries are done through extensive empirical trial and error, and they lead them to the construction of a diverse catalogue of roughly 15,000 snowflakes [28], the parameter distribution of which is displayed in 4.13. We include a sampling method based on Krzywinski's approach, which involves randomly picking one of Krzywinski's snowflakes, and then using a slightly perturbed version of its parameters.

4.3.1 Other Implementation Details

If we aim to model this simulator using a Markov model, we must ensure that subsequent states are distinct. This is crucial because the Markov property relies on the assumption that the next state is dependent only on the current state. Identical subsequent states indicate no change or transition, failing to capture the dynamics of the system. Since our data-driven simulator focuses solely on the attachment field, we maintain this property by saving a simulation state only when the attachment field changes. In other words, we record only those steps where the crystal grows by at least one cell. This approach ensures that each saved state reflects a meaningful transition in the system.

Furthermore, to properly capture the stochasticity of the snow crystal growth process in the dataset, we implement an option to repeat simulation parameter sets while generating the dataset. This approach allows us to include multiple random simulations from the same starting conditions in the dataset.

For further implementation details, we refer to appendix A.

Chapter 5

Problem Formulation and Solution Approach: Convergence to the Identity Function

In this chapter, we delve into the first research question:

RQ1 How can we best model systems characterized by very small state changes using first-order Markov autoregressive models, while avoiding convergence to the simplistic identity function?

5.1 Problem Formulation: Modeling Snow Crystal Growth

Before this, however, we specify the general problem formulation of modeling snow crystal growth. The first-principles simulator models the dynamical system of snow crystal growth. This dynamical system evolves the simulation state over time in discrete time steps based only on the current state, and is therefore Markovian.

We consider only the evolution of the attachment field. Formally, at time t the system is in state (x_t, y), where x_t is a binary function on a fixed grid and y is a real-valued vector of static environmental parameters. The system evolution is specified with a sequence of states $(x_{0:T}, y)$. Our goal is to fit a simulation model p_θ to the data by maximizing the log-likelihood over the training set $\{(x_{0:T}, y)_i\}_{i=1}^N$:

$$\frac{1}{N}\sum_{i=1}^{N} \log p\left((x_{0:T}, y)_i\right), \tag{5.1}$$

where the training set is generated from the ground-truth simulator. We choose to reformulate the problem into an autoregressive problem. We have:

$$\begin{aligned}
\log p\left(x_{0:T}, y\right) &= \log p\left(x_{0:T} \mid y\right) + \log p(y) \\
&= \log p\left(x_0, x_1, \ldots, x_{T-1}, x_T \mid y\right) + \log p(y) \\
&= \sum_{t=0}^{T-1} \log p\left(x_{t+1} \mid x_t, y\right) + \log p\left(x_0 \mid y\right) + \log p(y)
\end{aligned}$$

Since $p(x_0 \mid y)$ and $p(y)$ are known, maximizing Equation 5.1 over the training set is equivalent to maximizing the one-step autoregressive log-likelihood with respect to θ:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\log p\left(\left(x_{t+1}\mid x_t, y\right)_i\right). \tag{5.2}$$

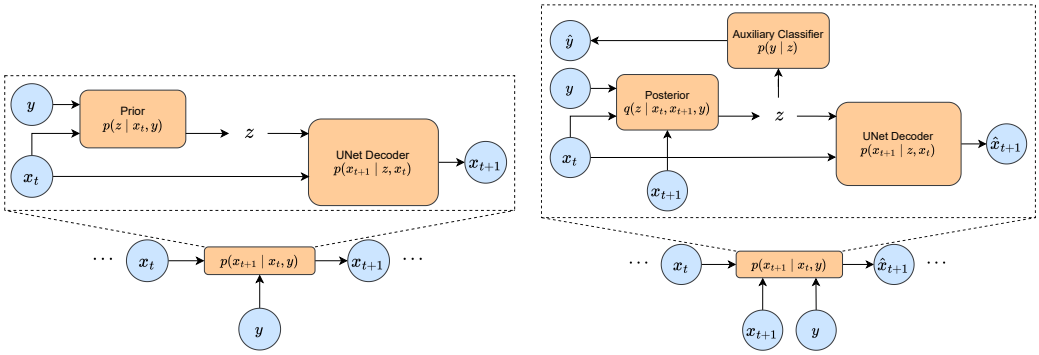
Since LCA models are autoregressive, this reformulation more closely aligns with the dynamics of the ground-truth simulator. After fitting a model $p_\theta\left(x_{t+1}\mid x_t, y\right)$ to maximize Equation 5.2 with respect to θ, we can generate new snow crystal growth sequences by applying the model autoregressively.

5.2 Approach

5.2.1 Model Design

The goal is to develop a probabilistic model that autoregressively produces realistic snow crystal growth sequences. To this end, we base the model off the PNS architecture by Minartz et al. [38] discussed in section 3.1.3, an autoregressive probabilistic approach for modeling dynamical systems. The framework follows a Conditional Variational Autoencoder (CVAE) [50] architecture where the conditional prior is conditioned on the current simulation state $p\left(z\mid x_t\right)$, and the posterior is additionally conditioned on the future state $q\left(z\mid x_t, x_{t+1}\right)$. A decoder $p\left(x_{t+1}\mid z, x_t\right)$ conditioned on the latent variable and the current state of the system then predicts the future state of the system to complete an autoregressive model $p\left(x_{t+1}\mid x_t\right)$.

However, to model snow crystal growth, we require an additional conditioning variable y, to represent the environmental parameters. We therefore reformulate the prior and posterior to additionally be conditioned on the environmental parameters $p\left(z\mid x_t, y\right)$, $q\left(z\mid x_t, x_{t+1}, y\right)$, which changes the model to $p\left(x_{t+1}\mid x_t, y\right)$. The full structure is displayed in Figure 5.1



(a) Sample pass of the model.

(b) Training pass of the model.

Figure 5.1: Model overview.

To preserve local translation symmetries in the data domain of fixed-grid simulation states (images), we design the model to be fully convolutional. This design is realized through implementing a spatial latent space and utilizing a UNet architecture [48] for the model decoder. The latent variable is incorporated into the UNet decoder at each layer via concatenation along the channel dimension, as illustrated in Figure 5.2, where the latent variable is spatially upsampled with a nearest-neighbor interpolation strategy to match the size for each respective layer. The choice of a UNet is particularly advantageous for simulating snow crystal growth, as the growth dynamics are only active at the crystal's boundary. Consequently, most of the image remains unchanged between successive timesteps, making the skip connections in the UNet architecture especially effective for capturing this

behavior. The decoder predicts a probability $\hat{p}$ per grid cell (i, j), which is discretized with binary thresholding:

$$\tilde{x}_{t+1}^{(i,j)} = \begin{cases} 1 & \text{if } \hat{p}_{t+1}^{(i,j)} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

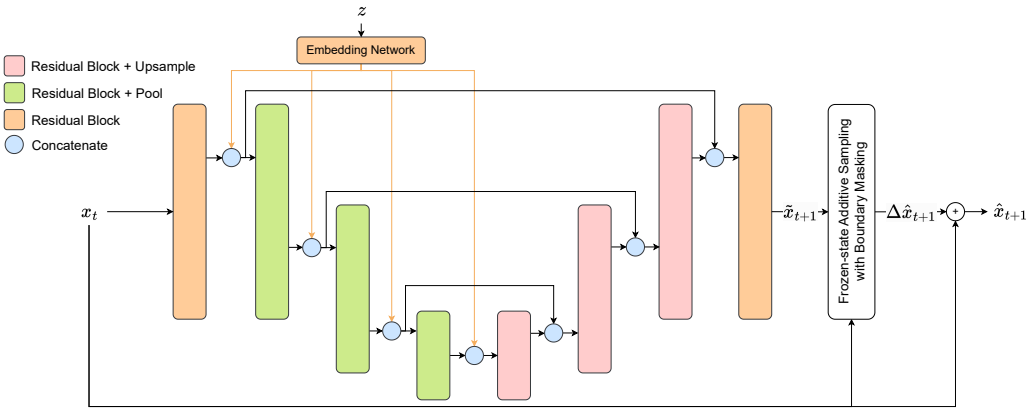


Figure 5.2: Overview of the model's decoder, including the frozen-state additive sampling strategy with the eligible growth area restricted to the crystal boundary.

Additionally, inspired by Ilse et al. [19], we use a small auxiliary classifier $p(y \mid z)$ that reconstructs y from the latent variable during training, improving the representation of the environmental parameters in the latent space.

Finally, we directly incorporate the physical properties of snow crystal growth into the model through several strategies. The first strategy is frozen-state additive sampling, which incorporates the property that crystallization is a unidirectional process, with no melting or sublimation. In this approach, the solidified portion from the previous step x_t is preserved and combined with the model's prediction for only the non-solidified region of the prior step $\Delta\hat{x}_{t+1}$. This enforces that only grid cells within the non-solidified region of x_t are eligible for growth. We define this set of grid cells as the "eligible growth area" $\mathcal{E}$. Formally,

$$\Delta\hat{x}_{t+1}^{(i,j)} = \begin{cases} \tilde{x}_{t+1}^{(i,j)} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\hat{x}_{t+1}^{(i,j)} = \Delta\hat{x}_{t+1}^{(i,j)} + x_t^{(i,j)} = \begin{cases} \tilde{x}_{t+1}^{(i,j)} & \text{if } (i,j) \in \mathcal{E} \\ x_t^{(i,j)} & \text{otherwise.} \end{cases}$$

This strategy ensures a physically accurate unidirectional solidification process.

Secondly, we enforce that growth dynamics only occur on the snow crystal boundary by use of a pixel-wide mask along the snow crystal boundary. The mask is computed using a hexagonal edge detection kernel K,

$$K = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -6 & 1 \\ 0 & 1 & 1 \end{bmatrix},$$

on x_t, and is used to mask off the model's prediction $\Delta\hat{x}_{t+1}$ to limit the eligible growth area $\mathcal{E}$ within the frozen-state additive sampling method. This only allows the model to

classify new solidified areas directly on the crystal boundary. Furthermore, in the generation of the dataset, we also ensure this property of growth dynamics occurring exclusively on the crystal boundary holds by saving the growth steps that advance the attachment field, as explained in section 4.3.1.

Finally, similar to the strategy used by the first-principles model explained in section 4.2, we incorporate the six-fold rotational and reflective symmetry of the snowflake into the model by only simulating one-third of the snowflake, or a quarter of the Cartesian grid representation.

To improve the stability of sampled trajectories in the autoregressive modeling paradigm, we utilize the pushforward training strategy described in section 3.1.4. Furthermore, we employ gradient clipping [44] and stochastic weight averaging [20] to improve training stability.

5.2.2 Convergence to the Identity Function

(a) Ground truth LCA sample. (b) Baseline model sample. (c) Area growth plot of the sampled snowflakes.

Figure 5.3: Preliminary snowflake samples, and a plot displaying their growth over time. Three phases of the snowflakes' growth are displayed in three shades of blue.

However, preliminary results of the model clearly reveal the problem of RQ1: In first-order Markovian autoregressive models, when differences Δx_t between sequential states are marginal, the model may converge to the identity function. This issue expresses itself in our model as a deadlock early in the growth process, where it continuously predicts no growth, leaving the simulation state unchanged. This deadlock is visible in Figure 5.3c, which plots the size of snowflakes over time for both the LCA model and our model. This results in the sampled snow crystals being far smaller than their counterparts sampled from the LCA model, as displayed in Figure 5.3b. We formalize the following problem:

5.2.3 Problem Formulation: RQ1

Given a sequence of states $(x_{0:T}, y)$ with marginal sequential differences $||\Delta x_t|| \approx 0$, the first-order Markov model p_θ trained to maximize the one-step autoregressive log-likelihood described in Equation 5.2 tends to converge to a simplistic identity function where

$$p_\theta(\hat{x}_{t+1} = x_t \mid x_t, y) \approx 1.$$

Our goal is to develop a strategy to mitigate this issue and prevent the model from converging to this failure mode, ensuring it accurately captures the dynamics of a system characterized by small state changes.

