Methods in Exploration and Interpretability for Automating Discovery

Habilitation à Diriger des Recherches

ISAE-Supaero, Université de Toulouse

Dennis G. Wilson

Acknowledgements

Research is often said to take place on the shoulders of giants. While mine has, I have also had the luck to meet these giants, discuss with them, learn from them, and to be held up not only by their shoulders, but by their encouragement and wisdom.

First, the PhD students whose works this manuscript discusses. Kaitlin Maile, Mahmoud Al Najar, Paul Templier, Paul-Antoine le Tolguenec, Estelle Chigot, Camilo de la Torre: thank you for trusting me, for spending this important period of your lives with me, and for all of the brilliant ideas you shared.

My understanding of advising, of research, and of teaching at the operational, daily, realistic level was shaped by the ADO team at ISAE-Supaero. Emmanuel Rachelson, Thomas Oberlin, Elise Vareilles, Zoé Krug, Alain Haït, Laurent Houssin: thank you for supporting me, working with me, and showing me what it means to be a researcher and teacher.

The work in this manuscript is the result of a number of collaborations; one of the joys of research is getting to learn from colleagues who master their own fields. Rafael Almar, Erwin Bergsma, Giorgia Nadizar, Eric Medvet, Una-May O'Reilly, Erik Hemberg, Wolfgang Banzhaf, Yuri Lavinas, Patrick Forré, Takeshi Doi, Swadhin Behera: thank you for sharing your expertise with me, for indulging my ideas, and for building exciting projects together. I hope we can continue to work together on the foundations you've made.

The process of writing and defending this manuscript was guided and made possible through colleagues that inspired much of its research. Claire Monteleoni, Bing Xue, Stéphane Doncieux, Marc Schoenauer, Frédéric Dehais: thank you for encouraging me, for sharing the vision of this manuscript, and for motivating directions for my own career.

Finally, I would maybe not be doing research, or at least not in the way I have, without key advisors who have molded my path. Sylvain Cussat-Blanc and Hervé Luga: thank you for bringing me into a thesis, for directing it, and for maintaining close ties and counsel since. Una-May O'Reilly: thank you for taking me into your group, for keeping that invitation open, and for the advice and inspiration you've given over the years. Finally, I extend a deep gratitude to Julian Miller, who taught me to think for myself, to research what I find most interesting, and to approach research with compassion.

Preface

This document is the manuscript for my application to the *Habilitation à Diriger des Recherches* (HDR). It is intended to demonstrate that I am able to direct research, building on the research activities which I have advised. As I was educated in the American university system, this step of the academic process was first foreign to me. However, I have greatly appreciated the opportunity to pause at this point in my career, to reflect, and to write.

It appears that there are countless ways to format and write an HDR manuscript. The goal is to represent an overview of my research, which motivates the inclusion of the *curriculum vitae* following this preface. In the rest of the manuscript, I focus on my work, in detail, as the main examples to motivate a thesis. However, the format of this evaluation, including this manuscript, allow me an additional liberty rarely taken in scientific writing. Rather than comprehensively detailing all the work under my supervision, I have chosen to advance a point of view.

My thesis was shaped by the books "Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems" (Downing 2015), "The Selfish Gene" (Dawkins 2016), and "Principles of Neural Design" (Sterling and Laughlin 2015). This manuscript was heavily influenced by the writings of Carl Sagan, Yuval Noah Harari, and Melanie Mitchell. I don't claim to place this manuscript amongst these works, but rather offer them as an explanation. Accessible, observant, and even opinionated scientific writing has long been an inspiration for me. I hope that, with this manuscript, I can provide perspective on my research, beyond what is found in the articles referenced herein.

Curriculum vitae

Personal Details

Name: Dennis G. Wilson
Email: dennis.wilson@isae.fr
Personal website: https://d9w.github.io/
ORCID: 0000-0003-2414-0051
Date of birth: April 17, 1991
Nationality: American

Education

PhD in Computer Science 2016 - 2019
Evolving Principles of Artificial Neural Design
Institut de Recherche en Informatique de Toulouse (IRIT)
Université Toulouse III - Paul Sabatier, France
Director: Prof. Hervé Luga, Université Toulouse - Jean Jaurès, IRIT
Co-supervisor: Prof. Sylvain Cussat-Blanc, Université Toulouse Capitole, IRIT

Bachelor of Science in Electrical Engineering and Computer Science 2010 - 2014
Massachusetts Institute of Technology (MIT), USA

Professional positions

Associate Professor *(Enseignant-chercheur)* 2019 - present
Department of Complex Systems Engineering, ISAE-Supaero, Toulouse, France

Co-founder and CTO 2019 - 2021
Nautilia Computing, Toulouse, France

Postdoctoral researcher 2019 - 2019
IRIT, Toulouse, France

Computer Science Lecturer 2016 - 2019
Université Toulouse 1 Capitole, Toulouse, France

Software engineer 2014 - 2016
Infinidat LTD, Israel

Key figures

54 publications, including Nature Communications
576 citations, h-index 13
6 theses co-advised, 3 defended
2 postdocs co-advised
100+ hours of teaching per year since 2019
6 international workshops organized, 2 competitions

Research achievements

The selected publications featured below are works in which I had a leading role or significant contribution. Out of the 10 publications, 4 were with either of my PhD supervisors (S. Cussat-Blanc, H. Luga), and 5 are from PhDs which I supervised. My work spans evolutionary algorithms, machine learning, and the application of AI to climate science. In the following works, I have advanced the following ideas: genetic programming offers an interpretable and competitive alternative to deep learning. Developmental neural networks provide insights into the learning process. Exploration in search and learning can improve AI robustness. Finally, machine learning can help us understand climate change.

Graph-based Genetic Programming

1. **Wilson, D. G.**, Cussat-Blanc, S., Luga, H. & Miller, J. F. *Evolving Simple Programs for Playing Atari Games* in *Proceedings of the Genetic and Evolutionary Computation Conference* (ACM, 2018). This conference paper demonstrated the competitiveness of Graph-based Genetic Programming (GGP) to state-of-the-art deep reinforcement learning methods on the Atari benchmark. It laid the foundation for subsequent advancements in interpretable genetic programming.
2. Nadizar, G., Medvet, E. & **Wilson, D. G.** *Naturally Interpretable Control Policies via Graph-Based Genetic Programming* in *European Conference on Genetic Programming* (2024). This conference paper showed that GGP creates interpretable control policies that rival deep reinforcement learning for standard robotic tasks. (Best Paper Award)
3. Cortacero, K., ..., **Wilson, D. G.**, *et al.* Evolutionary Design of Explainable Algorithms for Biomedical Image Segmentation. *Nature Communications*, 7112 (2023). This Nature Communications article presents a GraphGP method for optimizing interpretable analysis pipelines for biomedical image segmentation, addressing critical needs for explainable AI in healthcare and achieving performance competitive with state-of-the-art neural networks. (Humies Gold Award)

Developmental Neural Networks

1. Miller, J. F., **Wilson, D. G.** & Cussat-Blanc, S. Evolving Developmental Programs That Build Neural Networks for Solving Multiple Problems. *Genetic Programming Theory and Practice XVI* (2019). This book chapter explores the use of genetic programming to evolve rules of neural development to create flexible networks capable of solving multiple problems.

2. Maile, K., Rachelson, E., Luga, H. & **Wilson, D. G.** *When, Where, and How to Add New Neurons to ANNs* in *International Conference on Automated Machine Learning* (PMLR, 2022). This conference paper builds on the previous work of developmental neural networks, presenting strategies for dynamically expanding neural network architectures by determining precise measures for when to trigger artificial neurogenesis.

Exploration in Search and Learning

1. Templier, P., Grillotti, L., Rachelson, E., **Wilson, D. G.** & Cully, A. *Quality with Just Enough Diversity in Evolutionary Policy Search* in *Proceedings of the Genetic and Evolutionary Computation Conference* (2024). This work introduces a novel evolutionary policy search algorithm that balances exploration and exploitation, leading to more robust and diverse policy generation in complex environments. (Best Paper Award)
2. Le Tolguenec, P.-A., Teichteil-Koenigsbuch, F., Besse, Y., **Wilson, D. G.** & Rachelson, E. *Exploration by Learning Diverse Skills through Successor State Measures* in *The Thirty-eighth Annual Conference on Neural Information Processing Systems* (2024). This paper presents LEADS, an exploration algorithm that leverages state estimates to learn diverse skills, enhancing the exploration capabilities of reinforcement learning agents in multi-task environments.

Machine Learning for Climate Science

1. Al Najar, M., Thoumyre, G., Bergsma, E., Almar, R., Benshila, R. & **Wilson, D. G.** Satellite Derived Bathymetry Using Deep Learning. *Machine Learning* (2021). This journal publication introduces a deep learning model for estimating bathymetry from satellite data, significantly improving coastal geography analysis and supporting environmental monitoring efforts.
2. Disdier, E., Almar, R., Benshila, R., Al Najar, M., Chassagne, R., Mukherjee, D. & **Wilson, D. G.** Predicting beach profiles with machine learning from offshore wave reflection spectra. *Environmental Modelling & Software*, 106221 (2024). This journal publication details a machine learning approach to forecasting beach profiles using wave reflection data, demonstrating a new data source for coastal monitoring and management.
3. Al Najar, M., Almar, R., Bergsma, E. W., Delvit, J.-M. & **Wilson, D. G.** *Improving a Shoreline Forecasting Model with Symbolic Regression* in *Tackling Climate Change with Machine Learning*, ICLR 2023 (2023). Presented at the CCAI Workshop at ICLR 2023, this article uses genetic programming to improve existing shoreline models, demonstrating significant improvements in forecasting accuracy for climate-related coastal changes.

Teaching

I have taught over 100 hours of courses per year at ISAE-Supaero since 2019. From 2020 to 2024, I was in charge of the Data and Decision Science master-level program at ISAE-Supaero, which covers machine learning, data engineering, and counts 60 students over 240 hours. In this program, I have created new courses on data engineering, data privacy, and deep learning. I have also created a new 30 hour elective course on evolutionary algorithms.

Advising

I have co-advised 6 PhD students, 3 of which have now successfully defended. I have also supervised one postdoctoral researcher and assisted in the supervision of another, and I have supervised 8 Master's students for their thesis projects. Below, I list the PhDs which I have advised with the details of their financing and co-advising.

Kaitlin Maile
Title: Dynamic Architectural Optimization of Artificial Neural Networks
Financing: EDMITT Scholarship
Advisors: Hervé Luga, Sylvain Cussat-Blanc, Dennis G. Wilson
Dates: 01/11/2020 - 04/10/2023

Mahmoud Al Najar
Title: Modelling coastal evolution with machine learning
Financing: Scholarships from CNES and Région Midi-Pyrénées
Advisors: Rafael Almar, Dennis G. Wilson
Dates: 09/11/2020 - 30/11/2023

Paul Templier
Title: Leveraging Structure in Evolutionary Neural Policy Search
Financing: Scholarships from ISAE-Supaero and Région Midi-Pyrénées
Advisors: Emmanuel Rachelson, Dennis G. Wilson
Dates: 11/01/2021 - 22/04/2024

Paul Antoine le Tolguenec
Title: Exploration Driven Reinforcement Learning For Testing Critical Systems
Financing: CIFRE ANITI with Airbus
Advisors: Emmanuel Rachelson, Dennis G. Wilson, Yann Besse
Dates: 02/01/2022 - 05/2025 (provisional)

Estelle Chigot
Title: Maximizing the robustness of data-driven functions for aircraft autonomy through synthetic data generation
Financing: CIFRE with Airbus
Advisors: Thomas Oberlin, Dennis G. Wilson, Meriem Ghrib
Dates: 01/04/2023 - 04/2026 (provisional)

Camilo de la Torre
Title: Hybridization between Cartesian Genetic Programming and Machine Learning
Financing: EDMITT scholarship
Advisors: Sylvain Cussat-Blanc, Dennis G. Wilson, Hervé Luga
Dates: 01/10/2023 - 10/2026 (provisional)

Peer recognition

- **ACM SIGEVO Human Competitive Competition Gold Award** 2024
  First place in the "Humies" Competition at GECCO 2024 for our work on interpretable image analysis. This prestigious award recognizes results that are competitive with human performance.
- **Best Paper Award - GECCO Complex Systems Track** 2024
  For "Quality with Just Enough Diversity in Evolutionary Policy Search."
- **ACM SIGSOFT Distinguished Paper Award** 2024
  Awarded at the International Symposium on Software Testing and Analysis for "Exploration-Driven Reinforcement Learning for Avionic System Fault Detection."
- **Best Paper Award - EuroGP** 2024
  For "Naturally Interpretable Control Policies via Graph-Based Genetic Programming."
- **ANITI Affiliate Member** 2024 - present
  I am associated with the Artificial and Natural Intelligence Toulouse Institute (ANITI), being awarded the status of Affiliate Member for my contributions on interpretable machine learning.
- **CIFRE thesis grants** 2021-2025
  Two industrial grants for collaborations with Airbus on AI research applied to aerospace engineering.
- **Région Occitanie grants** 2019-2023
  Two thesis grants for research on combining evolutionary algorithms and machine learning, and on applying machine learning to coastal science.
- **Invited Speaker on Evolutionary Reinforcement Learning** 2021 - 2023
  Invited to present my work on interpretable reinforcement learning at the Evolutionary Reinforcement Learning Workshop, 2023, and the Reinforcement Learning Virtual School, 2021.
- **ACM SIGEVO Best Dissertation Award** 2020
  This yearly award recognizes the best doctoral dissertation in the field of evolutionary computation.
- **SIGAI Essay Contest on Ethics and AI** 2017
  I won the ACM SIGAI Essay Contest for my essay on "The ethics of automated behavioral microtargeting" where I identified the ethical problems of using AI for targeted advertising.
- **CIMI Doctoral Fellowship recipient** 2015
  My thesis was financed through a competitive fellowship from CIMI at the University of Toulouse.

Other contributions

Editorial activities

I am an editorial board member for the ACM Transactions on Evolutionary Learning and Optimization journal, where I led the journal's policy on the use of large language models and organized a Special Issue on Integrating Evolutionary Algorithms and Large Language Models. I served as Track Chair for Complex Systems at GECCO from 2020 to 2022 and

will serve as Track Chair for the Neuroevolution Track from 2025 to 2027; this two-year mandate is similar to Area Chair. I am also a regular reviewer for top AI conferences and journals, including NeurIPS, ICLR, ICML, GECCO, and the IEEE Transactions on Evolutionary Computation.

Workshop organization

I organized a workshop on Developmental Neural Network workshop from 2018 to 2020, and co-organize a workshop on Graph-based Genetic Programming since 2023. These workshops were hosted at international conferences such as GECCO. I also organized a local workshop in Toulouse in 2024 on Evolutionary Machine Learning, bringing international researchers together to discuss the latest advances in machine learning and evolutionary computation.

Competition organization

I organized a competition on Wind Farm Layout Optimization at GECCO 2014-2016, culminating in an article in *Renewable Energy*. I am currently organizing a competition on Interpretable Control Policies, which was held for the first time at GECCO 2024.

AI for Climate

I am involved in several initiatives to apply AI to climate problems. I am a member of the AI for the Environment Committee (ENVIA) in Toulouse, which promotes collaboration between AI researchers and environmental scientists. From 2022 to 2024, I served as a faculty representative for the Horizons committee at ISAE-Supaero, which promotes sustainable development at ISAE. I have also been involved in the Climate Change AI Mentorship program since 2021, supporting early-stage researchers in AI and climate science. I organized a special issue in the Remote Sensing Journal on integrating satellite remote sensing with AI for coastal issues and organized a special session at the IEEE World Congress on Computational Intelligence on "AI for Climate Science".

Diversity and inclusion

I am committed to promoting diversity in the research community. I was Secretary of the Diversity, Equity, and Inclusion Committee of the International Society of Artificial Life from 2021 to 2022. I served as co-organizer of the ANITI Diversity Commission since 2022 to 2024, focused on encouraging women in AI.

Public outreach

I have been active in public outreach, giving talks on AI in the greater Toulouse region. I have been invited to speak on the impact of generative AI on education and on the general impact of AI on society. I also maintain a newsletter, which has over 130 subscribers, on recent trends in and the societal impacts of AI.

Contents

1 Introduction

Models have long been used to help us understand our world. Over two thousand years ago, the astronomers of ancient Greece sought to explain the motions of the planets through geometry and mathematics. The model of nested crystal spheres, proposed by Aristotle and refined by Ptolemy, placed Earth at the center of the universe. The planets, sun, and stars were thought to be mounted on transparent spheres, rotating in perfect circles around our world. These circles were chosen not just for their mathematical simplicity - they were considered divine, the only shape worthy of the celestial bodies' perfect nature (Sagan 1981).

There were alternative models, such as the heliocentric model of Aristarchus of Samos, who proposed that our planet was just one of many that orbited the Sun. But most astronomers held to the established geocentric model, believing the perfect circles of planetary motion reflected God's design of the cosmos. These models, while imperfect by modern standards, represented the best understanding possible given the observational capabilities of the time.

This changed with the detailed observations of Tycho Brahe in the 16th century. Working before the invention of the telescope, Brahe built massive instruments to precisely measure the positions of planets and stars. Over decades of careful observation, he assembled the most accurate astronomical data ever collected. His measurements were accurate to within a few arc minutes - a remarkable achievement that would prove crucial for understanding planetary motion. But Brahe was primarily an observer, collecting data without a proper mathematical framework to explain it.

Johannes Kepler, Brahe's assistant, inherited this vast collection of astronomical measurements. Kepler spent years trying to reconcile these observations with the perfect circles of classical astronomy. But the data refused to fit - Mars' observed position differed from the predictions by eight arc minutes. Where others might have dismissed this small discrepancy, Kepler recognized its significance (Kepler and Donahue 1992), writing:

> Since the divine benevolence has vouchsafed us Tycho Brahe, a most diligent observer, from whose observations the 8' error in this Ptolemaic computation is shown, it is fitting that we with thankful mind both acknowledge and honor this benefit of God... For if I had thought I could ignore eight minutes of longitude, in bisecting the eccentricity I would already have made enough of a correction in the vicarious hypothesis found in Ch. 16. Now, because they could not have been ignored, these eight minutes alone will have led the way to the reformation of all of astronomy, and have constituted the material for a great part of the present work.

After years of mathematical labor, he discovered that planets move in ellipses, not circles, with the Sun at one focus. The collaboration of Brahe's precise observations with Kepler's mathematical genius revolutionized our understanding of the solar system, and demonstrated a fundamental principle of science: models must be confronted with reality, and data must be interpreted through theoretical frameworks. Neither alone is sufficient for discovery.

1.1 Models

On a spring morning in 1915, Lewis Fry Richardson undertook an extraordinary experiment. Working with just pen and paper, he attempted the first numerical weather forecast by dividing the atmosphere into a grid and calculating how conditions would change over six hours. The forecast took him six weeks to complete - far too slow for practical use, but revolutionary in its approach.

In 1922, Richardson published his vision of a "forecast factory" where 64,000 human computers would work in parallel, each responsible for calculations in their grid cell, passing information to their neighbors. A conductor at the center would coordinate their efforts, assembling a complete picture of tomorrow's weather. While fantastical at the time, Richardson had outlined the basic architecture of modern weather prediction: parallel computation across a discretized atmosphere (Richardson and Ashford 1993).

This vision began to materialize in the 1950s when the first digital computers appeared. In 1960, Edward Lorenz at MIT encoded the physics of the atmosphere into twelve differential equations that his Royal McBee computer could simulate (Lorenz 1982). Though primitive by today's standards - the machine could perform just sixty multiplications per second - it represented a fundamental shift in how we model nature. Rather than seeking perfect mathematical forms like Kepler's ellipses, modern scientific models would use computation to handle complexity (North 1975; Gould et al. 1996).

Today's weather centers employ massive supercomputers, dividing the globe into millions of grid cells and tracking hundreds of variables at each location. Yet even with this computational might, these models face fundamental limits. They require immense computing power, making long-range forecasts prohibitively expensive. More crucially, they may not fully capture the patterns hidden in the wealth of observational data now available from satellites, weather stations, and other sensors (Eyring et al. 2024).

A new approach to modeling physical systems has recently taken the world by storm: machine learning models that can learn patterns directly from observational data. These advances have been largely driven by a type of machine learning, deep learning (LeCun, Bengio, and Hinton 2015). Unlike traditional models that encode physical equations, the deep neural networks of deep learning consist of many, sometimes millions, of interconnected artificial neurons. The parameters of this Artificial Neural Network (ANN), being the strength of connections between neurons and their activation bias, are adjusted through a process of trial-and-error referred to as training or learning. A deep neural network makes a prediction or estimate based on an example from a dataset, then the error in this prediction is used to correct the many parameters of the network (Rumelhart, Hinton, and Williams 1986). This approach has revolutionized fields like computer vision (Krizhevsky, Sutskever, and Hinton 2012) and natural language processing (Brown et al. 2020). Just as a

meteorologist develops intuition by observing weather patterns over many years of experience, these networks learn to recognize complex relationships in data through this process of training.

GraphCast, developed by Google DeepMind, exemplifies this shift (Lam et al. 2023). Rather than simulating atmospheric physics step by step, it learns weather patterns from historical data. The results are impressive - not only does it match or exceed the accuracy of traditional numerical models, but it produces global forecasts in under a minute, a task that takes supercomputers hours. This dramatic speedup could enable more frequent forecasts and longer prediction windows, potentially revolutionizing our ability to prepare for extreme weather.

Similar advances are occurring across scientific domains. Machine learning models now predict protein structures (Jumper et al. 2021), discover new materials (Merchant et al. 2023), and forecast floods (Nearing et al. 2024) with impressive accuracy. Were Kepler alive today, he might be astounded to learn that a telescope bearing his name has used these techniques to discover planets in other solar systems (Valizadegan et al. 2022).

