Ministry of Education and Science of Ukraine

National University of "Kyiv-Mohyla Academy"

Department of Informatics of the Faculty of Informatics

Extending MorphoNAS With Multidimensional Developmental Processes for Searching Neural Architectures

Text part to the thesis in the specialty "Software Engineering" - 121

Thesis Supervisor:
Sergii Medvid
(Signature)
" 11 " May 2026

Done by Student:
Maksym Loshak
" 11 " May 2026

Kyiv, 2026

Національний університет «Києво-Могилянська академія»

Факультет Інформатики
Кафедра Інформатики
Освітній ступінь бакалавр
Спеціальність 121 Інженерія програмного забезпечення
Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри А. М. Нагірна

Доцент, кандидат фізико-математичних наук

«___» _____ 20___ року

З А В Д А Н Н Я
ДЛЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Лошака Максима Івановича

1. Тема роботи Розширення MorphoNAS багатовимірними процесами розвитку для пошуку нейронних архітектур
керівник роботи Медвідь Сергій Олександрович, старший викладач
затверджені наказом вищого навчального закладу від «__» _____ 20__ року № _____

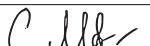
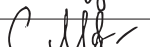
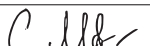
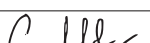
2. Строк подання студентом роботи 15 травня 2026 р.

3. План роботи:

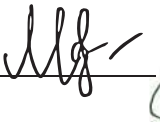
Abstract
1. Introduction
   1.1. Background and Motivation
   1.2. Research Question and Hypotheses
   1.3. Thesis Structure
2. Theoretical Foundations
   2.1. Neural Architecture Search
   2.2. Morphogenetic Development and MorphoNAS
   2.3. Evolutionary Algorithms for NAS
3. Methodology
   3.1. 3D Developmental Environment
   3.2. Neuron Migration Mechanism

Виконавець _____ Науковий керівник _____

ГРАФІК ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника. Узгодження календарного графіка підготовки кваліфікаційної роботи	жовтень 2025	11.09.2025		
2.	Вивчення джерел літератури з морфогенетичного пошуку архітектур нейронних мереж (MorphoNAS), 3D розробки, еволюційних алгоритмів	жовтень – листопад 2025	11.09.2025		
3.	Складання плану дипломної роботи та узгодження з науковим керівником	листопад 2025	2.11.2025		
4.	Розробка 3D розширення фреймворку MorphoNAS: реалізація 3D сотової сітки, 3D дифузії морфогенів	листопад 2025 – січень 2026	2.11.2025		
5.	Проміжний контроль виконання роботи. Розробка механізму міграції нейронів	до 10 лютого 2026	10.12.2025		
6.	Реалізація повного 3D+migration фреймворку.	лютий 2026	10.12.2025		
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літератури).				
	Розділ 2 (аналітично-дослідницька частина: методика експериментів)				
7.	Проведення еволюційних експериментів: 2D baseline, 3D static, 3D+migration.	до 15 березня 2026	10.12.2025		
	Розділ 3 (аналіз результатів експериментів)				
8.	Повне завершення написання кваліфікаційної роботи, оформлення згідно з вимогами й подання на відгук науковому керівнику	до кінця березня 2026	10.12.2025		
9.	Попередній захист кваліфікаційної роботи на засіданні кафедри	до 1 квітня 2026	10.12.2025		
9.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів на відповідність вимогам академічної доброчесності	кінець квітня 2026	10.12.2025		
10.	Подання на зовнішню рецензію	до 15 травня 2026	10.12.2025		
11.	Підготовка до захисту кваліфікаційної роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу	до 20 травня 2026	10.12.2025		
13.	Подання кваліфікаційної роботи на кафедру з усіма супроводжувальними документами	до 25 травня 2026	10.12.2025		
14.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	до 27 травня	10.12.2025		

Графік узгоджено «10» грудня 2025 р.

Науковий керівник Медвідь Сергій Олександрович 

Виконавець кваліфікаційної роботи Лошак Максим Іванович 

Contents

Анотація

Ця робота розширює MorphoNAS, фреймворк морфогенетичного пошуку нейронних архітектур, від двовимірних до тривимірних процесів розвитку. Розширення замінює плоску сітку 32×32 на об'ємну решітку $10 \times 10 \times 10$, збільшуючи локальну зв'язність з 8 до 26 сусідніх клітин. Масштабний порівняльний експеримент із 1 000 незалежних еволюційних запусків (50 цільових конфігурацій, 10 випадкових ініціалізацій кожна, 292,7 годин обчислень) надає свідчення того, що тривимірний морфогенетичний розвиток перевершує двовимірний за кількома метриками: загальний рівень успішності (68,6% проти 60,0%, $p = 0{,}006$), умовний рівень успішності серед запусків без відмов (77,6% проти 66,2%, $p = 0{,}0002$), рівень стагнації (19,8% проти 30,6%, $p = 0{,}0001$) та швидкість збіжності (в середньому 114 проти 138 поколінь, $p = 0{,}048$), тоді як рівні відмов не мають статистично значущої різниці ($p = 0{,}302$). Розроблено умовний статистичний фреймворк для аналізу бімодальних розподілів пристосованості, за яких стандартні рангові критерії дають хибні результати.

Інженерний внесок полягає у єдиній кодовій базі, в якій режими 2D та 3D співіснують із повною зворотною сумісністю, тому існуючі експерименти не потребують змін. Розширення охоплює основну симуляцію, серіалізацію геному, еволюційні оператори та експериментальну інфраструктуру. Окремий конвеєр візуалізації, побудований на анімаційному рушії Manim, дозволяє переглядати об'ємні поля морфогенів, відображати ортогональні проєкції та анімувати міграцію нейронів крізь решітку розвитку.

Abstract

This thesis extends MorphoNAS, a morphogenetic neural architecture search framework, from two-dimensional to three-dimensional developmental processes. The extension replaces the flat 32×32 grid with a volumetric $10 \times 10 \times 10$ lattice, increasing local neighborhood connectivity from 8 to 26 cells. A large-scale comparative experiment comprising 1,000 independent evolutionary runs (50 target configurations, 10 seeds each, 292.7 hours of computation) demonstrates that 3D morphogenetic development outperforms 2D across multiple metrics: overall success rate (68.6% vs. 60.0%, $p = 0.006$), conditional success rate among non-failed runs (77.6% vs. 66.2%, $p = 0.0002$), stagnation rate (19.8% vs. 30.6%, $p = 0.0001$), and convergence speed (mean 114 vs. 138 generations, $p = 0.048$), while failure rates show no statistically significant difference ($p = 0.302$). A conditional statistical framework was developed to address bimodal fitness distributions that render standard rank-sum tests misleading.

The engineering contribution is a single codebase in which the 2D and 3D modes coexist with full backward compatibility, so existing experiments require no modification. The extension touches the core simulation, genome serialization, evolutionary operators, and experiment infrastructure. A separate visualization pipeline was built on the Manim animation engine to inspect volumetric morphogen fields, display orthogonal projections, and animate neuron migration through the developmental lattice.

1. Introduction

Relevance. Neural architecture search (NAS) automates the design of neural networks, but conventional approaches rely on explicit architecture descriptions and gradient-based or reinforcement-learning search over hand-crafted search spaces. Early reinforcement learning-based approaches required approximately 22,400 GPU-days [1], while evolutionary algorithm approaches like AmoebaNet consumed up to 3,150 GPU-days [2]. Morphogenetic NAS is a distinct paradigm: compact genomes encode biological developmental rules, and neural architectures emerge through self-organization on a spatial grid. MorphoNAS demonstrated this approach in two dimensions, achieving 100% success on structural targeting tasks [3]. Extending morphogenetic development to three dimensions is motivated by biological evidence that dimensionality strongly affects cell signaling, spatial patterning, and emergent tissue organization [4, 5]. Despite advances in 3D neural cellular automata [6] and 3D evolutionary NAS [7], 3D morphogenetic development has not yet been applied to neural architecture search.

Goals and objectives. The goal of this thesis is to extend the MorphoNAS framework to three-dimensional developmental processes and to empirically evaluate whether the additional spatial dimension improves the evolutionary search for neural architectures. The objectives are:

1. Extend MorphoNAS from 2D to 3D within a single codebase, maintaining full backward compatibility.
2. Design and conduct a large-scale comparative experiment (1,000 runs) with fair, unbiased initialization conditions.
3. Develop a conditional statistical framework suitable for the bimodal fitness distributions observed in evolutionary NAS.
4. Build a 3D visualization pipeline for volumetric morphogen fields and neuron migration.

Object of research: the MorphoNAS morphogenetic neural architecture search framework and its extension to three-dimensional developmental processes.

Subject of research: the effect of spatial dimensionality on the efficiency and reliability of morphogenetic neural architecture search.

Methods. The research employs evolutionary optimization via a genetic algorithm with tournament selection and block-preservation crossover, developmental simulation on toroidal grids with reaction-diffusion dynamics, non-parametric statistical analysis (Mann-Whitney U, Fisher's exact test, Cliff's delta), and 3D scientific visualization using the Manim animation engine.

Scientific novelty. This thesis extends the morphogenetic neural architecture search paradigm from planar to volumetric development by implementing MorphoNAS on a fully three-dimensional toroidal lattice, while preserving the compact, biologically inspired genome representation introduced in the original 2D framework [3]. Unlike existing 3D NAS methods, which operate over explicitly parameterized search spaces for convolutional backbones [8, 9, 7], our system derives complete neural architectures as emergent products of a 3D morphogen-guided developmental process. At the same time, it differs from prior 3D self-organizing systems such as neural cellular automata, which grow volumetric artefacts but do not perform task-driven architecture search [6]. The thesis further introduces a conditional statistical analysis framework tailored to bimodal fitness landscapes in evolutionary NAS, decomposing outcomes into failure, conditional success, and stagnation components and thereby avoiding the misleading conclusions that arise when standard rank-based tests are applied to such mixtures [10, 11, 12].

Practical significance. The engineering contribution is a reusable, backward-compatible 3D extension of the MorphoNAS codebase. The 3D visualization pipeline provides tools for inspecting volumetric developmental dynamics. The conditional statistical framework is applicable to any evolutionary optimization comparison with bimodal outcome distributions.

1.1. From Traditional NAS to Morphogenetic Discovery

White et al. [13] conducted a comprehensive survey analyzing over 1,000 NAS papers, categorizing approaches by search strategy, search space design, and evaluation mechanisms. They revealed that while early NAS required thousands of GPU-days, modern techniques reduced costs dramatically through weight sharing and performance prediction. Critically, their analysis showed that search space design fundamentally constrains which architectures can be discovered, since cell-based spaces enable transferability but may sacrifice novelty [13]. This obser-

vation motivates exploring alternative search spaces fundamentally different from conventional cell-based or layer-based designs.

On the other hand, MorphoNAS [3] abandons explicit architecture design entirely. Instead, it encodes biological developmental rules in compact genomes that specify only morphogen parameters (diffusion properties, secretion rates, cross-inhibition) and cellular response thresholds. Operating on a 2D toroidal grid, a single progenitor cell self-organizes into complete neural networks through local interactions governed by chemical gradients. Despite this genomic simplicity, MorphoNAS achieved 100% success on structural targeting for graphs with 8-31 nodes and 94% on CartPole control, with 72% of solutions using only 6-7 neurons when optimizing for compactness [3]. This demonstrated that morphogenetic self-organization produces both complex and efficient architectures, establishing the foundation upon which we build.

1.2. Scaling Morphogenesis to Higher Dimensions

The question naturally arises of what happens when we extend morphogenetic development beyond 2D. Biological evidence suggests this transition is fundamental, as studies comparing 2D and 3D tissue organization demonstrate that dimensionality profoundly affects cell signaling pathways, behavior, and emergent structures [4]. Sudhakaran et al. [6] addressed this for neural cellular automata (NCAs), extending them to 3D using 3D convolutional kernels. Their system grew complex volumetric structures in Minecraft environments with over 3,000 blocks. Remarkably, these 3D structures retained regenerative properties and could regrow when damaged through continued local iterations [6].

This work proved that self-organizing developmental processes scale to volumetric spaces despite increased computational complexity. However, their focus was visual pattern generation rather than functional neural architectures. Concurrently, Yang et al. [7] tackled 3D architecture search from a different angle through SHSADE-PIDS, combining Success-History-based Adaptive Differential Evolution with Point Interaction Dimension Search for 3D point cloud processing. Their key innovation was encoding discrete architecture choices into continuous vectors suitable for evolutionary optimization, achieving 64.51% mIoU on SemanticKITTI with only 0.55M parameters (22-26$\times$ reduction vs. competitors) [7]. Together,

these works establish that both morphogenetic systems and evolutionary NAS scale effectively to 3D, suggesting their combination is computationally feasible.

Yet a critical question remains unanswered, namely whether 3D morphogenetic development produces qualitatively different and superior neural architectures compared to 2D? The biological evidence suggests it should. Gui et al. [5] studied morphogen signaling during Drosophila wing development and found that the transition from 2D to 3D tissue architecture fundamentally alters how signals propagate between cells. This demonstrates that 3D morphogenesis is not merely more complex, but enables fundamentally different developmental phenomena. Applied to neural development, 3D space enables novel network topologies, hierarchical layering, and spatial organization impossible in 2D.

1.3. Introducing Neuron Migration

Beyond dimensionality, biological neural development incorporates mechanisms entirely absent from MorphoNAS, most notably neural migration. During embryonic development, neurons undergo extensive migration along chemical gradients (chemotaxis) to reach their functional destinations, a process mediated by chemoattractant and chemorepulsive signals [14, 15]. This process is crucial for forming layered structures, achieving wiring economy, and enabling proper circuit formation. The biological importance of migration raises a complementary question of whether incorporating neuron migration into morphogenetic NAS could enable more efficient network organization and better performance.

Current morphogenetic NAS systems, including MorphoNAS, treat cells as static once placed in the developmental field. This assumption contradicts biological neural development, where migration is a fundamental developmental process. The gap in the literature is notable: while 3D cellular automata have been explored [6] and 3D NAS methods have been developed [7], none of these systems incorporate neuron migration into a morphogenetic developmental process. This thesis takes a first step toward addressing this gap by building a 3D visualization pipeline that simulates migration dynamics (Section 4.6), while leaving integration into the evolutionary framework as future work (Section 7.3).