5.2.4 RQ1 Strategies

5.2.4.1 Forced Boundary Growth

The issue of modeling the identity function expresses itself in our model by predicting no growth at a certain point in the sequence. Given that the dataset consistently exhibits growth between consecutive steps, predicting no growth at any stage is inherently incorrect. To address this, we introduce the Forced Boundary Growth rule, which ensures that growth always occurs.

This rule operates as follows: Let $\hat{p}_{t+1}^{(i,j)}$ represent the predicted probability of growth for the grid cell at position (i, j). If no grid cells have a growth probability greater than 0.5, the grid cell with the highest growth probability within the eligible growth area $\mathcal{E}$ as defined by the Frozen-state Additive Sampling procedure will be selected to grow. Formally, this is defined as:

$$\Delta\hat{x}_{t+1}^{(i,j)} = \begin{cases} 1 & \text{if } \max_{(i,j)\in\mathcal{E}} \hat{p}_{t+1}^{(i,j)} \leq 0.5 \text{ and } (i,j) = \arg\max_{(i,j)\in\mathcal{E}} \hat{p}_{t+1}^{(i,j)} \\ \tilde{x}_{t+1}^{(i,j)} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

This rule ensures that the model adheres to the inherent growth characteristics of the dataset by always enforcing growth within the eligible growth area.

5.2.4.2 Boundary Reinforcement Loss

Again motivated by trying to prevent the failure mode of the growth probability of all eligible grid cells being below 0.5, we introduce the Boundary Reinforcement Loss strategy. This strategy involves placing a term in the loss that encourages the maximum eligible grid cell probability to remain above a threshold T. Formally, the Boundary Reinforcement Loss can be defined as follows:

$$\mathcal{L}_{\text{BR}} = \max\left(0, T - \max_{(i,j)\in\mathcal{E}} \hat{p}_{t+1}^{(i,j)}\right)$$

where $\mathcal{E}$ denotes the set of grid cells eligible for growth and $\hat{p}_{t+1}^{(i,j)}$ represents the predicted probability of growth for the grid cell at position (i, j). The term $\mathcal{L}_{\text{BR}}$ is added to the overall loss function with weight $W_{\mathcal{L}_{BR}}$ to penalize the model whenever the maximum growth probability within the eligible area $\mathcal{E}$ falls below the threshold. This encourages the model to maintain sufficiently high growth probabilities, thereby preventing stagnation in the growth process.

5.2.4.3 Boundary Count Loss

Like the Boundary Reinforcement Loss strategy, the Boundary Count Loss strategy attempts to prevent the model's failure mode through a loss term. However, this strategy attempts to place a loss term on the number of grid cells the model grows at each step. We calculate the number of growing grid cells $N(t + 1)$ from the ground truth data by summation over the binary Δx_{t+1}:

$$N(t+1) = \sum_{(i,j)\in\mathcal{E}} \Delta x_{t+1}^{(i,j)} = \sum_{(i,j)\in\mathcal{E}} \left(x_{t+1}^{(i,j)} - x_t^{(i,j)}\right)$$

For the model's prediction, a differentiable method of transforming the logits to near binary values is required to calculate $|\Delta\hat{x}_{t+1}|$ in the same manner. For this, we multiply the

logits with a high inverse temperature parameter α before passing them through the logistic sigmoid function to create a differentiable approximation of a step function at 0.5. Formally,

$$\hat{N}(t+1) = \sum_{(i,j)\in\mathcal{E}} \sigma\left(\alpha z_{t+1}^{(i,j)}\right)$$

where σ is the logistic sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, and $z_{t+1}^{(i,j)}$ represents the logits for the grid cell at position (i,j) at time $t+1$.

The Boundary Count Loss is then defined as the L_2 distance between the ground truth and predicted growth counts of growing grid cells:

$$\mathcal{L}_{\text{BC}} = \left(\hat{N}(t+1) - N(t+1)\right)^2$$

This loss term is again added to the overall loss function with weight $W_{\mathcal{L}_{BR}}$ to encourage the model to grow the correct number of grid cells at each step, therefore ensuring that the model's predictions more closely match the growth dynamics expected from the training data.

5.2.4.4 Bernoulli Output Sampling

One key observation is that the problem only arises later in the growth process, once the eligible growth area $\mathcal{E}$, which equals the crystal boundary, grows large enough. Since growth usually occurs at only a handful of pixels at once, as $\mathcal{E}$ grows larger, eventually we run into the scenario where the predicted growth probabilities could add up to be 1.0 or greater, but individually are all below 0.5. Perhaps the model places a growth probability of $\frac{1}{3}$ on three different candidates. In this scenario, binary thresholding predicts no growth, and therefore is not an optimal sampling method. Instead, we experiment with treating the model output as probabilities of a Bernoulli distribution from which we sample to derive the binary output. Specifically, let $\hat{p}_{t+1}^{(i,j)}$ represent the predicted probability of growth for the grid cell at position (i,j) at time $t+1$. We sample the binary output $\Delta\hat{x}_{t+1}^{(i,j)}$ from a Bernoulli distribution parameterized by $\hat{p}_{t+1}^{(i,j)}$:

$$\Delta\hat{x}_{t+1}^{(i,j)} \sim \text{Bernoulli}(\hat{p}_{t+1}^{(i,j)})$$

where:

$$\text{Bernoulli}(\hat{p}_{t+1}^{(i,j)}) = \begin{cases} 1 & \text{with probability } \hat{p}_{t+1}^{(i,j)} \\ 0 & \text{with probability } 1 - \hat{p}_{t+1}^{(i,j)} \end{cases}$$

This approach ensures that the model can predict growth even when individual probabilities are below the threshold of 0.5, thus more accurately reflecting the probabilistic nature of the growth process.

5.3 Experiment Setup

5.3.1 Dataset

We conducted the experiments using a dataset of snow crystal growth sequences generated by the LCA model described in chapter 4. Each data point is a variable-length sequence of 64×64 pixel images, with sequence lengths ranging from approximately 200 to 500 steps. The dataset contains 20480 data points, with 90% allocated for training and 10% for validation. Additionally, we have a separate test dataset of 2048 datapoints on which all the following experiments are conducted. The snowflakes are generated with all parameters fixed except for ρ, which corresponds to ambient water vapor density. We sampled the

parameter ρ uniformly between 0.35 and 0.65, generating four snowflakes for each sampled ρ to account for the stochastic nature of the LCA model. These repeated snowflakes are not mixed between train, validation and test splits.

5.3.2 Metrics

Because the growth process is stochastic, snow crystals generated from the same environmental parameters can differ significantly at the pixel level. While low ρ tends to generate thin crystals with few side branches, and a high ρ typically generates a thick crystal with many side branches, the exact location where these branches attach to the crystal can vary within the same ρ. Consequently, a simple pixel-wise accuracy metric is insufficiently useful for model validation. Instead, we opt to validate the model by investigating the high-level morphological properties of the generated structure.

Specifically, we investigate if the joint distribution $p(a, b \mid \rho)$ of the model's generated snowflakes, conditioned on the environmental parameter ρ, matches the joint conditional distribution of the numerical LCA simulator $p_{\mathrm{LCA}}(a, b \mid \rho)$. Where a is the area of the snowflake and b is the boundary length. We quantify the match of these distributions using the expected value of the Wasserstein distance between the conditional distributions, where the expectation is taken over the environmental parameter ρ:

$$\mathrm{EWD} = \mathbb{E}_{\rho \sim p(\rho)} \left[W_2 \left(p(a, b \mid \rho), p_{\mathrm{LCA}}(a, b \mid \rho) \right) \right]. \tag{5.3}$$

In Equation 5.3, W_2 denotes the type-2 Wasserstein distance. Intuitively, the EWD is small if the conditional distributions $p(a, b \mid \rho)$ overlap well with the ground-truth distribution $p_{\mathrm{LCA}}(a, b \mid \rho)$. As a second metric, we consider the Evidence Lower Bound (ELBO) of the model on the held-out data. The ELBO is a lower bound on the log-likelihood, defined in Equation 5.2, where a higher value indicates that the model allocates more probability mass to real samples, and suggests a better fit to the data.

5.3.3 Experiments

We evaluate the model in five configurations: a baseline configuration and four additional configurations, each incorporating a single training strategy added to the baseline model. The Boundary Reinforcement Loss and Boundary Count Loss strategies both have tuneable hyperparameters. For the Boundary Reinforcement Loss, we set the threshold hyperparameter $T = 0.9$ and the loss weight $W_{\mathcal{L}_{BR}} = 1.0$. For the Boundary Count Loss strategy, we set the inverse temperature hyperparameter $\alpha = 100$, and the loss weight $W_{\mathcal{L}_{BC}} = 1.0$. We train every model for 18 epochs, or seven days on a single GTX 1080Ti GPU with batch size 64. All snowflakes generated in the results section are generated with a maximum of 500 steps.

5.4 Results

5.4.1 Qualitative Results

Figure 5.4 presents a matrix of simulated snow crystals for five values of ρ across all five trained models, compared to the ground-truth LCA simulator. Given the stochastic nature of snow crystal growth, as evidenced by the two different ground-truth snowflakes for $\rho = 0.625$, we don't expect a well-performing model to generate identical snow crystals. Instead, we evaluate whether the model produces realistic-looking snowflakes with a style that corresponds to the conditioning parameter ρ. Notably, lower ρ values result in thinner snow crystals with thinner branches.

Neither the Boundary Reinforcement Loss nor the Boundary Count Loss significantly enhances the quality of snowflakes generated by the model. Snowflakes generated with the

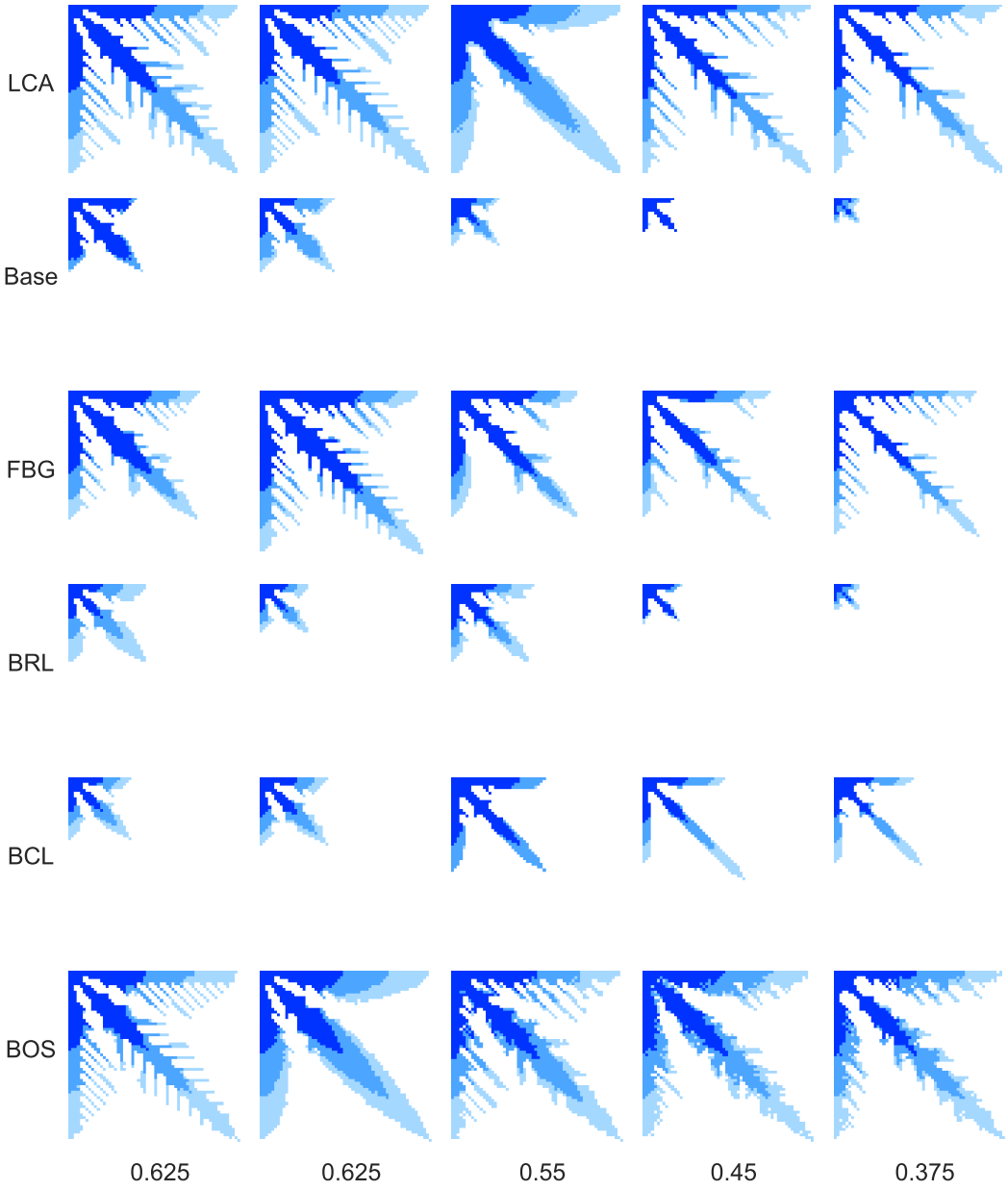


Figure 5.4: Simulated snow crystals for five values of ρ for all five trained models compared to the ground-truth LCA simulator. Three stages of growth are displayed with different shades of blue.