However, these models function as "black boxes." Unlike Kepler's laws, their internal logic remains opaque. This raises questions: How can we trust them in critical decisions? How do we ensure they learn genuine patterns? Most importantly, how can they advance scientific understanding rather than just prediction?

These questions drive this manuscript's focus: developing artificial intelligence methods that drive discovery through the creative exploration of machine learning models that are interpretable and understandable by human researchers.

1.2 Exploration

Discovery often demands deviation. Scientific breakthroughs, like Kepler's shift from circles to ellipses, require exploring beyond existing assumptions. Modern machine learning, however, excels at interpolation within known data rather than extrapolation to novel solutions.

Consider how a typical deep learning model works: it learns to recognize cats by averaging across millions of cat photos, extracting the statistical patterns that define "catness." But crucially, such models cannot imagine a fundamentally new kind of cat—they can only interpolate between examples they've seen. They are, in a sense, prisoners of their training data.

Scientific discovery, in contrast, often requires extrapolation—the ability to conceive of possibilities that lie outside known examples. When Einstein imagined riding alongside a beam of light, he was not averaging across prior theories but exploring an entirely new way of thinking about space and time (Norton 2012). This raises a fundamental question: how can we create artificial systems capable of such creative exploration?

The answer may lie in moving beyond traditional objective-based optimization. Just as evolution does not proceed with a predetermined goal but rather through the continuous exploration of novel forms, we can design algorithms that prioritize discovering new behaviors over optimizing specific metrics. This approach presents a radical departure from conventional machine learning and optimization.

Traditional optimization methods all share a common assumption: that the best way to reach a goal is to continuously measure and reward progress toward that goal. Yet this assumption can be deceptive. Like a climber fixated on reaching the summit who becomes trapped on a local peak, objective-based search can become stuck in suboptimal solutions precisely because it is too focused on improvement rather than exploration (Lehman and Stanley 2011a).

Research that I have participated in or advised advances algorithmic exploration through several complementary approaches. First, through Quality-Diversity optimization (Cully et al. 2015a; Mouret and Clune 2015), we develop methods that evolve populations of solutions that maximize both performance and behavioral novelty (Templier, Grillotti, et al. 2024). Rather than seeking a single optimal solution, these algorithms maintain archives of diverse, high-performing solutions—much like nature maintains a diversity of species rather than converging on a single "best" organism.

Second, through curiosity-driven search (Le Tolguenec et al. 2022), we create systems that are intrinsically motivated to explore their environment by seeking out novel states and interactions. This approach mirrors how human scientists often make discoveries through playful experimentation rather than directed search.

Finally, we apply these exploration techniques to stress-test critical systems, systematically searching for edge cases and failure modes that traditional testing might miss (Le Tolguenec, Rachelson, Besse, et al. 2024). Like scientists probing the boundaries of a theory, these methods actively seek out the unexpected rather than confirming the expected.

The key insight unifying these approaches is that meaningful discovery often requires us to temporarily abandon the pursuit of obvious objectives. Just as Kepler had to be willing to discard the perfect circles that had dominated astronomy for millennia, our artificial discovery systems must be willing to explore seemingly unpromising directions that might, paradoxically, lead to breakthrough solutions.

As we tackle increasingly complex challenges in science—from climate modeling to drug discovery—this capacity for principled exploration becomes ever more crucial. We need systems that can not only optimize within known parameters but also help us discover entirely new possibilities. The methods presented in this manuscript demonstrate how we can move beyond simple optimization toward true exploration-driven discovery.

1.3 Interpretability

Understanding requires transparency. When Kepler derived his laws of planetary motion from Brahe's observations, he could explain every step of his reasoning—from the rejection of perfect circles to the mathematical properties of ellipses that made them a better fit for the data. This transparency was not merely incidental to his discovery; it was essential to the advancement of astronomy. Other scientists could verify his calculations, challenge his assumptions, and ultimately build upon his insights to develop an even deeper understanding of celestial mechanics.

Today, however, we face a paradox in scientific modeling. Our most powerful prediction engines—deep neural networks—operate as black boxes, their internal workings obscured behind billions of inscrutable parameters (Rudin 2019; Tjoa and Guan 2020).

Consider AlphaFold (Jumper et al. 2021), which revolutionized protein structure prediction by achieving unprecedented accuracy in determining how proteins fold. While this breakthrough has enormous practical value, the model itself offers little insight into the underlying physical principles governing protein folding. Scientists can use AlphaFold's predictions, but they cannot easily extract new biological understanding from how it makes those predictions.

This opacity creates a fundamental limitation for scientific discovery. When a neural network learns to predict El Niño events (Ham, Kim, and Luo 2019) or forecast extreme weather (Bi et al. 2022) with high accuracy, it may have discovered important patterns in the data—but if we cannot understand these patterns, we have not fully advanced our scientific knowledge. We are left with a powerful oracle that can make predictions but cannot explain the mechanisms behind them.

The standard response to this challenge has been to develop post-hoc explanation methods that attempt to interpret already-trained neural networks (Tjoa and Guan 2020). These approaches use techniques like saliency maps or feature attribution to suggest which inputs were most important for particular predictions. However, such explanations are often unreliable and can be misleading (Rudin 2019). More fundamentally, they attempt to reverse-engineer understanding from models that were not designed to be understood in the first place.

My work proposes a different approach: developing machine learning models that are interpretable by design. Rather than attempting to explain black-box models after the fact, I advance methods that maintain transparency throughout the learning process. The key insight is that interpretability need not come at the cost of performance—with proper design, models can be both accurate and understandable.

Through genetic programming techniques like Cartesian Genetic Programming (CGP) (Julian Francis Miller 2020), we have demonstrated that interpretable models can solve complex control tasks while remaining fully decomposable and analyzable. Unlike neural networks, these models represent solutions as computational graphs that can be read and understood like computer programs. For instance, we have shown that CGP can learn to play video games (Wilson et al. 2018) and control robots (Nadizar, Medvet, and Wilson 2024a) with performance competitive with deep learning approaches, while maintaining complete transparency about their decision-making process.

Moreover, we have developed novel methods that extend these interpretable approaches to handle diverse types of input data, from images to numerical measurements (De La Torre et al. 2024). This enables interpretable modeling of complex scientific phenomena that involve multiple data modalities—a crucial capability for modern scientific discovery.

My latest work explores the frontier of interpretable scientific modeling through Language Model Genetic Programming (LMGP) (Hemberg, Moskal, and O'Reilly 2024). This approach combines the pattern-recognition capabilities of large language models, massive deep neural networks capable of high-level language use (Brown et al. 2020), with the interpretability of genetic programming. While preliminary, this work shows that the direct optimization of computer code can enable the automated discovery of scientific models that can be easily understood and validated by human experts.

The implications of this research extend beyond any single scientific domain. By developing methods that can discover interpretable models from data, we enable a virtuous cycle

of scientific understanding: models that can not only make accurate predictions but also advance our theoretical understanding, leading to better models and deeper insights. This represents a step toward automated scientific discovery that augments rather than replaces human scientific reasoning.

1.4 Outline

This manuscript examines the automation of discovery through two key capabilities: exploration and interpretability. While the ultimate goal is to advance scientific discovery, particularly in climate science, the methodology is first developed and demonstrated using standard benchmarks in artificial intelligence research, such as simulated robotic control tasks.

The document is structured as follows: this chapter introduced the concept of discovery as a process requiring both deviation from established solutions and understanding of new findings. Chapter 2 focuses on exploration, detailing how optimization can be reimagined as a search process. Through evolutionary algorithms and reinforcement learning, the chapter develops methods that encourage systematic exploration of unknown solution spaces. These concepts are demonstrated through robotic control tasks, where agents must discover novel behaviors to achieve objectives.

Chapter 3 examines interpretability, showing how genetic programming can create models that perform complex tasks while remaining fully transparent. Using examples from robotic control and game environments, the chapter demonstrates that interpretable models can match the performance of black-box approaches while providing clear insights into their decision-making processes.

Chapter 4 shifts focus to scientific applications, particularly in climate science. This chapter transitions from methodology development using AI benchmarks to addressing real-world challenges. Applications in shoreline forecasting and El Niño prediction are analyzed, illustrating how exploration and interpretability advance our understanding of complex environmental systems.

Finally, in Chapter 5, I reflect on my research directions and the research that has been, in part, under my direction. I motivate my current direction of climate modeling and conclude with a proposed research project to apply AI methods to climate science.

The progression of focus presented in this work, from traditional benchmark tasks, to a specific application in climate science, reflects a broader pattern in artificial intelligence research. The field has often focused on specific, well-defined challenges to develop and validate new methods. Chess served as an early benchmark, culminating in Deep Blue's victory over world champion Garry Kasparov (Seirawan, Simon, and Munakata 1997). Image recognition became a key test through the ImageNet challenge, leading to breakthroughs that transformed computer vision (Krizhevsky et al. 2012). More recently, Go provided the stage for AlphaGo to demonstrate how neural networks could master complex strategic thinking (Silver et al. 2016). Alan Turing anticipated this approach in 1950, writing in Turing (1950) that:

> We may hope that machines will eventually compete with men in all purely intellectual fields. But which are the best ones to start with? Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best. It can also be maintained

> that it is best to provide the machine with the best sense organs that money can buy, and then teach it to understand and speak English. This process could follow the normal teaching of a child. Things would be pointed out and named, etc. Again I do not know what the right answer is, but I think both approaches should be tried.

Today, both paths suggested by Turing have yielded fruit. Machines have mastered chess and Go (Silver et al. 2016), while large language models can understand and generate human language (Brown et al. 2020). Yet these achievements, while impressive, represent only a narrow slice of human intellectual capability. In this manuscript, I argue that the next frontier lies in scientific discovery itself.

This manuscript proposes a course toward that ambitious goal. While much of our methodology development uses traditional AI benchmarks - simulated robots and Atari games - these serve as controlled environments to develop the fundamental capabilities needed for scientific discovery: systematic exploration and interpretable modeling. In Chapter 4, we apply these methods to real scientific challenges in climate science, demonstrating how techniques refined on benchmark tasks can advance our understanding of complex natural systems. Finally, in Chapter 5, I detail directions to respond to the central question of this manuscript - can artificial intelligence help us understand complex natural phenomena and advance human knowledge?

2 Exploration

Most AI algorithms are objective-based. They seek to maximize or minimize some quantifiable goal - modeling planetary movement in the solar system, predicting tomorrow's weather, or controlling a robotic arm. These objectives can be broken down into intermediate goals: how accurately does the planetary model match observed data? Did it rain as much as predicted? How close did the robot get to picking up the object? These evaluations guide algorithms toward solutions that better achieve their goals, predicting rain more accurately or getting the robot to successfully grasp objects.

Natural evolution serves as a powerful inspiration for algorithmic optimization. Life demonstrates a remarkable ability to find solutions in even the most extreme conditions. In the darkness of Antarctic waters, evolution produced aquatic birds capable of incubating their young (Li et al. 2014). In fire-prone forests, plants evolved seeds that require burning to germinate (Lamont, He, and Yan 2019). Even in the scalding temperatures of volcanic vents, certain bacteria thrive (Temple and Colmer 1951). The process of natural selection consistently discovers ways to pass on genetic information across generations, even in seemingly impossible circumstances.

The biological process of natural selection inspired the field of Evolutionary Computation (EC) (Fogel, Owens, and Walsh 1965; Holland 1975; Rechenberg 1978). Algorithms in ECs, which I'll call Evolutionary Algorithms (EAs), simulate key aspects of natural selection by maintaining a population of candidate solutions, often called individuals. These solutions are evaluated according to predefined objectives that quantify their fitness. The most successful individuals are selected and modified through combination or mutation - typically random processes, though they can incorporate problem-specific knowledge. This process repeats iteratively until meeting stopping criteria or computational limits (Mitchell 1998; De Jong 2017).

This manuscript focuses on EAs for two reasons. The first reason is due to their flexibility. The same algorithm can be used to optimize any sort of solution representation towards any sort of goal. The only constraints are that the solution representation can be modified automatically and that the objective function can be used to rank solutions. If those criteria are met, the full family of EAs is open.

The second reason is that, of the many types of algorithms that optimize towards a goal, evolutionary algorithms are quite good at exploring inherently (Lehman, Clune, and Misevic 2018). There are multiple sources of randomness that encourage exploration: the starting population is often generated randomly; the selection of successful individuals is informed by their evaluation and random noise; and the modification of an individual is

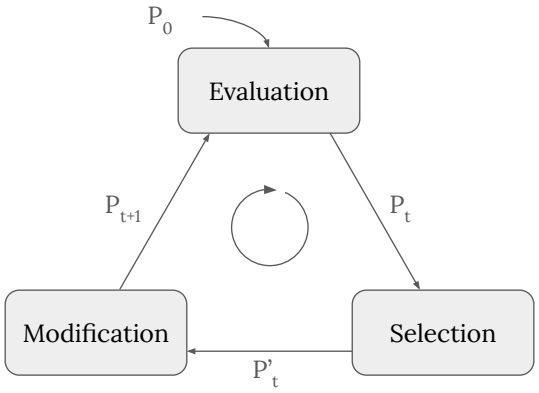


Figure 2.1: Evolution Algorithm loop. An initial population P_0 is evaluated, then certain individuals are selected based on the evaluation. The selection determines a set of individuals to modify, which then creates the next population.

often random. All of this noise makes for good exploration, even if optimization is shooting for a single objective. It is not so surprising that algorithms inspired by the unwieldy force of natural selection rely so heavily on chance.

However, EAs still aim to progress towards some objective, and steady progress toward a goal does not always lead to optimal solutions. Consider again the historical models of planetary motion that relied on circular orbits. Astronomers spent centuries refining these models, adding more circles to correct for observed errors. Yet no amount of incremental improvement would have arrived at Kepler's insight that the orbits were elliptical - this required stepping back to consider a fundamentally different approach. Similarly, objective-based search can become trapped in local optima, dead ends in the search space where small changes no longer improve the evaluation function. The more ambitious the goal, the more difficult it becomes to define an appropriate objective function and the more likely the search is to be deceived by these local optima.

The core issue is that the objective function may not reward the intermediate steps that ultimately lead to the best solutions. Consider a Chinese finger trap - while the goal is to free one's fingers, pulling them apart (the most direct action) yields no progress. Counter-intuitively, one must first push the fingers together, seemingly moving away from the goal, to solve the puzzle. The trap is deceptive because progress requires temporarily moving further from the apparent objective (Lehman and Stanley 2011a).

An alternative to objective-based search is open exploration (Taylor et al. 2016). Methods like open-ended evolution and Novelty Search (Lehman and Stanley 2011a) maximize behavioral diversity rather than a specific objective. They operate with minimal constraints (Soros, Cheney, and Stanley 2016), allowing properties to emerge naturally - similar to biological evolution, which has no explicit objective beyond the propagation of genetic information. This approach often leads to creative solutions that objective-based methods might never discover.

However, pure exploration has its own limitations. While it can generate endless novelty, much of that novelty may be irrelevant to solving practical problems. Consider an algorithm that is generating bowls of oatmeal by manipulating the placement of oat flakes. One could create an infinite number of unique oatmeal configurations by varying the position of each flake. While each bowl would be technically novel, these differences are not interesting if our goal is simply to make edible oatmeal (Earle, Togelius, and Soros 2021; Compton 2016). Pure exploration can be inefficient, spending computational resources on variations that don't advance useful capabilities.

This tension between exploring new possibilities and exploiting known solutions is known as the exploration-exploitation trade-off (Audibert, Munos, and Szepesvári 2009; Sutton and Barto 2018). Exploitation - moving toward an objective - risks getting stuck in suboptimal solutions. Exploration - seeking novelty - might never find practically useful solutions. Different fields of AI handle this trade-off in different ways, from planning algorithms to Reinforcement Learning (RL). In this manuscript, I focus on this trade-off in EC, where exploration, creativity, and innovation in solutions has long been a focus of the field (Lehman et al. 2020).

In this chapter, I illustrate the current challenges around exploration using two detailed examples of exploration algorithms. These examples are drawn from PhD theses under my supervision: those of Paul Templier and Paul-Antoine le Tolguenec. I begin with a background on exploration algorithms in EC, then present a contribution that achieves quality with just enough diversity for exploration. This approach requires defining what type of exploration is desired, but such definitions aren't always obvious. To address this, I also present Curiosity-ES, a method that uses artificial curiosity to guide exploration. Finally, I give an example of how these exploration techniques can be applied to find software flaws in critical aviation systems, showing how principled exploration enables practical innovation.

2.1 Quality and Diversity

A core challenge in exploration is how to measure it. How does one express the set of solutions already explored? How does one express that one solution is different than another, or by how much they differ? One common approach is to first characterize a solution through its behavior profile, a description of how it solves the problem. Consider a robot navigating a maze: a complete description of its behavior would encompass the full path taken. However, this level of detail provides an unwieldy amount of information that may not serve the ultimate goal of reaching the maze exit. A more practical approach is to use a simpler behavioral descriptor, such as the robot's final position.

We can formally express this by defining a behavioral function $b : \theta \mapsto \mathbb{R}^m$ that maps an individual θ to a set of m features relevant to the task. For maze navigation, the behavior of an individual controller, also termed a "policy," can be characterized by its terminal coordinates $b(\theta) = (x_{\text{final}}, y_{\text{final}})$. This behavioral representation enables measuring the distance between two behaviors through a distance function $d : \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$.

With behavior characterized and distance defined, exploration can be achieved by searching for solutions that are different from previously found solutions. This is the idea behind Novelty Search algorithms; these methods search by computing a novelty score

for each individual (Lehman and Stanley 2011a). This score quantifies how different an individual's behavior is compared to others in the population or archive. Typically, this is calculated as the average distance to the k-nearest neighbors in behavior space:

$$\text{novelty}(\theta) = \frac{1}{k}\sum_{i=1}^{k} d(b(\theta), b(\theta_i)) \tag{2.1}$$

where $b(\theta_i)$ represents the k-nearest neighbors of $b(\theta)$ in behavior space. Novelty search algorithms use this score for selection, focusing purely on maximizing behavioral diversity without considering the original objective function.

However, as discussed above, exploration alone can lead to inefficiently looking through solutions that, while different or novel, don't help in solving a problem. In most cases, there is some defined goal that an algorithm is used for, and exploration should be used to innovate towards that goal. An early approach along this line, built on Novelty Search, introduced local competition: similar individuals, based on their behavior, will compete based on their objective fitness (Lehman and Stanley 2011b). In that way, both the quality and diversity of solutions increase over time; these methods are known as Quality Diversity (QD) algorithms (Pugh, Soros, and Stanley 2016; Cully and Demiris 2017).

MAP-Elites (Cully et al. 2015b) is a foundational QD algorithm that builds an archive of high-performing but behaviorally diverse solutions. The algorithm partitions the behavior space into discrete cells. When evaluating an individual θ, it is placed into the cell corresponding to its behavior $b(\theta)$, say cell b_i. If another individual ϕ already occupies cell b_i with a similar behavior, the two compete based on their fitness f on the main optimization objective. If the new individual has superior fitness (for maximization, for example, $f(\theta) > f(\phi)$), it replaces the incumbent in cell b_i. To generate new individuals, the algorithm randomly selects a parent from a filled behavior cell, improves upon it, and repeats this process.

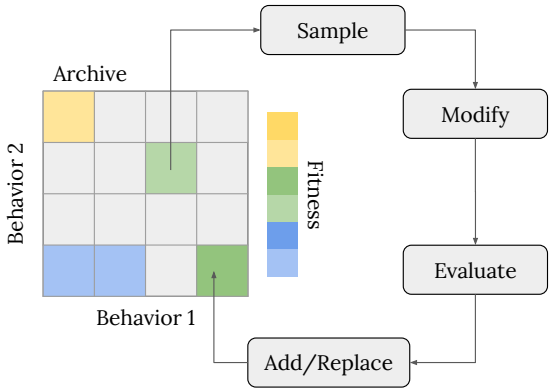


Figure 2.2: The MAP-Elites algorithm. Individuals are sampled from an archive, then modified and evaluated. The new individual is placed into the archive cell corresponding to its behavior if the cell is empty or the new individual has a better fitness than the individual occupying the cell.

MAP-Elites was first demonstrated for the exploration of the full range of possible walking behaviors for a robot. When the robot was later damaged, the diverse archive of walking behaviors enabled rapid adaptation by switching to an alternative gait (Cully et al. 2015b). This success spawned numerous derivative methods and applications in domains from locomotion to robotic arm control (Cully and Demiris 2017; Pugh, Soros, and Stanley 2016; Sigaud 2023). However, while MAP-Elites excels at exploring the full spectrum of possible solutions, it can be inefficient at optimizing solution quality.

Consider learning to play chess. One could study every possible movement pattern, including clearly suboptimal ones. Take the "Bongcloud" opening, where a player advances the pawn in front of their King and then moves the King forward early in the game. This is widely considered one of the worst possible openings - it reduces winning chances and creates numerous ways to lose. Exploring all possible winning strategies from this position would be far less efficient than focusing on more promising openings. For effective chess mastery, one would rather explore a limited diversity of useful possibilities. This idea motivates the algorithm we'll examine next, Quality with Just Enough Diversity (JEDi) (Templier, Grillotti, et al. 2024).