1.4. Scope and Contributions

This thesis addresses the gap by extending MorphoNAS to three-dimensional development and empirically validating whether the additional spatial dimension improves the evolutionary search for neural architectures. The 3D extension is the first step toward the broader program of multidimensional developmental processes suggested by the thesis title. Neuron migration, the second such process, was explored through a custom 3D visualization system built with Manim (Section 4.6) and is discussed as future work (Section 7.3), but was not integrated into the evolutionary framework for this thesis.

The core engineering contribution is a single-codebase 3D extension maintaining full backward compatibility with existing 2D experiments. The core scientific contribution is a comparative experiment demonstrating that 3D development outperforms 2D in success rate, convergence speed, and stagnation avoidance, while failure rates show no statistically significant difference under fair experimental conditions.

1.5. Research Hypotheses

This thesis tests the hypothesis that extending MorphoNAS from two-dimensional to three-dimensional development will improve the evolutionary search for neural architectures. Specifically, we predict that the 3.25-fold increase in neighborhood connectivity (26 neighbors in 3D vs. 8 in 2D) will allow the developmental process to produce target architectures more reliably and with faster convergence, because the richer spatial structure provides more degrees of freedom for morphogen-guided self-organization.

This hypothesis is grounded in biological observations that three-dimensional morphogenesis fundamentally shifts spatial relationships and diffusion dynamics, enabling qualitatively different tissue organizations compared to 2D systems [5]. MorphoNAS demonstrated that compact genomic encodings combined with reaction-diffusion dynamics can generate complex neural architectures through self-organization in 2D, achieving 100% success rates on structural targeting tasks and 94% on CartPole [3]. Extending the developmental field to 3D introduces exponentially more geometric possibilities for cellular arrangement and connectivity

patterns, while maintaining the same compact genome representation. We test this through a large-scale comparative experiment (1,000 runs) with fair experimental conditions designed to isolate the effect of dimensionality from other confounds such as initialization bias.

1.6. Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 reviews background literature and theoretical foundations, covering neural architecture search, morphogenetic approaches, and the biological principles underlying the developmental model. Chapter 3 presents the software design and architecture of the 3D extension. Chapter 4 details the mathematical formalization and engineering implementation. Chapter 5 describes the experimental methodology, including the benchmark task, developmental environments, and statistical analysis framework. Chapter 6 presents the results of the 1,000-run comparative experiment. Chapter 7 discusses the interpretation and limitations of the findings. Chapter 8 concludes with a summary of contributions and directions for future work.

2. Background and Theoretical Foundations

This chapter presents the background material on neural architecture search methodologies, morphogenetic design principles, multidimensional search spaces, and biological mechanisms relevant to three-dimensional development, followed by the core theoretical principles underlying the MorphoNAS developmental process.

2.1. Neural Architecture Search: Foundations and Evolution

Neural architecture search has emerged as a critical subfield of automated machine learning, aiming to automate the design of optimal neural network architectures tailored to specific tasks and datasets [16]. The most fundamental challenge in NAS is to navigate vast combinatorial search spaces efficiently while discovering high-performing architectures without prohibitive computational costs [13, 17].

2.1.1. Search Strategies in NAS

The evolution of NAS methodologies can be broadly categorized by their search strategies. Early approaches utilized reinforcement learning (RL), where controllers generate architecture descriptions and receive rewards based on validation performance [1]. However, these methods often required thousands of GPU-days for a single search (for instance, the original NAS method consumed 22,400 GPU-days [1]), necessitating the development of more efficient alternatives. Evolutionary algorithms have emerged as a particularly promising alternative due to their population-based search mechanism, which naturally suits exploring vast combinatorial spaces of neural architectures [16]. Notable evolutionary NAS techniques include NeuroEvolution of Augmenting Topologies (NEAT), Genetic CNN, and AmoebaNet [2], which have demonstrated the ability to discover highly competitive architectures with greater stability, lower computational requirements, and less hyperparameter tuning compared to RL-based methods [13].

More recent paradigms have shifted toward efficiency through weight-sharing

and gradient-based optimization. Differentiable architecture search (DARTS) formulates NAS as a continuous optimization problem, treating the architecture search space as a continuous domain where operations are mixed and gradually discretized [18]. This approach dramatically reduced search costs from thousands of GPU-days to approximately 0.2–4 GPU-days [18, 19], a reduction of several orders of magnitude.

2.1.2. Search Space Design

The design of the search space itself has proven to be a critical factor in NAS effectiveness [13]. Cell-based search spaces, where small computational cells are discovered and then stacked to form the full network, dominate the NAS literature due to their transferability across datasets [13]. However, recent work has demonstrated that these cell-based spaces may inadvertently constrain the diversity of discoverable architectures, as the fixed stacking pattern limits the overall topology to repeated motifs [16].

2.2. Morphogenetic Approaches to Neural Network Design

The application of morphogenetic principles to neural network design represents a paradigm shift from explicit architectural specification to emergent self-organization [3]. Rather than defining network components directly, morphogenetic approaches encode developmental rules, and the architecture emerges as the result of a simulated biological growth process.

2.2.1. MorphoNAS Framework

MorphoNAS introduces a morphogenesis-inspired NAS that uses compact genomes to specify reaction-diffusion parameters (diffusion rates, secretion profiles, cross-inhibition matrices) and thresholds for cell behaviors (division, differentiation, axon growth) [3]. A developmental simulation on a 2D toroidal grid starts from a single progenitor cell. Over many timesteps morphogens diffuse, cells divide to fill space, differentiate into neurons, and extend axons, eventually producing a complete neural network. The approach achieved 100% success on graph-property

targeting (8–31 nodes) and 94% on CartPole, with the majority of solutions using compact architectures (6–7 neurons) [3].

The MorphoNAS search space is fundamentally different from conventional NAS: instead of choosing from predefined operations, the genome encodes physical parameters of a biological-like process. This means the search space is continuous and the architecture is an emergent property of the developmental dynamics rather than a directly specified design.

The framework was designed with an abstract optimizer interface, allowing different search strategies to be substituted without modifying the developmental simulation or fitness evaluation pipeline.

2.3. Multidimensional NAS and Spatial Architectures

2.3.1. 3D Neural Architecture Search

The adaptation of NAS to 3D domains has gained traction, particularly for applications in medical imaging, LiDAR-based perception, and volumetric data analysis. Methods such as C2FNAS (Coarse-to-Fine NAS) [8] and NG-NAS (Node Growth NAS) [9] have been developed specifically for 3D medical image segmentation, incorporating specialized 3D convolution operators and search spaces. However, these approaches typically search for optimal 3D convolutional architectures through conventional NAS paradigms rather than through developmental processes [9].

The work by Yang et al. on SHSADE-PIDS [20] represents a significant contribution to evolutionary NAS for 3D point cloud analysis. This framework bridges discrete and continuous optimization spaces by encoding discrete neural network architectures into continuous vectors suitable for differential evolution. The system achieved state-of-the-art results on 3D semantic segmentation (64.51% mIoU on SemanticKITTI with only 0.55M parameters) and classification (93.4% accuracy on ModelNet40 with 1.31M parameters), demonstrating that evolutionary algorithms can effectively navigate complex 3D architecture search spaces while optimizing for both performance and efficiency.

2.3.2. Challenges in High-Dimensional Search Spaces

Extending NAS to higher-dimensional spaces introduces several challenges. The search space grows exponentially with dimensionality, making exhaustive or even comprehensive sampling infeasible [7]. Computational costs increase substantially as 3D operations (convolutions, pooling, etc.) require significantly more memory and computation than their 2D counterparts, necessitating specialized hardware-aware search strategies [21]. Evaluation becomes more expensive, necessitating more sophisticated performance predictors or zero-shot proxies. Additionally, the increased representational capacity of 3D networks may require larger training datasets to avoid overfitting, further complicating the search process.

2.4. Biological Mechanisms in Morphogenesis and Development

2.4.1. 3D Morphogenesis and Spatial Patterning

Biological morphogenesis inherently occurs in three dimensions, with the spatial organization of cells fundamentally shaping tissue and organ function. Studies comparing 2D and 3D tissue organization demonstrate that dimensionality profoundly affects cell signaling pathways, behavior, and emergent structures [4]. Gui et al. [5] showed that the transition from 2D to 3D tissue architecture during Drosophila wing development fundamentally alters how morphogen signals propagate between cells.

Sudhakaran et al. [6] demonstrated that neural cellular automata can be extended to 3D, growing complex volumetric structures with over 3,000 blocks in Minecraft environments while retaining regenerative properties. This established that self-organizing developmental processes can scale to volumetric spaces, though their work focused on visual pattern generation rather than functional neural architectures.

Three-dimensional Turing patterns exhibit different behaviors compared to their 2D counterparts. Pattern formation in 3D depends more critically on initial conditions and domain growth, with mode competition playing a more significant role [22, 23]. The additional spatial dimension enables richer pattern diversity and more complex spatial hierarchies, but also increases the parameter sensitivity of

pattern formation [24].

The following sections present the core theoretical principles underlying the MorphoNAS developmental process.

2.5. Free Energy Principle in Morphogenesis

The Free Energy Principle (FEP) formalizes the tendency of living systems to minimize variational free energy, defined as the discrepancy between predicted and sensed signals, in order to maintain homeostasis [25]. This principle establishes that physical systems minimize surprisal (the negative log probability of outcomes) by maintaining internal models that predict sensory inputs through hierarchical predictive coding [26]. Within a morphogenetic setting this provides a normative lens for understanding how cells evaluate their local environment and elect the actions that reduce surprise. MorphoNAS leverages this idea by encoding thresholds that represent expected morphogen concentrations. When observations diverge from those expectations, progenitor cells respond by dividing, differentiating, or remaining quiescent, while neurons adjust secretion profiles or initiate growth. This implicit minimization of surprise guides the developmental trajectory toward low-entropy anatomical configurations without requiring explicit global coordination. [3]

2.6. Reaction-Diffusion Systems

Reaction-diffusion (RD) systems capture how interacting diffusive chemicals can spontaneously break symmetry and produce spatially structured patterns [27]. Originally proposed by Alan Turing in his seminal 1952 paper ‘The Chemical Basis of Morphogenesis,’ this mechanism demonstrates how morphogen gradients with differential diffusion rates can generate periodic patterns such as stripes and spots from initially homogeneous states [22]. MorphoNAS instantiates RD dynamics by allowing each cell type to secrete morphogens with configurable diffusion coefficients, decay rates, and cross-inhibition matrices. The resulting concentration gradients provide positional information that cells exploit when evaluating developmental rules. Because gradients emerge from local interactions, complex organizations such as layered or modular tissues can arise from compact

genomic encodings, and extending the developmental field to three dimensions simply augments the spatial degrees of freedom over which these gradients form.

2.7. Gene Regulatory Networks

Gene regulatory networks (GRNs) interpret morphogen cues and trigger discrete developmental programs. [3] In MorphoNAS the genome specifies GRN-like logic through thresholded response functions: when morphogen concentrations fall within certain intervals, cells adopt new identities, alter secretion behavior, or initiate axonal extension. [3] This abstraction captures the essence of transcriptional regulation while remaining computationally tractable. By coordinating RD-guided positional information with GRN decision boundaries, the framework can orchestrate multistage development and synthesize neural architectures whose topology and connectivity reflect the encoded developmental rules.

2.7.1. Chemotaxis and Neuronal Migration

During embryonic neural development, neurons undergo extensive migration along chemical gradients (chemotaxis) to reach their target destinations. This process involves multiple mechanisms: radial migration, where neural progenitors move outward from the neural tube; tangential migration, where neurons move parallel to the brain surface guided by chemotactic gradients [14]; and chain migration, where neurons follow pathways established by pioneer cells. Chemotaxis is mediated by chemoattractant and chemorepulsive signals that create concentration gradients guiding cell movement, with neuronal responses exhibiting gradient steepness-dependent regulation [28].

The biological rationale for neuronal migration is multifaceted. Migration enables cells born in one location to function in another, allowing developmental programs to separate neurogenesis sites from final functional positions. This separation facilitates the formation of laminar structures (like the six-layered neocortex) where neurons born at different times migrate past earlier-born neurons to form inside-out or outside-in layers [29]. Migration also contributes to wiring economy by allowing neurons to position themselves optimally relative to their synaptic partners, minimizing axonal wiring lengths [14].

3. Software Design and Architecture

This chapter describes the software engineering aspects of the three-dimensional MorphoNAS extension, including the requirements that drove the design, the system architecture, technology choices, key design decisions, and the approach to testing and validation.

3.1. Requirements Specification

The 3D extension was designed to satisfy the following functional and non-functional requirements.

Functional requirements:

FR1: Support three-dimensional developmental grids with $10 \times 10 \times 10$ toroidal lattices, including 3D morphogen diffusion, 26-neighbor Moore neighborhoods, and volumetric cell positioning.

FR2: Maintain full backward compatibility with existing 2D experiments, so that all prior results remain reproducible under the same codebase.

FR3: Enable large-scale comparative experiments between 2D and 3D development with configurable target specifications, random seeds, and evolutionary parameters.

FR4: Provide 3D visualization of volumetric morphogen fields, neuron positions, connectivity, and neuron migration trajectories.

Non-functional requirements:

NFR1: Single codebase: both 2D and 3D modes must be supported by the same source files, with no code duplication across separate implementations.

NFR2: Reproducibility: all experiments must be deterministic given a fixed random seed and configuration file.

NFR3: Reasonable performance: a single evolutionary run (500 generations, population size 50) should complete within 30 minutes on commodity hardware.

3.2. System Architecture

The MorphoNAS system is organized into six major modules, illustrated in Figure 3.1. The `is_3d` flag, set when the grid depth exceeds one, controls data shapes throughout the system.

Grid manages the developmental field, including morphogen concentration tensors, cell occupancy arrays, and neighbor offset tables. In 2D the morphogen tensor has shape (K, N_x, N_y); in 3D it becomes (K, N_x, N_y, N_z).

Genome encodes all developmental parameters as a structured collection of scalars, vectors, and tensors. The genome supports flattening to a continuous vector for optimizer consumption and reconstruction back to structured form, with the flattening length varying by dimensionality.

DevelopmentalSimulation executes the morphogenetic process: morphogen secretion, cross-inhibition, diffusion, cell division, cell differentiation, and axon growth. Each operation reads from and writes to the Grid.