Boundary Count Loss enabled grow slightly larger at lower ρ values compared to the baseline but still fail to grow to completion and lack side-branching. Figure 5.5 indicates that the BRL and BCL models still fall into the same failure mode of stagnated growth, with their growth plots flat-lining early into the process.

The Forced Boundary Growth (FBG) rule notably improves the model's ability to simulate realistic snow crystals. While the baseline model generates only tiny snow crystals, the FBG-enabled model produces larger crystals that are nearly complete. The crystals from the FBG model capture the morphological traits of ground-truth snowflakes well across the entire range of ρ, producing fuller snow crystals at higher values and thinly branched

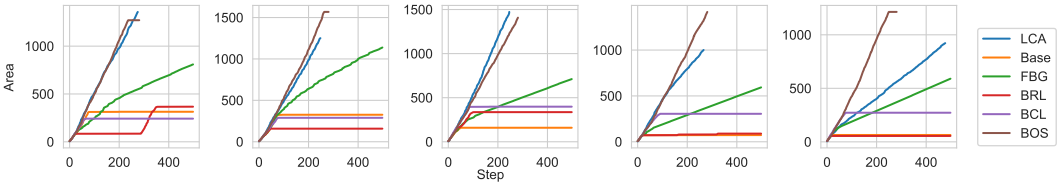


Figure 5.5: Area growth plot of the sampled snowflakes.

snow crystals at lower values. Importantly, these snowflakes appear reasonably realistic, with no obvious artifacts. The primary flaw is their size; although larger than those from the baseline model, they are still smaller than the ground-truth samples from the LCA model. Additionally, Figure 5.5 shows that the FBG model's snowflakes grow significantly slower than those from the LCA simulator, indicating the growth dynamics are not properly captured. This slow growth is due to the Forced Boundary Growth rule only forcing one pixel to grow per step if no growth is predicted. Consequently, after growth probabilities drop below 0.5, the crystal grows only one pixel per step, while it should grow closer to four pixels per step, particularly at higher ρ values.

The model with Bernoulli Output Sampling (BOS) enabled accurately captures the speed of growth. Except at the lowest ρ value, where snow crystal growth is very slow, the area growth plots in Figure 5.5 show a good match between the BOS and LCA traces. Sampling the output values from a Bernoulli distribution enables the number of predicted growth pixels per step to be correctly modeled on aggregate. However, on a per-pixel level, this sampling method introduces noise. Consequently, most snowflakes produced by the BOS model in Figure 5.4 exhibit significant artifacting, such as noisy branches and crystal boundaries, leading to unrealistic snowflakes. While sampling from a distribution may be the correct approach for capturing system dynamics on aggregate, a naive independent distribution like the Bernoulli distribution fails to capture the more complex joint distribution over the growing grid cells.

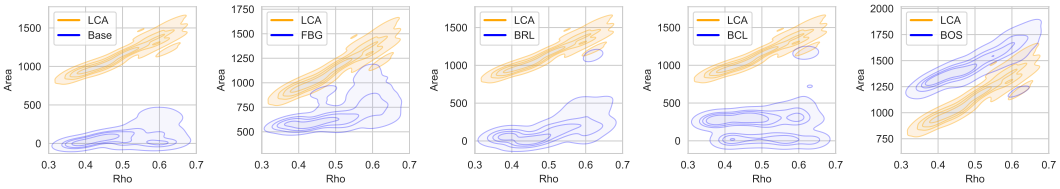
5.4.2 Distribution Overlap

The distribution plots reinforce these findings. In Figure 5.6 and Figure 5.7, it is evident that the baseline, BRL, and BCL models deviate significantly from the ground-truth distributions of snowflake areas and boundary lengths. The FBG model, while closer in terms of area, is biased towards lower values, displayed in Figure 5.6b. This bias is due to the slow growth of the FBG model's snowflakes, resulting in incomplete snowflakes with a lower average area. Similarly, the FBG model's boundary length is closer to the ground-truth, as depicted in Figure 5.7b, but still skewed towards lower values. This can be attributed to the incomplete growth leading to shorter branches and, consequently, a shorter boundary length.

In Figure 5.6e and Figure 5.7e, the Bernoulli Output Sampling (BOS) model tends to overshoot the snowflake area and exhibits a bias towards lower boundary lengths. This suggests that the BOS model creates snowflakes with less branching, favoring more solid crystal structures. We speculate that the noisy output sampling often disrupts the formation of finer branches, leading to these characteristics.

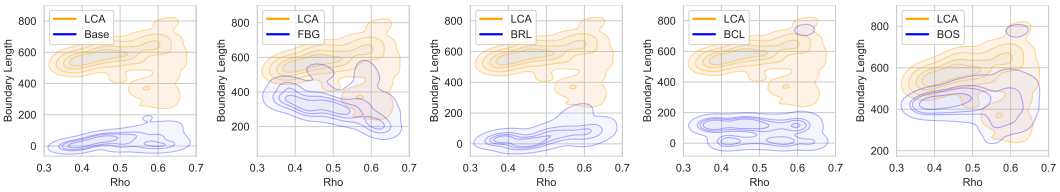
5.4.3 Quantitative Results

The quantitative results align with previous findings. The expected Wasserstein distance described in Equation 5.3 is only slightly lower for the models with Boundary Reinforcement Loss and Boundary Count Loss compared to the baseline model. Models with Forced Boundary Growth and Bernoulli Output Sampling show significantly improved EWD, with the BOS model exhibiting the best fit according to this metric. Similar to the distribution



(a) Baseline and LCA. (b) Forced Boundary Growth and LCA. (c) Boundary Reinforcement Loss and LCA. (d) Boundary Count Loss and LCA. (e) Bernoulli Output Sampling and LCA.

Figure 5.6: Joint distributions over area and environmental parameters for the five trained models, compared with the actual joint distribution produced by the LCA model.



(a) Baseline and LCA. (b) Forced Boundary Growth and LCA. (c) Boundary Reinforcement Loss and LCA. (d) Boundary Count Loss and LCA. (e) Bernoulli Output Sampling and LCA.

Figure 5.7: Joint distributions over boundary length and environmental parameters for the five trained models, compared with the actual joint distribution produced by the LCA model.

overlap plots, the primary reason for the FBG model's poorer performance is its inability to grow a full snowflake within the 500 maximum steps used for this evaluation. In contrast, the BOS model excels on the EWD metric simply due to its ability to grow a complete snow crystal. However, despite the BOS model's superior performance, an EWD of 335.3 remains high, indicating a poor fit to the underlying system dynamics.

The Evidence Lower Bound of the log-likelihood (ELBO) is similar across all models except the Boundary Count Loss enabled model. Since the Forced Boundary Growth and Bernoulli Output Sampling strategies do not alter the loss, they are not expected to to impact the models' predicted probabilities, and thus their ELBO scores. The Boundary Reinforcement Loss strategy does alter the model's loss function, but seems to be a completely ineffective strategy. The Boundary Count Loss strategy, which is the only strategy that integrates into the model's loss function and affects the qualitative results, shows the highest ELBO score. This indicates that this model allocates more probability mass on real data, suggesting a better fit.

5.5 RQ1 Conclusion

While some strategies show improvements over the baseline model, no proposed strategy sufficiently enables the model to capture the system dynamics while preventing the identity function failure mode. We conclude that the best way to model a system characterized by small state changes while preventing this failure mode is by decreasing the temporal resolution. By decreasing the temporal resolution, the differences between sequential states Δx_t are increased from the model's perspective, and the identity function $p_\theta(\hat{x}_{t+1} = x_t \mid x_t, y) \approx 1$ no longer maximizes Equation 5.2. Therefore, the model p_θ that maximizes Equation 5.2

Table 5.1: Quantitative results. The lower expected Wasserstein distance EWD indicates that simulations generated by the Forced Boundary Growth and Bernoulli Output Sampling enabled models match the ground-truth data-generating process better than the other models, while the higher ELBO on the test set for the Boundary Count Loss model indicates a more strongly peaked probability distribution on real crystallization processes.

Method	EWD ↓	ELBO ↑
Baseline	1184.9	-0.0139
Forced Boundary Growth	541.1	-0.0139
Boundary Reinforcement Loss	1126.8	-0.0140
Boundary Count Loss	1047.8	**-0.0131**
Bernoulli Output Sampling	**335.3**	-0.0144

can model an approximation of dynamical systems characterized by small state changes without converging to the identity function, as long as the temporal resolution is sufficiently downsampled.

We implement this by increasing the step size with a stride of 10. However, aggregating 10 growth steps into one means that growth dynamics might extend beyond the pixel-wide crystal boundary. Therefore, limiting the eligible growth area $\mathcal{E}$ to the crystal boundary is no longer valid. Instead, we restrict the eligible growth area to the non-solidified region of x_t:

$$\mathcal{E} = \{(i, j) \mid x_t^{(i,j)} = 0\}. \tag{5.4}$$

As a result, simulated snow crystals can grow more than one pixel per step. However, this also means the simulation can produce floating crystal pieces, as nothing physically prevents this. While such artifacts are undesirable, a model that accurately captures the growth dynamics should not produce them.

Chapter 6

Problem Formulation and Solution Approach: Latent Variable Neglect

In this chapter, we delve into the second research question:

RQ2 What are the specific conditions and model classes where the issue of latent variable neglect arises, and what strategies can effectively address this challenge for such models under these conditions?

6.1 Problem Formulation: RQ2

Convergence to the identity function is not the only challenge with autoregressive latent variable models. A common pitfall of latent variable models, as described in section 3.1.5.1, is posterior collapse. This failure mode occurs when the KL term of the loss function dominates the overall loss, especially during the early parts of training when the signal-to-noise ratio for the reconstruction gradients is very low. In this scenario, the model posterior gets stuck in a local minimum where it fully collapses onto the prior, thereby converging any information difference between the prior and posterior. In the case of an uninformed (Gaussian) prior, this means that the encoder is fully collapsed, and the posterior simply models a fully random Gaussian that is independent from the encoder input. In the case of a conditional prior, such as our model, this means that only the information gain of the posterior collapses to an uninformative random variable. More specifically, we have that:

$$q(z \mid x_{t+1}, x_t, y) \approx p(z \mid x_t, y) \quad \text{for all } x_{t+1}. \tag{6.1}$$

Another failure mode more specific to our model type is Latent Variable Neglect. Since subsequent simulation states are very similar, with only small differences Δx_t, predicting x_{t+1} is possible to a large degree based only on x_t. Therefore, since the decoder $p(x_{t+1} \mid x_t, z)$ is partly conditioned on x_t, the model is able to largely predict x_{t+1} without the use of the latent variable z.