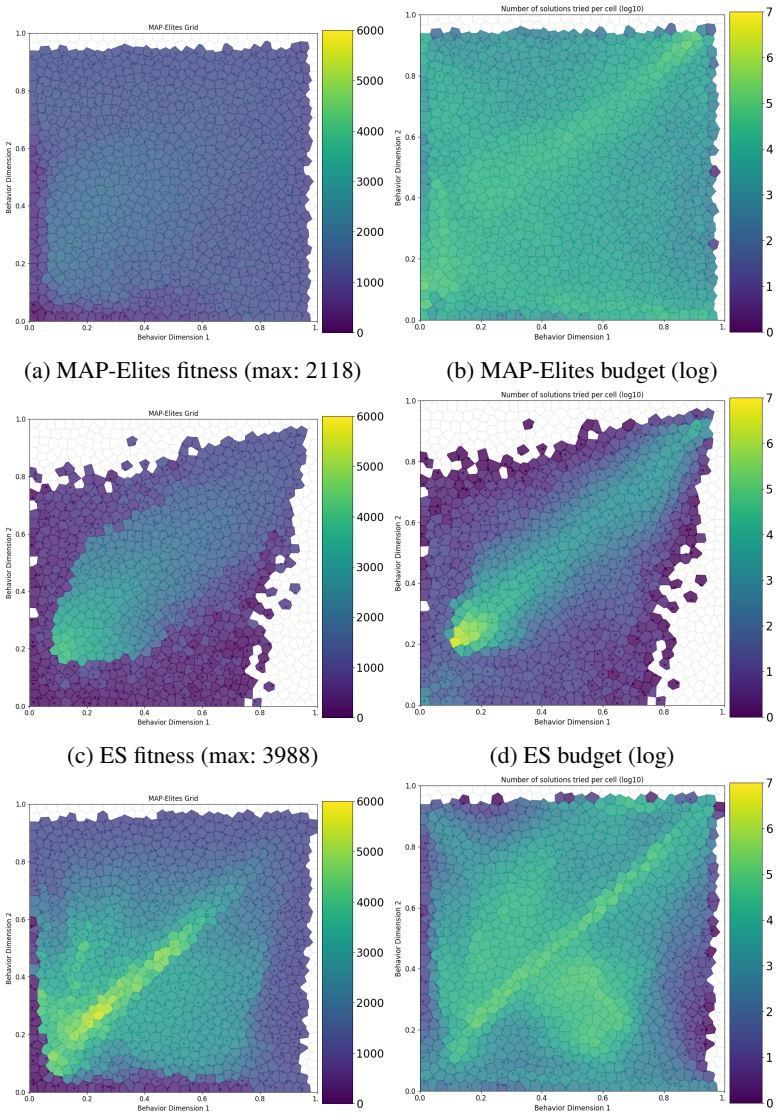
2.2 Just Enough Diversity

In MAP-Elites, exploration is driven by randomly selecting "parent" solutions from the archive as starting points for new variations. This approach ensures broad coverage of possible behaviors, as even rarely-seen behaviors have an equal chance of being selected for further exploration. In contrast, purely objective-focused algorithms like Evolutionary Strategy (ES) concentrate on finding better solutions according to the fitness function (Hansen and Ostermeier 2001). ESs are a type of EAs that work especially well for continuous optimization, as they estimate and refine the distribution of a population around the objective function (Rechenberg 1978). As such, they explore around high-performing individuals, potentially missing behaviors that don't immediately appear promising.

To illustrate this difference, consider how these algorithms perform on a walking robot task. MAP-Elites discovers a wide variety of walking gaits, from slow but stable movements to rapid but unstable ones. It maintains this diversity because it gives equal opportunity to all discovered behaviors. An ES, however, quickly converges to a single type of movement that maximizes forward speed, discarding alternative gaits that might prove useful in different circumstances.

This difference is visualized in Figure 2.3, which shows the behavior archives of the two methods after optimization, as well as the number of evaluations used in each part of the behavior archive. While MAP-Elites achieves a higher coverage of all possible behaviors, it spends a large amount of evaluations on behaviors that do not have high fitness. The ES, on the other hand, doesn't evaluate most behaviors for long, instead concentrating evaluations at one behavior point. While this isn't highly exploratory, it does result in a much higher maximum fitness for this task.

What if we could combine the best aspects of both approaches? Rather than maintaining equal diversity everywhere or focusing solely on optimization, we could intelligently decide which behaviors are worth exploring further. This is what happens in JEDi, an



(a) MAP-Elites fitness (max: 2118)

(b) MAP-Elites budget (log)

(c) ES fitness (max: 3988)

(d) ES budget (log)

(e) JEDi fitness (max: 5535)

(f) JEDi budget (log)

Figure 2.3: Fitness and number of solutions tried in each behavior cell for MAP-Elites, ESs, and JEDi on a walking robot task. Figure from Templier, Grillotti, et al. (2024).

algorithm that learns which areas of the behavior space tend to yield useful solutions and focuses exploration around those behaviors.

2.2.1 JEDi

The JEDi algorithm operates in cycles of exploration and optimization. Like MAP-Elites, it maintains an archive of diverse solutions, but it uses this archive in a more strategic way. Each cycle begins by analyzing the relationship between behaviors and performance using

machine learning - specifically, a Gaussian Process model (Williams and Rasmussen 1995) that learns to predict how well a solution might perform based on its behavior. This learned model then helps select promising new behaviors to target.

Think of this process like a scientist deciding which experiments to run next. Rather than trying everything possible or focusing only on what's already known to work, the scientist uses their understanding of previous results to make educated guesses about which new directions might be most fruitful. Similarly, JEDi uses its learned model to identify behaviors that show promise, either because they're predicted to perform well or because there's uncertainty about their potential.

Once target behaviors are selected, JEDi launches focused optimization efforts around solutions in the archive that are similar to these targets. These local searches balance two objectives: improving the solution's performance on the main task while steering it toward the desired target behavior. This dual-objective optimization is handled through a weighted scoring system that can emphasize either exploration of new behaviors or exploitation of known good solutions. The full algorithm of JEDi is described in Algorithm 1, and in Templier, Grillotti, et al. (2024) for more detail.

The balance between exploration and exploitation is controlled by a parameter α. When α is high, the algorithm prioritizes reaching new target behaviors; when low, it focuses more on improving performance. This parameter can be gradually adjusted during the search, similar to how a scientist might start with broad exploration and then narrow their focus as they better understand the problem.

Algorithm 1 JEDi

Require: iterations L, emitters n_{ES}, generations N, target selection $\mathcal{T}$, repertoire R, evaluation F, population λ, solution θ, parameter distribution σ, target weight α

Initialize R with n_{init} random genomes
for $l = 1$ to L **do** ▷ JEDi iterations
  for $i = 1$ to n_{ES} **do** ▷ ES Emitters
    $b_i = \mathcal{T}_l(R)$ ▷ Sample target behavior
    $\theta_1 = \arg\min_{\theta, b \in R} \|b_\theta - b_i\|$ ▷ Select start solution from the repertoire
    for $k = 1$ to N **do** ▷ N generations
      for $i = 1, \lambda$ **do** ▷ λ individuals
        $\theta_k^i \sim \mathcal{N}(\theta_k, I\sigma)$ ▷ sample individual
        $f_{\theta_k^i}, b_{\theta_k^i} = F(\theta_k^i)$ ▷ Evaluate solutions
        $R \leftarrow (\theta_k^i, f_{\theta_k^i}, b_{\theta_k^i})$ ▷ Update the repertoire
        $s_k^i = f_{\theta_k^i} + \alpha \|b_{\theta_k^i} - b_i\|$ ▷ Compute scores
      end for
      $\theta_{k+1} \leftarrow \theta_k + \frac{1}{\sigma\lambda} \sum_{j=1}^{\lambda} (\theta_k^j - \theta_k) s_j$ ▷ Update ES center
    end for
  end for
  $\mathcal{T}_{l+1} \leftarrow (\mathcal{T}_l, R)$ ▷ Update target behavior model
end for

Consider training a robot to walk efficiently while maintaining stability. Traditional optimization might focus solely on speed, leading to fast but risky gaits. MAP-Elites would explore all possible walking styles equally, including many impractical ones. JEDi, however, would learn that certain combinations of stability and speed tend to yield good results, and would focus its exploration around these promising regions while still maintaining enough diversity to discover unexpected solutions.

This approach has proven particularly effective on complex control tasks where the relationship between behavior and performance isn't obvious in advance. By learning this relationship during the search process, JEDi can guide exploration toward behaviors that are both novel and likely to be useful.

2.2.2 Experimental evaluation

The effectiveness of JEDi was evaluated on two types of challenges: maze navigation and robotic control tasks. These environments represent different aspects of the exploration-exploitation challenge. In maze navigation, the algorithm must discover paths through complex environments where the most direct route may be blocked. In robotic control, it needs to find stable and efficient ways to move various types of robots, from a simple half-cheetah model to more complex walking robots.

An experimental comparison of JEDi to other algorithms is presented in Figure 2.4. JEDi shows competitive performance to exploration-focused algorithms like MAP-Elites on tasks that require high exploration, the maze navigation tasks. When the task requires less exploration and more exploitation, JEDi is competitive with the ES that is fully objective-based.

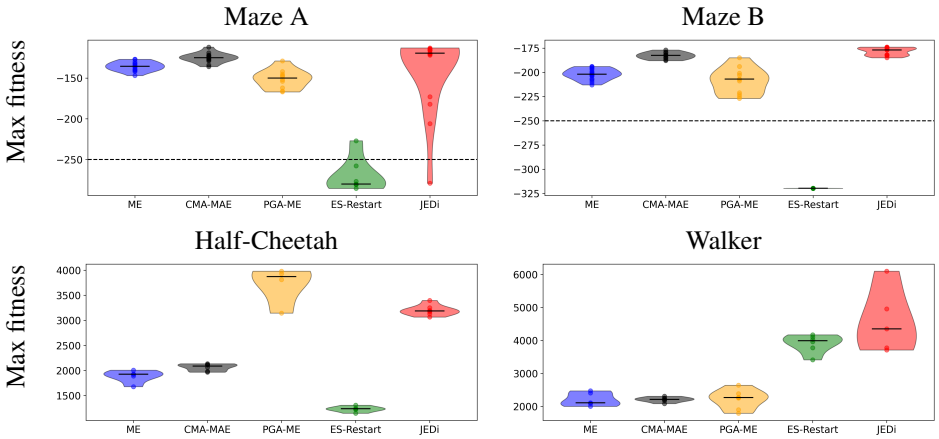


Figure 2.4
Violin plots of final max fitness results for maze exploration (row 1) and robotics control tasks (row 2). The solid line in each violin is the median. Reaching the dotted line at -250 in a maze means an agent has reached the target. Figure adapted from Templier, Grillotti, et al. (2024).

A particularly interesting case emerged in the "Walker" environment, in which the solution controls the movement of a robot, with the goal of making the robot walk forward. Both ESs and JEDi discovered solutions that achieved high performance, but through

markedly different means. The ES converged to a single, highly optimized walking gait. JEDi, in contrast, found multiple distinct ways of stable walking, each with its own advantages. This diversity provides robustness, offering alternative strategies if conditions change.

The budget distribution visualizations reveal how JEDi allocates its computational resources more intelligently than either MAP-Elites or ESs. As shown previously in Figure 2.3, MAP-Elites spreads its budget evenly and ES concentrates it narrowly. JEDi adapts its exploration based on learned patterns of success, focusing more resources on behaviors that show promise while maintaining enough diversity to discover new possibilities.

2.2.3 Discussion

JEDi represents a step toward automated discovery by demonstrating how algorithms can learn to explore efficiently. Like a skilled researcher, it doesn't simply try everything possible or focus solely on what's already known to work - it learns which directions are worth investigating and adapts its strategy accordingly. This balance between broad exploration and focused optimization mirrors how human scientists work, combining systematic investigation with informed intuition about promising research directions.

Other researchers have recognized this need for intelligent exploration in automated discovery. The Covariance Matrix Adaptation MAP-Elites (CMA-ME) algorithm (Fontaine et al. 2020) balances exploration from MAP-Elites with exploitation from the precise local optimization method Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen and Ostermeier 2001). Fontaine and Nikolaidis (2023) builds on CMA-ME by introducing an annealing method that regulates between exploration and exploitation. Paolo et al. (2021) tackles the same balancing issue in cases where the objective function only rewards individuals in rare conditions, termed sparse rewards. Each of these approaches shares JEDi's core insight: effective discovery requires both breadth of exploration and depth of understanding.

Looking forward, several promising directions could further enhance JEDi's capabilities for automated discovery. One is the integration of more sophisticated machine learning models to better predict which behaviors warrant investigation. Another is the development of adaptive methods for managing the exploration-exploitation trade-off, perhaps drawing inspiration from how human researchers adjust their research strategies as they gain understanding of a problem domain.

The interaction between human actors and algorithms like JEDi also deserves further exploration. Currently, human expertise is integrated into JEDi through the definition of problem representation, the objective function, and the behavior descriptions. While JEDi discovers solutions autonomously within these definitions, its discoveries might be even more valuable when combined with human insight. For instance, when JEDi finds multiple ways to solve a problem, human experts might recognize patterns or principles that could guide future research.

In the context of automated scientific discovery, JEDi demonstrates a crucial principle: effective exploration isn't just about trying everything possible, but about learning where to look. By combining the breadth of QD algorithms with the focused optimization of ES, it

points toward a future where automated systems can not only assist in scientific discovery but actively guide it in promising directions.

This theme of guided exploration sets up our next investigation into curiosity-driven search, where we'll examine how intrinsic motivation can drive discovery even without explicit behavior descriptions. The transition from JEDi's learned guidance to curiosity-based exploration represents another step toward more autonomous discovery systems.

2.3 Curiosity

Children explore their world through play, driven by an innate curiosity about things they don't yet understand. When a toddler encounters a new toy, they might shake it, taste it, or drop it repeatedly - not to achieve any particular goal, but to learn how it behaves. This intrinsic motivation to explore the unknown turns out to be a good principle for machine learning as well.

Traditional approaches to exploration in RL and EC often rely on explicit rewards or behavior descriptions to guide search. But what if we could create algorithms that, like children, are naturally drawn to explore what they don't understand? This insight led to the development of curiosity-driven exploration methods (Schmidhuber 1991; Pathak et al. 2017).

Machine learning algorithms are usually trained to predict some quantity, so the expression of curiosity can be formulated as situations that an agent finds difficult to predict. Just as a child might be fascinated by a toy with unexpected behaviors, these algorithms are drawn to parts of their environment where their predictions fail. This creates a natural drive to explore novel situations without requiring any external guidance about what constitutes interesting or valuable behavior.

Early implementations of this concept used simple counting methods - rewarding agents for visiting previously unseen states (Brafman and Tennenholtz 2002). More sophisticated approaches emerged with the development of prediction-based curiosity (Burda et al. 2018; Eysenbach et al. 2019). In Pathak et al. (2017), an agent maintains an internal model that tries to predict how the environment will respond to its actions. When the model's predictions are inaccurate, it signals that the agent has found something novel worth exploring.

This predictive approach to curiosity has compelling advantages. First, it scales naturally to complex environments where explicitly counting states or categorizing behaviors becomes impractical. Second, it creates a curriculum where agents progress from simple to complex behaviors as their predictive models improve.

The algorithm we developed, called Curiosity-ES, combines these ideas with ESs. This method uses the formulation of Curiosity from Pathak et al. (2017), an Intrinsic Curiosity Module (ICM) - a neural network that learns to predict the consequences of the agent's actions. The prediction error from this module serves as an bonus fitness value for the ES, encouraging the evolution of policies that seek out novel, unpredictable situations.

2.3.1 Curiosity-ES

Curiosity-ES combines the exploratory nature of curiosity-driven learning with the robust optimization capabilities of ESs. The algorithm works by maintaining two parallel systems:

an evolutionary process that generates and improves possible solutions, and a curiosity module that guides exploration toward novel behaviors.

The evolutionary component maintains a population of candidate solutions and gradually improve them through the previously described process inspired by natural selection. In our case, each "individual" in the population is an Artificial Neural Network (ANN) that controls how an agent acts in its environment. The parameters of the neural network are optimized by the ES, modifying the agent's behavior. We use a classic ES for continuous optimization, CMA-ES (Hansen and Ostermeier 2001), in its separable form (Ros and Hansen 2008).

The innovation in Curiosity-ES comes in how we evaluate the solutions. Traditional evolutionary algorithms measure how well each solution achieves the desired goal - for instance, how quickly a robot reaches its destination. Curiosity-ES, however, considers two factors: the traditional "extrinsic" reward for achieving goals, and an "intrinsic" reward based on Curiosity.

This curiosity reward is computed using the ICM from Pathak et al. (2017), which consists of three ANNs working together to predict the next state. The three networks are an encoder that converts raw environmental states into a more manageable form, a forward model that tries to predict how actions will affect the environment, and an inverse model that helps ensure the encoder learns meaningful features.

As an agent interacts with its environment, the ICM attempts to predict each transition - each change from one state to another resulting from the agent's actions. The prediction error serves as our measure of curiosity. Just as a child might be more interested in a toy that behaves unexpectedly, our algorithm assigns higher intrinsic reward to transitions it finds difficult to predict. The full algorithm of Curiosity-ES is presented in Algorithm 2.

Algorithm 2 Curiosity-ES

Require: $\mu, \lambda, \sigma, \alpha, \alpha_{ICM}, \beta, \gamma, m, p, N, \varphi, w_j$
  Initialize : $\theta_0, w_f, w_i, w_e, \mathcal{D}$
  for $k = 1, N$ **do** ▷ N generations
    for $i = 1, \lambda$ **do** ▷ λ individuals
      $\theta_k^i \sim \mathcal{N}(\theta_k, I\sigma)$ ▷ sample individual
      $f_e, \tau = f(\theta_k^i), \Gamma(\theta_k^i)$ ▷ fitness and trajectory
      $f_i = \sum_{t=0}^{T-1} \gamma^{T-1-t} \parallel F_{w_f}(\phi_{w_e}(s_t), a_t) - \phi_{w_e}(s_{t+1}) \parallel_2$ ▷ curiosity over trajectory
      $f_{\theta_k^i} = \varphi(f_e - \mu_{f_e})/\sigma_{f_e} + (1-\varphi)(f_i - \mu_{f_i})/\sigma_{f_i}$ ▷ global fitness
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_{j+1}, a_j, s_j) \sim U(\tau) | j \in [0, m]\}$ ▷ add m transition to $\mathcal{D}$
    end for
    $\{\theta_j | j \in [1, \lambda]\} = sort((\theta_k^i, f_{\theta_k^i}), i \in [1, \lambda])$ ▷ sort individuals by fitness
    $\nabla_{\theta_k} = \frac{1}{\sigma\mu} \sum_{j=1}^{\mu} (\theta_j - \theta_k) w_j$ ▷ estimate gradient
    $\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_{\theta_k}$ ▷ update θ
    for $l = 1, p$ **do**
      $L_{ICM}(w_i, w_f, w_e) = (1-\beta)L_I + \beta L_F$ ▷ ICM loss over $\mathcal{D}$
      $w \leftarrow w + \alpha_{ICM} \nabla_w L_{ICM}, w \in (w_e, w_i, w_f)$ ▷ update ICM weights
    end for
  end for

The combination of traditional rewards and curiosity is controlled by a single parameter φ, which determines how much weight to give each factor. When φ is close to 1, the system focuses mainly on achieving explicit goals. When it's closer to 0, exploration through curiosity takes precedence. In practice, we found that a value of 0.8 works well - enough emphasis on goals to make progress, but sufficient curiosity to maintain exploration.

The result is an algorithm that can discover solutions in environments where traditional approaches struggle - particularly in situations with sparse rewards, where feedback about success is rare and difficult to find. By following its curiosity, the system can learn to navigate complex environments even before it discovers how to achieve its ultimate objectives.

2.3.2 Experimental Evaluation

To understand how curiosity-driven exploration works in practice, we tested Curiosity-ES on two types of challenges: maze navigation and robotic control. These environments represent different aspects of the exploration challenge. Maze navigation requires discovering paths through complex environments where the most direct route may be blocked. Robotic control tasks demand finding stable and efficient ways to move various types of robots, from a simple half-cheetah model to more complex walking robots.

The results on the maze environments show how Curiosity leads to exploration. Figure 2.5 shows the final states reached by CMA-ME (Fontaine et al. 2020), Novelty Search Evolutionary Strategy (NS-ES) (Conti et al. 2018), and Curiosity-ES. Consider in particular the "SNAKE" maze, where the path to the goal follows a winding S-shaped pattern. Traditional optimization methods tend to get stuck here because moving directly toward the goal (which seems like the right thing to do) actually leads to dead ends. Quality-diversity algorithms like CMA-ME can eventually find solutions through extensive exploration, but they spend considerable computational effort exploring areas that don't lead to successful paths.

Curiosity-ES approached these mazes differently. Rather than being drawn toward the goal or trying to catalog all possible behaviors, it was naturally attracted to areas where its predictions failed. This led to an interesting pattern of exploration: the system quickly became "bored" with the repetitive corridors of the SNAKE maze where movement was predictable. Instead, exploration is focused on corners and the end of the maze, where the consequences of actions were harder to predict.

The effectiveness of this approach becomes clear when we look at not just whether solutions were found, but how efficiently they were discovered. While MAP-Elites would often find paths to the goal first, Curiosity-ES discovered notably more efficient routes. This makes intuitive sense - by maintaining curiosity about transitions through the maze, the system naturally seeks out smoother, more efficient paths rather than being satisfied with the first solution it finds.