FitnessEvaluation extracts a neural network graph from the final developmental state and computes fitness as the weighted deviation from target graph properties (neuron count, connection count, source count).

Optimizer implements the evolutionary search. The abstract base class supports pluggable strategies; this thesis uses a genetic algorithm (GA) with tournament selection and block-preservation crossover.

ExperimentRunner orchestrates multi-configuration, multi-seed experiments, managing parallel execution, result aggregation, and progress logging.

The data flow proceeds as follows: the ExperimentRunner loads a JSON configuration specifying target graph properties and grid parameters, then invokes the Optimizer. Each fitness evaluation creates a Grid from the Genome, runs the DevelopmentalSimulation for a fixed number of timesteps, and passes the result to FitnessEvaluation. The Optimizer receives fitness scores and produces the next generation of genomes. Figure 3.2 shows the sequence of operations within a single developmental timestep.

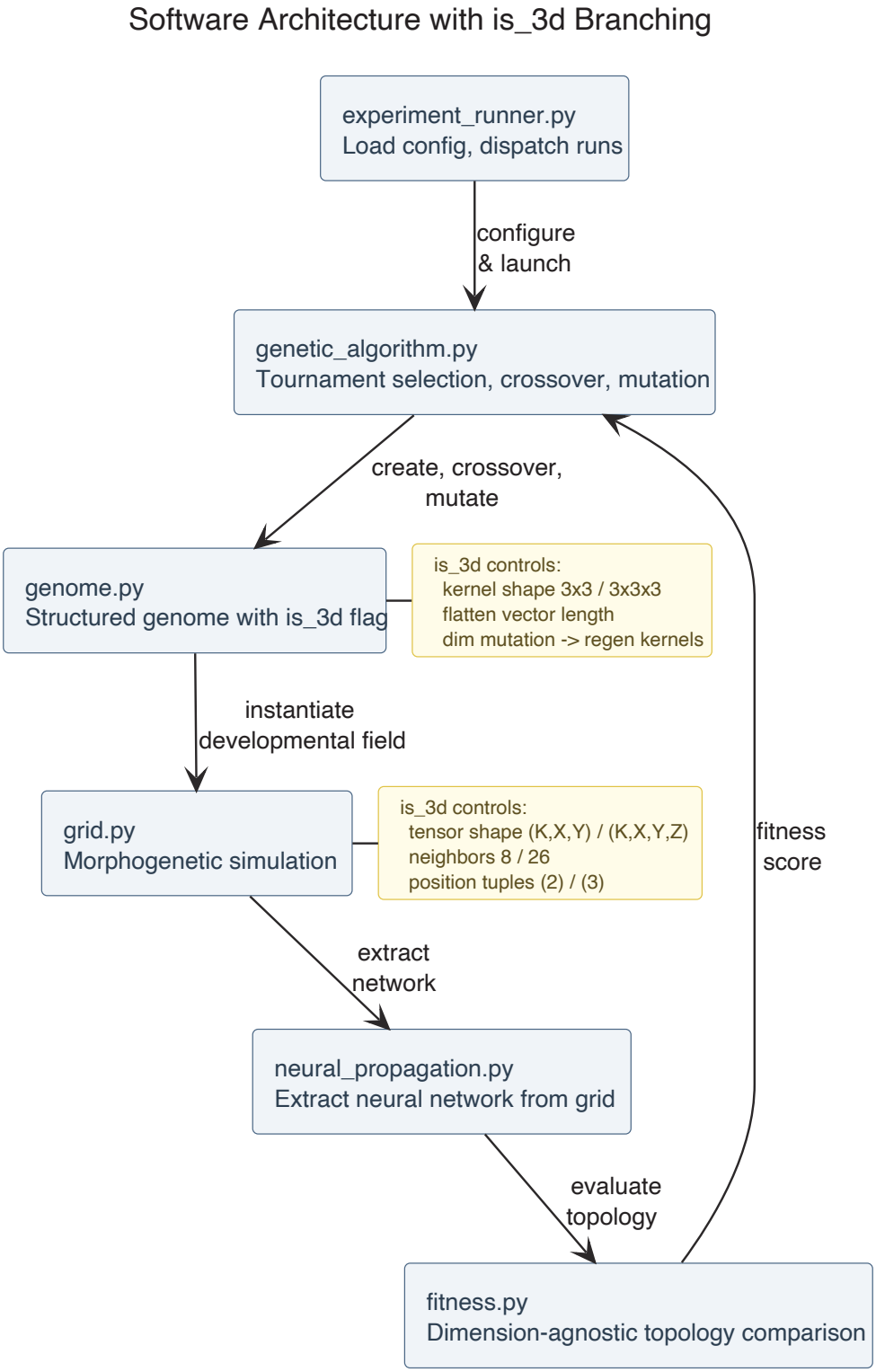


Figure 3.1. Software architecture. The `is_3d` flag controls data shapes in the genome and grid layers.

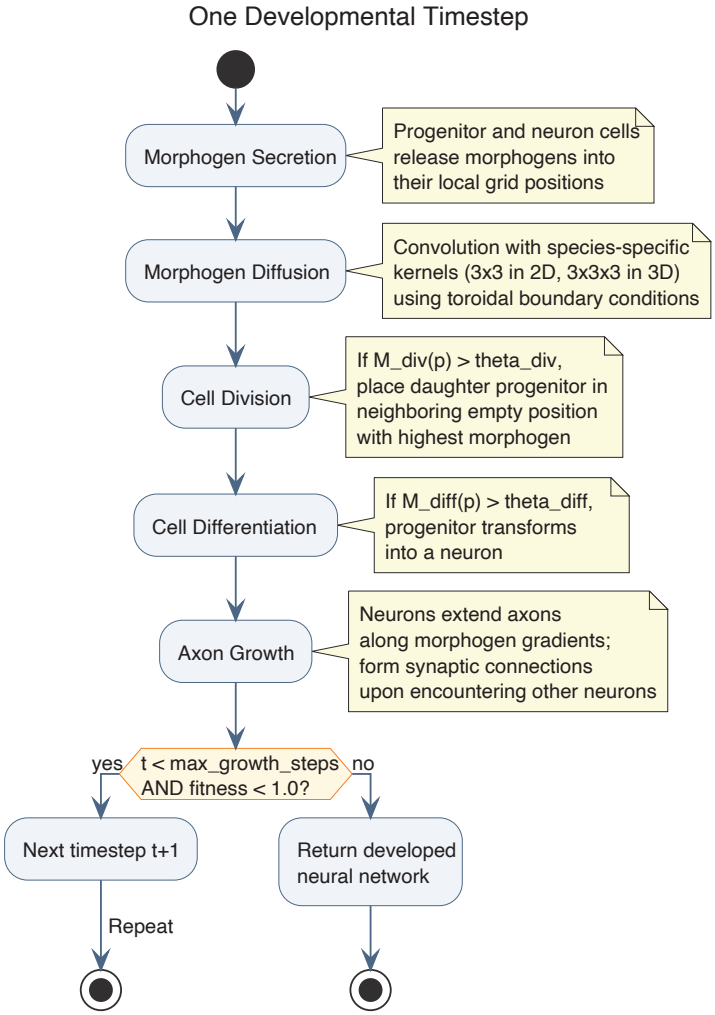


Figure 3.2. One developmental timestep: secretion, diffusion, division, differentiation, and axon growth.

3.3. Technology Stack

The system is implemented in Python, chosen for compatibility with the existing MorphoNAS codebase and the scientific computing ecosystem.

NumPy provides the foundation for all array operations, including morphogen tensors, cell occupancy grids, and vectorized neighbor lookups. The transition from 2D to 3D required no changes to the NumPy-based computation patterns because NumPy natively supports n-dimensional arrays.

SciPy is used for morphogen diffusion via `scipy.ndimage.convolve`, which accepts kernels of arbitrary dimensionality and supports wrapping boundary conditions. This replaced a 2D-specific convolution function from `scipy.signal`, eliminating the need for dimension-dependent branching in the diffusion step.

Manim (manimgl). The 3D visualization pipeline uses the original manimgl version of the Manim animation engine rather than the community fork, for two reasons: native OpenGL-based 3D scene support with built-in camera control, and real-time preview capability essential for iterative visualization development. The community fork primarily targets 2D animations and offers limited support for interactive 3D scenes.

3.4. Design Decisions

Dimensional branching via `is_3d` flag. Rather than creating a separate 3D implementation, a single codebase serves both dimensionalities. A boolean flag, set when the grid depth exceeds one, controls all dimension-dependent behavior: data structure shapes, neighbor offset tables, and diffusion kernel sizes. This ensures bug fixes apply to both modes and guarantees identical code paths for the 2D baseline in comparative experiments. The implementation details are presented in Chapter 4.

Backward-compatible JSON serialization. The genome serialization format was extended with a depth parameter that defaults to one, so existing 2D experiment files load correctly without modification. The binary and flat-vector representations were likewise extended with variable-length encoding (Section 4.2).

Grid size matching. For fair comparison, grid dimensions were chosen to yield approximately equivalent total positions: $32 \times 32 = 1{,}024$ in 2D versus $10 \times 10 \times 10 = 1{,}000$ in 3D. This controls for total developmental capacity while

allowing the neighborhood connectivity difference (8 vs. 26) to be the primary experimental variable.

Self-describing flat genome encoding. The genome is flattened to a continuous vector for optimizer consumption, with scalar parameters first in a fixed order followed by array parameters. The vector length varies by dimensionality; the reconstruction procedure determines genome structure from the vector length and known dimensionality alone, without external metadata. The full genome structure is illustrated in Figure 3.3.

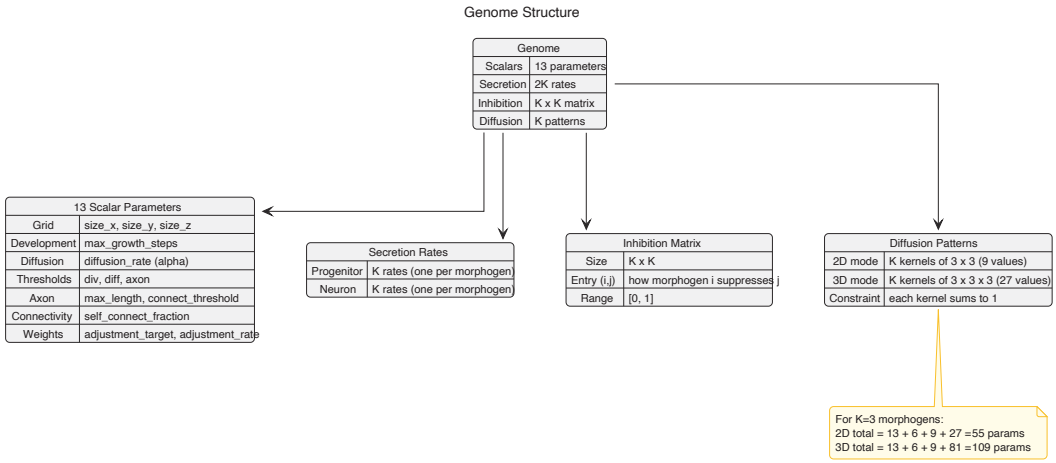


Figure 3.3. Genome structure. For K=3 morphogens: 55 parameters in 2D, 109 in 3D.

3.5. Testing and Validation

Backward compatibility is ensured by design: the single codebase means that 2D experiments execute the same code paths as before the 3D extension, with the `is_3d` flag set to false. All prior 2D experiment configurations were re-run after the extension to verify identical results.

Correctness invariants. The diffusion kernel normalization constraint ($\sum_{i,j,l} D_k[i,j,l] = 1$) is enforced at genome construction time, ensuring mass conservation. Toroidal boundary conditions are verified by checking that cells at grid edges correctly wrap to the opposite side. Neighbor offset generation is validated by confirming that 2D produces exactly 8 offsets and 3D produces exactly 26.

Reproducibility. All experiments use fixed random seeds specified in JSON configuration files. The experiment runner logs the full configuration, random seed, and software version for each run, enabling exact reproduction of any result. Genome checksums (via binary serialization and hashing) provide an additional

verification layer.

4. Implementation

This chapter details the mathematical formalization and engineering implementation of the three-dimensional extension to MorphoNAS. The developmental process, genome representation, evolutionary search, and visualization pipeline are described in turn.

4.1. Extension to Three-Dimensional Development

Moving from a flat grid to a volumetric one changes more than just the shape of the developmental field. It alters how cells perceive their surroundings, how chemical signals spread, and ultimately what network topologies can emerge. We formalize these changes below.

The developmental field is defined as a discrete lattice $\Omega \subset \mathbb{Z}^d$ where $d \in \{2,3\}$ denotes the spatial dimensionality. For three-dimensional development, the domain takes the form $\Omega = \{0,1,\ldots,N_x-1\} \times \{0,1,\ldots,N_y-1\} \times \{0,1,\ldots,N_z-1\}$ with toroidal boundary conditions implemented through modular arithmetic:

$$(x,y,z) \sim (x \bmod N_x,\ y \bmod N_y,\ z \bmod N_z) \tag{4.1}$$

This periodic boundary eliminates edge effects that would otherwise introduce artificial spatial biases into the developmental process. For comparative experiments, grid dimensions are chosen to yield approximately equivalent total positions: the two-dimensional configuration uses a 32×32 grid providing 1,024 positions, while the three-dimensional configuration uses a $10 \times 10 \times 10$ grid providing 1,000 positions.

The transition to three dimensions fundamentally alters the local connectivity available to developing cells. In two dimensions, the Moore neighborhood of a position $\mathbf{p}$ comprises all positions within Chebyshev distance one:

$$\mathcal{N}_8(\mathbf{p}) = \{\mathbf{q} \in \Omega : \|\mathbf{p}-\mathbf{q}\|_\infty = 1,\ \mathbf{q} \neq \mathbf{p}\} \tag{4.2}$$

yielding $|\mathcal{N}_8| = 8$ neighbors. The three-dimensional generalization extends this to

a cubic neighborhood:

$$\mathcal{N}_{26}(\mathbf{p}) = \{\mathbf{q} \in \Omega : \|\mathbf{p} - \mathbf{q}\|_\infty = 1,\ \mathbf{q} \neq \mathbf{p}\} \tag{4.3}$$

with $|\mathcal{N}_{26}| = 26$ neighbors, a 3.25-fold increase in local connectivity. This expansion has profound implications for developmental dynamics, since progenitor cells have more candidate division sites, differentiating neurons can establish connections with a larger pool of neighbors, and morphogen gradients propagate through a richer spatial structure.