During the early stages of training, the signal-to-noise ratio for z is very low, while the signal-to-noise ratio for x_t is very high. This is particularly exacerbated by the skip connections within the UNet decoder and the high similarity between x_t and x_{t+1}. A simple approximate mapping $p(x_{t+1} \mid x_t)$ exists, which is relatively close to maximizing Equation 5.2, the one-step autoregressive log-likelihood. As a result, the gradients associated with x_t tend to dominate those associated with z, leading the model to ignore the latent variable. Consequently, the latent variable z does not contribute meaningfully to the prediction, which undermines the purpose of incorporating latent variables and leads to the model capturing only the simple approximate mapping. While this approximate mapping is relatively accurate

in the one-step prediction, and therefore might be sufficient in one-step feedforward models, in autoregressive models this results in compounding errors, leading to a poor fit on the stochastic growth dynamics. Formally,

$$p(x_{t+1} \mid x_t, z) \approx p(x_{t+1} \mid x_t) \quad \text{for all } z. \tag{6.2}$$

The result of this is a model that ignores the latent variable, and likely fails to construct a meaningful representation of the encoded information in the latent space, leading to a model where the latent variable mechanism cannot function properly. Consequently, the model captures only a simple approximation of the ground-truth distribution $p^*(x_{t+1} \mid x_t, y)$, leading to a poor fit on the growth dynamics in autoregressive settings.

While the outcome is similar to posterior collapse, the cause is entirely different. Whereas posterior collapse is characterized by the KL divergence loss term limiting the expressive power of the posterior, latent variable neglect is characterized by the gradients to the latent variable being too noisy, resulting in the failure to establish informative pathways. Although nothing actively limits the expressive power of the latent variable, the gradients of x_t into the decoder are sufficiently dominant that they create a local minimum from which the model does not escape. Consequently, even with mechanisms in place to prevent posterior collapse, the latent variable can still be ignored, leading to a model where the latent variable mechanism cannot function properly. This problem is not exclusive to our model, as Bowman et al. [6] observe the same behavior in their latent variable model.

6.2 Formalization of Models Susceptible to LVN

We formally define the model class where LVN can arise: Let $p_\theta(x \mid y)$ be a latent variable model with decoder $p_\theta(x \mid u, z)$ conditioned on a signal $u \subseteq y$ in addition to the latent variable z. Latent variable neglect (LVN) can arise in the model if there exists a simple approximate mapping from u to x:

$$\exists p'_\theta \quad \text{such that} \quad p'_\theta(x \mid u) \approx p^*(x \mid y) \tag{6.3}$$

Where p'_θ is a simple mapping, and $p^*(x \mid y)$ is the ground-truth distribution of x w.r.t. y. This indicates that the ground truth distribution from y to x can be approximated by a simple mapping from $u \subseteq y$ to x, rendering the influence of the latent variable z negligible. In these scenarios, the gradients of the conditioning signal u are significantly stronger than the gradients of the latent variable z, causing latent variable neglect.

Given a model p_θ susceptible to LVN, our goal is to develop a strategy or set of strategies that mitigates the possibility of latent variable neglect.

6.3 Approach: Posterior Collapse

To prevent posterior collapse, we employ beta-annealing, as introduced in section 3.1.5.2. This is implemented with a simple multiplier on the KL term of the loss function, which transitions from 0 to 1 over a set number of epochs.

Additionally, we use the free bits technique described in section 3.1.6.1. Crucially, we follow the instruction of Kingma et al. [27] to first aggregate the KL divergence over the batch dimension, then clamp it to the `free_bits` parameter, and finally aggregate over the remaining dimensions to obtain a scalar value.

$$\mathcal{L}_{KL} = \frac{1}{c \cdot h \cdot w} \sum_{j=1}^{c} \sum_{k=1}^{h} \sum_{l=1}^{w} \max\left(\frac{1}{b} \sum_{i=1}^{b} D_{\mathrm{KL}}^{(i,j,k,l)}, \texttt{free\_bits} \right) \tag{6.4}$$

where b is the batch size, c is the number of channels, h and w are the height and width of the spatial dimensions, and $D_{\mathrm{KL}}^{(i,j,k,l)}$ is the KL-divergence for the i-th sample in the batch and the (j,k,l)-th element in the spatial dimensions. This aggregation order ensures that the free bits trick sets a lower bound on the KL term per spatial dimension. Suppose the lower bound is taken after aggregation of the spatial dimensions. In that case, a scenario can occur where the KL term of most dimensions collapses to zero, while a few dimensions have a very high KL divergence. This effectively causes a large misalignment between the prior and posterior distributions, as all information is encoded in these misaligned dimensions. With the appropriate aggregation order, the KL divergence is controlled for each spatial dimension, ensuring each dimension has enough freedom to encode information while aligning with the prior distribution. By combining these training strategies, we mitigate the risk of posterior collapse.

6.4 Approach: Latent Variable Neglect

We address the issue of Latent Variable Neglect through two strategies: an alternative strategy of injecting the latent variable into the decoder, and more importantly, decoder dropout.

6.4.1 Latent Variable Summation

Typically, we inject the spatial latent variable into the decoder by concatenating it over the channel dimension, with the spatial dimensions upsampled using a nearest-neighbor interpolation strategy at each layer. However, this simple concatenation makes it easy for the model to ignore the latent variable, as it can simply ignore the entire channels where the latent variable is added.

To address this, we experiment with injecting the spatial latent variable into each layer through summation instead of concatenation. While concatenation only requires matching spatial dimensions, summation requires both the spatial and channel dimensions to match between the latent variable and the corresponding decoder layer. To achieve this, we continue to upsample the spatial dimensions using nearest-neighbor interpolation. To match the channel dimensions, we replicate the channels of the latent variable as needed for each layer of the decoder, ensuring the channels of the latent variable never exceed those of any decoder layer.

By summing the latent variable directly into the model's activations, we make it more challenging to disentangle the latent variable information from the model's activations. Consequently, the model cannot easily ignore the latent variable, which mitigates the risk of latent variable neglect.

6.4.2 Decoder Dropout

In addition to summing the latent variable into the decoder, we also experiment with weakening the conditioning pathway of the decoder during training. Since latent variable neglect stems from the gradients of the conditioning pathway overpowering those of the latent variable pathway in the decoder, a reasonable way to mitigate latent variable neglect is by weakening the conditioning pathway.

Similarly to Bowman et al. [6], we implement this weakening by means of dropout. Specifically, we implement a samplewise decoder dropout where x_t is input to the decoder. Whereas regular dropout acts independently over both batches and channels, samplewise decoder dropout only acts independently over only the batch dimension and therefore deactivates the whole input image with probability p_{DD}. This dropout is only used during training. A higher dropout probability forces the model to rely more on the latent variable

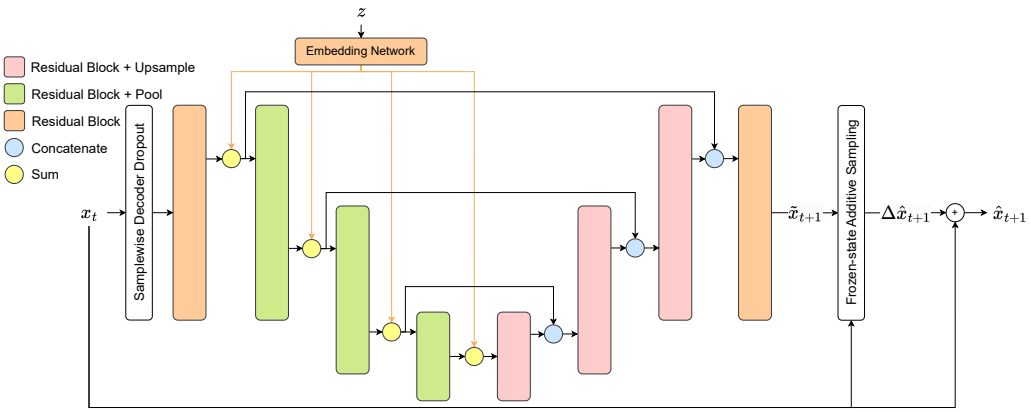


Figure 6.1: Revised decoder architecture with boundary masking removed and Z summation and Decoder Dropout added.

to predict x_{t+1}, thereby reinforcing the VAE reconstruction pathway over x_{t+1} through the posterior:

$$p(x_{t+1} \mid z), \quad \text{with} \quad z \sim q(z \mid x_{t+1}, x_t, y), \tag{6.5}$$

ensuring the formation of an informative latent space and robust pathways from the latent space to the decoder.

The formation of these informative pathways is particularly important during the early stages of training. Once these pathways are established, the signal-to-noise ratio of the latent variable gradients is much higher, reducing the risk of latent variable neglect. We therefore employ a schedule on the dropout probability. During the first stage of training, we fully disable the conditioning input to the decoder by setting $p_{DD} = 1$. Then, we linearly decrease p_{DD} to $p_{DD} = 0.1$, after which it remains at that level for the remainder of training. We maintain this low dropout probability to prevent the model from reverting to latent variable neglect.

6.5 Experiment Setup

6.5.1 Dataset and Metrics

We conduct the following experiments on a downsampled version of the dataset described in section 5.3.1, where we sample the sequences from the original sequences with a stride of 10. This reduces the sequence length tenfold, now ranging from approximately 20 to 50 steps. All other settings remain identical. Furthermore, we re-use the expected Wasserstein distance (EWD) and Evidence Lower Bound (ELBO) metrics described in section 5.3.2. All experiments are conducted only on the test data. All of the subsequent models are trained for 125 epochs, or seven days on a single GTX 1080Ti GPU with a batch size 64. All snowflakes generated in the results section are generated with a maximum of 500 steps.

6.6 Results

6.6.1 Free Bits and Beta Annealing

We begin by examining the effectiveness of strategies to mitigate posterior collapse. Specifically, we compare the baseline model against two configurations that incorporate both the

free bits trick and beta annealing. The first configuration, referred to as "F", optimally sets the `free_bits` parameter to 0.001. At this setting, the KL term stabilizes at 0.001, or one tenth of the overall loss. The second configuration, "HIF", increases the `free_bits` parameter to 0.1 to evaluate the impact of an elevated free bits setting where the KL divergence is ten times the overall loss.

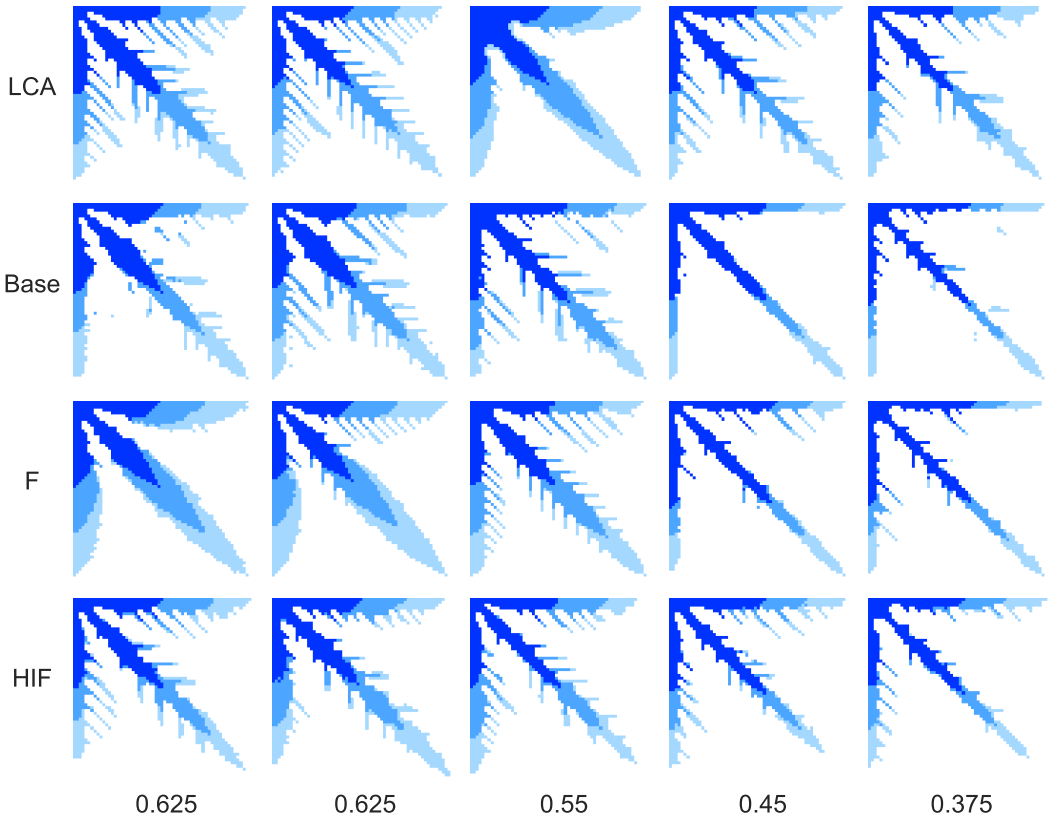


Figure 6.2: Simulated snow crystals for five values of ρ for three trained models compared to the ground-truth LCA simulator. Three stages of growth are displayed with different shades of blue.