In Figure 2.6, we can see the total reward gained by policies optimized by various methods on robotic control tasks, as well as the percentage of possible terminal states reached. In these environments, agents needed to learn complex behaviors like manipulating objects or maintaining balance while walking. Traditional optimization methods typically converge to a single type of movement that maximizes immediate performance. ESs, for instance, might find one stable walking gait and continuously refine it. Curiosity-ES, in contrast,

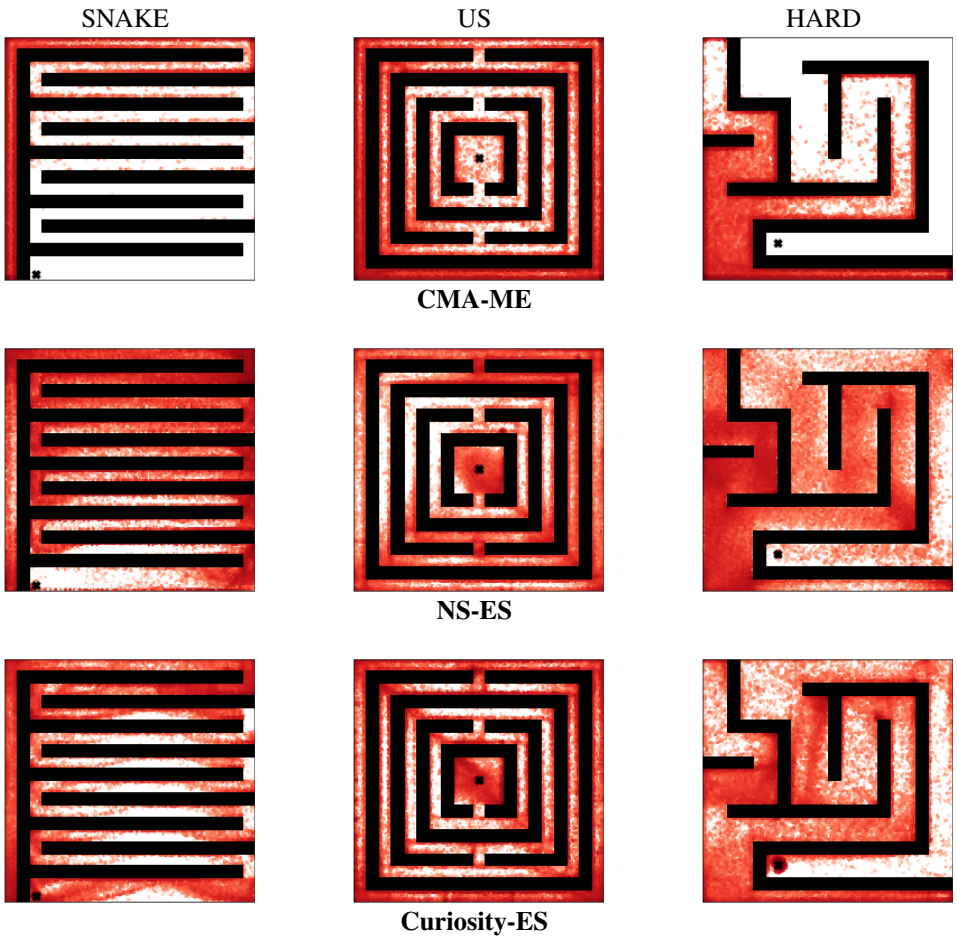


Figure 2.5: Final states reached in the Maze Navigation task by CMA-ME (top), NS-ES (middle), and Curiosity-ES (bottom). Individual points are semi-transparent; color indicates density. Figure adapted from Le Tolguenec et al. (2022).

discovered multiple distinct ways of achieving its goals. Surprisingly, that results not only in greater exploration, but in a quicker convergence to high-reward behaviors.

These results point to a broader insight about automated discovery: the most valuable solutions aren't always the most obvious ones. By maintaining curiosity about unexplored possibilities, systems can discover innovative approaches that might be overlooked by more directly goal-focused methods. This capability becomes particularly important as we move from controlled environments to real-world applications, where adaptability and robustness are crucial.

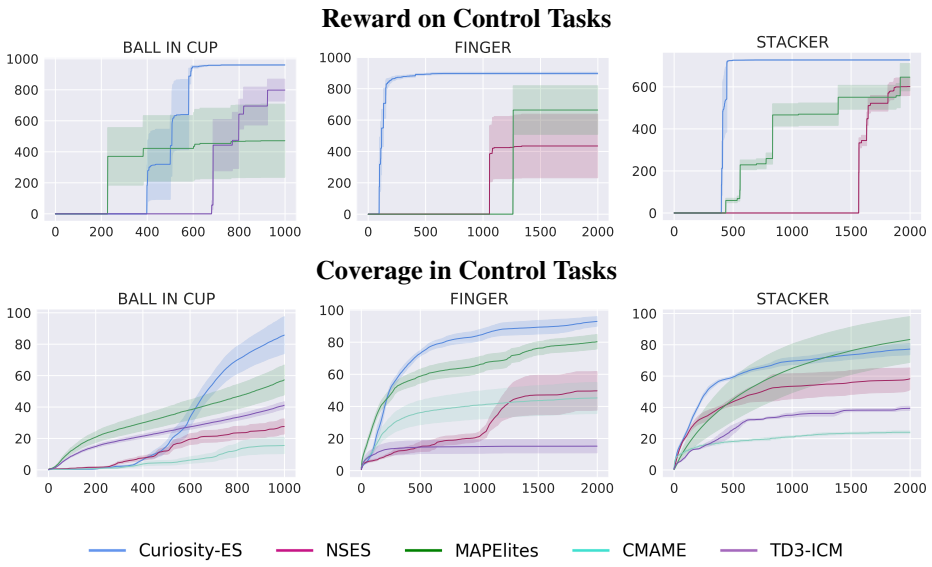


Figure 2.6: Top: the sum reward of the best solution in each generation on the three control task environments. Bottom: The percentage of discrete terminal behaviors reached throughout search. Figure adapted from Le Tolguenec et al. (2022).

2.4 Exploring Critical Systems

While Curiosity-ES showed promising results on standard robotics and navigation tasks, it was developed for the testing of critical systems. Specifically, the goal was to use exploration to find flaws in an airplane software system. This work is presented fully in Le Tolguenec, Rachelson, Besse, et al. (2024).

High-assurance systems, like those in aviation, must function reliably even under extreme conditions. These systems typically employ redundancy and protective mechanisms to prevent failures. Yet this very complexity, intended to improve safety, can paradoxically introduce new vulnerabilities. Consider automated vehicles experiencing "phantom braking" - engaging emergency brakes when no actual danger exists - or railway systems where redundant safety units unexpectedly interfere with each other. These examples highlight how dependability mechanisms, despite their crucial role in safety, can create new and subtle failure modes.

In this study, we apply Curiosity-ES to search for potential failures in an aircraft control system that uses paired command and monitoring units. This architecture employs two independent units to compute control orders, comparing their outputs to detect faults. While the units may disagree slightly due to normal variations, excessive disagreement triggers a fault detection. The challenge lies in finding corner cases where properly functioning units might still produce concerning disagreements - a task that has proven particularly difficult for traditional testing approaches.

This scenario presents an ideal test case for curiosity-driven exploration. The system's complexity - arising from multiprocessor architecture, varying computation periods, and

floating-point calculations - makes traditional formal analysis impractical. Curiosity-ES, with its ability to systematically explore unfamiliar system states, offers a fresh approach to discovering potential vulnerabilities that might be overlooked by conventional testing methods.

2.4.1 Critical software system

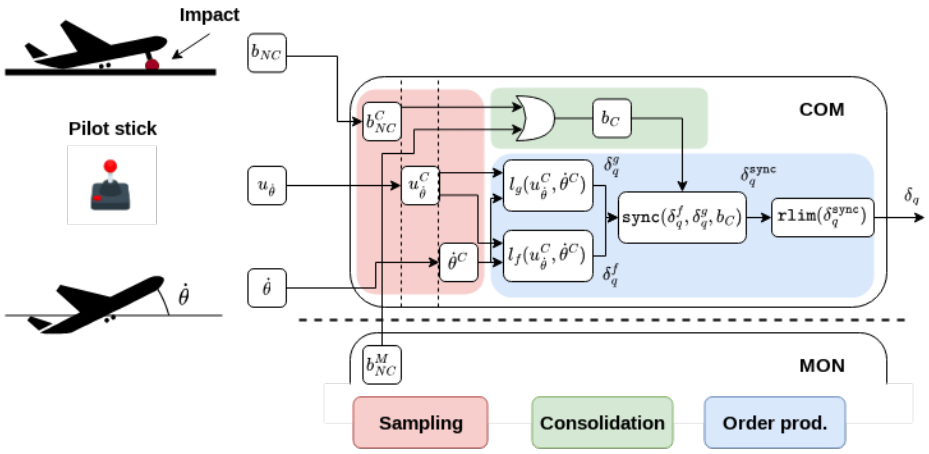


Figure 2.7: The COM architecture, which calculates control orders for the elevator surface. The MON architecture is equivalent and provides the binary variable of ground contact to the COM. Figure from Le Tolguenec, Rachelson, Besse, et al. (2024).

Aircraft control systems must balance performance with safety. The Command/Monitor (COM/MON) architecture achieves this through a system of checks and balances. As shown in Figure 2.7, each critical control function is performed by two independent units - a command unit that issues control orders and a monitoring unit that verifies these decisions.

Both units independently compute control orders for the aircraft's surfaces based on sensor data. In our study, we focus on the elevator control during landing, where precise adjustments are crucial. While the units perform identical calculations, they operate on slightly different time scales due to their independent clocks. This asynchrony, though minor, creates subtle differences in how each unit samples and processes incoming data.

The specific scenario we examine, illustrated in Figure Figure 2.8, involves the critical transition between flight and ground modes during landing. When the aircraft touches down, the control system must rapidly switch from flight-optimized to ground-optimized parameters. This transition relies on detecting ground contact through sensors in the landing gear. The scenario includes the possibility of a "bounce" - where the aircraft briefly returns to air after initial ground contact, requiring another transition between control modes.

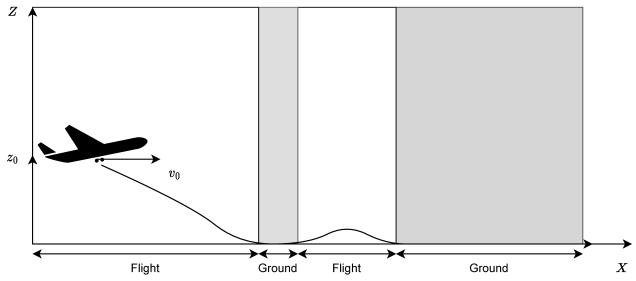


Figure 2.8: The flight scenario. The aircraft is initialized in a landing pattern where bouncing is possible based on the given controls. The different activation phases of the flight control law and ground control law are shown in background color. Figure from Le Tolguenec, Rachelson, Besse, et al. (2024).

Each unit samples various flight parameters - pitch rate commands, actual pitch rates, and ground contact signals - at specific intervals. They then process this data through control laws that differ between flight and ground modes. A synchronization mechanism helps manage transitions between these modes, and rate limiters ensure smooth control surface movements. While this architecture provides robust fault detection, it also creates opportunities for the units to temporarily disagree about the aircraft's state, particularly during mode transitions.

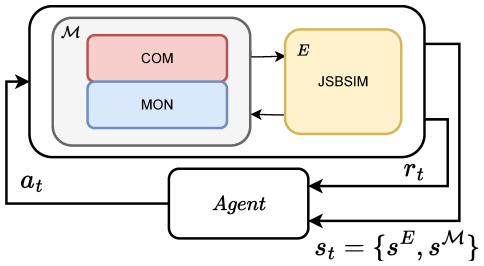


Figure 2.9: Test architecture overview. Figure from Le Tolguenec, Rachelson, Besse, et al. (2024).

To systematically search for potential disagreements between the COM and MON units, we developed a testing framework based on the principles of Adaptive Stress Testing (R. Lee et al. 2015). As shown in Figure 2.9, this framework combines our COM/MON system model with a high-fidelity aircraft simulator, JSBSim (Berndt 2004). An intelligent agent interacts with this integrated simulation, playing two roles simultaneously: it acts as the pilot by controlling the aircraft's surfaces, and it manipulates the timing differences between the COM and MON units.

The JSBSim flight dynamics model provides realistic aircraft behavior, while our COM/-MON implementation captures the intricacies of the control system architecture. These components interact in a closed-loop system, with the simulator operating synchronously

and deterministically. The artificial timing differences between units are created by adjusting their internal clocks within the simulation - a crucial capability for exploring potential failure modes.

Finding edge cases in this system is particularly challenging because potential failures emerge from complex interactions between multiple elements: timing differences in how units sample sensor data; mode transitions triggered by ground contact; the continuous physics of aircraft motion; and multiple control laws operating at different frequencies. This combination of discrete mode switches and continuous dynamics creates a vast space of possible behaviors that must be systematically explored to ensure safety.

2.4.2 Finding failure cases

We tasked different search methods with finding scenarios that could cause significant disagreement between the COM and MON units. The methods included a simple Monte Carlo baseline that randomly samples control inputs (Mooney 1997), a standard Evolution Strategy (sCMA-ES) (Ros and Hansen 2008), and Curiosity-ES (Le Tolguenec et al. 2022). Each method could modify two key parameters: the pilot's pitch rate commands and the timing difference between the COM and MON units. Success was measured by the magnitude of disagreement between the units' computed control orders.

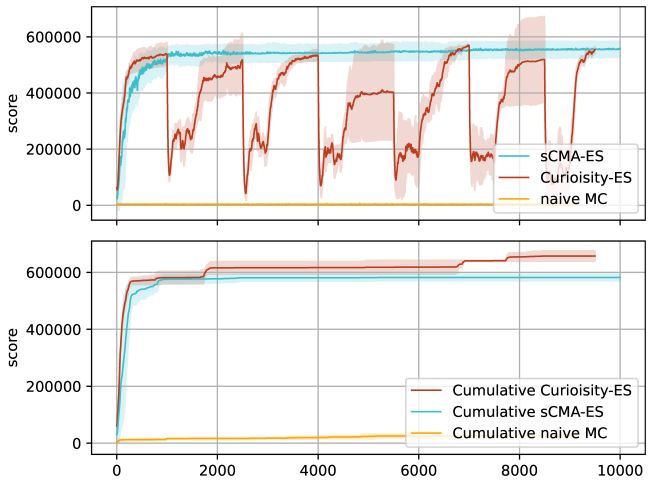


Figure 2.10: (top) Maximum fitness per generation during evolution for the three different methods. (bottom) Maximum fitness accumulated over evolution. Lines indicate the mean over 5 trials, and ribbons the standard deviation. Figure from (Le Tolguenec, Rachelson, Besse, et al. 2024).

Both evolutionary approaches significantly outperformed random sampling, but in different ways. As shown in Figure 2.10, sCMA-ES quickly discovered policies that created large disagreements, showing consistent performance across multiple trials. Curiosity-ES, while more variable in its exploration, ultimately found policies that generated even larger

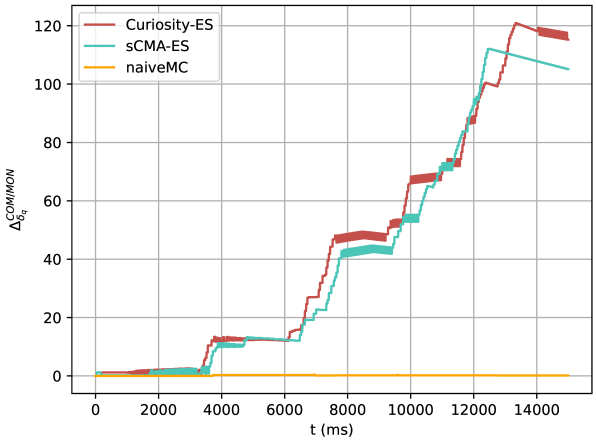


Figure 2.11: Temporal progression of the disagreement $\Delta\delta_q^{C/M}$ for the best policy by each method. Figure from (Le Tolguenec, Rachelson, Besse, et al. 2024).

disagreements. Figure 2.11 illustrates how the best policy from Curiosity-ES maintained higher levels of COM/MON disagreement throughout the landing sequence.

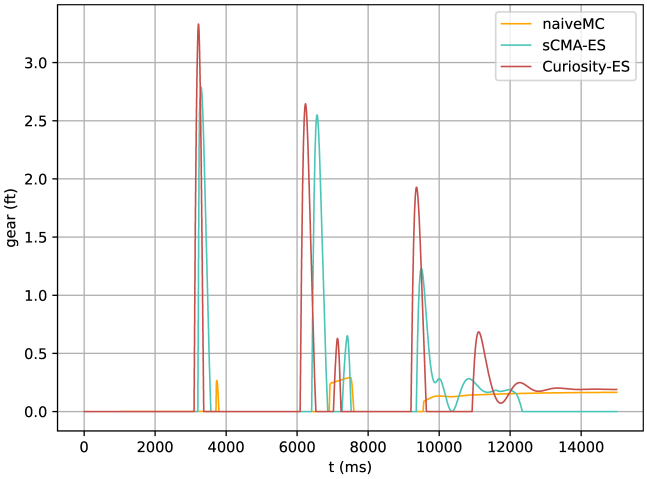


Figure 2.12: Temporal progression of the front gear compression for the three best policies. Both ES methods create many bounces to exploit the synchronization function. Figure from (Le Tolguenec, Rachelson, Besse, et al. 2024).

Analysis of the successful policies revealed an intriguing pattern: the evolutionary methods had discovered that rapid bouncing during landing could exploit the system's

mode-switching mechanism. As shown in Figure 2.12, both ES methods learned to create multiple bounces, unlike the smoother landings produced by random sampling. These bounces forced frequent transitions between flight and ground control modes. Since the COM and MON units operated on slightly different timescales, they could temporarily disagree about which mode to use, amplifying their differences in computed control orders.

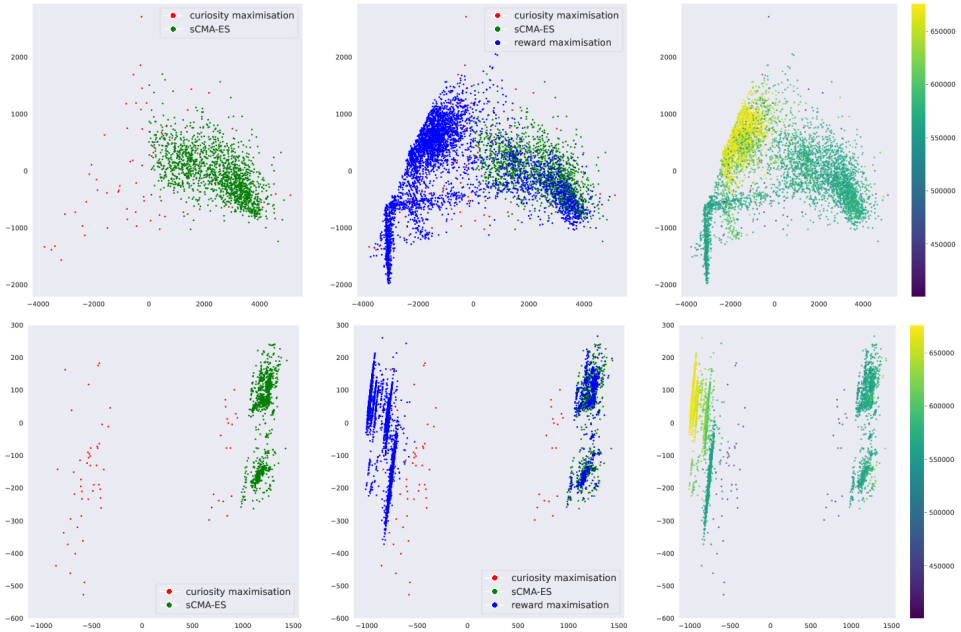


Figure 2.13: Visualization of policy diversity using dimensionality reduction. Each point represents a different policy. Top row shows the distribution of control strategies when analyzed by their frequency components. Bottom row shows how these policies affect the system state. Colors in the rightmost plots indicate the level of COM/MON disagreement achieved. Figure from (Le Tolguenec, Rachelson, Besse, et al. 2024).

While both evolutionary methods found the bouncing strategy, Curiosity-ES discovered a broader range of failure cases. To visualize this diversity, we analyzed the policies in two ways, as shown in Figure 2.13. First, we characterized each policy by the frequencies present in its control signals, reducing this complex frequency spectrum to two dimensions through principal component analysis. This revealed that sCMA-ES policies cluster tightly together, suggesting they all use similar control patterns. In contrast, Curiosity-ES policies spread across the space, indicating a variety of different control strategies.

We also examined how these policies affected the system state by analyzing the most critical states visited during each policy's operation. This analysis revealed that Curiosity-ES not only found diverse control strategies but also explored a wider range of system behaviors. Notably, some of the highest-disagreement scenarios were discovered not through direct optimization but during Curiosity-ES's exploratory phases. These diverse

failure cases provide engineers with a more comprehensive understanding of potential system vulnerabilities, rather than just a single worst-case scenario.

This broader exploration highlights a key advantage of curiosity-driven testing: rather than converging on a single exploit, it continues to search for qualitatively different ways the system might fail. Such comprehensive exploration is particularly valuable in safety-critical systems, where understanding the full range of potential failure modes is as important as finding the most severe ones.

2.5 Perspectives

This chapter has demonstrated two complementary approaches to exploration in automated discovery. JEDi showed how learning the relationship between behaviors and performance can guide exploration toward promising areas, while Curiosity-ES revealed how intrinsic motivation can drive systematic exploration without requiring explicit behavior descriptions. Both methods achieved state-of-the-art results on standard benchmarks, and Curiosity-ES proved particularly effective at finding diverse failure cases in critical aviation software.

In this chapter, I focused heavily on evolutionary computation for simplicity, as this manuscript as a whole mostly covers evolutionary methods. However, the field of RL offers increasingly sophisticated approaches to exploration that are worth discussing. Furthermore, the complementary strengths of EC and RL point toward hybrid approaches that combine the two. Finally, and perhaps most importantly, these exploration methods show great promise for scientific discovery, where finding diverse solutions and unexpected phenomena is crucial for advancing understanding.

2.5.1 Exploration in Reinforcement Learning

While this chapter has focused on evolutionary approaches to exploration, RL offers complementary insights into how artificial systems can systematically explore their environments. Like a student trying different approaches to solve a math problem, RL agents must balance trying new strategies against exploiting what they already know works well.

Early RL methods approached exploration simply by counting which states had been visited, encouraging agents to visit unexplored areas (Brafman and Tennenholtz 2002; Kearns and Singh 2002). However, this becomes impractical in complex environments where the number of possible states is enormous. Imagine trying to count every possible configuration of a robot's joints - the numbers quickly become unmanageable.