Morphogen concentrations are represented as a tensor $\mathbf{M} \in \mathbb{R}^{K \times N_x \times N_y \times N_z}$ where K denotes the number of distinct morphogen species. Each morphogen k diffuses according to a species-specific kernel $\mathbf{D}_k \in \mathbb{R}^{3\times3\times3}$ subject to the normalization constraint $\sum_{i,j,l} D_k[i,j,l] = 1$ to ensure mass conservation. The diffusion update at each developmental timestep follows:

$$M_k^{t+1} = (1-\alpha)M_k^t + \alpha\left(M_k^t * \mathbf{D}_k\right) \tag{4.4}$$

where $*$ denotes convolution with periodic boundary conditions and $\alpha \in [0,1]$ is the diffusion rate controlling the balance between retention and spreading. When $\alpha = 0$, concentrations remain stationary; when $\alpha = 1$, the field is entirely replaced by the convolved result. The implementation employs `scipy.ndimage.convolve` with `mode='wrap'` to handle toroidal boundaries efficiently.

4.2. Genome Representation and Parameter Space

The genome encodes all parameters governing the developmental process as a structured collection of scalars, vectors, and tensors. A binary flag `is_3d` is set when $N_z > 1$, triggering three-dimensional behavior throughout the system. Key parameters include grid dimensions (N_x, N_y, N_z), diffusion rate (α), thresholds for division, differentiation, and axon growth $(\theta_{\text{div}}, \theta_{\text{diff}}, \theta_{\text{axon}})$, maximum axon length $(L_{\max})$, and morphogen count (K).

The diffusion kernels constitute the largest component of the genome. In two dimensions, each morphogen requires 9 coefficients for its 3×3 kernel; in three dimensions, this increases to 27 coefficients for the $3 \times 3 \times 3$ kernel. For K morphogens, the three-dimensional representation thus requires $18K$ additional

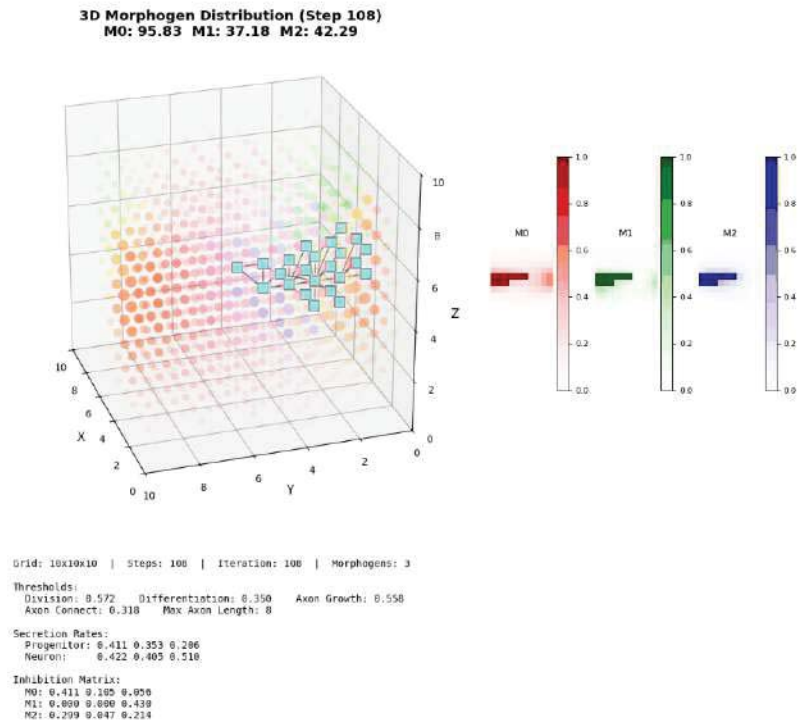


Figure 4.1. 3D morphogen distribution with neurons and connections on a $10 \times 10 \times 10$ grid.

parameters compared to the two-dimensional case. Additionally, the genome encodes morphogen secretion rates for each cell type (progenitor and neuron), yielding $2K$ secretion parameters, and a $K \times K$ cross-inhibition matrix governing morphogen interactions.

For evolutionary optimization, the genome is flattened into a continuous vector suitable for derivative-free optimization algorithms. Integer parameters are normalized to the unit interval via linear scaling, while array parameters are concatenated in a fixed order: scalar parameters, progenitor secretion rates, neuron secretion rates, inhibition matrix elements, and finally diffusion pattern coefficients. This flattening enables the application of continuous optimization methods while the reconstruction process ensures valid discrete parameters through appropriate rounding and clamping. The full genome structure is shown in Figure 3.3.

4.3. Developmental Dynamics in the Extended Framework

The developmental process unfolds through iterative application of cell behaviors governed by local morphogen concentrations. At each timestep, cells

evaluate their chemical environment and execute appropriate responses according to genomically-encoded thresholds.

Cell division occurs when the local concentration of the division morphogen exceeds the division threshold: $M_{\text{div}}(\mathbf{p}) > \theta_{\text{div}}$. Upon meeting this condition, the progenitor cell at position $\mathbf{p}$ selects a target position from its neighborhood according to:

$$\mathbf{p}^* = \arg \max_{\mathbf{q} \in \mathcal{N}(\mathbf{p}),\, \text{empty}(\mathbf{q})} M_{\text{div}}(\mathbf{q}) \tag{4.5}$$

where $\text{empty}(\mathbf{q})$ indicates that position $\mathbf{q}$ contains no cell. The daughter progenitor is placed at $\mathbf{p}^*$, inheriting the same developmental potential as its parent. In three-dimensional development, the expanded 26-cell neighborhood provides substantially more candidate positions for division compared to the 8-cell neighborhood in two dimensions, potentially enabling faster tissue growth and more varied spatial arrangements.

Differentiation transforms progenitor cells into neurons capable of forming synaptic connections. This transition is triggered when the differentiation morphogen concentration exceeds the differentiation threshold: $M_{\text{diff}}(\mathbf{p}) > \theta_{\text{diff}}$. Unlike division, differentiation depends only on the local concentration at the cell's position rather than on neighborhood properties. The mechanism operates identically in two and three dimensions, ensuring that the spatial complexity introduced by volumetric development does not alter the fundamental biochemical logic of cell fate determination.

Neurons extend axons through chemotactic guidance along the axon morphogen gradient. At each growth step, the axon tip moves toward the neighboring position with highest morphogen concentration, provided it exceeds the growth threshold θ_{axon}. Growth continues until the axon reaches its maximum length L_{max} or encounters another neuron. In the latter case, a synaptic connection forms with weight determined by the local morphogen concentration and the Euclidean distance between the connected neurons. The 26-directional choice available in three-dimensional development enables substantially richer connectivity patterns than the 8-directional choice in two dimensions, potentially yielding network topologies with greater architectural diversity. The full developmental timestep sequence is shown in Figure 3.2.

4.4. Evolutionary Search with a Genetic Algorithm

The search for optimal genomes employs a custom genetic algorithm (GA) with tournament selection, block-preservation crossover, and multi-type mutation operators. The framework is built around an abstract `Optimizer` base class with pluggable strategy components, allowing different optimization strategies to be used without changing the rest of the pipeline. All experiments in this thesis use the GA because it operates on structured genomes directly, preserving functionally coupled parameter groups during crossover, which is more appropriate when genome parameters have strong interdependencies (e.g., spatial dimensions must remain consistent).

Each generation proceeds through four stages. First, parents are selected via tournament selection, where a small subset of the population is sampled and the fittest individual is chosen as a parent. Second, pairs of parents undergo block-preservation crossover, which groups functionally related parameters (grid width, height, depth, and maximum growth steps) into an atomic block inherited from a single parent, preventing offspring with mismatched dimensional parameters. Third, offspring are mutated through several operators: parameter mutation (small perturbations to scalar values), morphogen count mutation, diffusion matrix mutation, and growth step mutation, each with configurable probability. Fourth, the developmental simulation runs for each individual, producing a neural network whose fitness is evaluated against the target specification (detailed in Section 5.1).

Evolutionary parameters were chosen based on preliminary experimentation. The population size of 50 individuals balances exploration breadth against computational cost. Evolution proceeds for a maximum of 500 generations, with early termination when fitness reaches 1.0 (perfect match). Elitism preserves the best individual across generations to prevent regression.

4.5. Engineering the 2D-to-3D Extension

While the mathematical formulation of the three-dimensional extension was presented in Section 4.1, this section describes the practical software engineering work needed to implement it. The changes span the core simulation, genome representation, evolutionary operators, and experiment infrastructure.

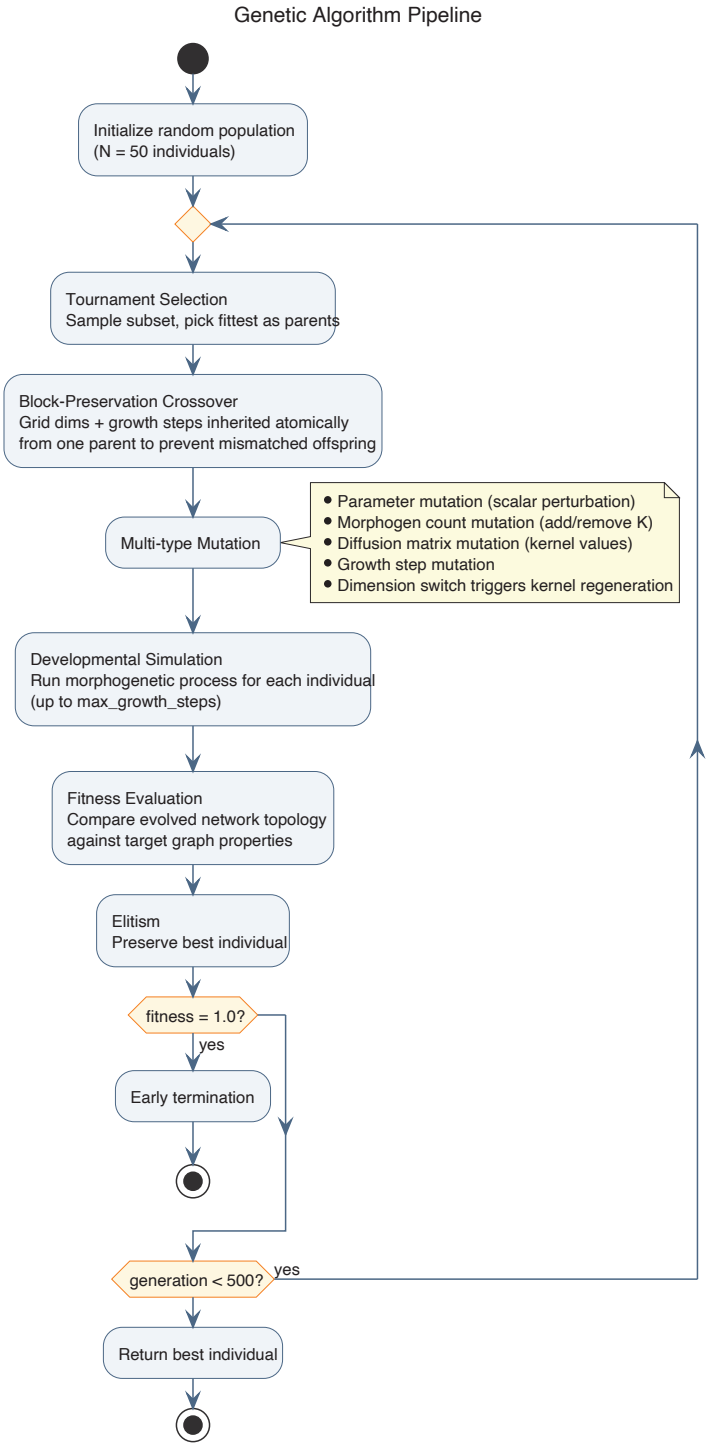


Figure 4.2. Genetic algorithm pipeline with early termination on perfect fitness.

4.5.1. Diffusion and Neighbor Operations

The morphogen diffusion step required swapping the 2D-only convolution function from SciPy's signal processing module for a more general one from SciPy's n-dimensional image processing module. The replacement accepts kernels of any dimensionality and supports wrapping boundaries, so the same code works for both 3×3 and $3 \times 3 \times 3$ diffusion kernels without any branching.

The neighborhood calculation, by contrast, did need explicit branching. In two dimensions, the Moore neighborhood has 8 offsets; in three dimensions this grows to 26, generated by iterating over all combinations of $\{-1, 0, 1\}$ across three axes and excluding the origin. The cell division operation, which is the most performance-sensitive part, uses NumPy broadcasting to avoid slow Python loops: all eligible cell positions are combined with the offset array at once to produce a full table of neighbor coordinates, and modular arithmetic handles toroidal wrapping. Array indexing then gathers morphogen concentrations and occupancy for all neighbors of all eligible cells in one step. This vectorized approach keeps the 3D extension fast despite the 3.25-fold increase in neighbors per cell.

4.5.2. Genome Serialization

The genome gained a depth parameter alongside width and height, increasing the total number of evolvable parameters. The biggest expansion is in the diffusion kernels, where each morphogen species needs 9 coefficients for a 3×3 kernel in 2D but 27 for the $3 \times 3 \times 3$ kernel in 3D. With three morphogen species, that means 54 extra parameters in the genome.

Saving and loading the genome required changes at multiple levels. The binary format (used for hashing and checksums) was extended with an extra integer field and conditional packing for the diffusion patterns. The JSON format added the depth parameter with a default value of one for backward compatibility, so existing 2D experiment files still load correctly. The trickiest part was the flattening operation that converts the structured genome into a flat vector for the optimizer, since the diffusion section now has a variable length. Going the other way, converting a flat vector back into a structured genome, requires knowing the dimensionality before the number of morphogens can be determined. The solution

reads the depth parameter first, uses it to decide the pattern size (9 or 27), and then solves a quadratic equation to figure out how many morphogens the vector encodes.