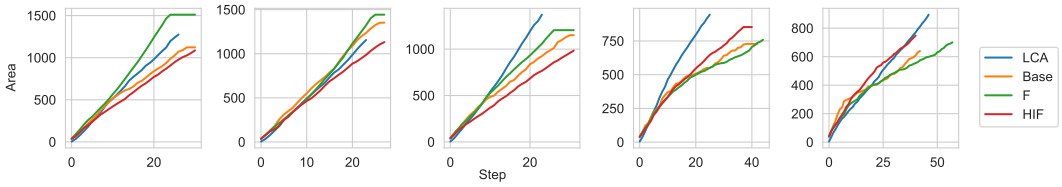
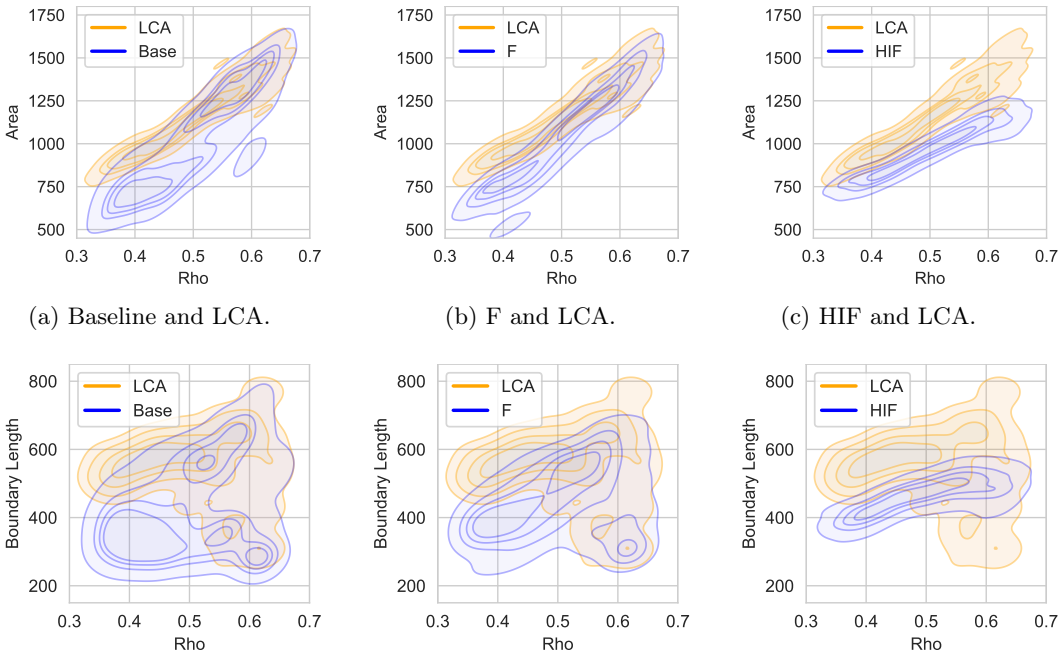


Figure 6.3: Area growth plot of the sampled snowflakes. Growth traces look normal, with no abrupt stops before the snowflake has reached full size, including for the baseline model. This indicates the convergence to the identity function is no longer an issue.

Qualitatively, we see the baseline model trained on a lower temporal resolution and no additional training strategies already shows a great improvement in quality compared to snowflake samples from models without temporally subsampled data displayed in Figure 5.4. Snowflakes now grow to full size, at reasonable speeds and without abrupt stops or deadlocks, as visible in Figure 6.3. Furthermore, the crystals show no artifacts in the form of noisy boundaries or biases toward thicker branches. However, we observe a new form of artifacts

in the form of floating crystal pieces. This is made possible by the fact that we no longer limit $\mathcal{E}$ to the crystal boundary.

Enabling the posterior collapse mitigation methods of beta annealing and the free bits trick seems to improve this artifacting. Both the "F" and "HIF" models show significantly less floating crystal pieces in their sampled snowflakes than the baseline model. Furthermore, at lower ρ, the models with the posterior collapse mitigation strategies enabled are more able to reconstruct fine side branches, whereas the baseline model struggles to grow side branches. Notably, we observe a lack of diversity in the HIF model where the `free_bits` parameter is set higher than optimal.



(a) Baseline and LCA. (b) F and LCA. (c) HIF and LCA.

(d) Baseline and LCA. (e) F and LCA. (f) HIF and LCA.

Figure 6.4: Joint distributions over area and boundary length and environmental parameters for the three trained models, compared with the actual joint distribution produced by the LCA model.

All density plots in Figure 6.4 show a far better match than any of the models evaluated in chapter 5, including the baseline model. The probability mass of the baseline model distributions now significantly overlap the ground truth distributions, especially for higher values of ρ. This suggests the learned snow crystal growth process closely aligns with the ground-truth data generating process for those parameter ranges. For lower values of ρ, we see the distributions skew toward lower values of crystal area and boundary length, which is consistent with the qualitative findings of the model struggling with side-branch formation for this parameter range.

While F shows the best distribution alignment, we again see the HIF model lacks the diversity to match the ground-truth distribution, which is most prominently reflected in the boundary length plot Figure 6.4f. The qualitative results reflect these findings, with the F model performing optimally, and the HIF model performing quantitatively worse than the baseline. We see a trend among these plots that higher values of the `free_bits` parameter result in lower variance in these distribution plots. This means some skew is traded in for bias: whereas we don't see the distribution so far skewed toward lower crystal area and boundary lengths for lower values of ρ, we now do notice the overlap in probability mass starts to decrease for higher values of crystal area and boundary length.

The F model strikes the best middleground between bias and skew, and hence aligns its crystal area and boundary length distributions closest to the ground-truth distributions, resulting in the lowest EWD as displayed in Table 6.1. Although the F model outperforms both other models, crystal areas are still skewed toward lower values, and boundary lengths are both skewed and biased toward lower values, as shown in Figure 6.4b and Figure 6.4e. This suggests the model still struggles with side-branch formation for lower values of ρ, while it now additionally biases against fully branched structures for higher values of ρ. Incidentally, Figure 6.2 displays two solid snowflake samples generated by the F model for the highest ρ values, while the LCA model generates fully branched structures here. While a solid snowflake morphology is possible for this parameter value in the LCA model, Figure 6.4e suggests the F model does not reproduce the most branched snowflake morphologies present in the dataset.

Table 6.1: Quantitative Results: The lower Expected Wasserstein Distance (EWD) demonstrates that the simulations generated by the model with beta annealing and an optimal free bits setting more closely align with the ground-truth data-generating process compared to other models. Conversely, the higher ELBO on the test set for the baseline model suggests a more sharply defined probability distribution for real crystallization processes. It is important to note that the Free Bits trick weakens the KL divergence term in the loss function, which can adversely affect the ELBO, making it a less reliable metric.

Method	EWD ↓	ELBO ↑
Baseline	205.9	**-0.0669**
(F) Free Bits + Beta Annealing	**138.0**	-0.0773
(HIF) Elevated Free Bits + Beta Annealing	282.3	-0.1385

6.6.2 Z Summation

Next, we examine the effectiveness of Z Summation, both in a standalone setting (Z) and combined with the posterior collapse mitigation methods (FZ). We compare these two model configurations against both the baseline model and the best model so far (F). Qualitatively, we observe in Figure 6.5 that the Z model with Z Summation enabled performs better than the baseline model. We observe less floating crystal artifacts, and generated snowflakes look more natural throughout the parameter range, except for lower values of ρ where the Z model struggles with side-branch formation. While better than those of the baseline model, the samples generated by the Z model are not as good as those generated by model F. This model manages to construct realistic looking snow crystals without artifacting throughout the entire parameter range, and struggles less with side-branch formation for lower values of ρ. Clearly the best performing model however, is the FZ model, which combines both the free bits and beta annealing from the F model, as well as the Z summation strategy from the Z model. Since free bits and beta annealing are strategies targeting posterior collapse, and Z summation targets latent variable neglect, these two strategies are able to complement each other by individually tackling the two separate problems. As a result, the samples produced by the FZ model are the best observed yet, with rich diversity and faithful representations of the growth process throughout the entire parameter range.

The quantitative results, displayed in Table 6.2, support this narrative. According to the EWD metric, the best performing model is FZ, followed by F, then Z, then the baseline model. The one-step ELBO is also highest for the model with both the posterior collapse mitigating strategies and Z summation enabled, suggesting a more peaked probability distribution over the ground-truth crystallization process.

Looking at the density plots in Figure 6.6 we again observe that the addition of free bits and beta annealing decreases variance in the crystal area and boundary length distributions,

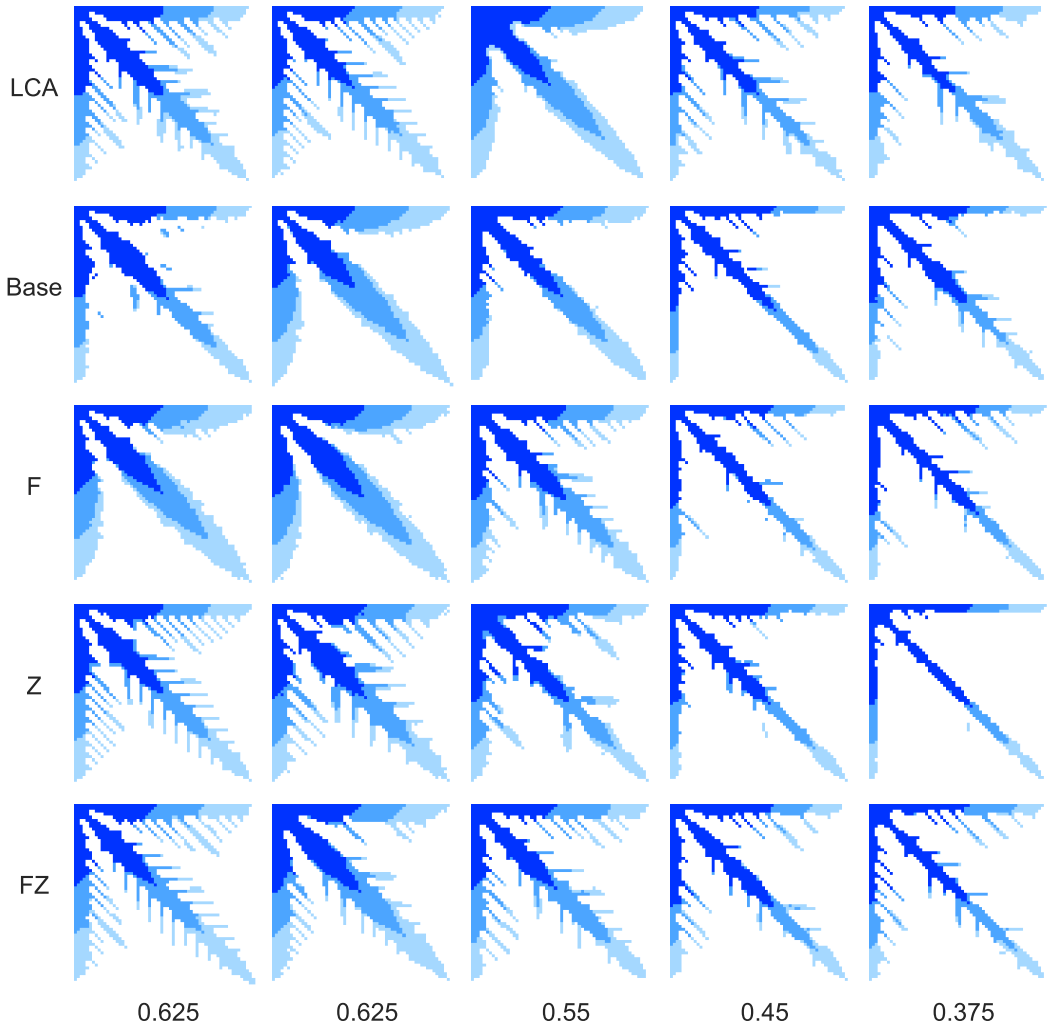
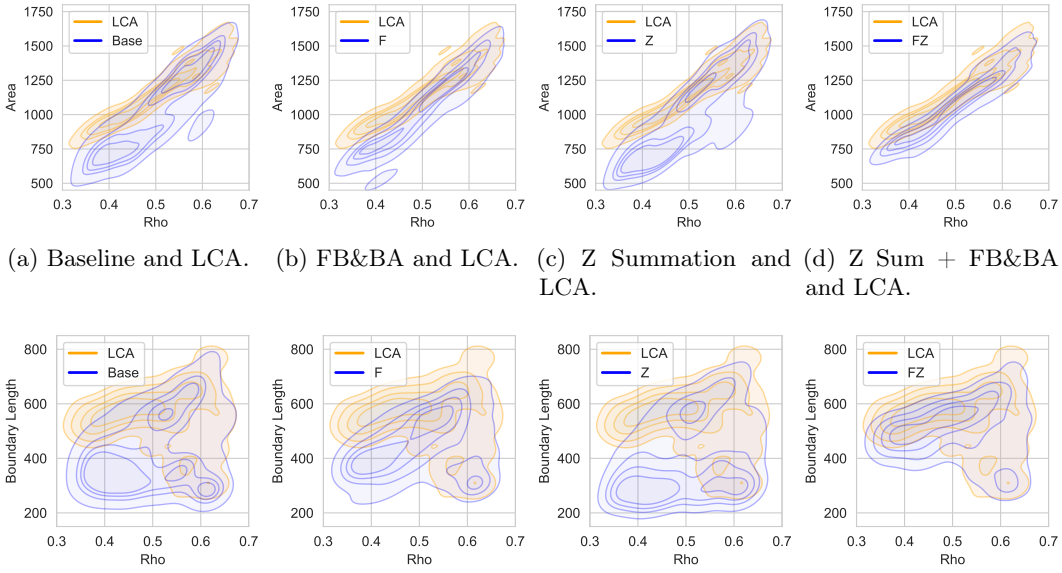