Recent approaches have taken inspiration from human learning. Just as curiosity drives children to explore their environment, some RL methods give agents intrinsic rewards for discovering new or unexpected situations (Pathak et al. 2017; Burda et al. 2018). Others focus on developing diverse skills, much like how a musician might practice different techniques to master an instrument (Eysenbach et al. 2019).

Our work has advanced these ideas through the Learning Diverse Skills through Successor State Representations (LEADS) algorithm, a method that helps agents learn a diverse set of skills that comprehensively cover their environment (Le Tolguenec, Teichteil-Koenigsbuch, et al. 2024). Rather than just maximizing performance on a specific task, LEADS encourages agents to develop different ways of interacting with their environment.

For example, a robot might learn not just to walk forward, but also to walk backwards, sideways, or even jump - creating a rich repertoire of movements.

Modern RL methods like those mentioned above typically maintain a “replay buffer” that stores past experiences, allowing them to learn from their entire history of exploration. This approach encourages sophisticated, data-driven characterizations of exploration. Different behaviors can be defined by measuring distances between distributions of states in the replay buffer, allowing behavior descriptions to emerge naturally from the data rather than being manually specified.

This is a direction that evolutionary methods are beginning to adopt, although not without difficulties. It is, for example, one of the reasons we didn’t directly relate JEDi to Curiosity-ES: while both methods guide exploration, Curiosity-ES requires implementing mechanisms for storing historical data and training neural networks that aren’t typically present in evolutionary frameworks. Traditional evolutionary algorithms discard most information about past individuals, keeping only their fitness and behavior characterization. A promising direction is methods that bridge this gap by incorporating replay buffer mechanisms into evolutionary frameworks to enable more data-driven approaches to characterizing and guiding exploration.

2.5.2 Combining ES and RL

The previous section highlighted how RL’s data-driven approach to exploration differs from evolutionary methods. Yet these approaches are complementary - evolution excels at maintaining diverse solutions, while RL efficiently learns from individual experiences. This suggests that combining them could create more effective methods for exploration and discovery.

Early attempts to merge these approaches focused on running them in parallel, with occasional exchange of information between them (Sigaud 2023). The most straightforward form of exchange is "actor injection," where solutions found by RL are inserted into the evolutionary population (Bodnar, Day, and Lió 2020; Khadka and Tumer 2018). The intuition is simple: if RL finds a good solution through gradient-based learning, why not share it with the evolutionary process?

However, our research revealed a subtle problem with this approach. When using Evolution Strategies (ES), which maintains a statistical distribution over solutions rather than just a population of individuals, we found that RL solutions tend to drift away from this distribution over time. This "genetic drift" means that injecting RL solutions can actually harm the evolutionary process rather than help it.

To address this, we developed Genetic Drift Regularization (GDR), which adds a simple constraint to the RL training process (Templier, Rachelson, et al. 2024). GDR keeps RL solutions close to the ES distribution while still allowing them to improve. This enables effective collaboration between the two methods - RL can efficiently optimize solutions while evolution maintains diversity and explores alternative possibilities.

This hybrid approach points toward a broader opportunity in automated discovery. RL’s ability to learn efficiently from experience could enhance evolutionary exploration, while evolution’s capacity for maintaining diverse solutions could help RL avoid getting stuck in local optima. Nilsson and Cully (2021) offers an example of this combination, using

gradient-based learning on top of a MAP-Elites archive. Future work could extend this integration, perhaps using RL's sophisticated exploration mechanisms to guide evolutionary search, or using evolutionary diversity maintenance to help RL develop robust solutions.

2.5.3 Exploration for Scientific Discovery

The methods developed in this chapter, while demonstrated on robotics and software testing, point toward a broader vision of automated discovery. Just as JEDi and Curiosity-ES found diverse solutions in their respective domains, similar approaches could help scientists explore complex scientific phenomena and discover unexpected patterns.

Our work on aircraft safety systems particularly illustrates this potential. Traditional testing might focus on finding the single worst-case scenario, but Curiosity-ES revealed multiple, qualitatively different ways the system could fail. This diversity of cases provided engineers with a richer understanding of potential vulnerabilities than any single failure case could offer. Similarly, in scientific exploration, finding multiple different explanations for a phenomenon often proves more valuable than finding a single, seemingly optimal solution.

The flexibility of these exploration methods is crucial for scientific applications. Scientists can guide exploration by defining behavior descriptions that match their research interests, just as we defined relevant behaviors for robot locomotion or software testing. When using evolutionary methods, the final result is not a single solution but a population of alternatives, allowing researchers to examine different possibilities and choose those that warrant further investigation.

Exploration encourages algorithms to uncover the unexpected. In our robotics experiments, JEDi discovered solutions that surprised us. On the ant maze environment, an ant robot is expected to navigate a maze. Rather than traversing a path through the maze, exploration lead to a faster solution - jumping over the walls. Similarly, Curiosity-ES found numerous failure cases in a thoroughly-tested aviation system that traditional methods like Monte Carlo sampling missed. This ability to discover unexpected phenomena is perhaps the most valuable feature for scientific applications, where breakthrough insights often come from observing the unexpected.

However, for these methods to truly advance scientific understanding, humans must be able to comprehend and learn from the solutions they discover. A robot that solves a maze in an unexpected way is interesting, but its applicability is limited if we don't understand how it works. This need for comprehensible results motivates the focus of our next chapter: developing interpretable models that can help explain the patterns and solutions discovered through automated exploration.

3 Interpretability

Since the groundbreaking application to image classification, (Krizhevsky, Sutskever, and Hinton 2012), deep learning has driven remarkable results across a wide range of domains, from natural language processing, to image generation, to robotic control. These successes stem, in part, from the universal approximation properties of Artificial Neural Networks (ANNs) - given sufficient size and training data, neural networks can learn to represent virtually any function (Funahashi 1989; Lu and Lu 2020). In the previous chapter, we saw how neural networks serve as flexible function approximators in control tasks, enabling complex behaviors to emerge through optimization.

However, these capabilities come at a cost: neural networks operate as black boxes, their internal decision-making processes opaque even following complex analysis (Rudin 2019). When AlphaGo defeated world champion Lee Sedol at Go, it made moves that expert players initially thought were mistakes, only to later recognize their profound strategic value (Silver et al. 2017). But even after analysis, the exact reasoning behind these moves remained unclear. The neural network had learned effective strategies, but could not be used to explain the reasoning behind them.

The standard response to this challenge has been to develop post-hoc explanation methods that attempt to interpret already-trained neural networks (Angelov et al. 2021). These approaches use techniques like saliency maps or feature attribution to suggest which inputs were most important for particular predictions. However, such explanations are often unreliable and can be misleading. More fundamentally, they attempt to reverse-engineer understanding from models that were not designed with the goal of being understood.

This opacity becomes particularly problematic in critical applications like autonomous vehicles or medical diagnosis, where understanding the reasoning behind decisions is crucial for safety and accountability. A self-driving car that makes the right decisions 99.9% of the time may still be unacceptable if we cannot verify why it occasionally makes dangerous choices. Post-hoc explanations that merely highlight which visual features influenced a decision provide limited assurance about the underlying logic.

An alternative approach, which has been a major focus of my research, is to develop models that are interpretable by design. Rather than attempting to explain black-box models after the fact, we can create models whose decision-making process is transparent and decomposable into understandable components. Even if the complete model is complex, we can analyze its individual parts and their interactions to build confidence in its behavior.

Lipton (2018) offers the following components for defining interpretability:

1. *Simulatability*: the decision process could be reproducible in a reasonable amount of time by a user;
2. *Decomposability*: the decision process could be decomposed in several atomic operators that are interpretable;
3. *Transparency*: the training process should feature convergence guarantees.

Among these, I find decomposability to be particularly crucial. Consider an operating system - while its total complexity may exceed what any single person can fully comprehend in detail, its decomposable architecture allows us to understand and modify specific components. The July 2024 CrowdStrike incident illustrates this principle: a faulty kernel configuration file caused widespread system crashes, but because operating systems are built from separable components, engineers could isolate the problem to specific driver files and develop targeted fixes. Had the operating system been a monolithic black box, diagnosing and repairing the issue would have been far more challenging.

In contrast, many modern AI systems lack this crucial property of decomposability. Large Language Models (LLMs), for instance, distribute their knowledge across billions of parameters in ways that resist clear decomposition. Consider updating the physical location of something, like where the Eiffel Tower is. An ANN like an LLM can be trained to include such information, storing it in parameters in the network. What if the location changes, like the highly unlikely event that the Eiffel Tower is relocated to Rome? In Meng et al. (2022), they make this precise manipulation, identifying the parameters that encode the Eiffel Tower's location and modifying them in order to change this specific fact. However, some of the other information stored in the LLM was modified as a result; information that is related to neither the Eiffel Tower nor Rome (Meng et al. 2022). Furthermore, while the location of the Eiffel Tower was modifiable though these methods, other information, like what sport Michael Jordan plays, wasn't. In large ANNs, information and decision factors can be stored in a way that is too fundamentally entangled to be decomposed and cleanly separated into interpretable components.

More classic information storage mechanisms like databases rather keep information in ways that allow for individual modification. While the location of the Eiffel Tower may be represented in multiple database entries, if all the database entries about the Eiffel Tower are updated, a software system using the database will respond with the new location. Furthermore, no other information will have been modified; only the Eiffel Tower's location. Complex software systems can have complex information dependencies, but they are made of modular components (Parnas 1972). The information and decision factors can be decomposed from the full model in order to modify them.

Through techniques in Genetic Programming (GP), my work focuses on developing AI systems that are natively decomposable. By evolving programs as compositions of well-understood functions, we ensure that if each component function is interpretable, the complete model remains amenable to analysis. For instance, in robotic control tasks where reinforcement learning typically employs neural networks, we have shown that graph-based genetic programming can discover policies that are both effective and transparent. These evolved programs reveal not just what information they use, but precisely how they process it to make decisions.

In this chapter, we explore two contributions to interpretable machine learning. First, through a collaboration with Giorgia Nadizar, we demonstrate how genetic programming naturally produces interpretable solutions, achieving comparable performance to deep reinforcement learning on continuous control tasks while remaining fully understandable. Then, in the thesis of Camilo de la Torre, we show how these interpretable approaches can be extended to handle complex visual inputs, providing transparent alternatives to deep learning for real-world applications.

This investigation into interpretability mirrors key themes from our exploration of quality-diversity algorithms in the previous chapter. Just as we found that focusing solely on optimization can lead to suboptimal solutions, we will see that the assumed trade-off between performance and interpretability is often false. By carefully designing our methods to be transparent while maintaining high performance, we can create models that are both performant and understandable.

3.1 Native Interpretability

The need for interpretable AI has led to numerous methods for explaining the decisions of black-box models (Landajuela et al. 2021). A common approach is to train a simpler, interpretable model like a decision tree to mimic the behavior of a complex neural network, a process known as policy distillation (Verma et al. 2018). While this can provide insights into what patterns the neural network has learned, it adds an extra layer of approximation and uncertainty - we cannot be sure if the explanatory model truly captures the neural network's decision process. Indeed, studies have shown that such imitation learning approaches often yield less effective policies than direct optimization (Hein, Udluft, and Runkler 2018), and can even fail entirely when the underlying behavior is too complex to capture through imitation (Medvet and Nadizar 2023).

GP offers a compelling framework for evolving interpretable models directly (Zhou and Hu 2023). By representing solutions as explicit computational graphs or programs, GP methods naturally create models whose decision-making process can be analyzed and understood. This approach dates back to the early work of Koza and Rice (1992), who demonstrated how GP could be used to evolve interpretable robot control policies. While these evolved programs may become complex, their step-by-step operation remains traceable, unlike the distributed representations learned by neural networks. In this section, we examine how graph-based GP can produce effective solutions for challenging control tasks while maintaining natural interpretability. We demonstrate that this interpretability emerges without explicitly optimizing for it, suggesting that certain representations may inherently lead to more understandable models.

3.1.1 Genetic Programming

GP is a type of Evolutionary Computation (EC) focused on the evolution of computer programs (Koza 1994). Just as biological evolution optimizes organisms through selection and variation, GP iteratively improves candidate programs by selecting the most successful solutions and modifying them to create new variants. This process has proven remarkably effective across diverse domains, from circuitry design to medical diagnosis (Julian Francis Miller 2020; Cava et al. 2021).

The challenge in evolving programs lies in representing them in a way that can be systematically modified while maintaining valid syntax. Traditional GP uses tree structures to represent programs (Koza 1994), where each node in the tree represents a function and its children represent the arguments to that function. While this approach is intuitive and mirrors the structure of mathematical expressions, it can be limited in its ability to reuse computed values and represent complex program flows.

Graph-based Genetic Programming (GGP) addresses these limitations by representing programs as directed graphs rather than trees (Julian Francis Miller 2020). In this work, we'll see two prominent variants of GGP: Cartesian Genetic Programming (CGP) and Linear Genetic Programming (LGP). CGP arranges nodes in a grid-like structure, with each node representing a function that can take inputs from previous nodes or program inputs (Julian F Miller 2011). The program's outputs are then collected from specified nodes in this grid. LGP, in contrast, represents programs as sequences of instructions that operate on a set of registers (Brameier, Banzhaf, and Banzhaf 2007), though the flow of information between instructions forms an implicit directed graph.

Both CGP and LGP encode programs as sequences of bounded integers that specify function choices and connections. In CGP, each node is defined by a tuple $(h, i_1, \ldots, i_m)$, where h selects a function from a predefined set and i_k specify the inputs to that function. LGP similarly encodes each instruction with a destination register, a function choice, and source registers for the function's arguments. This integer encoding enables efficient mutation operators that can explore the space of possible programs while maintaining valid syntax.

As we'll explore in the next section, these graph-based representations naturally lead to interpretable solutions, even without explicitly optimizing for simplicity. We will first explore this property on robotic control tasks, showing how GGP can produce transparent yet effective solutions.

3.1.2 Interpretable control

To demonstrate the natural interpretability of graph-based genetic programming, we conducted a study comparing GGP-based controllers with state-of-the-art deep Reinforcement Learning (RL) on continuous control tasks in Nadizar, Medvet, and Wilson (2024a). We selected eight environments from the Brax physics simulation suite (Freeman et al. 2021), ranging from simple balancing tasks like inverted pendulum to complex locomotion challenges like the walker and ant robots. These environments are standard benchmarks in RL, allowing direct comparison with established methods. For these methods, we used two leading algorithms: Proximal Policy Optimization (PPO) (Schulman et al. 2017) and Soft Actor-Critic (SAC) (Haarnoja et al. 2018).

The results, presented in Figure 3.1, challenge the common assumption that interpretable models must sacrifice performance. Graph-based policies achieved comparable or superior performance to deep RL in most environments. In four out of eight tasks (inverted pendulum, inverted double pendulum, swimmer, and hopper), GGP matched or exceeded the performance of both deep RL algorithms. In three additional tasks (reacher, walker2d, and half-cheetah), GGP remained competitive with at least one of the deep RL methods.

Only in the ant environment did GGP consistently underperform compared to deep learning approaches. The ant robot has many sensors which provide input data and many

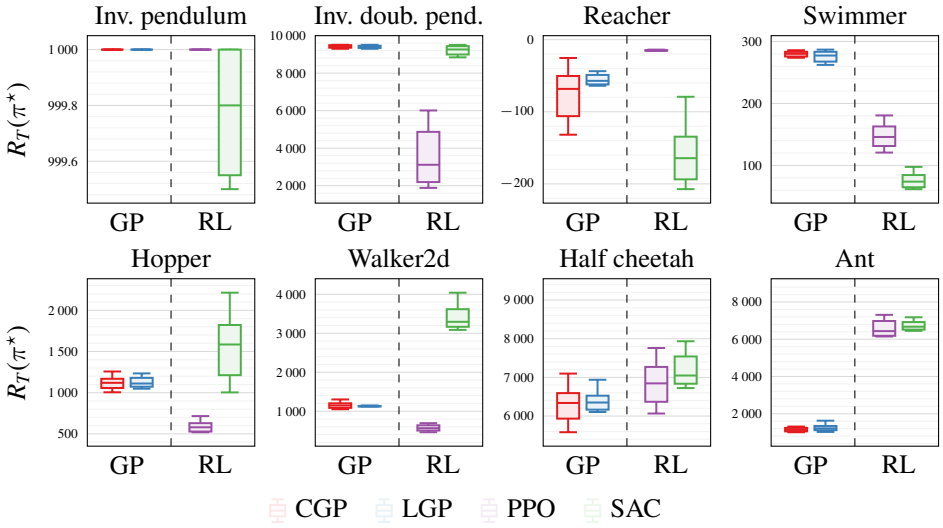


Figure 3.1: Box plots of the cumulative reward $R_T(\pi^\star)$ found by the best policy $\pi^\star$ of each method. Figure from Nadizar, Medvet, and Wilson (2024a).

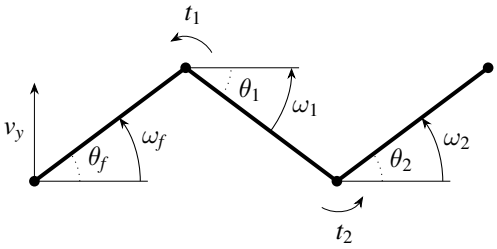
joints to control, making the decision variables more complex. Later in this chapter, we'll look at an adaptation of CGP for complex data.

The swimmer environment provides a clear example of the interpretability of models found by GGP. The task requires coordinating two joints to propel a snake-like robot through water. While a deep neural network might use hundreds of parameters to solve this task, the evolved programs uses just a handful of mathematical operations. A representation of the swimmer environment, as well as the full solutions found by CGP and LGP, is presented in Figure 3.2. The CGP solution computes the torque for each joint using simple trigonometric and arithmetic operations on the robot's state variables. Most notably, each joint's control takes into account the state of the opposite joint, revealing a clear coordination strategy that emerges from the optimization process.

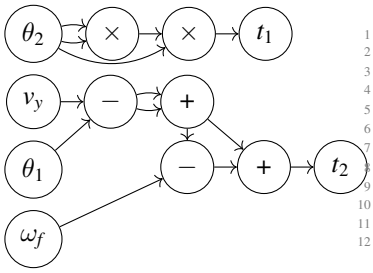
Analyzing the swimmer policy reveals not just what information is being used, but precisely how it is processed to generate actions. This level of transparency is impossible with deep neural networks, where the relationship between inputs and outputs is distributed across many layers of neurons. The graph structure makes it clear that the policy has learned to coordinate the joints by creating a feedback loop between their states, a strategy that is both effective and intuitive from a control theory perspective.

Interestingly, this interpretability emerges naturally from the graph-based representation, without explicitly optimizing for it. Our analysis shows that successful policies consistently use only a small fraction of the available computational nodes, typically less than one-third of the maximum possible complexity. This tendency toward simplicity persists even as performance improves during evolution, suggesting that the representation itself biases the search toward interpretable solutions.

We verified this finding by comparing our single-objective optimization to a multi-objective approach that explicitly rewarded simpler solutions. Using NSGA-II (Deb et al. 2002) to simultaneously optimize for performance and simplicity, we found that the



(a) Swimmer schematic.




```python
def controller(inputs, r):
  # inputs to registers
  r[0:10] = inputs
  # perform computation
  r[13] = r[1] - r[5]
  r[16] = r[16] - r[7]
  r[15] = r[13] - r[6]
  r[15] = r[15] + r[2]
  r[16] = r[16] - r[1]
  # registers to output
  return r[-2:]

```

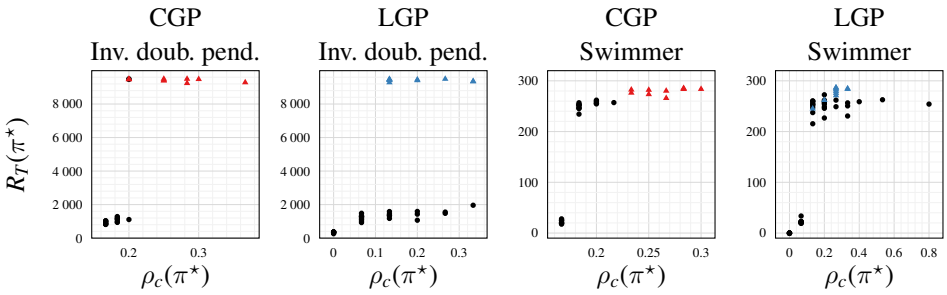
$$t_1 = \theta_2^3$$
$$t_2 = 4(v_y - \theta_1) - \theta_f$$

(b) CGP solution.

$$t_1 = \theta_1 - \omega_f - \omega_1 + \theta_2$$
$$t_2 = -\omega_2 - \theta_1$$

(c) LGP solution.

Figure 3.2: (a) Schematic representation of the swimmer environment, showing the observed variables $\theta_f, \theta_1, \theta_2, \omega_f, \omega_1, \omega_2, v_y$ and the control variables t_1, t_2. (b) The policies found with CGP and (c) LGP, shown in their original form and as formulae. Figure from Nadizar, Medvet, and Wilson (2024a).



▵ Standard Genetic Algorithm (GA) ● NSGA-II

Figure 3.3: Cumulative reward $R_T(\pi^\star)$ vs. fraction of complexity $\rho_c(\pi^\star)$ for each policy $\pi^\star$ in the final population of a single-objective GA and dual-objective NSGA-II. Figure from Nadizar, Medvet, and Wilson (2024a).

explicit promotion of interpretability was largely unnecessary. As we can see in Figure 3.3, while the multi-objective optimization did find simpler solutions, it often failed to discover the most effective policies. Single-objective GGP found solutions that were both simple and effective without explicitly searching for simplicity. This suggests that for graph-based

GP, interpretability is an inherent property of the representation rather than a competing objective that must be explicitly optimized.

3.1.3 Diversity of solutions

In the previous chapter, we explored how quality-diversity optimization can discover diverse solutions to complex problems. This approach becomes particularly useful when combined with graph-based genetic programming, as it allows us to explore not just different behaviors but also different levels of interpretability. Rather than seeking a single optimal solution, we can discover a range of interpretable policies that solve the task in different ways. This diversity is valuable both for understanding the problem space and for providing alternatives when certain solutions exhibit undesirable properties.