4.5.3. Evolutionary Operator Adaptations

Grid dimensionality is fixed for the entire evolutionary run: all genomes in a population share the same grid depth, set at initialization. Diffusion kernels are sized accordingly (3×3 for 2D, $3 \times 3 \times 3$ for 3D) and remain consistent throughout evolution.

Crossover operators group the three spatial dimensions (width, height, depth) together with the maximum growth steps as a "Size and Growth" block that always comes from one parent, preventing offspring from getting mismatched dimensional parameters. The diversity maintenance strategy, which replaces weak individuals with random genomes to avoid premature convergence, was also updated to pass the depth parameter to the random genome generator, so replacement genomes match the population's dimensionality.

4.5.4. Experiment Infrastructure

A dedicated script was created to generate three-dimensional experiment configurations. It produces randomized target graphs (varying numbers of neurons, connections, and source nodes) and pairs them with 3D grid parameters: a $10 \times 10 \times 10$ grid with 1,000 positions, chosen to be comparable to the 32×32 grid (1,024 positions) used in 2D experiments. Each configuration is saved as a JSON file that the experiment runner loads at runtime.

The experiment runner itself needed several 3D-specific changes. The existing 2D morphogen display, which shows concentration fields as a colored heatmap, cannot handle three-dimensional data and was disabled for 3D grids. Two new visualization modes replaced it: 3D scatter plots showing each morphogen species as a separate colored point cloud, and 2D projection views that average concentrations along each axis to produce XY, XZ, and YZ heatmaps. Progress logging was updated to show dimensions in the right format, and the convergence strategy gained a minimum fitness threshold to avoid stopping too early on zero-fitness runs, which happen more often in 3D because the larger parameter space makes bad

initializations more likely.

4.6. 3D Visualization Engineering

Visualizing three-dimensional morphogenetic development presents significant engineering challenges that required custom tooling. While two-dimensional developmental fields can be rendered straightforwardly using standard plotting libraries, volumetric morphogen gradients with simultaneous cell positioning and connectivity demand specialized 3D animation capabilities.

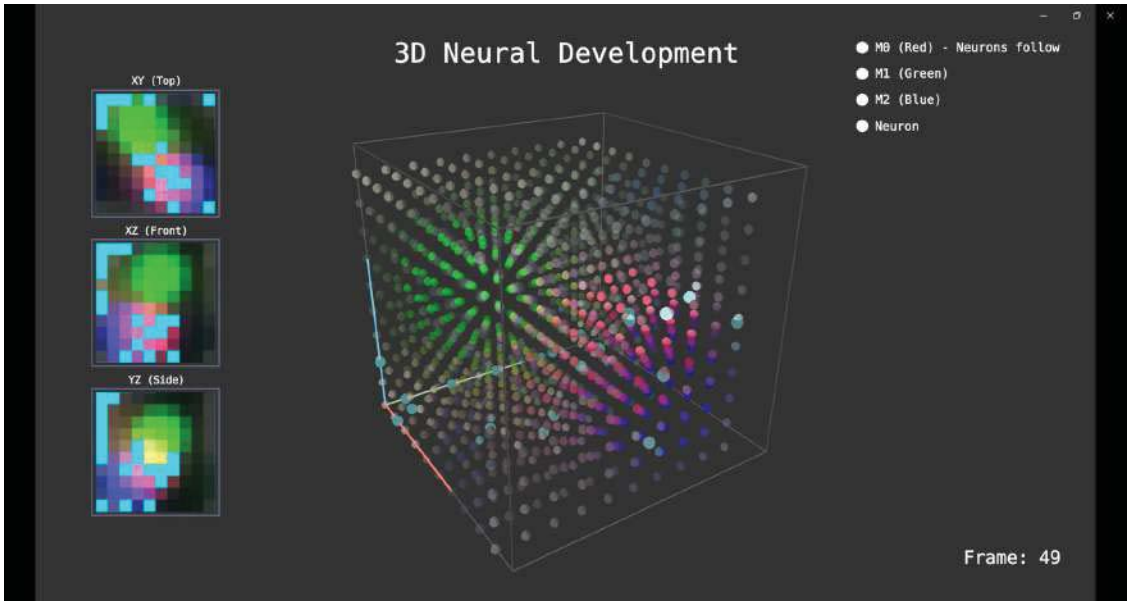


Figure 4.3. 3D morphogenetic development visualization with morphogen concentrations, neurons, and projection panels.

4.6.1. Manim Library Challenges

The visualization system was built using Manim, the mathematical animation engine originally developed by Grant Sanderson for the 3Blue1Brown educational mathematics videos. Two versions of Manim exist: the original `manimgl` (OpenGL-based) and the community fork `manim` (Cairo-based). Neither version was designed for scientific volumetric visualization, presenting several obstacles.

The community edition suffered an infrastructure compromise in which the project's GitHub organization, Discord server, and social media accounts were hijacked, raising concerns about the integrity of community-maintained releases. This, together with the original `manimgl` version's native three-dimensional scene

support with built-in camera control and OpenGL-accelerated rendering, motivated our adoption of `manimgl` despite its less extensive documentation. The community fork primarily targets two-dimensional animations and offers limited support for interactive 3D scenes. The OpenGL renderer also enables real-time preview, which proved essential during iterative visualization development where rapid feedback on camera angles and object placement significantly reduced the design cycle.

The manimgl API differs substantially from the community edition's widely-documented tutorials, creating a steep learning curve. Key differences include how objects are pinned to screen coordinates so they do not rotate with the 3D camera, how numeric trackers drive animation progress, and how objects receive per-frame update functions. These differences meant that most community-maintained examples were not directly applicable, requiring extensive experimentation with the less-documented original API.

Several limitations of Manim required creative workarounds. Manim does not support split-screen or multi-viewport layouts, since the entire scene shares a single camera. Showing flat projection panels alongside the 3D view required generating image textures and pinning them to fixed screen positions, effectively faking a split-screen layout within the single-camera design. Another challenge was that transitioning between discrete evolutionary steps produced jarring visual jumps where cells would teleport to new positions between frames. Achieving smooth animation required separating the simulation from rendering: trajectories are pre-computed for all cells, and a numeric tracker smoothly interpolates positions between discrete steps using cubic spline curves. Rendering performance with large numbers of 3D objects also posed a constraint. Each sphere at default resolution generates roughly 200 triangles; placing 1,000 spheres for every grid position would produce 200,000 triangles per frame, causing severe slowdown. The solution reduced sphere detail levels for morphogen dots and neuron objects separately, trading some visual quality for usable frame rates.

A further challenge arose when multiple morphogen species overlap in the same region. With three or more morphogens shown as colored spheres, areas of high combined concentration became visually opaque and indistinguishable, making it impossible to see individual gradients. The solution was to apply selective transparency: positions where one morphogen clearly dominates receive higher opacity, while positions with roughly equal concentrations from multiple mor-

phogens are made more transparent. This allows viewers to see through crowded regions and identify the distinct colored gradients underneath, keeping the spatial structure of each morphogen field readable even in dense parts of the developmental field.

4.6.2. Volumetric Morphogen Rendering

Rendering morphogen concentration fields in 3D required representing up to 1,000 spatial positions ($10 \times 10 \times 10$ grid) with RGB color encoding for three morphogen species. A naïve approach creating individual sphere objects for each position proved computationally prohibitive for animation.

The solution employed an object pooling strategy where a fixed pool of sphere primitives is pre-allocated at scene construction, then dynamically repositioned, recolored, and opacity-adjusted each frame based on current morphogen concentrations. Positions with total concentration below a threshold have their spheres hidden (opacity set to zero) rather than removed, avoiding expensive object creation and destruction during animation.

Color blending presented additional challenges. Direct RGB mapping from morphogen concentrations produced muddy brown regions where multiple morphogens overlapped. The implemented solution applies a power-curve transformation ($c^{0.7}$) to emphasize dominant colors, followed by saturation boosting that pushes non-dominant channels toward the average, maintaining vivid distinct hues even in regions of morphogen overlap.

Raw morphogen concentrations can vary by several orders of magnitude, so visualizing them directly is not useful since a few extreme values wash out everything else. To address this, values are divided by the 98th percentile rather than the global maximum. This keeps the relative spatial structure of gradients visible and ensures good contrast across most of the field, with only the most extreme outliers reaching full brightness.

During longer animations, diffusion gradually blends all morphogen species into a uniform mixture, producing a featureless brown field with no useful information. To prevent this, the visualization uses moving injection sources that continuously emit one morphogen at a time from bouncing positions within the grid, keeping distinct colored regions visible throughout the animation. A 5%

per-frame decay keeps concentrations from building up endlessly while letting the injected gradients last long enough to be seen clearly.

4.6.3. Projection Panels

To provide interpretable 2D views alongside the 3D visualization, orthogonal projection panels (XY, XZ, YZ planes) are rendered as texture-mapped image objects. These panels display morphogen concentrations averaged along each axis, with cell positions overlaid as colored markers. The projections update every fifth frame to balance visual feedback against rendering performance, using temporary image files with unique timestamps to circumvent Manim's texture caching behavior.

Each projection panel is generated as an image using the Python Imaging Library. The three-dimensional morphogen data is averaged along one axis (for example, averaging over Z to produce a top-down view), producing a two-dimensional heatmap. The first three morphogen species are mapped to the red, green, and blue channels, giving a compact view of all three in a single image. Cell positions are drawn on top as colored markers: cyan for neurons and yellow for progenitor cells, so viewers can quickly find where cells are within the morphogen landscape.

Because Manim caches loaded textures by file path, reusing the same filename when updating a panel causes the renderer to show old data from the previous frame. The workaround is to append the current frame number to each filename, which forces the engine to load the new version. Old files are deleted after a short delay to avoid filling up disk space during long animations.

The projection panels are placed at fixed screen positions so they stay still as the camera rotates around the 3D scene. This creates a heads-up display alongside the main view, letting the viewer compare flat cross-sections with the rotating volume at the same time. The three projections (top, front, and side views) are stacked vertically along the left edge of the frame, giving complementary perspectives on the morphogen distribution without blocking the 3D view.

4.6.4. Neuron Migration Animation

Animating neuron migration required smooth interpolation between discrete grid positions. Each cell object maintains a target position updated when the underlying simulation moves the neuron. A per-frame updater function interpolates between current and target positions using linear interpolation with a configurable rate factor, producing fluid motion even when neurons traverse multiple grid cells between rendered frames.

The migration direction is guided by morphogen gradients, specifically following the concentration gradient of M0 (encoded as red). A momentum system accumulates directional bias over time, preventing erratic oscillation and producing biologically plausible drift patterns toward high-concentration regions.

An improved version of the migration animation uses cubic spline curves to smoothly connect discrete grid positions instead of straight-line interpolation. This produces paths with continuous direction changes, removing the visible sharp turns that occur at grid boundaries when using linear interpolation. The spline needs four nearby points to calculate each curved segment; at the start and end of a trajectory where not enough points are available, the missing ones are estimated by extending the direction of the nearest known segment.

Rather than computing migration during rendering, which would tie the simulation to the frame rate, complete trajectories for all neurons are calculated across all simulation steps before the animation begins. A time tracker maps the animation clock to trajectory progress, and each neuron's position is interpolated along its pre-computed path every frame. Separating the simulation from the rendering this way ensures consistent visual quality regardless of how fast the frames are drawn, and lets the same trajectory data be replayed at different speeds without changing the simulation results.

The migration model combines gradient following with random exploration and momentum. At each step, candidate positions in the 26-cell neighborhood are scored by two things: the morphogen concentration at the target position, which pulls neurons toward regions of strong chemical signal, and how well the proposed move aligns with the previous movement direction, which biases neurons to keep drifting the same way. The scores are turned into probabilities and sampled randomly, producing drift patterns that generally head toward high-concentration

areas while still showing the kind of random variation seen in real biological neural migration.

5. Experimental Methodology

This chapter describes the experimental design used to evaluate the 3D extension, including the benchmark task, developmental environments, and statistical analysis framework.

5.1. Benchmark Task: Graph Property Targeting

The primary benchmark task evaluates the system's ability to evolve neural network architectures matching specific topological properties. Following the MorphoNAS validation approach [3], the fitness function measures how closely the evolved network matches target graph properties, specifically the number of neurons (N_{target}), the number of connections (E_{target}), and the number of input neurons without incoming connections (S_{target}). This structural targeting task provides a controlled evaluation of the developmental process independent of downstream task performance, enabling direct comparison of 2D and 3D morphogenetic dynamics.

The fitness function computes a weighted sum of normalized deviations from each target:

$$f = 1 - \frac{1}{3} \sum_{i \in \{N,E,S\}} \frac{|x_i - x_i^*|}{\tau_i} \tag{5.1}$$

where x_i is the achieved value, x_i^* is the target, and τ_i are tolerance parameters ($\tau_N = 5$, $\tau_E = 25$, $\tau_S = 2$). A fitness of 1.0 indicates perfect matching.

5.2. Developmental Environments

The core experimental comparison evaluates morphogenetic development in two spatial environments:

2D Environment: A 32×32 toroidal grid providing 1,024 spatial positions. Each cell has 8 neighbors (Moore neighborhood), and morphogen diffusion follows a 3×3 diffusion kernel. This configuration replicates the original MorphoNAS setup [3]. Figure 5.1 illustrates the developmental field with morphogen gradients and cellular connectivity.