Figure 6.5: Simulated snow crystals for five values of ρ for two Z Summation enabled models compared to the ground-truth LCA simulator as well as the baseline model and the best model so far (F). Three stages of growth are displayed with different shades of blue.

Table 6.2: Quantitative results with Z Summation experiments added.

Method	EWD ↓	ELBO ↑
Baseline	205.9	-0.0669
(F) FB&BA	138.0	-0.0773
(Z) Z Summation	184.0	-0.0726
(FZ) FB&BA + Z Sum	**129.6**	**-0.0637**

and trades in skew for bias. The FZ model has the greatest distribution overlap, aligning with the qualitative results in Table 6.2. However, a slight bias toward lower area and boundary length values is still apparent.



(a) Baseline and LCA. (b) FB&BA and LCA. (c) Z Summation and LCA. (d) Z Sum + FB&BA and LCA.

(e) Baseline and LCA. (f) FB&BA and LCA. (g) Z Summation and LCA. (h) Z Sum + FB&BA and LCA.

Figure 6.6: Joint distributions over area and boundary length and environmental parameters for two models with Z Summation enabled, as well as the baseline model and the so far best performing model (F) for comparison, compared with the actual joint distribution produced by the LCA model.

6.6.3 Decoder Dropout

Decoder dropout delivers superior results over any previously evaluated strategy in terms of qualitative improvements. Previous models would on occasion struggle with growth of side-branches with extreme values of ρ, and would instead simply grow thin centre branches. However, Figure 6.8 shows that model D with decoder dropout enabled no longer exhibits difficulties with side-branch growth across the entire parameter range. The level of diversity among the sampled snowflakes is promising, although assessing how well the level of diversity matches the ground-truth data generating process is challenging with only a few samples. Crucially, the model generates different snow crystals for the same ρ, as seen by the two leftmost samples for model D. A light amount of artifacting in the form of floating crystal pieces is observed with this model.

This artifacting disappears when beta annealing and the free bits trick, which mitigate posterior collapse, are also enabled, as seen in the samples generated by model FD. The FZD model, which includes Z summation in addition to decoder dropout and the posterior collapse mitigating strategies, represents the most complete version of the model with all strategies enabled. There is no discernible difference between the FD and FZD models based on the qualitative results alone.

The density plots convey a similar message. The density plots for model D, which only has decoder dropout enabled, already align significantly better with the ground-truth distribution compared to previous examples. However, the FD and FZD models demonstrate an even closer match. In model D, we still observe a slight skew towards lower crystal areas and boundary lengths for lower values of ρ, indicating a tendency towards less branching under these environmental parameters. This bias is not present in the FD and FZD models. The density plots for crystal area and boundary length match the ground-truth distributions exceptionally well, with no apparent flaws. Based on the density plots, there is no discernible difference between the FD and FZD models, suggesting that the effect of Z Summation is

negligible when decoder dropout is active.

Table 6.3: Complete quantitative results. The lower expected Wasserstein distance EWD indicates that simulations generated by the models with Decoder Dropout and Free Bits + Beta Annealing enabled match the ground-truth data-generating process better than the other models, while the higher ELBO on the test set for these models indicates a more strongly peaked probability distribution on real crystallization processes.

Method	EWD ↓	ELBO ↑
Baseline	205.9	-0.0669
(F) FB&BA	138.0	-0.0773
(Z) Z Sum	184.0	-0.0726
(FZ) FB&BA + Z Sum	129.6	-0.0637
(D) Dropout	71.8	-0.0594
(ZD) Dropout + Z Sum	72.3	-0.2614
(FD) Dropout + FB&BA	**51.8**	-0.0591
(FZD) Dropout + FB&BA + Z Sum	56.3	**-0.0464**

Referring to Table 6.3, we can indeed see that Z Summation has no significant impact on the EWD score for models with decoder dropout enabled. We speculate that this is because Z Summation is a strategy to mitigate latent variable neglect, which decoder dropout also addresses. Decoder dropout is a more effective solution and can mitigate latent variable neglect on its own, rendering Z Summation unnecessary. In contrast, free bits and beta annealing address posterior collapse, a different problem. As evidenced by the improvements in the EWD scores in Table 6.3, models with decoder dropout enabled still benefit from the free bits trick and beta annealing, as decoder dropout does not affect posterior collapse.

6.6.4 Inference Speedup

We choose model configuration FD, which strikes the best balance between simplicity and performance, to be our final model. We refer to this model as the Crystal Growth Neural Emulator (CGNE).

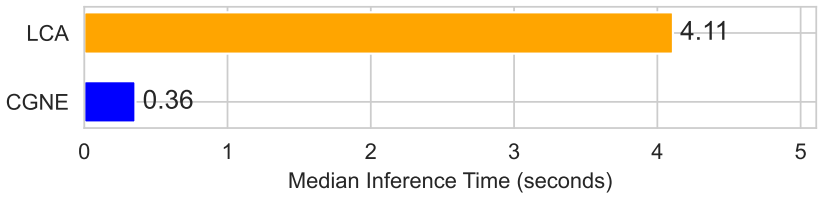


Figure 6.7: Median inference time of a full simulation trajectory for LCA and CGNE.

CGNE effectively increases simulation speed of snow crystal growth. Compared to the GPU-accelerated LCA simulator, CGNE achieves 11× faster inference of snow crystal growth trajectories on average, as displayed in Figure 6.7. We evaluate this by taking the median inference time of 1000 simulation trajectories with ρ linearly spaced between 0.35 and 0.65, where both models use a batch size of 1.

6.7 RQ2 Conclusion

CGNE is able to simulate realistic and probabilistic trajectories of snow crystal growth more efficiently than classical numerical modeling methods. In latent variable models

such as ours, where the decoder is conditioned on a signal that explains a significant amount of output variability, posterior collapse and latent variable neglect are a common issue. The combination of strategies to mitigate both these issues results in a model that effectively captures and closely aligns with the complex process of snow crystal growth. The effectiveness of beta annealing and the free bits trick against posterior collapse is already widely established, and is again apparent in our experiments. While Z summation is effective against latent variable neglect, decoder dropout is so effective against this problem that it renders Z summation obsolete. Importantly, the independent effectiveness of both free bits with beta annealing and decoder dropout clearly displays the difference between the two individual issues of latent variable neglect and posterior collapse. While the symptoms of these issues are similar, their causes and remedies are entirely different. These results are not only significant for simulating crystal growth, but they may also have considerable implications for neural simulation tasks beyond crystal growth, as well as for conditional autoregressive latent variable models more broadly.

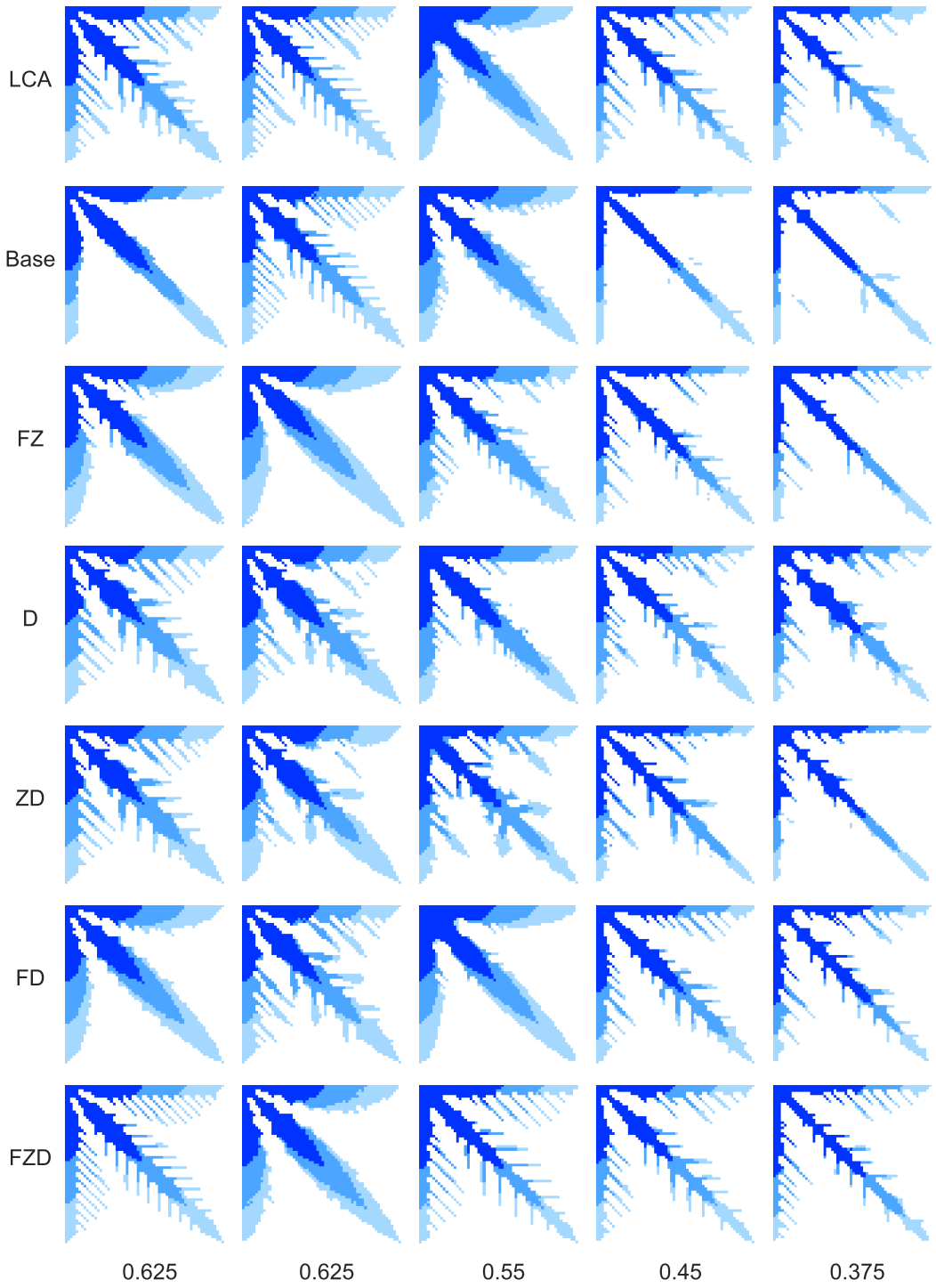
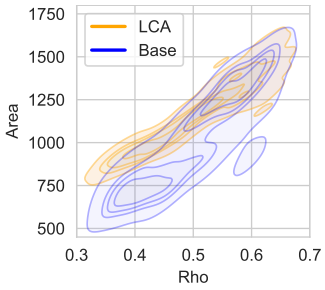
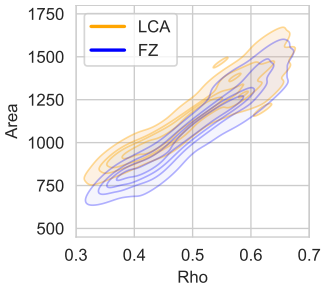