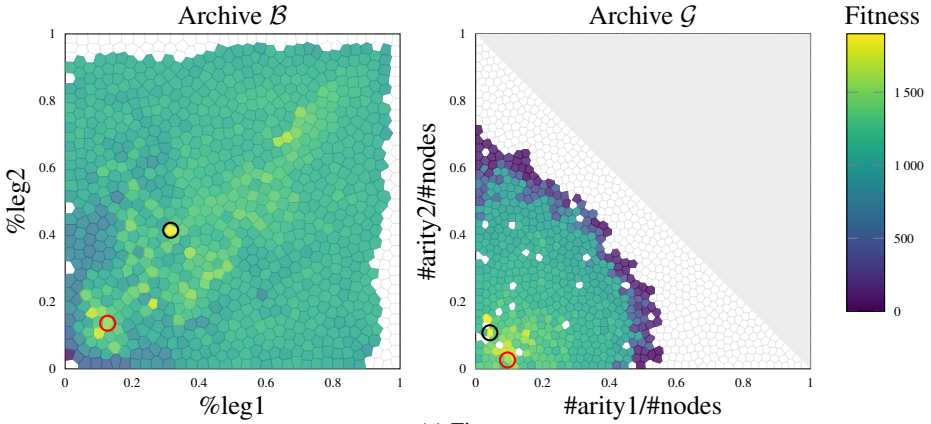
To achieve this, we extended our graph-based GP framework with MAP-Elites style quality-diversity optimization (Nadizar, Medvet, and Wilson 2024b). The key innovation was the introduction of a dual-archive system: one archive maintains diversity in the behavioral space $\mathcal{B}$ (how the robot moves), while the other preserves diversity in the structural space of the graphs themselves $\mathcal{G}$ (how the solution is composed).

This dual-archive approach allows for independent exploration of both the behavioral and structural aspects of solutions. Solutions can be selected from either archive during evolution, promoting diversity over both how the agent acts and how its controller policy is structured.

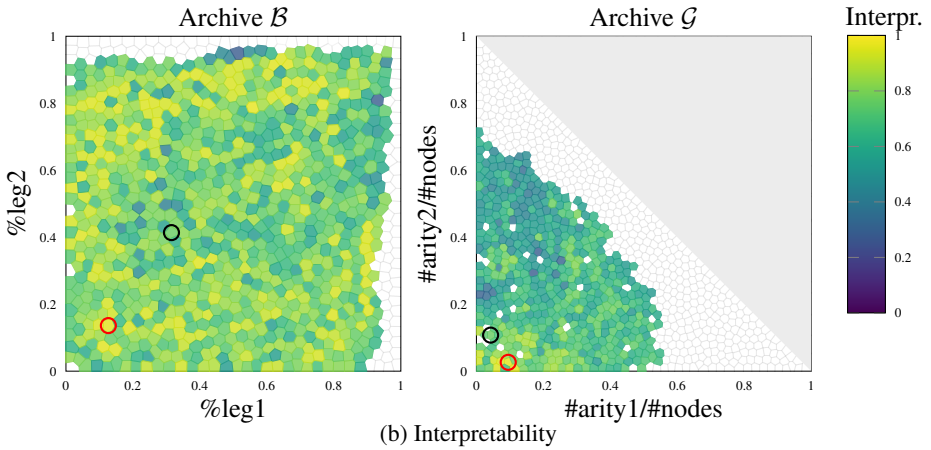
The archives in Figure 3.4 reveal the relationship between solution structure, performance, and interpretability. In the walker task, for example, we found that the most effective gaits cluster around balanced leg usage in the behavioral space. Meanwhile, the structural archive showed that simpler solutions (those using fewer operations) often achieved better performance than more complex ones. This pattern, consistent across different tasks, suggests that the search naturally gravitates toward simple yet effective solutions when given the freedom to explore both behavioral and structural spaces.

Most notably, the archives demonstrate that similar behaviors can be achieved through structurally different solutions, and conversely, similar graph structures can produce different behaviors. This diversity of graph structures leads to a wide array of interpretability across behaviors. We find that there is no negative relationship between interpretability and performance, as discussed in Rudin (2019). Rather, interpretable behaviors are found throughout the two archives, and across the full range of policy performance.

These results reinforce a key theme of this section: interpretability need not be explicitly optimized but can emerge naturally from appropriate representations and search methods. By combining graph-based GP with quality-diversity optimization, we can discover not just individual interpretable solutions but entire families of them, each offering different trade-offs between complexity, performance, and robustness. In other words, if we use these algorithms to solve a problem, the result will be a diverse set of different interpretable solutions, each presenting a different insight that can help our understanding of the problem.



(a) Fitness



(b) Interpretability

Figure 3.4: Archives from the robotic locomotion task walker2d. Behavior archives $\mathcal{B}$ organize policies based on their behavior. Graph archives $\mathcal{G}$ organize policies based on their graph structure. The upper-right portion of the graph archive $\mathcal{G}$ is not reachable. The black circles ○ highlight the best performing policies, the red circles ○ highlight the most interpretable one. Figure adapted from Nadizar, Medvet, and Wilson (2024b).

3.2 Learning On Complex Data

The previous section demonstrated how graph-based genetic programming can create interpretable solutions for continuous control tasks. However, many of today's most challenging problems require handling multiple types of high-dimensional data simultaneously. Medical diagnosis combines image data with patient histories, autonomous vehicles fuse data from cameras, LIDAR, and other sensors, and scientific modeling often requires integrating measurements across different modalities and scales. While deep learning has shown impressive capability in processing such complex data, the resulting models remain opaque black boxes.

Can we maintain interpretability when working with such rich, multi-modal data? The key, I argue here, lies in decomposability - breaking down complex operations into

well-understood building blocks. Just as an operating system manages diverse data types through clearly defined interfaces and operations, we can design genetic programming systems that handle complex data while keeping each operation transparent and analyzable.

Through careful design of high-level functions specialized for different data types, we can evolve programs that process complex inputs while remaining interpretable. The functions themselves may perform sophisticated operations, but their effects are well-defined and their interactions can be traced. This allows us to maintain the benefits of interpretability we saw with control tasks, even as we tackle problems involving images, text, and other rich data types.

In this section, we will examine this approach through three complementary studies. First, I introduce Multimodal Adaptive Graph Evolution (MAGE), a framework for evolving programs that handle multiple data types while maintaining interpretability through type-aware operations. I will then show a demonstration of its effectiveness for biomedical image analysis, where interpretability is crucial for clinical applications. Finally, we will see how this approach enables transparent visual processing in Atari game-playing agents, comparing program-level interpretations with traditional explainability methods.

3.2.1 MAGE

MAGE (De La Torre et al. 2024) extends CGP (Julian F Miller 2011) by organizing functions into type-specific groups, allowing programs to process multiple data types while maintaining the interpretability advantages of graph-based evolution. This concept builds on previous work in applying CGP to multi-type data (Harding et al. 2012). Unlike standard CGP, where all functions exist in a single pool, MAGE separates functions by their return type into distinct "chromosomes" - rows of nodes that can only contain functions returning a specific type. As shown in Figure 3.5, this creates a layered structure where each row specializes in processing one type of data.

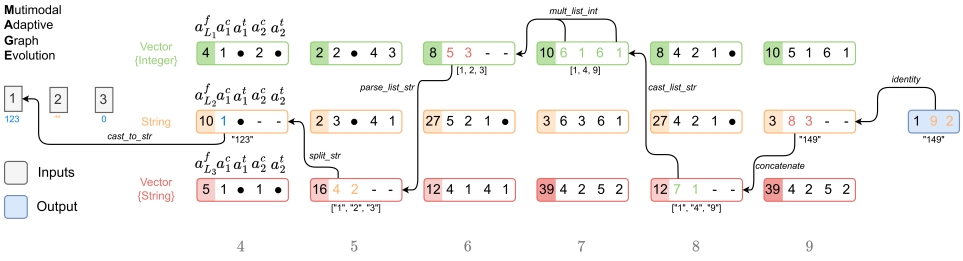


Figure 3.5: Multimodal Adaptive Graph Evolution. Figure from De La Torre et al. (2024).

The goal of MAGE is to connect functions across different types of data. While nodes within a chromosome all return the same type, they can take inputs from any other chromosome, allowing complex data transformations to be built up from simpler, type-safe operations. For example, a string-processing node might take input from both string and integer nodes, combining them to produce a new string. These cross-type connections

are explicitly tracked during evolution, ensuring that all operations remain type-safe and interpretable.

This type-aware structure offers two key advantages. First, it constrains the search space by only allowing connections that make sense from a typing perspective, making evolution more efficient. Second, it maintains interpretability by organizing operations into clear functional groups - we can easily trace how data flows between different types of processing. When combined with carefully chosen high-level functions for each type, this allows MAGE to tackle complex data processing tasks while keeping the resulting programs analyzable and understandable.

3.2.2 Biomedical image analysis

Cortacero et al. (2023) demonstrated that CGP can create effective image segmentation pipelines for biomedical applications. This work showed that evolved programs could match the performance of deep learning approaches while remaining fully interpretable. Notably, this was demonstrated on images used in cancer detection, highlighting the importance of understanding the full diagnosis pipeline, including AI image analysis tools. This method of image segmentation operated primarily on a single data type - 2D matrices representing image data or derived features. The success of interpretable segmentation raises a question: can we achieve similar results for more complex tasks like classification that traditionally require multiple types of processing?

Classification presents a challenge for interpretable models. While segmentation has a clear decomposition into pixel-level operations, classification requires both feature extraction and decision-making components. This dual nature of classification helped drive the success of Convolutional Neural Networks (CNNs), which seamlessly integrate feature extraction and classification through learned convolutional filters and fully-connected layers (Krizhevsky, Sutskever, and Hinton 2012). Can a functional graph discover an effective and understandable decomposition of these components?

Our recent work (De La Torre, Nadizar, Lavinas, Schwob, et al. 2025) suggests that for biomedical images, which exhibit more constrained patterns than general natural images, interpretable classification is achievable. We tested this approach on the PatchCamelyon (PCAM) dataset (Veeling et al. 2018), which contains patches from lymph node biopsies labeled as either malignant or benign. These patches present significant variability in tissue appearance and staining, making classification challenging even for trained pathologists.

When compared to standard CNNs, our evolved classifiers achieved competitive performance. While CNNs reached higher accuracy (84.5% versus 78% for MAGE), the margin remained small enough to demonstrate the viability of interpretable classification. More importantly, the evolved programs generalized well to unseen data, suggesting they captured meaningful patterns rather than superficial correlations.

The true advantage of our approach lies in its interpretability. As shown in Figure 3.6, we can trace exactly how the program processes an image to reach its classification decision. For example, one successful program discovered that counting the number of unique pixel values in the hue channel provides a strong signal for malignancy. This feature was not hand-engineered but emerged through evolution, yet remains completely understandable.

The full decomposition of a model also reveals the limitations of the model. By analyzing the cases it classifies correctly and incorrectly, we can discern scenarios where the

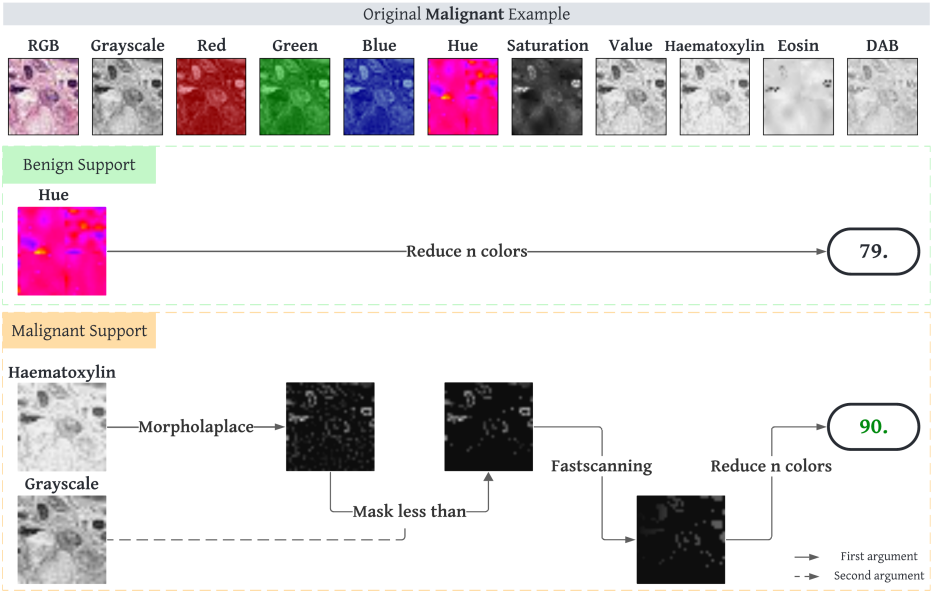


Figure 3.6: Example of a correctly classified image by an optimized classification program. Original channels are shown at the top. The program produces two outputs, the first (79.) is the support for the benign class and the second the support for the malignant class (90.). The highest number is chosen as the predicted class, in this case, a correct prediction of malignant tissue. Outputs are the results of a series of operations applied to inputs and intermediary outputs. Figure from De La Torre, Nadizar, Lavinas, Schwob, et al. (2025).

model may be less effective. For instance, the model reasons largely on the whole patch level. Because of this, if a clear malignant nucleus appears in an otherwise white patch, the model will be overconfident about its benign prediction. Detection of such failure cases in ANN has required the development of post-hoc analysis tools, which only provide a limited view of the full range of possible failure cases (Zheng and Hong 2018; Singla et al. 2021). With interpretable models, we can arrive at a full understanding simply by decomposing and analyzing their internal structure.

3.2.3 Visual control

One of the early successes of deep RL was a demonstration that these algorithms could be trained to compete with humans on Atari games (Mnih et al. 2015). The Arcade Learning Environment became a standard benchmark in AI for simulating Atari games as a metric of artificial intellience (Bellemare et al. 2013). In earlier work during my thesis, we demonstrated that CGP could solve some of the Atari tasks, reaching levels above human performance (Wilson et al. 2018). This work used Mixed-Type CGP (Harding et al. 2012), an approach similar to MAGE but with less constrained interactions between functions. The evolved programs could achieve performance competitive with deep reinforcement learning methods, suggesting that interpretable approaches need not sacrifice effectiveness.

Figure 3.7: The Atari game of Freeway. The goal is to cross the highway while avoiding moving cars. Figure adapted from De La Torre, Nadizar, Lavinas, Luga, et al. (2025)

An unexpected finding emerged from this work: many of the evolved policies achieved strong performance without processing the visual input at all. Instead, they discovered simple, fixed output patterns that exploited regularities in the games. Some of these minimal strategies even outperformed both human players and deep learning approaches. While these solutions might seem trivial, their discovery was only possible because the interpretable nature of the programs allowed us to understand exactly how they worked.

However, our goal extends beyond finding clever shortcuts - we want to understand how programs can process complex information to make decisions. We therefore applied MAGE to the Atari benchmark. By constraining function interactions to be type-safe, MAGE encourages the evolution of programs that actually process the visual input rather than ignoring it. Our preliminary work on three games - Pong, Freeway, and Bowling - shows that this approach can create effective visual control policies (De La Torre, Nadizar, Lavinas, Luga, et al. 2025). In the next section, I will focus on the evolved policy on the Freeway game to illustrate the advantage of interpretability.

3.2.4 Beyond explanations

Understanding how a program processes visual information to make decisions is inherently difficult. Each frame contains multiple objects and features that could influence the choice of action. How can we determine which elements matter and how they factor into the decision?

Current approaches to this problem rely heavily on statistical analysis. Methods like RISE and occlusion sensitivity aim to identify which parts of the input most strongly influence the output (Petsiuk, Das, and Saenko 2018; Zeiler and Fergus 2014). These techniques have become standard tools for understanding deep learning models in applications from autonomous vehicles to satellite image analysis. When applied to our Atari agents, they highlight regions of the screen that appear important for decision-making.

Consider the game of Freeway, shown in Figure 3.7. Freeway is a simple game where you control a chicken trying to cross a busy highway without getting hit by cars. The agent can only move up or down, and each successful crossing earns a point. The goal is to cross

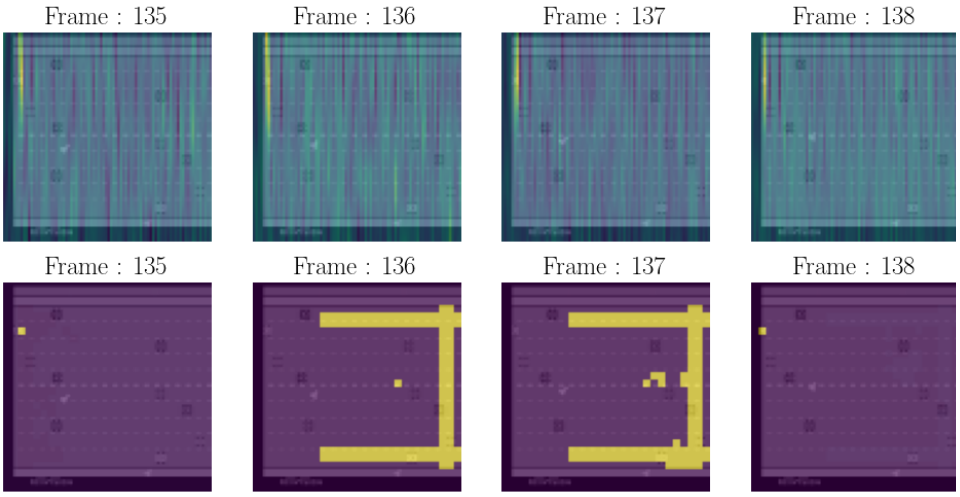


Figure 3.8: Saliency produced by RISE (top) and occlusion sensitivity (bottom) produced by perturbations over the fourth frame for (a), third frame for (b) and fourth frame for (c). Figure adapted from De La Torre, Nadizar, Lavinas, Luga, et al. (2025).

as many times as possible before time runs out while avoiding traffic. As such, Freeway has a variety of visual information that might influence the policy.

Explanation methods like RISE and occlusion sensitivity should help identify which visual features are used by a given Freeway policy. As shown in Figure 3.8, both methods highlight a car at the top of the screen, indicating its importance. While we can determine that this car influences the policy, it is not clear how Furthermore, it is unclear if other visual features, which are given less but non-zero importance in the saliency maps, count in the decision process. If we watch the video of these saliency maps, we might arrive at an explanation of how the policy functions in general. However, this won't be a complete picture and will include assumptions on our part. Explanations like these are strongly influenced by our biases and rarely offer rigorous insights, limiting their utility for validation (Atrey, Clary, and Jensen 2020).

In contrast, examining the evolved program reveals the complete decision process. The full code is presented in Figure 3.9. It is long and uses many functions to arrive at its decision, which is to be expected. It makes decisions per frame based on multiple sources of information, pulled from a screen through visual analysis functions. But it is decomposable. We can analyze it, line by line, and understand the complete logic of how it works.

From this analysis, we learn that the agent indeed tracks the top car, but its behavior is more nuanced than the explanations suggest. It moves upward when the top car is far from the left corner, but it also monitors a car in the middle lane. This middle lane car barely appears in the saliency maps, and isn't significant enough to differentiate it from other cars. However, we can clearly see its use in the policy program. The program calculates distances between these cars, the screen center, and the player's position to determine when to pause movement. This full picture of the decision pipeline - including the role of the middle car

```
ACTIONS = [0,1,2]
function evolved_freeway_policy(frame1, frame2, frame3, frame4)
    # output NO OP
    car_in_the_middle = sobel(frame2, border = 50.)
    x_y_car = argmax_position(car_in_the_middle) # the car in the middle is usually
     highlighted because of the road lines
    blurry_frame3 = dilation(frame3, k = vertical_argmax(frame1))
    blurry_frame3 = gaussian_blur(blurry_frame3)
    x_y_center = center_of_mass(blurry_frame3) # ~ midpoint of the screen
    dir_car_to_center = direction_from_to(x_y_car, x_y_center)
    extract_chicken = tophat(frame3,  k = 100)
    x_y_chicken = argmax_position(extract_chicken)
    d_chicken_to_dir_car_to_center = direction_from_to(x_y_chicken, dir_car_to_center)
    x_y_chicken_second_frame = argmax_position(frame2)
    x_y_chicken_first_frame = argmax_position(frame1)
    d = direction_from_to(x_y_chicken_second_frame, x_y_chicken_first_frame) # has the
     chicken move up/down or not moved
    output1 = true_gt_or_eq(d, d_chicken_to_dir_car_to_center) # checks if the middle car
     has crossed the x coordinate of the chicken

    # Output UP
    from = exp(vertical_relative_argmax(sobely(sobelx(frame4))))
    k = vertical_argmax(frame4)
    dilated_once = dilation(frame3, k)
    dilated_twice = dilation(dilated_once, k)
    edges = sobely(dilated_twice)
    upper_edges = notmaskfromto_vertical(edges, from, 10) # gets the top part of the screen
     who can have some pixels if the top car is there
    y_pos_edge = vertical_argmax(upper_edges)
    output_2 = y_pos_edge * 0.5 # if the top car "disappears" from the top, this value will
     be low, high otherwise (unless the chicken is also there).

    # Output DOWN
    edges = sobelm(frame2, 50)
    opened = opening(edges, 50)
    edges_2 = sobelm(opened, 40)
    output3 = horizontal_relative_argmax(edges_2) # usually a very low value

    outputs = (output1, output_2, output_3)
    return ACTIONS[argmax(outputs)]
end
```

Figure 3.9: Code for the Freeway player, from De La Torre, Nadizar, Lavinas, Luga, et al. (2025).

that wasn't clearly highlighted by statistical explanations - emerges only through direct interpretation of the program.