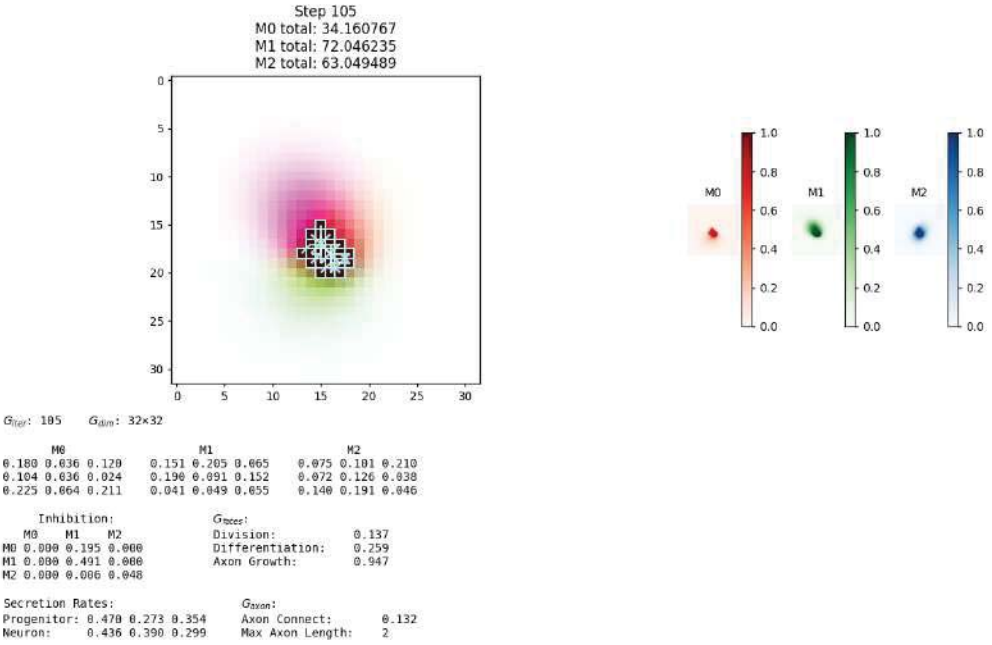


Figure 5.1. 2D developmental field (32×32) showing morphogen concentrations and neural connectivity.

3D Environment: A $10 \times 10 \times 10$ toroidal grid providing 1,000 spatial positions (comparable volume to 2D). Each cell has 26 neighbors (3D Moore neighborhood), and morphogen diffusion follows a $3 \times 3 \times 3$ diffusion kernel. The increased neighborhood connectivity fundamentally alters the developmental dynamics by providing more potential interaction pathways. Figure 5.2 shows orthogonal projections of the 3D morphogen field.

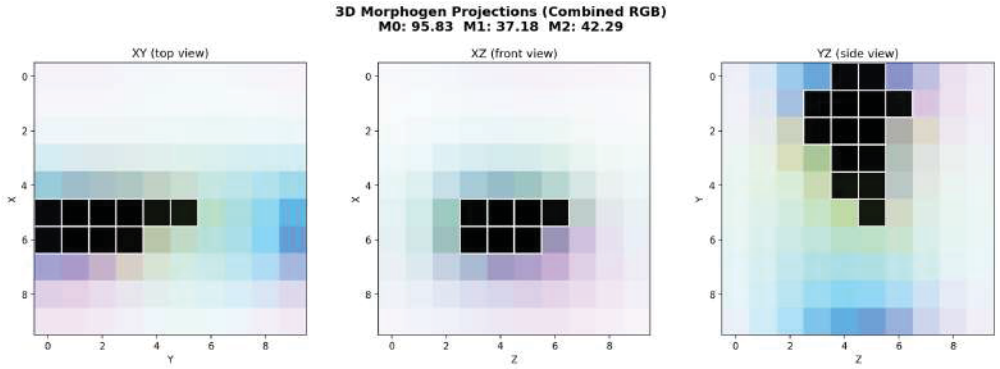


Figure 5.2. Orthogonal projections (XY, XZ, YZ) of 3D morphogen fields with cell positions.

Both environments use identical evolutionary parameters: population size of 50, maximum 500 generations, and 3 morphogen types. To ensure a fair comparison, all genome parameters were initialized from fully open ranges ($[0, 1]$), avoiding any bias toward either dimension from previously tuned initialization values (see Section 6.6). Fifty target configurations with varying complexity (12–25

neurons, 19–167 connections) were each evaluated with 10 independent runs per environment, yielding 1,000 total runs (500 2D, 500 3D) and 292.7 hours of total computation. Non-parametric statistical tests (Mann-Whitney U [10], Fisher's exact [11]) were used given the non-normal distributions, with significance assessed at $\alpha = 0.05$.

5.3. Statistical Analysis Framework

Given the stochastic nature of evolutionary optimization and the non-normal distributions typically observed in fitness outcomes, the experimental analysis employs non-parametric statistical methods throughout.

Comparison of continuous metrics between two-dimensional and three-dimensional development uses the Mann-Whitney U test, [10] a rank-based test that assesses whether observations from one group tend to be larger than observations from the other without assuming any particular distributional form. For binary outcomes such as success rates (achieving perfect fitness) and failure rates (fitness below a threshold), Fisher's exact test [11] provides exact p-values for 2×2 contingency tables without relying on asymptotic approximations that may be unreliable with moderate sample sizes.

Effect sizes are quantified using Cliff's delta, [12] a non-parametric measure that estimates the probability that a randomly selected observation from one group exceeds a randomly selected observation from the other group, minus the reverse probability. Values range from -1 to $+1$, with magnitudes below 0.147 considered negligible, 0.147–0.33 small, 0.33–0.474 medium, and above 0.474 large. For location estimates, 95% bootstrap confidence intervals are computed for medians using 10,000 resamples. All hypothesis tests use a significance threshold of $\alpha = 0.05$.

6. Results and Analysis

The experimental comparison encompasses 1,000 independent runs (500 2D, 500 3D) across 50 target configurations with 10 random seeds each, representing 292.7 hours of computation and 8.5 million fitness evaluations.

6.1. Outcome Distribution

Runs were classified into three outcome categories based on final fitness: *success* (fitness ≥ 0.95), *failure* (fitness < 0.1, indicating initialization failure where development never produces a viable network), and *stagnation* ($0.1 \leq$ fitness < 0.95, where evolution finds a partial solution but cannot reach the target). Three-dimensional development achieved a higher success rate (68.6% vs. 60.0%), a substantially lower stagnation rate (19.8% vs. 30.6%), and no statistically significant difference in failure rate (11.6% vs. 9.4%).

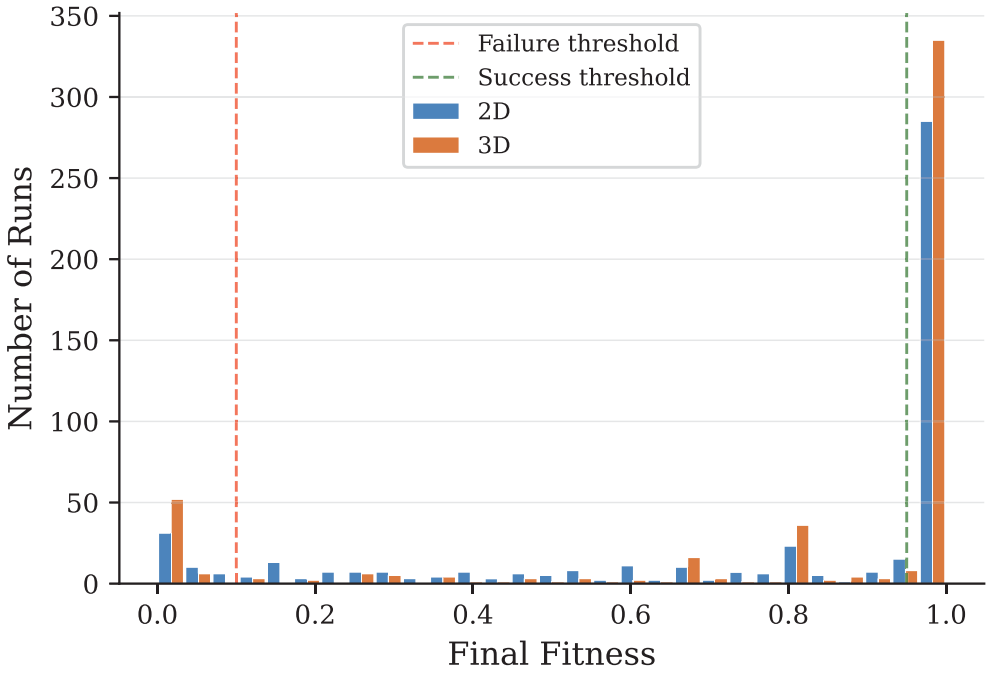


Figure 6.1. Final fitness distribution. Dashed lines mark failure (< 0.1) and success (≥ 0.95) thresholds. Both conditions exhibit bimodal distributions, but 3D shows fewer stagnated runs at intermediate fitness values.

Figure 6.2 quantifies these differences directly, showing all four outcome rate comparisons with statistical significance annotations. The overall success rate

difference is significant ($p = 0.006$), while the stagnation rate difference is highly significant ($p = 0.0001$). Failure rates show no significant difference ($p = 0.302$).

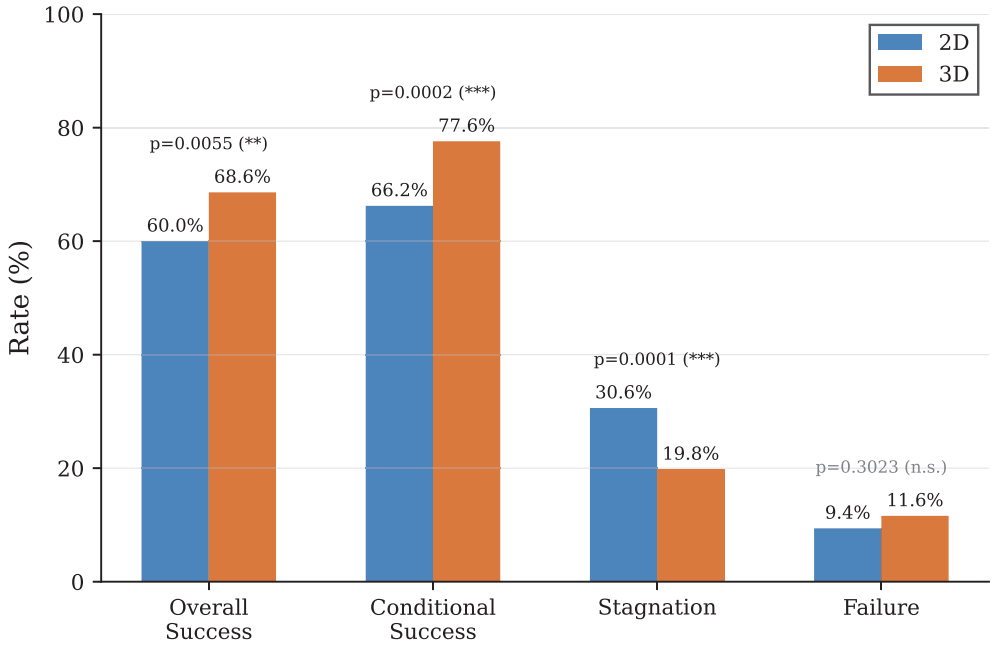


Figure 6.2. Outcome rate comparison with Fisher's exact test p-values. 3D shows significantly higher success and lower stagnation, with no significant difference in failure rate.

The per-configuration breakdown in Figure 6.3 shows that 3D equals or exceeds 2D success rates across the majority of the 50 target configurations.

6.2. Conditional Analysis

The bimodal fitness distributions make aggregate comparisons (mean or median fitness) misleading, as low-fitness failures and high-fitness successes cancel out in rank-based statistics. We therefore decompose the comparison into two independent questions: (1) does 3D fail at initialization more often than 2D? and (2) among runs that do not fail, does 3D reach the target more often?

Failure rates are not significantly different (9.4% vs. 11.6%, Fisher's exact, $p = 0.302$), indicating that 2D and 3D are equally susceptible to initialization failures under unbiased parameter ranges. Among non-failed runs, however, 3D achieves success significantly more often, at 77.6% vs. 66.2% (Fisher's exact, $p = 0.0002$). This 11.4 percentage point advantage represents the core finding of the experiment. When both environments successfully initialize, 3D discovers target architectures more reliably.

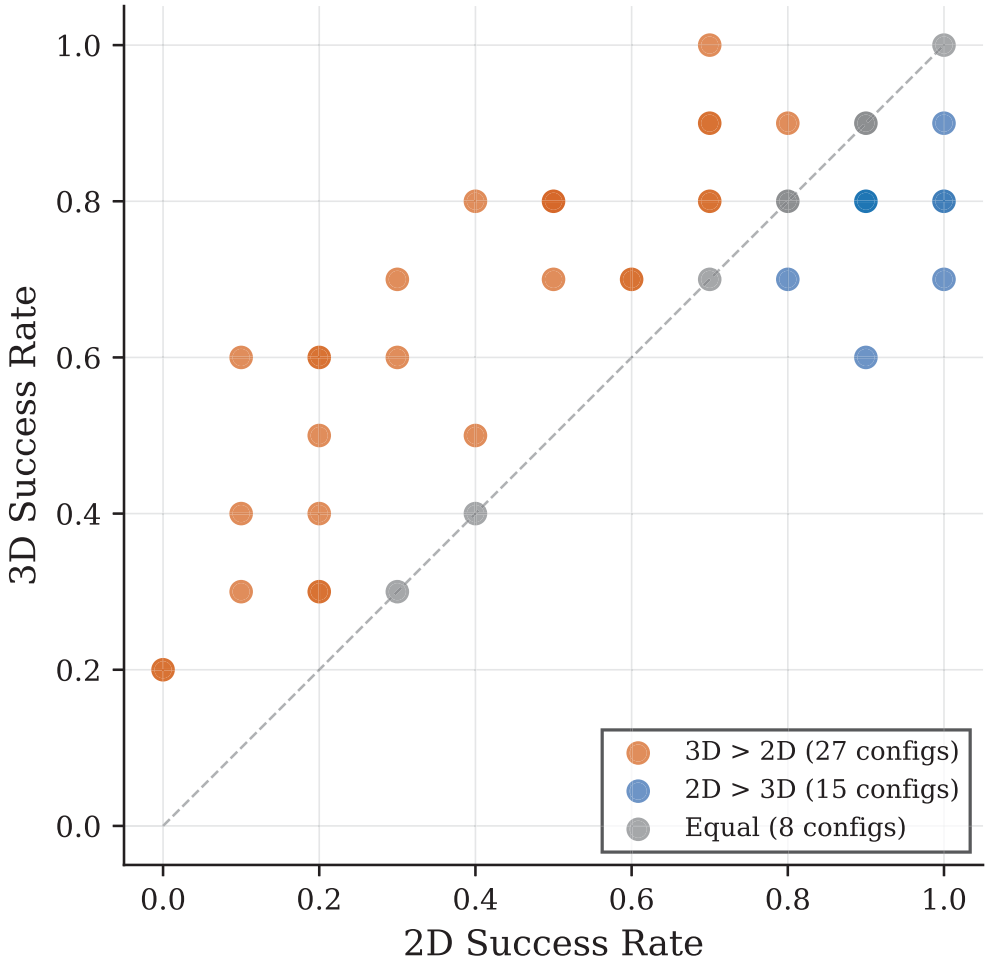


Figure 6.3. Per-configuration success rate. Each point is one of 50 target configurations. Points above the diagonal indicate 3D outperforming 2D.