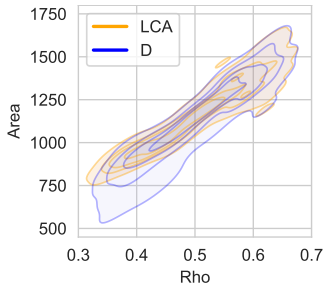
Figure 6.8: Simulated snow crystals for five values of ρ for all four models with decoder dropout compared to the ground-truth LCA simulator as well as the baseline model and the best performing model so far (FZ). Three stages of growth are displayed with different shades of blue.



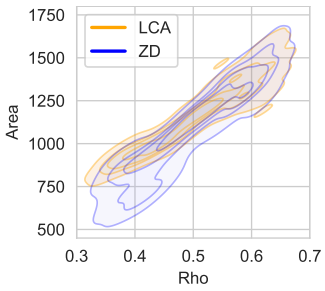
(a) Baseline and LCA.



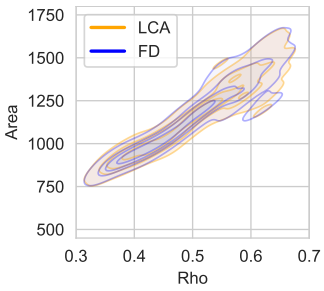
(b) Z Sum + FB&BA and LCA.



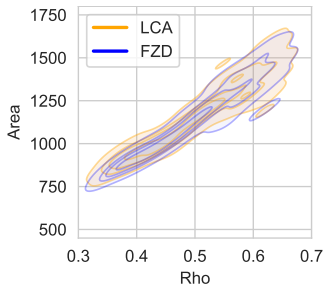
(c) Dropout and LCA.



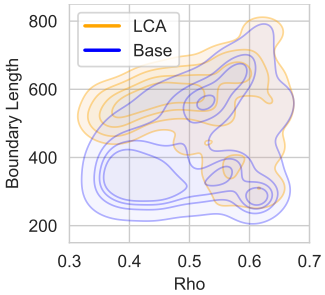
(d) Dropout + Z Sum and LCA.



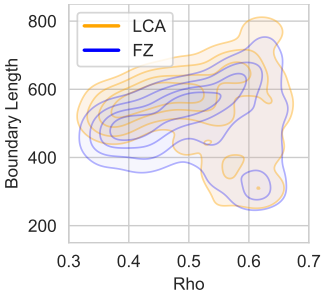
(e) Dropout + FB&BA and LCA.



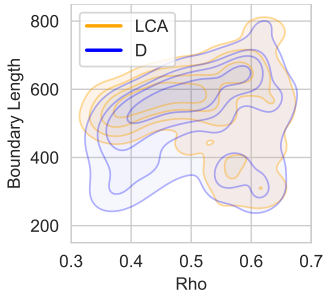
(f) Dropout + Z Sum + FB&BA and LCA.



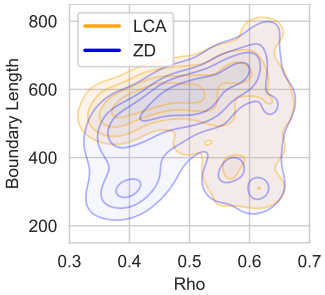
(g) Baseline and LCA.



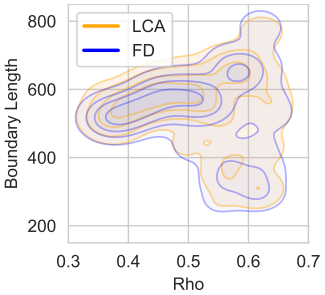
(h) Z Sum + FB&BA and LCA.



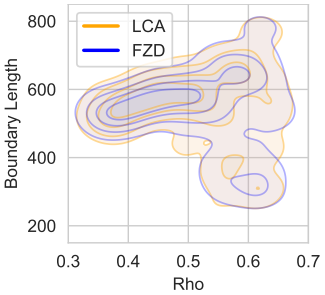
(i) Dropout and LCA.



(j) Dropout + Z Sum and LCA.



(k) Dropout + FB&BA and LCA.



(l) Dropout + Z Sum + FB&BA and LCA.

Figure 6.9: Joint distributions over area and boundary length and environmental parameters for the three trained models, compared with the actual joint distribution produced by the LCA model.

Chapter 7

Conclusions

7.1 Summary of the Results

In this thesis, we introduced the Crystal Growth Neural Emulator (CGNE), a data-driven model for efficient probabilistic simulation of crystallization processes at the mesoscopic scale. Specifically, we developed an autoregressive latent variable model, which we evaluated on snow crystal growth. We discussed the challenges that come with training this model, such as convergence to the identity function, posterior collapse, and latent variable neglect. Consequently, we introduced the following two research questions:

RQ1 How can we best model systems characterized by very small state changes using first-order Markov autoregressive models, while avoiding convergence to the simplistic identity function?

RQ2 What are the specific conditions and model classes where the issue of latent variable neglect arises, and what strategies can effectively address this challenge for such models under these conditions?

Through our experiments, we discovered that the best method to avoid convergence to the identity function is to ensure a sufficiently large step size, which can be achieved by downsampling the training sequences temporally, thereby successfully addressing the first research question.

To address the second research question, we have formalized the problem of latent variable neglect, and the class of models in which this issue can occur. We have demonstrated the difference between latent variable neglect and posterior collapse, and have supplied solutions for both problems. Against latent variable neglect, we have introduced decoder dropout, which mitigates the problem very effectively. Against posterior collapse, we have used two established methods, namely beta annealing and the free bits trick, and demonstrated their effectiveness.

7.2 Summary of the Main Contributions

The main contributions of the thesis are as follows:

- We have introduced CGNE, a novel method for efficiently modeling crystallization processes at the mesoscopic scale. This method fills a gap, as no previous attempts have been made at modeling crystallization processes at the mesoscopic scale with probabilistic neural networks. We have demonstrated the effectivenes of this method on the problem of snow crystal growth.

- We have developed an open-source GPU accelerated simulator that enables the generation of snow crystal growth datasets, such as the dataset used in this thesis. This implementation improves on computational efficiency by several orders of magnitude over existing implementations.
- We have identified convergence to the identity function to be a possible failure mode for neural simulators, and have supplied an effective strategy to prevent this failure mode.
- We have formalized the problem of Latent Variable Neglect (LVN) in the context of conditional autoregressive latent variable models, and have introduced an effective strategy to mitigate this problem called Decoder Dropout.

7.3 Future Work

A potential direction for future research is to re-implement the physical property that restricts crystal growth to occur only when attached to the crystal. Due to increasing step size, we were forced to remove the masking system that limits the eligible growth area to the crystal boundary only. This system enforced the physical property of crystal growth that growth dynamics only occur on the boundary of a crystal, and hence prevented the formation of floating crystal artifacts. With this system removed, the currently proposed model no longer has a mechanism in place that physically prevents the model from creating floating crystal pieces. Incorporating this physical property into CGNE would align it more closely to the physical process of crystal growth.

A further possible direction for future work would be to validate CGNE on different mesoscopic crystallization processes, for example multigrain growth in metallurgy, or other dendritic growth processes, or perhaps simulation of snow crystal growth with a real-valued field of ice density, in addition to the attachment field we currently model. These experiments could demonstrate how well the method generalizes to different types of crystallization.

Appendix A

Further First Principles Simulator Implementation Details

The first-principles simulator is implemented in Python, a programming language widely used for scientific computing and machine learning. We use NumPy, a popular Python library for working with arrays and linear algebra, to manage the simulation state as a 2D array, or matrix. We store the dataset to HDF5 format using h5py. The generated dataset is a folder of HDF5 files, where each file contains a sequence of the simulation state, as well as the environmental parameters of the simulation.

A.1 Speed Efficiency Improvements

An important factor of this implementation of the Gravner-Griffeath algorithm over other existing implementations is its computational efficiency and scalability. This trait is achieved through several optimizations and efficiency improvements, most prominently GPU acceleration.

A.1.1 Batching

To maximize the GPU's capabilities and accelerate the generation of large datasets, we incorporate batching in the simulator. This technique is particularly effective for smaller snowflake sizes, such as 32x32 pixels, where the number of grid cells is less than the number of GPU compute units. The batch size can be adjusted by the user to optimize performance.

A.1.2 Simulation Stop Conditions

As the diffusive mass at the extreme edges of the simulation is kept to a constant ρ, the snowflake tends to distort if it grows too close to the simulation edges. Moreover, snowflakes tend to grow at wildly different speeds depending on their environmental parameters, so giving every snowflake the same amount of simulation steps would lead to wildly different outcomes in final crystal size. To prevent distortion, and allow all snow crystals to grow to an appropriate size, our implementation stops the growth of a crystal when it gets within a certain distance of the simulation boundary. When this happens, the simulation state freezes only for that crystal in the batch. The other crystals in the batch continue their growth, until all crystals have reached the boundary margin, or the maximum number of allowed simulation steps has been reached.

Not only do snowflakes grow at different speeds depending on their environmental parameters, but at extreme values the growth process can even be indefinite. To prevent the simulator from spending excessive time on extremely slow growing snowflakes, the simulator

requires an argument for the maximum number of allowed simulation steps. If this number is reached, the simulation is stopped, and crystals that have not finished growing are discarded.

A.1.3 Utilizing Temporary Directories

The simulator initially does its many write operations to a temporary directory. Only after the dataset generation is complete, is the folder of generated files moved to the desired destination. The use case for this is when the user wants to save the dataset to a slow storage HDD, the simulator can still benefit from the fast SSD storage that the operating system is installed on, assuming that the temporary directory is on this fast drive.

A.2 Storage Efficiency Improvements

Besides speed of generating the dataset, storage space of the dataset is also an important consideration. With thousands of snowflakes in a dataset, thousands of simulation steps per snowflake, and tens to hundreds of thousands of floating point values per step defining the state of the simulation, a naive approach would easily generate a dataset terabytes in size, compared to mere gigabytes required by our current approach.

One previously introduced speed improvement that also enhances space efficiency is the partial simulation method. This method effectively reduces the dataset size by a factor of four, as it involves simulating and saving only a quarter of the cells typically required.

A larger space reduction comes from enabling compression for the HDF5 format.

A.2.1 Pruning Simulation Steps

Additionally, the strategy of only recording simulation steps when the snowflake grows results in a great storage space reduction. Since one of the strategies to expedite the simulation is through reducing temporal resolution, the generated dataset also does not require an excessively high temporal resolution. Recording only the essential simulation steps necessary for the Markov process implies reducing temporal resolution. This modification leads to an approximate reduction in the number of steps saved by a factor of 10-30, thereby substantially decreasing the overall size of the dataset.