3.3 Perspectives

The work presented in this chapter demonstrates that interpretable methods can achieve performance competitive with black-box approaches while maintaining transparency in their decision-making processes. Through graph-based genetic programming, we showed that models can be both effective and understandable, challenging the common assumption that interpretability must come at the cost of performance. Furthermore, our extensions to

handle complex data types suggest that interpretability can be maintained even as we tackle increasingly sophisticated problems.

Next, we'll discuss how interpretable AI can aid in scientific discovery. Before that, I would like to highlight some challenges and opportunities for interpretable AI.

3.3.1 Measuring interpretability

One of the central challenges in developing interpretable AI systems is the inherently subjective nature of interpretability itself. While we can intuitively recognize when a solution is more or less interpretable, developing metrics to quantify this property remains an open challenge. Recent work by Virgolin et al. (2020) demonstrates a possible approach, using human feedback to develop quantitative measures of interpretability for mathematical formulas. This is the metric that we deployed in Figure 3.4 to measure interpretability, for example.

This data-driven approach to measuring interpretability suggests a broader methodology that could be applied across different types of models. By gathering human feedback on specific aspects of interpretability, such as simulatability and decomposability, we can develop metrics that align with human understanding. These metrics could then be incorporated into optimization objectives, guiding the search toward solutions that are not just mathematically sound but also naturally interpretable to human users.

However, as with all data-driven approaches, a consideration must be taken here for which data to use. Interpretability must be measured relative to the intended user of the system. A model that is interpretable to a domain expert may be opaque to a policymaker, and vice versa. Future work should focus on developing adaptive measures of interpretability that can be tailored to specific user groups and application contexts. Data-driven metrics of interpretability must adapt to the intended stakeholders of an AI model.

An alternative to quantifying interpretability would be to directly involve humans in the optimization process (Liu et al. 2019). Evolutionary algorithms are particularly well-suited for this type of interaction due to their population-based nature and iterative improvement process (Ortega et al. 2013). At each generation, human experts could review candidate solutions, providing feedback that helps guide the search toward more interpretable results.

This approach has several advantages over purely automated optimization. First, it allows for real-time adjustment of the search based on domain expertise and contextual understanding that might be difficult to encode in an objective function. Second, it creates an interactive learning process where the system can adapt to user preferences and requirements as they evolve during the search. Finally, it builds trust in the optimization process by making it more transparent and collaborative.

3.3.2 Solution representations

The methods presented in this chapter have relied primarily on graph-based representations, which provide a clear visual structure for understanding program flow. However, recent advances in LLMs present an opportunity to work directly with code as the solution representation. This shift towards code-based representations holds particular promise for interpretability, as computer code is already designed for human understanding.

Language Model Genetic Programming (LMGP) combines the strengths of genetic programming with the capabilities of LLMs to optimize code directly (Hemberg, Moskal, and

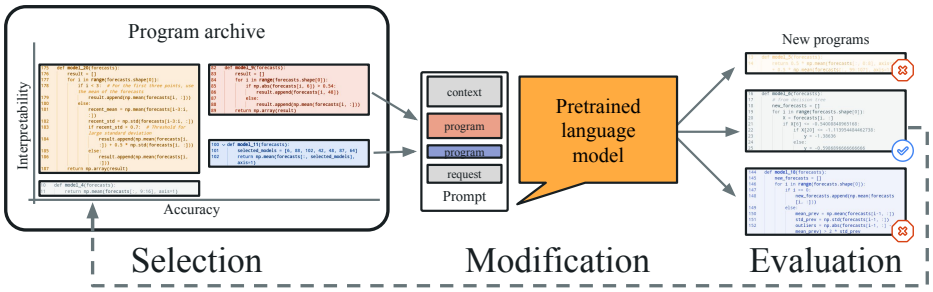


Figure 3.10: The LMGP framework. First, programs represented as computer code are either written by experts or generated, and then organized in an archive. Programs are selected from the archive and combined with a fixed context to form a prompt. This prompt is given to a pretrained language model, which generates new programs. The new candidate programs are evaluated and potentially placed into the archive. This process continues until a high-quality program is found.

O'Reilly 2024). Rather than evolving abstract syntax trees or computational graphs, LMGP maintains a population of actual program code, using LLMs to generate and modify solutions. This approach leverages the semantic understanding of code embedded in LLMs, which have been trained on vast repositories of human-written programs.

Recent applications demonstrate the potential of this approach. FunSearch (Romera-Paredes et al. 2024) uses LMGP to discover novel algorithms, maintaining an archive of generated programs and employing a LLM to create new candidate solutions. The system has successfully generated solutions to complex problems like bin-packing, producing code that is both effective and readable. Similar successes have been shown in optimizing neural network architectures (Chen, Dohan, and So 2024), creating search algorithms (Ma et al. 2024), and designing computer game levels (Sudhakaran et al. 2024).

The key advantage of LMGP for interpretability lies in its use of natural programming languages. While graph-based representations must be translated into a form that humans can understand, code is already expressed in a language designed for human comprehension. Furthermore, LLMs can leverage natural language descriptions and documentation, allowing the evolution process to be guided by human-provided specifications and constraints. This creates a more natural interface between human intent and program synthesis.

The combination of code and natural language also enables new forms of interpretability. For instance, LLMs can generate explanations of how code works, produce documentation, or refactor code into more readable forms. This capability creates a bridge between the program's functional representation and its human-understandable description. As noted in our work on shoreline forecasting (Al Najar et al. 2023), the ability to explain and document models is crucial for their adoption in scientific applications.

Looking forward, LMGP represents a promising direction for developing interpretable AI systems. The ability to work directly with text and code, combined with the semantic understanding provided by LLMs, could enable the evolution of programs that are not only

functionally correct but also naturally comprehensible to human users. This aligns with our broader goal of creating AI systems that serve as tools for human understanding.

4 Discovery

Before the invention of the microscope, an entire world remained hidden from human observation. Without the ability to see microorganisms, scientific understanding was constrained by the limits of human perception. The development of the microscope didn't just provide new data - it fundamentally transformed how scientists could interact with and understand the natural world. This revolutionary tool enabled the discovery of cells, bacteria, and countless other microscopic phenomena that reshaped our understanding of biology, medicine, and life itself (Harari 2014).

Today, machine learning offers a similar promise. Rather than uncovering previously invisible data, these computational tools help us make sense of data that overwhelms human analytical capabilities (Wang et al. 2023). Consider protein folding - while we have long had the amino acid sequences of proteins, understanding how these sequences translate into three-dimensional structures remained a grand challenge in biology. AlphaFold demonstrated how machine learning could transform this data into profound scientific insights, predicting protein structures with unprecedented accuracy and advancing our understanding of fundamental biological processes (Jumper et al. 2021).

This chapter examines how machine learning can serve as a tool for scientific discovery, particularly in environmental science. Drawing from the methods developed in previous chapters, we explore two case studies where computational techniques reveal new insights in complex natural systems. The first, drawn from the thesis of Mahmoud Al Najar, focuses on shoreline evolution, where we use genetic programming to discover interpretable models that capture the interplay between waves, sea levels, and coastal dynamics. The second, from a recent collaboration with the Japan Agency for Marine-Earth Science and Technology (JAMSTEC), examines prediction of El Niño-Southern Oscillation (ENSO), demonstrating how machine learning can improve our understanding of this crucial climate phenomenon.

Our approach builds directly on the themes developed earlier in this manuscript. The exploration techniques from Chapter 2 help us systematically search through vast spaces of possible scientific models. The interpretability methods from Chapter 3 ensure that our discoveries can be understood and validated by domain experts. Together, these capabilities enable a new approach to scientific discovery - one that combines the creativity of exploration with the rigorous demands of scientific understanding. At the end of this chapter, I offer a reflection on the broader implications of these approaches for automating scientific discovery. I argue that machine learning can augment rather than replace human

scientific reasoning, providing tools that expand our capacity for understanding complex natural phenomena.

4.1 Discovering Shoreline Models

Coastal zones face mounting pressures from climate change, urbanization, and human activity (H. Lee et al. 2023). Understanding how shorelines evolve is crucial for coastal management, infrastructure planning, and natural hazard assessment. Rising sea levels and increasing storm intensity make this understanding even more critical, as coastal communities must adapt to changing conditions (Reimann, Vafeidis, and Honsel 2023).

Shoreline prediction aims to forecast how the boundary between land and sea will change over time. This apparently simple interface represents the complex interplay of waves, tides, sediment transport, and other physical processes. While shoreline modeling focuses on understanding these underlying mechanisms, shoreline prediction addresses the practical challenge of forecasting future coastal states. These predictions operate across multiple time horizons - from days for storm response to years for coastal planning (Splinter et al. 2014).

Traditional approaches rely on physics-based models that simulate the fundamental processes driving coastal evolution. These models incorporate wave mechanics, sediment transport equations, and conservation laws to represent how beaches respond to changing conditions. The ShoreFor model exemplifies this approach, using concepts like wave energy flux and equilibrium beach states to predict shoreline change (Davidson, Splinter, and Turner 2013; Splinter et al. 2014). However, these physics-based models only make partial use of observational data, often by tuning model parameters with linear regression.

Recent work, particularly through the Shoreshop competition, has demonstrated the potential of machine learning for shoreline prediction (Montaño et al. 2020). Neural networks and random forests achieved high accuracy in this competition, especially for high-frequency shoreline changes. These results suggest that data-driven approaches can capture coastal dynamics that elude physical models (Beuzen and Splinter 2020). However, like many machine learning applications, these successful models operate as black boxes - they can predict effectively but offer limited insight into the physical processes they represent.

This raises a fundamental question for scientific discovery: can we develop models that match the predictive capacity of neural networks while maintaining the interpretability of physics-based approaches? Our work with genetic programming suggests a path forward, using evolutionary algorithms to discover mathematical models that both predict and explain shoreline evolution. These models emerge from data but take forms that coastal scientists can analyze and understand, potentially revealing new insights about coastal processes.

4.1.1 Local shoreline models

Our investigation centered on five diverse coastal sites, shown in 4.1: Grand Popo beach in Benin, Truc Vert beach in France, Narrabeen beach in Australia, and two sites in the United States - Duck, North Carolina and Torrey Pines, California. These locations represent different coastal environments, wave climates, and shoreline behaviors. Data collection

combined traditional and modern approaches, from GPS surveys to video monitoring systems, creating five varied datasets of shoreline positions over time (Almar et al. 2012; Bonou et al. 2018; Splinter et al. 2014). While physical models of shoreline evolution are largely based on wave physics, we also included data on sea level anomaly and river discharge to discovery relationships between these data and shoreline change.

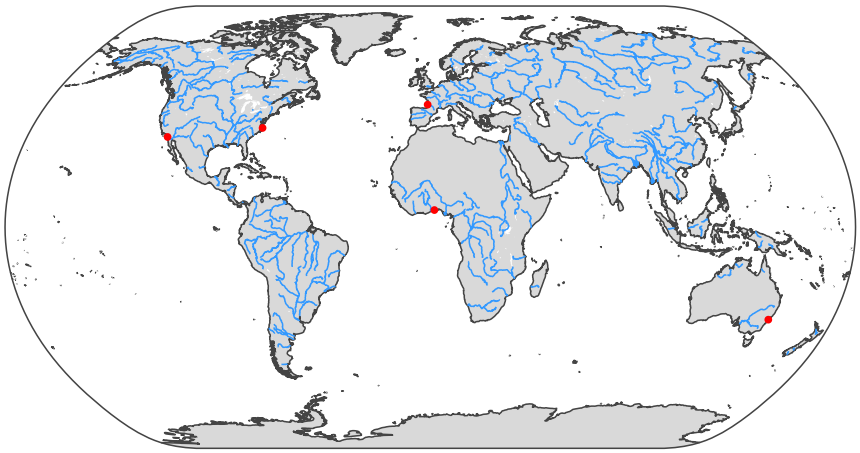


Figure 4.1: World map highlighting the locations of the five sites included in the multi-site study. Figure from Al najar (2023).

We applied Cartesian Genetic Programming (CGP), discussed in Chapter 3, to discover mathematical models that could predict shoreline evolution at these sites. Our function set incorporated operations from the ShoreFor model (Splinter et al. 2014), allowing evolution to build upon established physical principles. The evolutionary process optimized models for each site individually. Following evolution, we also analyzed the full archive from evolution to select models that generalize across all locations.

The evolved models demonstrated improved predictive capability over physical models, as shown in Figure 4.2. At most sites, they matched or exceeded the performance of the baseline ShoreFor model across different timescales. Significant improvements appeared in capturing short-term shoreline changes, where traditional physical models often struggle. The models showed different strengths at different locations - some excelled at predicting seasonal patterns, while others better captured long-term trends.

Evolution discovered several noteworthy models, including two "generalist" formulations that performed well across multiple sites. These models represent different hypotheses about how various physical processes combine to drive shoreline change. The first generalist model introduces a conditional response based on the relationship between sea level (S), wave power (P), and dimensionless fall velocity (Ω):

$$\frac{dx}{dt} = \begin{cases} \frac{1}{2}\bar{P}^{0.5}\frac{d}{dt}\sqrt{\frac{\phi^2}{2} + \frac{1}{4}(R-\Omega)^2 + S^2}, & \text{if } S \geq P^{0.5} + \Omega \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

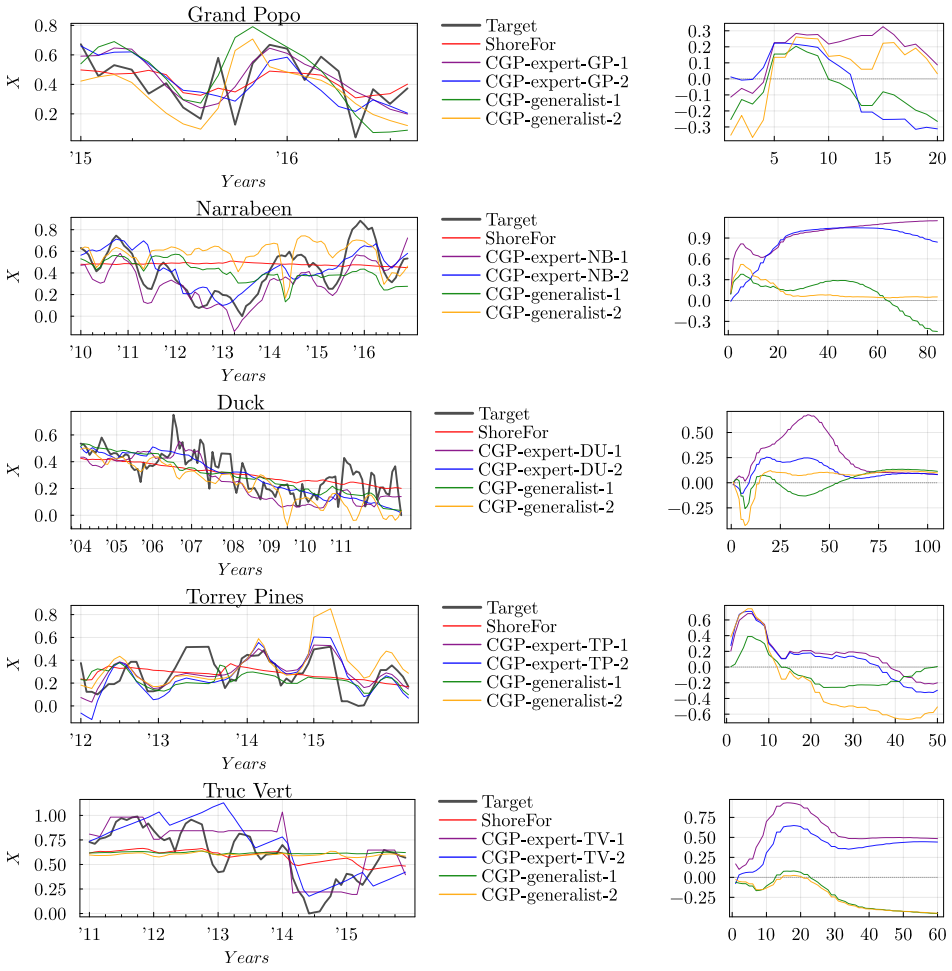


Figure 4.2: Forecast performances on the five sites (left) and the difference in the Mielke skill score at different timescales compared to the performance of ShoreFor at that timescale ($Y = 0$ represents ShoreFor and the X axis corresponds to the number of months at the evaluated timescale). Figure from Al najar (2023).

The second model suggests a simpler relationship where river discharge (R) modulates the influence of wave power, while sea level acts through an exponential term:

$$\frac{dx}{dt} = P^{\frac{R}{2}} \frac{d}{dt} 10^{S}. \tag{4.3}$$

These models differ significantly from traditional formulations like ShoreFor. Where ShoreFor emphasizes the concept of equilibrium beach states, our evolved models suggest alternative mechanisms. They incorporate factors not typically included in shoreline models, such as river discharge and sea level anomalies. This integration of multiple drivers aligns with growing evidence that shoreline evolution responds to a broader range of environmental forces than previously considered.

The mathematical structure of these models offers insights into coastal processes. The first generalist model suggests a threshold behavior - shoreline change occurs only when sea level exceeds a wave-dependent threshold. The second model proposes a more continuous response, where river discharge modulates the system's sensitivity to wave power. Both models point to the importance of interactions between different environmental factors, rather than treating them as independent influences.

Cross-validation between sites helped ensure the models captured genuine physical relationships rather than site-specific correlations. However, performance varied significantly between locations, suggesting that local factors still play a crucial role in shoreline evolution. The models performed particularly well at sites with strong seasonal patterns, indicating they effectively capture cyclic coastal behavior.

4.1.2 Global shoreline models

Our investigation expanded from local to global scale, analyzing satellite-derived shoreline positions across thousands of coastal locations worldwide. Each point in our dataset represents a unique waterline, tracked through 30 years of satellite observations (Almar et al. 2023). This global perspective required different approaches than our local studies - while we maintained the goal of interpretable modeling, we needed methods that could handle both the increased scale and diversity of coastal behaviors.

The genetic programming setup reflected these new challenges. Rather than optimizing models for specific locations, we evolved models using a subset of global points. This approach forced evolution to discover more general relationships between environmental drivers and shoreline change. Post-evolution evaluation then tested these models' performance across all available coastal points, assessing their true global applicability.

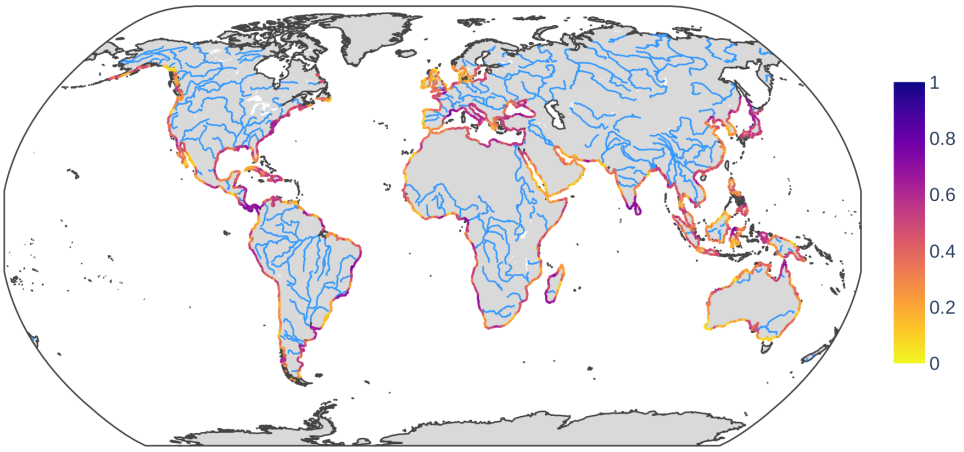


Figure 4.3: Global map of the correlation scores achieved by the baseline global model ($\bar{r} = 0.36$). *The color-scale corresponds to Pearson correlation. Figure from Al Najar, Almar, and Wilson (2025).

Our baseline global model achieved a mean correlation of $\bar{r} = 0.36$ in hindcasting shoreline change, as shown in Figure 4.3. This performance, while modest, demonstrates that

even a single interpretable model can capture some universal patterns in coastal evolution. The spatial distribution of model performance reveals where simple physical relationships hold and where more complex dynamics may be at play. Areas with strong seasonal patterns or dominated by a single physical process showed the strongest correlations.

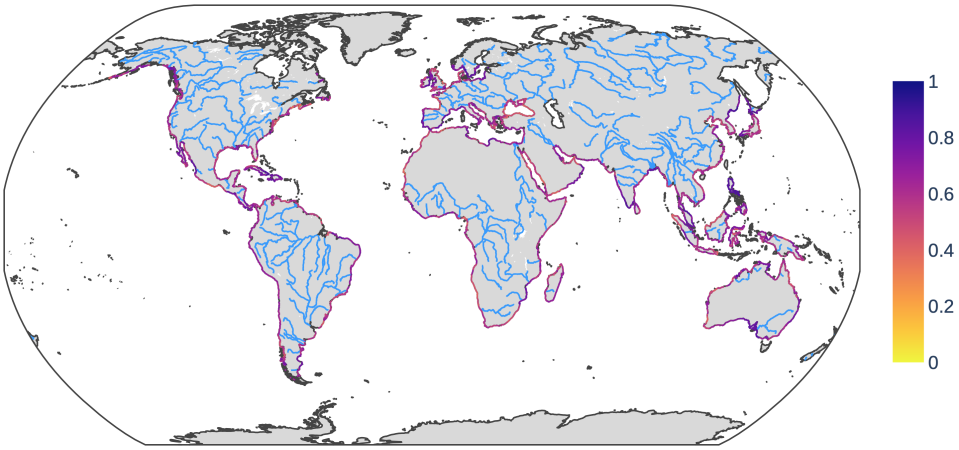


Figure 4.4: Global map of the ensemble-based shoreline hindcast correlations ($\bar{r} = 0.61$). *The color-scale corresponds to Pearson correlation. Figure from Al Najar, Almar, and Wilson (2025).