6.3. Convergence and Stagnation

Figure 6.4 shows that 3D converges faster, with the mean trajectory reaching higher fitness earlier in the optimization. Among successful runs, 3D required significantly fewer generations (mean 114 vs. 138, Mann-Whitney U, $p = 0.048$), though wall-clock time to success was similar (mean 516s vs. 511s, $p = 0.975$) due to the slightly higher per-evaluation cost of 3D convolutions (0.131s vs. 0.119s per evaluation).

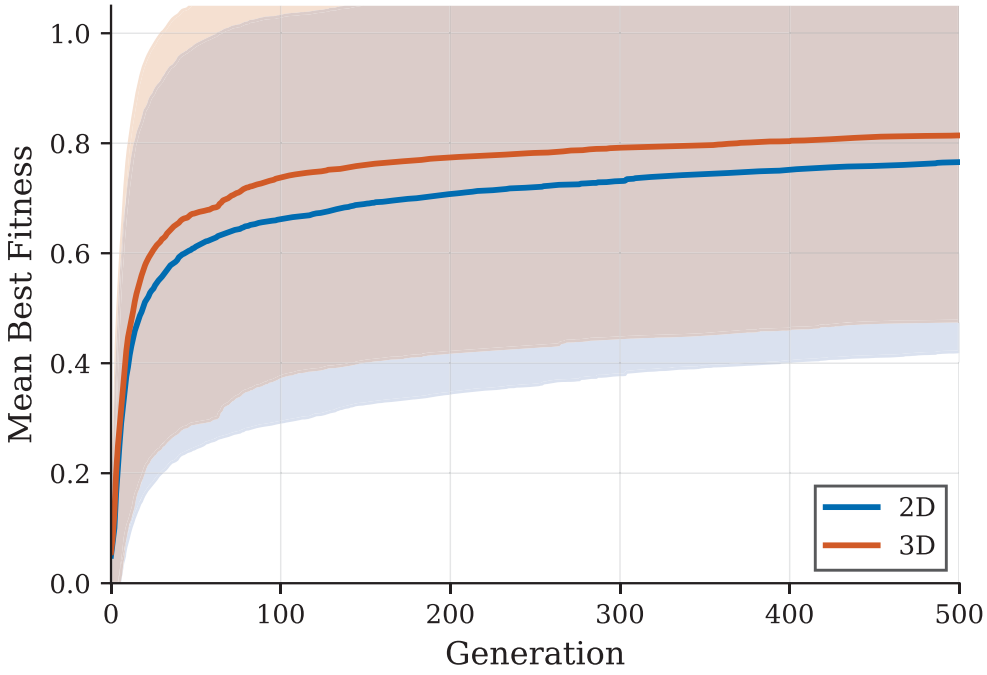


Figure 6.4. Mean convergence trajectories over generations with standard deviation shading.

The stagnation rate difference (30.6% vs. 19.8%, $p = 0.0001$) is the most pronounced behavioral divergence between the two environments. Rather than becoming trapped at intermediate fitness values, 3D runs tend to either converge decisively or fail at initialization (Figure 6.5), suggesting that the 26-neighbor connectivity creates sharper fitness gradients that prevent stagnation in local optima.

6.4. Computational Efficiency

Figure 6.6 compares computational cost across all 1,000 runs. The per-evaluation cost of 3D development is modestly higher than 2D due to the larger convolution kernel ($3 \times 3 \times 3$ vs. 3×3): mean 0.131s vs. 0.119s per fitness evaluation, approximately 10% overhead. Despite this, 3D runs are faster overall: mean

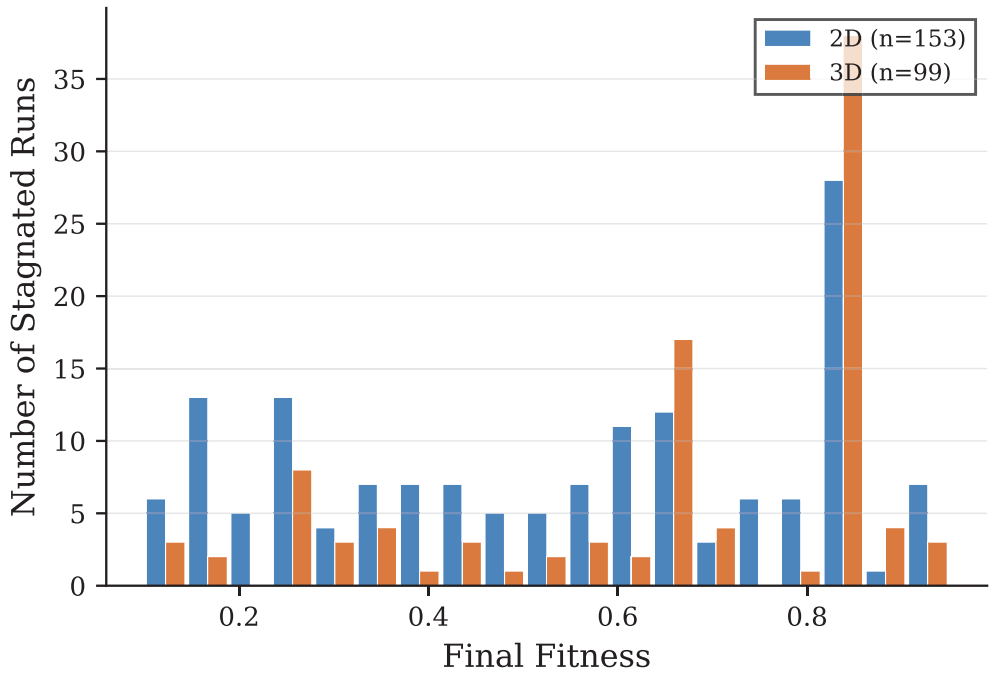


Figure 6.5. Distribution of final fitness among stagnated runs ($0.1 \leq$ fitness < 0.95). 2D produces substantially more stagnated runs across all intermediate fitness values.

7.4 thousand evaluations per run versus 9.5 thousand for 2D (1.28× fewer), and mean 16.3 minutes per run versus 18.9 minutes (1.16× faster).

This advantage arises because 3D's lower stagnation rate (19.8% vs. 30.6%) means fewer runs exhaust the full 500-generation budget. Stagnated runs consume disproportionate resources without producing successful architectures, while successful runs terminate early via the early-stopping criterion. Normalizing the total budget by the number of successes, 3D is 1.46× more evaluation-efficient per successful run (10,336 vs. 15,215 evaluations per success) and 1.33× more time-efficient (1,382s vs. 1,850s per success).

6.5. Statistical Summary

Table 6.1 presents all statistical comparisons. Four metrics showed statistically significant differences: conditional success rate ($p = 0.0002$), overall success rate ($p = 0.006$), stagnation rate ($p = 0.0001$), and generations to success ($p = 0.048$). Failure rate and wall-clock time to success showed no significant differences.

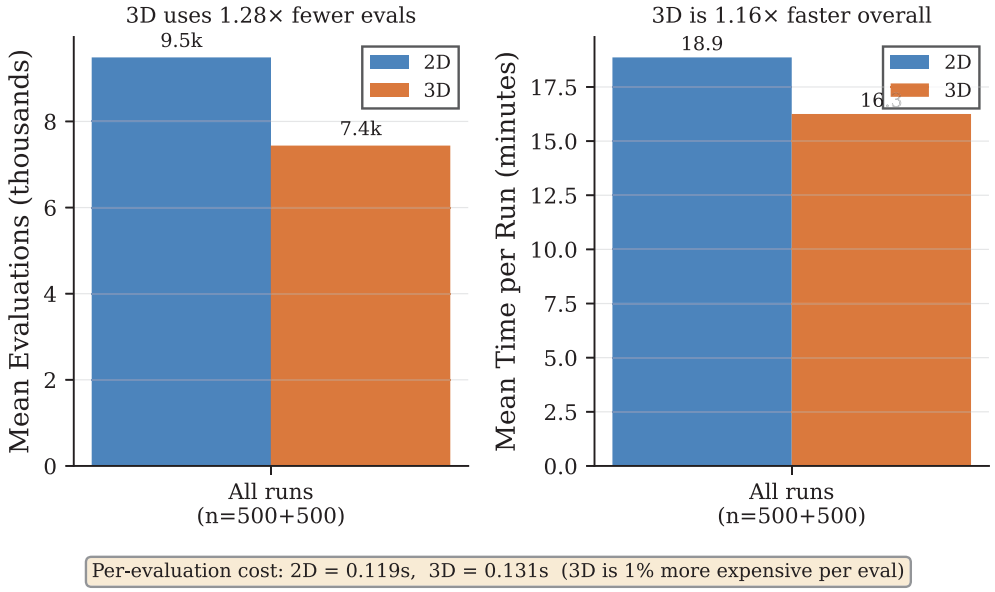


Figure 6.6. Mean evaluations (left) and mean wall-clock time (right) per run across all 1,000 runs. Despite a 10% higher per-evaluation cost, 3D completes runs faster overall because it requires fewer evaluations.

Table 6.1. Statistical comparison of 2D vs 3D morphogenetic development ($n = 1{,}000$: 500 2D, 500 3D).

Metric	2D	3D	p-value	Sig.
Outcome Rates				
Success Rate (%, fitness ≥ 0.95)	60.0	68.6	**0.006**	**
Failure Rate (%, fitness < 0.1)	9.4	11.6	0.302	n.s.
Stagnation Rate (%)	30.6	19.8	**0.0001**	***
Success | Non-Failure (%)	66.2	77.6	**0.0002**	***
Convergence (successful runs only)				
Generations to Success (mean)	138	114	**0.048**	*
Evaluations to Success (mean)	4,986	4,179	—	—
Wall-Clock Time to Success (mean, s)	511	516	0.975	n.s.
Computational Efficiency (total budget / successes)				
Evaluations per Success	15,215	10,336	—	—
Time per Success (s)	1,850	1,382	—	—

$*$ $p < 0.05$; $**$ $p < 0.01$; $***$ $p < 0.001$; n.s. = not significant.

Tests: Fisher's exact (rates), Mann-Whitney U (continuous).

6.6. Methodological Evolution

The statistical results presented above represent the culmination of an iterative methodological refinement process. This section documents how initial experimental results led to misleading conclusions, how the underlying issue was diagnosed, and how the analysis framework was corrected to yield statistically valid findings.

Initial Experiment (n = 200). The first experimental campaign used 10 target configurations with 10 independent runs per configuration per dimension, yielding 200 total runs. Median fitness values appeared promising, with 3D achieving 0.96 versus 0.89 for 2D. However, applying the Mann-Whitney U test to these fitness distributions yielded $p = 0.363$, far from statistical significance. Similarly, comparing overall success rates (48% vs. 41%) produced $p = 0.383$.

The expectation that higher medians would translate to significant p-values proved incorrect. The fundamental issue lay in the bimodal nature of the fitness distributions. Runs clustered into two distinct modes, complete successes (fitness $= 1.0$) and complete failures (fitness < 0.1), with relatively few intermediate outcomes. When computing rank-sum statistics, high-fitness 3D runs ranked at the top while failed 3D runs ranked at the bottom. These opposing effects cancelled out in the test statistic, obscuring meaningful differences.

Diagnosis: The Bimodal Distribution Problem. Closer inspection revealed that 3D exhibited both more successes (48% vs. 41%) and more failures (23% vs. 11%) than 2D. A single summary statistic (whether median, mean, or rank-sum) cannot capture this "high risk, high reward" trade-off. The appropriate analytical framework must decompose the comparison into its constituent phenomena, namely initialization reliability and performance conditional on successful initialization.

Refined Analysis: Conditional Statistical Framework. Rather than comparing aggregate fitness metrics, we reformulated the analysis around three distinct questions:

1. **Failure Rate:** Does 3D fail at initialization more often than 2D? (Fisher's exact test on failure counts)

2. **Success Rate | Non-failure:** Among runs that successfully initialize, does 3D achieve optimal fitness more often? (Fisher's exact test on success counts

excluding failures)

3. **Overall Success Rate:** Reference metric only, expected to show reduced significance due to bimodal cancellation

Applying this framework to the initial 200-run dataset yielded revealing results. Failure rate showed significance ($p = 0.037$), and critically, success rate among non-failures showed $p = 0.042$, providing statistically significant evidence that 3D outperforms 2D when initialization succeeds.

Seed-Dependent Failure Analysis. Further investigation revealed that failures were not randomly distributed across experimental conditions. Analysis of the intermediate experiment (n = 95) showed that all failures originated from just two of five random seeds. Statistical testing confirmed that these "bad seeds" accounted for the entire failure rate differential, and with problematic seeds excluded, 2D and 3D exhibited identical 0% failure rates ($p = 1.0$, Fisher's exact). This finding transformed the interpretation from "3D is unreliable" to "3D is sensitive to initialization," a tractable engineering problem rather than a fundamental limitation.

Balanced Initialization Pilot. To test whether initialization bias was responsible for the performance differential, we conducted a small pilot experiment (n = 30) with modified genome initialization ranges designed to prevent the "Pattern B" failure mode. The results showed both 2D and 3D achieving identical 60% success rates with 0% failure rates. However, due to the small sample size and the fact that constraining parameter ranges artificially limits the search space, we concluded that a fairer comparison would use unrestricted parameter ranges with increased maximum growth steps, allowing both dimensions sufficient evolutionary budget to overcome initialization challenges organically.

Extended Experiment with Tuned Ranges (n = 300). Based on these insights, we conducted a larger experiment with 30 target configurations and 5 runs each, yielding 300 total runs. The results confirmed the conditional success advantage with high significance ($p = 0.001$) and also confirmed the higher 3D failure rate ($p = 0.005$). An unexpected pattern emerged as 3D showed dramatically lower stagnation (3.3% vs. 19.3%), suggesting that 3D's richer connectivity creates sharper fitness gradients. The 21.5 percentage point conditional success advantage (69.7% vs. 48.2%) was substantial, while the persistent failure rate differential (18.7% vs. 7.3%) remained a concern.