A.2.2 Optimized Data Types

Additionally, we opt to store only the attachment and ice fields of the algorithm, rather than the entire simulation state. These two fields sufficiently describe the snow crystal while excluding the simulation of vapor surrounding the crystal and the quasi-liquid boundary layer, which are unnecessary for our purposes. Additionally, since the attachment field consists of binary values, we appropriately store this as an integer, instead of a floating point value. These changes further reduce the space requirements by a factor three.

Bibliography

[1] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press, 1987. ISBN 0934613427.

[2] John W. Barrett, Harald Garcke, and Robert Nürnberg. Numerical computations of faceted pattern formation in snow crystal growth. *Physical Review E*, 86(1), jul 2012. doi: 10.1103/physreve.86.011604. URL https://doi.org/10.1103%2Fphysreve.86.011604.

[3] Federico Bergamin, Cristiana Diaconu, Aliaksandra Shysheya, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E. Turner, and Emile Mathieu. Guided autoregressive diffusion models with applications to PDE simulation. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, 2024. URL https://openreview.net/forum?id=1avNKFEIOL.

[4] W. J. Boettinger, J. A. Warren, C. Beckermann, and A. Karma. Phase-field simulation of solidification. *Annual Review of Materials Research*, 32(1):163–194, 2002. doi: 10.1146/annurev.matsci.32.101901.155803. URL https://doi.org/10.1146/annurev.matsci.32.101901.155803.

[5] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, Jul 2023. URL https://proceedings.mlr.press/v202/bonev23a.html.

[6] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In Stefan Riezler and Yoav Goldberg, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. URL https://aclanthology.org/K16-1002.

[7] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

[8] Salva Rühling Cachay, Bo Zhao, Hailey James, and Rose Yu. DYffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=WRGldGm5Hz.

[9] Mathematical Games. The fantastic combinations of john conway's new solitaire game "life" by martin gardner. *Scientific American*, 223:120–123, 1970. URL https://web.stanford.edu/class/sts145/Library/life.pdf.

[10] Zhihan Gao, Xingjian Shi, Boran Han, Hao Wang, Xiaoyong Jin, Danielle C Maddix, Yi Zhu, Mu Li, and Bernie Wang. Prediff: Precipitation nowcasting with latent diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/pdf?id=Gh67ZZ6zkS.

[11] Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Ander Gray, Daniel Brennand, Nitesh Bhatia, Gregory Stathopoulos, Matt Kusner, Marc Peter Deisenroth, Anima Anandkumar, the JOREK Team, and MAST Team. Plasma surrogate modelling using fourier neural operators. *Nuclear Fusion*, 64(5), apr 2024. URL https://dx.doi.org/10.1088/1741-4326/ad313a.

[12] Janko Gravner and David Griffeath. Modeling snow crystal growth i: Rigorous results for packard's digital snowflakes. *Experimental Mathematics*, 15(4):421–444, 2006. doi: 10.1080/10586458.2006.10128978. URL https://doi.org/10.1080/10586458.2006.10128978.

[13] Janko Gravner and David Griffeath. Modeling snow crystal growth ii: A mesoscopic lattice map with plausible dynamics. *Physica D: Nonlinear Phenomena*, 237(3):385–404, 2008. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2007.09.008. URL https://www.sciencedirect.com/science/article/pii/S0167278907003387.

[14] Janko Gravner and David Griffeath. Modeling snow-crystal growth: A three-dimensional mesoscopic approach. *Phys. Rev. E*, 79:011601, Jan 2009. doi: 10.1103/PhysRevE.79.011601. URL https://link.aps.org/doi/10.1103/PhysRevE.79.011601.

[15] Jayesh K. Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint*, 2022. URL https://arxiv.org/abs/2209.15616.

[16] Takayuki Hirose and Tetsuo Sawaragi. Extended fram model based on cellular automaton to clarify complexity of socio-technical systems and improve their safety. *Safety Science*, 123:104556, 03 2020. doi: 10.1016/j.ssci.2019.104556.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[18] Shuhada A. Idrus-Saidi, Jianbo Tang, Stephanie Lambie, Jialuo Han, Mohannad Mayyas, Mohammad B. Ghasemian, Francois-Marie Allioux, Shengxiang Cai, Pramod Koshy, Peyman Mostaghimi, Krista G. Steenbergen, Amanda S. Barnard, Torben Daeneke, Nicola Gaston, and Kourosh Kalantar-Zadeh. Liquid metal synthesis solvents for metallic crystals. *Science*, 378(6624):1118–1124, 2022. doi: 10.1126/science.abm2731. URL https://www.science.org/doi/abs/10.1126/science.abm2731.

[19] Maximilian Ilse, Jakub M. Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, volume 121 of *Proceedings of Machine Learning Research*, Jul 2020. URL https://proceedings.mlr.press/v121/ilse20a.html.

[20] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019.

[21] Mohamad Ali Jaafar, Daniel R. Rousse, Stéphane Gibout, and Jean-Pierre Bédécarrats. A review of dendritic growth during solidification: Mathematical modeling and numerical simulations. *Renewable and Sustainable Energy Reviews*, 74:1064–1079, 2017. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2017.02.050. URL https://www.sciencedirect.com/science/article/pii/S136403211730268X.

[22] P Jarry and N Jakse. Medium range ordering in liquid al-based alloys: towards a machine learning approach of solidification. *IOP Conference Series: Materials Science and Engineering*, 1274(1), jan 2023. URL https://dx.doi.org/10.1088/1757-899X/1274/1/012001.

[23] Alain Karma and Wouter-Jan Rappel. Phase-field method for computationally efficient modeling of solidification with arbitrary interface kinetics. *Phys. Rev. E*, 53:R3017–R3020, Apr 1996. doi: 10.1103/PhysRevE.53.R3017. URL https://link.aps.org/doi/10.1103/PhysRevE.53.R3017.

[24] Alain Karma and Wouter-Jan Rappel. Quantitative phase-field modeling of dendritic growth in two and three dimensions. *Phys. Rev. E*, 57:4323–4349, Apr 1998. doi: 10.1103/PhysRevE.57.4323. URL https://link.aps.org/doi/10.1103/PhysRevE.57.4323.

[25] James G. Kelly and Everett C. Boyer. Physical improvements to a mesoscopic cellular automaton model for three-dimensional snow crystal growth, 2013. URL https://arxiv.org/abs/1308.4910.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

[27] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

[28] Martin Krzywinski, 2024. URL https://mk.bcgsc.ca/snowflakes.

[29] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=86dXbqT5Ua.

[30] Kenneth G. Libbrecht. Cylindrically symmetric green's function approach for modeling the crystal growth morphology of ice. *Phys. Rev. E*, 60:1967–1974, Aug 1999. doi: 10.1103/PhysRevE.60.1967. URL https://link.aps.org/doi/10.1103/PhysRevE.60.1967.

[31] Kenneth G Libbrecht. The physics of snow crystals. *Reports on Progress in Physics*, 68 (4):855, mar 2005. doi: 10.1088/0034-4885/68/4/R03. URL https://dx.doi.org/10.1088/0034-4885/68/4/R03.

[32] Kenneth G. Libbrecht. Physically derived rules for simulating faceted crystal growth using cellular automata, 2008. URL https://arxiv.org/abs/0807.2616.

[33] Kenneth G. Libbrecht. Incorporating surface diffusion into a cellular automata model of ice growth from water vapor. 09 2015.

[34] Kenneth G. Libbrecht. A quantitative physical model of the snow crystal morphology diagram. 2019. doi: 10.48550/ARXIV.1910.09067. URL https://arxiv.org/abs/1910.09067.

[35] Kenneth G. Libbrecht. *Snow Crystals: A Case Study in Spontaneous Structure Formation*. Princeton University Press, 2022. ISBN 9780691200378. URL http://www.jstor.org/stable/j.ctv1qdqztv.

[36] Kenneth G. Libbrecht. Snowcrystals.com. http://www.snowcrystals.com, 2024. Accessed: 01-2024.

[37] Jake Lever Martin Krzywinski. In silico flurries. *Scientific American Blog Network*, Dec 2017. URL https://blogs.scientificamerican.com/sa-visual/in-silico-flurries/.

[38] Koen Minartz, Yoeri Poels, Simon Koop, and Vlado Menkovski. Equivariant neural simulators for stochastic spatiotemporal dynamics. *arXiv*, May 2023. ISSN 2331-8422.

[39] W. W. Mullins and R. F. Sekerka. Morphological stability of a particle growing by diffusion or heat flow. *Journal of Applied Physics*, 34(2):323–329, February 1963. ISSN 1089-7550. doi: 10.1063/1.1702607. URL http://dx.doi.org/10.1063/1.1702607.

[40] Ukichiro Nakaya. *Snow Crystals: Natural and Artificial*. Harvard University Press, Cambridge, 1954. URL https://www.hup.harvard.edu/catalog.php?isbn=9780674182769.

[41] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023. URL https://arxiv.org/abs/2301.10343.

[42] Sukeharu Nomoto, Hiroshi Wakameda, Masahito Segawa, Akinori Yamanaka, and Toshiyuki Koyama. Solidification analysis by non-equilibrium phase field model using thermodynamics data estimated by machine learning. *Modelling and Simulation in Materials Science and Engineering*, 27(8), oct 2019. URL https://dx.doi.org/10.1088/1361-651X/ab3379.

[43] Norman H. Packard. Lattice models for solidification and aggregation. In *Proceedings of the First International Symposium for Science on Form*, Tsukuba University, 1986. Reprinted in Wolfram, Stephen. Theory and Applications of Cellular Automata. World Scientific (1986), pp. 305-310.

[44] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

[45] Yoeri Poels, Gijs Derks, Egbert Westerhof, Koen Minartz, Sven Wiesen, and Vlado Menkovski. Fast dynamic 1d simulation of divertor plasmas with neural pde surrogates. *Nuclear Fusion*, 63(12), December 2023. ISSN 0029-5515. doi: 10.1088/1741-4326/acf70d.

[46] Liangyuan Ren, Shaoning Geng, Ping Jiang, Song Gao, and Chu Han. Numerical simulation of dendritic growth during solidification process using multiphase-field model aided with machine learning method. *Calphad*, 78, 2022. URL https://www.sciencedirect.com/science/article/pii/S0364591622000554.

[47] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/rezende15.html.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[49] Karl Sims. Reaction diffusion tutorial, 2013. URL `https://www.karlsims.com/rd.html`.

[50] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf`.

[51] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/6ae07dcb33ec3b7c814df797cbda0f87-Paper.pdf`.

[52] Alan Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

[53] Tianju Xue, Zhengtao Gan, Shuheng Liao, and Jian Cao. Physics-embedded graph network for accelerating phase-field simulation of microstructure evolution in additive manufacturing. *npj Computational Materials*, 8(1), September 2022. URL `http://dx.doi.org/10.1038/s41524-022-00890-9`.

[54] Gefan Yang and Stefan Sommer. A denoising diffusion model for fluid field prediction. *arXiv preprint*, 2023. URL `https://arxiv.org/abs/2301.11661`.

[55] Etsuro Yokoyama. Formation of patterns during growth of snow crystals. *Journal of Crystal Growth*, 128(1):251–257, Mar 1993. ISSN 0022-0248. doi: https://doi.org/10.1016/0022-0248(93)90328-T. URL `https://www.sciencedirect.com/science/article/pii/002202489390328T`.

[56] Etsuro Yokoyama and Toshio Kuroda. Pattern formation in growth of snow crystals occurring in the surface kinetic process and the diffusion process. *Phys. Rev. A*, 41: 2038–2049, Feb 1990. doi: 10.1103/PhysRevA.41.2038. URL `https://link.aps.org/doi/10.1103/PhysRevA.41.2038`.

[57] Linshuang Zhang, Manyi Yang, Shiwei Zhang, and Haiyang Niu. Unveiling the crystallization mechanism of cadmium selenide via molecular dynamics simulation with machine-learning-based deep potential. *Journal of Materials Science & Technology*, 185:23–31, 2024. URL `https://www.sciencedirect.com/science/article/pii/S1005030223009623`.