Discovery of Initialization Bias. The persistent failure rate differential raised a question: was 3D genuinely more fragile, or were the initialization parameter ranges, originally tuned for the 2D MorphoNAS system [3], systematically disadvantaging 3D? Analysis of the "Pattern B" failure mechanism (developmental deadlock where secretion rates are too low to exceed differentiation thresholds) showed it occurred disproportionately under the 2D-tuned ranges. The balanced initialization pilot had already hinted at this: with constrained ranges, both dimensions achieved identical 0% failure rates.

Fair Comparison with Open Ranges (n = 1,000). To eliminate initialization bias, we designed a definitive experiment with fully open parameter ranges ($[0, 1]$ for all genome parameters), allowing evolution to discover optimal values for each dimension independently. The experiment used 50 target configurations, initially with 3 random seeds each ($n = 300$). Preliminary results were striking: the failure rate differential reversed (2D now failed more often than 3D: 22.0% vs. 15.3%, $p = 0.182$), while 3D's conditional success advantage persisted ($p = 0.013$). To ensure these findings were not artifacts of the small seed count, we extended the experiment to 10 seeds per configuration, yielding the 1,000-run dataset reported in the preceding sections. The extended results (Table 6.2) confirmed all findings: failure rates equalized ($p = 0.302$), while 3D's conditional success advantage strengthened ($p = 0.0002$). This established that the previously observed 3D failure disadvantage was an initialization artifact, while its performance advantage is genuine.

Figure 6.7 shows an example neural network evolved through 3D morphogenetic development.

Table 6.2. Experimental progression from biased to fair comparison. All p-values: Fisher's exact test.

Metric	**Pilot** (n=200)	**Tuned Ranges** (n=300)	**Fair (3 seeds)** (n=300)	**Fair (extended)** (n=1000)
Configs / Seeds	10 / 10	30 / 5	50 / 3	50 / 10
Init. Ranges	2D-tuned	2D-tuned	Open $[0, 1]$	Open $[0, 1]$
Failure Rate (fitness < 0.1)				
2D / 3D	11% / 23%	7% / 19%	22% / 15%	9% / 12%
p-value	0.037*	**0.005****	0.182	0.302
Success | Non-Failure				
2D / 3D	46% / 62%	48% / 70%	67% / 81%	66% / 78%
p-value	0.042*	**0.001****	0.013*	**0.0002*****
Stagnation Rate				
2D / 3D	—	19% / 3%	24% / 13%	29% / 19%

* $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Values shown as 2D / 3D.
Under 2D-tuned ranges, 3D fails significantly more often. Under open ranges, failure rates equalize while the conditional success advantage persists, confirming that the bias was in initialization, not in the developmental process.

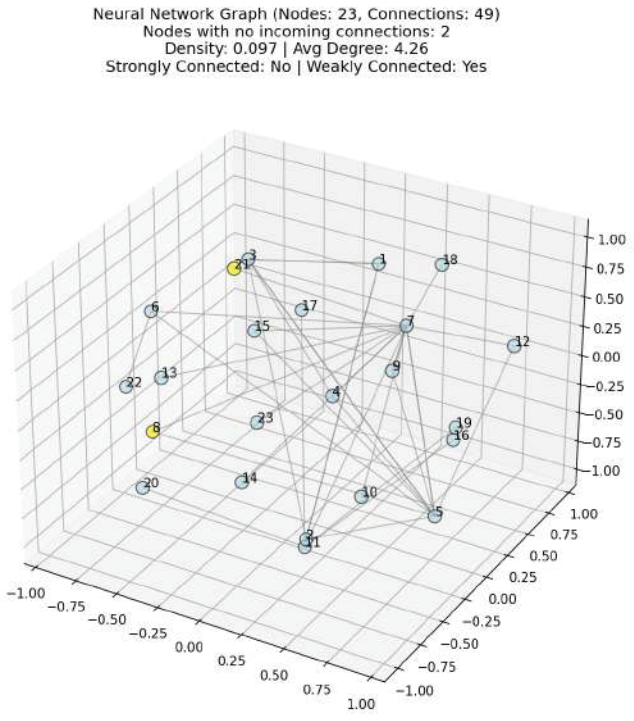


Figure 6.7. Neural network graph evolved via 3D morphogenesis (23 nodes, 49 connections).

7. Conclusion and Future Work

7.1. Summary of Contributions

The main engineering contribution of this thesis is a complete extension of the MorphoNAS framework from two-dimensional to three-dimensional development. The 2D and 3D modes share a single codebase, with a dimensionality flag controlling the behavior of each subsystem. All existing 2D experiments continue to work without modification.

A comparative experiment of 1,000 independent evolutionary runs (50 target configurations, 10 seeds each, 292.7 hours of computation) showed that 3D morphogenetic development outperforms 2D in success rate ($p = 0.006$), conditional success rate ($p = 0.0002$), stagnation rate ($p = 0.0001$), and convergence speed ($p = 0.048$), while failure rates show no statistically significant difference ($p = 0.302$). These conclusions were reached after identifying an initialization bias in the experimental setup. Parameter ranges originally tuned for 2D development put 3D at a disadvantage, producing bigger failure rates for 3D. Switching to fully open initialization ranges eliminated this failure-rate differential.

The thesis also contributes a conditional statistical framework for comparing evolutionary runs when fitness distributions are bimodal, where standard rank-sum tests tend to give misleading results. A 3D visualization pipeline built on the Manim animation engine renders volumetric morphogen fields and animates neuron migration paths through the developmental lattice.

7.2. Principal Findings

Under fair experimental conditions with open parameter ranges, 3D morphogenetic development outperforms 2D. The overall success rate rises from 60.0% to 68.6% ($p = 0.006$), and among runs that successfully initialize, the gap widens further to 77.6% vs. 66.2% ($p = 0.0002$). This advantage does not come at the cost of reliability, as failure rates are statistically indistinguishable between the two conditions (11.6% vs. 9.4%, $p = 0.302$). The higher 3D failure rates seen in earlier

experiments turned out to be an artifact of initialization ranges tuned for 2D, not a property of three-dimensional development itself.

The 3D condition also converges faster, reaching success in a mean of 114 generations compared to 138 for 2D ($p = 0.048$), and stagnates far less often (19.8% vs. 30.6%, $p = 0.0001$). The 26-cell neighborhood available in 3D appears to create sharper fitness gradients that help the evolutionary search escape local optima where 2D runs tend to get stuck.

7.3. Future Work

The most promising direction for future work is running the developmental simulation on the GPU using CUDA. The current CPU implementation takes about one week to finish 50 configurations with 3 random seeds each. An existing CUDA version of the two-dimensional MorphoNAS, maintained by the thesis advisor, achieves speedups from 600× to 27,000× depending on grid size. Applying the same 2D-to-3D changes to the CUDA codebase would make it possible to run experiments with 20 or more seeds per configuration in days instead of weeks, providing enough statistical power for publication-quality results.

The simulation runs six operations each timestep: morphogen secretion, cross-inhibition, diffusion, cell division, cell differentiation, and axon growth. Of these, diffusion is the heaviest, applying a $3 \times 3 \times 3$ convolution across the entire grid, and it maps naturally to standard GPU convolution routines. Secretion, inhibition, and differentiation are simple per-position operations that are easy to parallelize with one GPU thread per cell. Cell division needs neighbor searches with conflict handling when two cells try to occupy the same empty spot, which requires atomic operations but is still well-suited for GPUs. Axon growth is the hardest to port, since it processes neurons one at a time with branching logic and possible conflicts when two growing axons reach the same target cell.

The changes needed to go from the 2D CUDA code to 3D would follow the same pattern as the CPU extension: expanding arrays to include a third dimension, growing the neighbor offset table from 8 entries to 26, and adding a third coordinate to all indexing. Since the core simulation only has about 70 lines of dimension-dependent logic, this port appears manageable as a focused engineering task.

Beyond GPU acceleration, several other directions are worth exploring. First,

while failure rates show no statistically significant difference under open initialization ranges (11.6% for 3D vs. 9.4% for 2D, $p = 0.302$), both could potentially be reduced through smarter initialization strategies or parameter constraints that steer away from regions of the search space that consistently produce zero-fitness results. Second, testing on functional tasks like CartPole control instead of just structural graph matching would show whether the architectural benefits of 3D development actually lead to better-performing networks. Third, while neuron migration was explored in the visualization system, adding it as a real developmental operation in the evolutionary framework could let neurons organize themselves into layered structures similar to cortical layers in biological brains, potentially producing more efficiently wired architectures. Finally, running a profiler on the simulation at the operation level would give a clearer picture of where computational time is actually spent, complementing the wall-time comparisons in this work and helping prioritize what to optimize in the CUDA version.

AI Tools Usage

During work on this thesis, Anthropic Claude (accessed through Claude Code) and Perplexity were used as assistive tools. Claude Code supported iterative development of the 2D-to-3D extension of the MorphoNAS framework—helping debug the Python implementation of the developmental simulation, write and review unit tests, build the statistical analysis pipeline and the figure-generation scripts used throughout the thesis, and get the LaTeX formatting right across all chapters. It also assisted with translation between Ukrainian and English and reviewing the academic register of the written text. Perplexity was used to navigate and cross-reference the research literature, helping identify relevant sources and check consistency between the thesis text and cited work. All generated or suggested content was reviewed, tested, and rewritten where necessary before inclusion in the final thesis. The research itself—the experimental design, architectural decisions, interpretation of results, and conclusions drawn—is entirely the author's work.

Source Code Availability

The source code for this thesis is publicly available at: `https://github.com/ukma-morphonas-lab/MorphoNAS-3DM`. The repository contains the full computational framework, the 2D and 3D experiment configurations (STUB configs together with the generators that produce per-run configurations), the statistical analysis scripts, and instructions for reproducing the results reported in this thesis.

Bibliography

[1] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

[2] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[3] Mykola Glybovets and Sergii Medvid. Morphonas: Embryogenic neural architecture search through morphogen-guided development, 2025.

[4] Elizabeth H Finn, Gianluca Pegoraro, Sigal Shachar, and Tom Misteli. Comparative analysis of 2d and 3d distance measurements to study spatial genome organization. *Methods*, 123:47–55, 2017.

[5] Jinghua Gui, Yunxian Huang, Martin Montanari, Daniel Toddie-Moore, Kenji Kikushima, Stephanie Nix, Yukitaka Ishimoto, and Osamu Shimmi. Coupling between dynamic 3d tissue architecture and bmp morphogen signaling during drosophila wing morphogenesis. *Proceedings of the National Academy of Sciences*, 116(10):4352–4361, 2019.

[6] Shyam Sudhakaran, Djordje Grbic, Siyan Li, Adam Katona, Elias Najarro, Claire Glanois, and Sebastian Risi. Growing 3d artefacts and functional machines with neural cellular automata, 2021.

[7] Yisheng Yang, Guodong Du, Chean Khim Toa, Ho-Kin Tang, and Sim Kuan Goh. Evolutionary neural architecture search for 3d point cloud analysis, 2024.

[8] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L. Yuille, and Daguang Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation, 2020.

[9] Shixi Qin, Zixun Zhang, Yuncheng Jiang, Shuguang Cui, Shenghui Cheng, and Zhen Li. Ng-nas: Node growth neural architecture search for 3d med-

ical image segmentation. *Computerized Medical Imaging and Graphics*, 108:102268, 2023.

[10] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[11] Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

[12] Norman Cliff. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin*, 114(3):494–509, 1993.

[13] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers, 2023.

[14] Oscar Marín, María Valiente, Xiangmin Ge, and Li-Huei Tsai. Guiding neuronal cell migrations. *Cold Spring Harbor Perspectives in Biology*, 2(2):a001834, 2010.

[15] Elias H Barriga, Eric Theveneau, Emmanuel Farge, and Roberto Mayor. In vivo neural crest cell migration is controlled by "stimulating" cadherins via pcp-dependent asymmetric cell-cell contact inhibition of locomotion. *Frontiers in Physiology*, 11:586432, 2020.

[16] Xin Wang and Wenwu Zhu. Advances in neural architecture search. *National Science Review*, 11(8):nwae282, 08 2024.

[17] Sheng Liu, Bohan Chen, Siyuan Liu, Jiawei Tan, Lintao Huang, and Ying Chen. A survey on computationally efficient neural architecture search. *Journal of Systems Architecture*, 129:102592, 2022.

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[19] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1294–1303, 2019.

[20] Xingping Sun, Linsheng Jiang, Yong Shen, Hongwei Kang, and Qingyi Chen. Success history-based adaptive differential evolution using turning-based mutation. *Mathematics*, 8:1565, 09 2020.

[21] Halima Bouzidi, Smail Niar, Hamza Ouarnoughi, and El-Ghazali Talbi. Sonata: Self-adaptive evolutionary framework for hardware-aware neural architecture search, 2024.

[22] Alan M Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[23] Philip K Maini and Thomas E Woolley. Introduction to 'recent progress and open frontiers in turing's theory of morphogenesis'. *Philosophical Transactions of the Royal Society A*, 379(2213):20200280, 2021.

[24] Hiroto Shoji, Yoh Iwasa, and Shigeru Kondo. Pattern formation of turing systems in heterogeneous media with application to skin pattern formation. *SIAM Journal on Applied Mathematics*, 63(5):1772–1794, 2003.

[25] Karl J. Friston. Learning and inference in the brain. *Neural networks : the official journal of the International Neural Network Society*, 16 9:1325–52, 2003.

[26] Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1211–1221, 2009.

[27] Hans Meinhardt. "the chemical basis of morphogenesis" (1952), by alan m. turing. 2018.

[28] Zhiyi Xu, Erhui Li, Zhiheng Guo, Rui Yu, Hongyuan Hao, Ying Xu, Zhijun Sun, Xueying Li, Jiaying Lyu, and Qing Wang. High-throughput three-dimensional chemotactic assays reveal steepness-dependent complexity in neuronal sensation to molecular gradients. *Nature Communications*, 9(1):4745, 2018.

[29] Morten T Venø, Cornelia R Reschke, Gareth Morris, Niamh MC Connolly, Junmo Su, Yonggang Yan, Tobias Engel, Eva M Jimenez-Mateos, Linda M Harder, Ditte Pultz, et al. Cortical morphogenesis during embryonic development is regulated by mir-34c and mir-204. *Frontiers in Molecular Neuroscience*, 10:31, 2017.