We then developed an ensemble approach, combining multiple evolved models to capture different coastal behaviors. The ensemble achieved a global correlation of $\bar{r} = 0.61$, a substantial improvement over the baseline model. As shown in Figure 4.4, this improvement wasn't uniform - some regions saw dramatic gains while others showed more modest benefits. The varying performance helps identify where different physical processes dominate shoreline evolution.

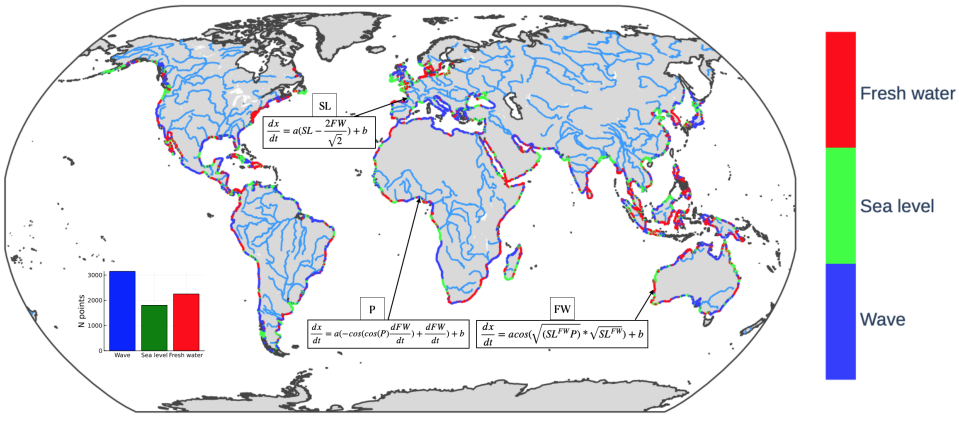


Figure 4.5: Global map of the drivers of the ensemble of 57 models, showing three example models. Figure from Al Najar, Almar, and Wilson (2025).

The global distribution of dominant shoreline drivers, presented in Figure 4.5, reveals regional patterns. Different models, each representing distinct physical hypotheses, dominate in different coastal regions. This pattern suggests that while some physical principles apply globally, local conditions often determine which processes most strongly influence shoreline evolution. The local models can thereby provide a new classification of coastal systems based on their governing dynamics.

The evolved equations show varying levels of complexity, from simple wave-driven models to more intricate formulations incorporating multiple environmental factors. Some coastal regions show clear dominance of a single driver - wave power in high-energy coasts, for example. Other areas exhibit more complex behaviors, where multiple drivers interact to determine shoreline position. This diversity of governing equations challenges the notion of a single, universal model for shoreline evolution.

This global analysis reveals patterns that would be difficult to discern from local studies alone. The spatial distribution of model performance and dominant drivers provides new insights into coastal classification. Regions with similar governing equations might suggest shared underlying physics, even in geographically distant locations. These patterns could help guide coastal management strategies and improve our understanding of how shorelines might respond to climate change.

4.1.3 Perspectives

The field of satellite-derived shoreline analysis remains young, with continuous improvements in data quality and processing methods (Almar et al. 2023). Our current work uses a new global dataset where quality assurance is still ongoing. This presents both challenges and opportunities - while we must be cautious about data reliability, the expanding availability of satellite observations promises increasingly robust analyses (Ibaceta et al. 2022). The distinction between modeling and prediction also merits attention. Where many studies focus on modeling shoreline states based on the corresponding drivers, we attempt the more challenging task of prediction, using historical drivers to forecast future shoreline positions.

A key limitation of current shoreline modeling approaches, including our own, is their strictly local nature. Each point along the coast is treated independently, even though coastal processes often operate at regional scales. This local focus means we potentially miss important information about sediment transport between neighboring beaches, regional wave patterns, and other spatially-connected processes (Almar et al. 2023). To examine the possibility of global influence, we performed a preliminary study with convolutional neural networks trained on the same global dataset discussed in this chapter (Riu et al. 2023). This study suggested promising directions for incorporating spatial relationships, although it was limited by the black-box nature of the model used.

An internship planned for 2025 will explore combining Language Model Genetic Programming (LMGP) with regional physical models. This work aims to bridge the gap between local and regional scales, potentially discovering models that can capture both spatial and temporal dynamics. The use of LMGP could help discover equations that better respect known physical principles while maintaining the flexibility to learn from data.

While our current models are interpretable in mathematical terms, they sometimes lack physical consistency. The evolved equations can produce accurate predictions but may

do so through mechanisms that don't align with physical understanding. Future work should focus on increasing the physical validity of these models. Several approaches could help: enforcing dimensional consistency in the evolved equations, adding physics-informed constraints during evolution, or incorporating principles from Physics-Informed Neural Networks (PINNs) (Cai et al. 2021; Cuomo et al. 2022).

Progress in this direction could yield models that not only predict coastal evolution but also advance our understanding of coastal processes. By incorporating spatial relationships and physical constraints, we might discover new patterns in how shorelines respond to environmental forcing. Such discoveries could prove particularly valuable as coastal communities face the challenges of climate change.

4.2 Discovering El-Niño Models

ENSO represents one of Earth's most influential climate patterns. Like a planetary heartbeat, it drives global weather variations through complex interactions between the ocean and atmosphere. When this pattern shifts toward El Niño conditions, the central and eastern Pacific warms abnormally, weakening trade winds and altering weather patterns worldwide. During La Niña, the opposite occurs - cooler waters in the eastern Pacific strengthen trade winds, creating distinct but equally far-reaching effects (Timmermann et al. 2018).

Despite its profound importance for agriculture, public health, and economic planning, accurate ENSO prediction remains challenging. Traditional approaches fall into two categories: dynamical models that simulate physical processes and statistical models that learn from historical patterns. Both face significant limitations. Physical models struggle to capture the full complexity of tropical climate patterns, while statistical approaches can miss crucial mechanisms that drive ENSO evolution (Ibebuchi and Richman 2024).

Most current prediction systems rely on ensemble forecasting, where multiple simulations run in parallel to account for uncertainties. The JAMSTEC Scale Interaction Experiment-Frontier (SINTEX-F) model exemplifies this approach, maintaining an ensemble of over 100 members to predict ENSO evolution (Doi, Behera, and Yamagata 2016, 2019). However, even with substantial computational resources, questions remain about how best to combine these ensemble predictions. The standard practice of simple averaging across ensemble members may not capture the full predictive power of these sophisticated models.

This section examines how machine learning can improve ENSO prediction by discovering more effective ways to combine ensemble forecasts. Working with JAMSTEC, we explored alternatives to simple ensemble averaging using LMGP. This approach allows us to evolve interpretable mathematical expressions for combining ensemble members, potentially revealing new insights about which model configurations and parameters matter most for accurate prediction. The work demonstrates how automated discovery methods can enhance our understanding and prediction of critical climate phenomena.

4.2.1 SINTEX-F model

The SINTEX-F model represents one of several major climate prediction systems used for seasonal forecasting. At its core, SINTEX-F employs 12 different model configurations that vary in their initialization data and physical parameterizations. These configurations

differ in three key aspects: the choice of observational sea surface temperature datasets used for initialization, the strength of temperature feedback in the initialization phase, and the parameterization of ocean vertical mixing.

To increase the robustness of predictions, this base ensemble of 12 configurations was expanded using lagged average forecasting. This method creates additional ensemble members by initializing the same model configurations on different nearby start dates. For each of the 12 base configurations, forecasts were initialized on eight consecutive days (June 1-8), resulting in a total of 108 ensemble members (Doi, Behera, and Yamagata 2019). While other methods exist for generating ensemble members, the lagged average approach has the advantage of preserving the underlying physical dynamics of the system.

Our study focuses specifically on the SINTEX-F ensemble's prediction of the Niño 3.4 index (Trenberth and Stepaniak 2001). This index, measuring sea surface temperature anomalies in the central equatorial Pacific (5°N-5°S, 170°W-120°W), serves as the primary metric for monitoring ENSO conditions. The choice of this region reflects its importance in ocean-atmosphere coupling - it captures the average equatorial temperatures from roughly the dateline to the South American coast. When these temperatures deviate significantly from normal conditions, they signal the development of El Niño or La Niña events.

The ensemble generates monthly forecasts of the Niño 3.4 index, with each member producing its own prediction trajectory. Currently, these 108 individual predictions are combined through simple averaging to create the final forecast. However, this straightforward approach may not optimally capture the information contained in the ensemble. Some members or combinations of members might provide more reliable predictions under certain conditions. Our work explores whether machine learning can discover more effective ways to combine these predictions, potentially revealing patterns in which ensemble members contribute most to accurate forecasts.

For this preliminary study, we focus solely on improving the ensemble prediction of the Niño 3.4 index rather than examining the full SINTEX-F forecast system. This simplified scope allows us to directly assess whether evolutionary computation can enhance the combination of ensemble predictions for this crucial climate indicator. The methods developed here could potentially be extended to other aspects of the SINTEX-F system in future work.

4.2.2 Code optimization

Building on the LMGP framework introduced in Chapter 3, we explored ways to improve the SINTEX-F ensemble predictions. Rather than simply averaging ensemble members, we sought to discover more sophisticated combination strategies that could better capture the system's predictive power. The search began with hand-written candidate solutions that implemented various aggregation methods, from weighted averages to conditional combinations based on model configurations.

The optimization process balanced three objectives: Root Mean Square Error (RMSE), correlation, and novelty. For overall prediction accuracy, we used the standard RMSE between predicted and observed Niño 3.4 values. We also incorporated correlation as a second objective to ensure the ensemble captured the temporal patterns of the Niño 3.4

index variations. These two metrics complement each other, as RMSE focuses on absolute accuracy while correlation addresses the synchronization of patterns regardless of magnitude.

Finally, to encourage exploration, we include novelty as a third objective, calculated as in Novelty Search (Lehman and Stanley 2011a). As in the Curiosity-ES method discussed in Chapter 2, this additional fitness balances the exploration drive with the two data-based objectives. The optimization was handled through NSGA-II (Deb et al. 2002), a multi-objective evolutionary algorithm that maintains a Pareto front of solutions trading off these objectives. NSGA-II has previously been used with novelty as an objective (Paolo et al. 2021) as a means of combining quality and diversity. This approach helps avoid premature convergence to a single solution type by rewarding novel approaches to the ensemble problem. Code solutions were generated using Language Model Crossover (Meyerson et al. 2024). To rigorously evaluate our solutions, we split the data into five folds, using the final fold (covering data from 2015 to 2022) for out-of-sample testing.

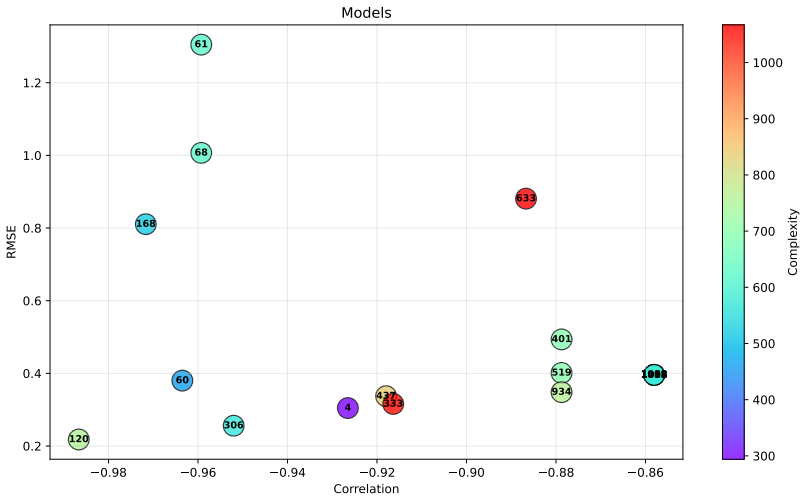


Figure 4.6: Evolved models organized by their correlation and RMSE scores relative to Niño 3.4 data, with different colors indicating program complexity.

The resulting Pareto front, shown in Figure 4.6, reveals interesting trade-offs between our objectives. Each point represents a different combination strategy, positioned according to its RMSE and correlation with observed data. The color of each point indicates program complexity, providing an additional dimension of information. This visualization demonstrates how different regions of the solution space offer a variety of models, which can be evaluated by experts for final selection.

From this Pareto front, we selected Solution 120 as a particularly promising candidate. This solution, shown in Figure 4.8, implements a sophisticated strategy for combining ensemble predictions. The solution applies a multi-stage weighting approach: first weighting ensemble members according to their metadata characteristics (SST observation dataset, SST-nudging feedback, and SVS mixing), then applying time-dependent blending

```python
def ensemble_forecast(forecast_metadata):
    '''
    Generate a forecast by combining the SINTEX ensemble predictions.

    Args:
        forecast_metadata: DataFrame containing forecast data and metadata
            - Metadata columns include: 'SST observation', 'SST-nudging feedback',
              'SVS mixing', and 'forecast start'
            - Remaining columns are the forecasts for each timestamp

    Returns:
        numpy.ndarray: Predicted Nino 3.4 index values across all timestamps
    '''
    # Find where the forecast data begins (after metadata columns)
    metadata_cols = ['SST observation', 'SST-nudging feedback', 'SVS mixing', 'forecast
     start']
    forecast_cols = [col for col in forecast_metadata.columns if col not in metadata_cols]

    # Strategy: Weighted averaging and time-dependent blending based on metadata,
     incorporating
    #           SST observation, SST-nudging feedback, forecast start date, and SVS mixing.
    #           Also, incorporates dynamic weighting for extreme events.

    # 1. Weighting based on SST observation dataset
    sst_weights = {
        'OISSTv2weekly': 0.4,
        'OISSTv2daily': 0.5,  # Favor OISSTv2daily slightly
        'ERSSTv5': 0.3
    }

    # 2. Weighting based on SST-nudging feedback
    nudging_weights = {
        -1200: 1.1,  # Favor -1200
        -2400: 0.9,  # Slightly penalize extreme nudging
        -800: 1.0
    }

    # 3. Prepare ensemble members based on forecast start date
    may1_members = forecast_metadata[forecast_metadata['forecast start'] == 'May1'][
     forecast_cols]
    jun1_members = forecast_metadata[forecast_metadata['forecast start'] == 'Jun1'][
     forecast_cols]

    # Handle missing start dates gracefully
    if len(may1_members) == 0 and len(jun1_members) == 0:
        return forecast_metadata[forecast_cols].mean(axis=0).values
    if len(may1_members) == 0:
        return jun1_members.mean(axis=0).values
    if len(jun1_members) == 0:
        return may1_members.mean(axis=0).values

    may1_forecasts = may1_members.values
    jun1_forecasts = jun1_members.values

    n_timesteps = len(forecast_cols)
    result = np.zeros(n_timesteps)

    # Apply weights to ensemble members before time-dependent blending
    may1_weighted_forecasts = np.zeros((may1_forecasts.shape[0], may1_forecasts.shape[1]))
    jun1_weighted_forecasts = np.zeros((jun1_forecasts.shape[0], jun1_forecasts.shape[1]))
```

Figure 4.7: First half of the code of the selected solution, number 120

```
    for i in range(may1_forecasts.shape[0]):
        # SST observation weight
        obs_type = forecast_metadata['SST observation'].iloc[forecast_metadata.index[
    may1_members.index[i]]]
        sst_weight = sst_weights.get(obs_type, 0.2)  # Default weight if not found

        # Nudging feedback weight
        nudging_feedback = forecast_metadata['SST-nudging feedback'].iloc[forecast_metadata.
    index[may1_members.index[i]]]
        nudging_weight = nudging_weights.get(nudging_feedback, 1.0) # Default weight

        # SVS mixing weight (if enabled, slightly boost)
        svs_mixing = forecast_metadata['SVS mixing'].iloc[forecast_metadata.index[
    may1_members.index[i]]]
        svs_weight = 1.1 if svs_mixing else 1.0

        may1_weighted_forecasts[i, :] = may1_forecasts[i, :] * sst_weight * nudging_weight *
     svs_weight

    for i in range(jun1_forecasts.shape[0]):
        # SST observation weight
        obs_type = forecast_metadata['SST observation'].iloc[forecast_metadata.index[
    jun1_members.index[i]]]
        sst_weight = sst_weights.get(obs_type, 0.2)  # Default weight if not found

        # Nudging feedback weight
        nudging_feedback = forecast_metadata['SST-nudging feedback'].iloc[forecast_metadata.
    index[jun1_members.index[i]]]
        nudging_weight = nudging_weights.get(nudging_feedback, 1.0) # Default weight

        # SVS mixing weight (if enabled, slightly boost)
        svs_mixing = forecast_metadata['SVS mixing'].iloc[forecast_metadata.index[
    jun1_members.index[i]]]
        svs_weight = 1.1 if svs_mixing else 1.0

        jun1_weighted_forecasts[i, :] = jun1_forecasts[i, :] * sst_weight * nudging_weight *
     svs_weight

    # Time-dependent blending of May1 and Jun1 forecasts
    for t in range(n_timesteps):
        june_weight = min(1.0, t / (n_timesteps / 2))  # Increasing weight for June start
        may_weight = 1.0 - june_weight

        may1_avg = np.mean(may1_weighted_forecasts[:, t])
        jun1_avg = np.mean(jun1_weighted_forecasts[:, t])

        # Dynamic weighting for extreme events (El Nino/La Nina)
        # This section is designed to amplify extreme events.
        extreme_threshold = 0.7  # Define a threshold for extreme events
        may1_extreme_count = np.sum(np.abs(may1_weighted_forecasts[:, t]) >
    extreme_threshold)
        jun1_extreme_count = np.sum(np.abs(jun1_weighted_forecasts[:, t]) >
    extreme_threshold)

        # Boost the influence of ensemble members predicting extreme events
        may1_extreme_weight = 1.0 + (may1_extreme_count / may1_forecasts.shape[0]) # More
    extreme members -> higher weight
        jun1_extreme_weight = 1.0 + (jun1_extreme_count / jun1_forecasts.shape[0])

        may1_avg_extreme = may1_avg * may1_extreme_weight
        jun1_avg_extreme = jun1_avg * jun1_extreme_weight

        result[t] = (may_weight * may1_avg_extreme + june_weight * jun1_avg_extreme)

    return result
```

Figure 4.8: Second half of the code of the selected solution, number 120.

between different forecast initialization dates, and finally implementing a dynamic weighting mechanism that amplifies predictions of extreme events. While complex, this solution remains interpretable through decomposition—each section performs a clear function that can be analyzed individually.

The logic of the solution reflects domain knowledge about the SINTEX model. It assigns higher weights to configurations using newer observational datasets (OISSTv2daily) and certain physical parameterizations, suggesting these choices may be particularly important for prediction accuracy. The time-dependent blending recognizes that forecasts initialized at different dates have varying skill over the forecast horizon, with gradually increasing weight given to later initializations as the forecast extends further in time. Perhaps most interestingly, the solution incorporates special handling for extreme events, boosting the influence of ensemble members that predict anomalies exceeding a threshold. This represents a sophisticated insight about ensemble behavior during El Niño and La Niña events.

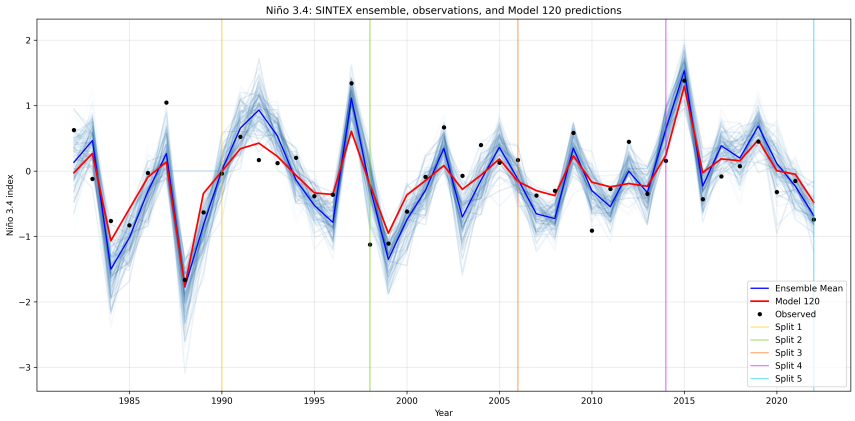


Figure 4.9: Comparison of Solution 120 to the ensemble mean and members on Niño 3.4 prediction.

Figure 4.9 compares the predictions of Solution 120 with the standard ensemble mean and observed values. It is notable that in periods where the ensemble mean overshoots the Niño 3.4 prediction, Solution 120 better matches the observational data. However, this conservative approach comes at the cost of not predicting outlier years like 1997 and 2010. An analysis of the optimized solutions based on extreme events using a metric like Symmetric Extremal Dependence Index (SEDI) (Ferro and Stephenson 2011) could provide complementary insight to this more conservative model.

Table 4.1 we compare Solution 120 to other ensemble methods on the held-out split of 2015 to 2022. Solution 120 outperforms both the traditional ensemble mean and the machine learning approaches, including neural networks and random forests. The comparison with a Bayesian Neural Network is particularly noteworthy, as this approach has been used for ensemble weighting of geophysical models (Sengupta et al. 2020).

Model	RMSE	Correlation
Solution 120	0.2179	0.9866
Ensemble Mean	0.2637	0.9623
Random Forest	0.3788	0.9139
Bayesian NN	0.4595	0.8784
Neural Network	0.5237	0.8156

Table 4.1
Machine learning methods compared to the ensemble mean and Solution 120.

The diversity of viable solutions in our NSGA-II Pareto front suggests there may not be a single "best" way to combine ensemble predictions. Different combination strategies may excel under different conditions or for different aspects of prediction. This finding aligns with work by Monteleoni et al. (2011), who demonstrated adaptive tracking of climate model performance over time. Our approach of discovering ensemble strategies that extract information in diverse ways could be combined with an adaptive tracking mechanism to propose different solutions from the Pareto front over time. Furthermore, given that the models are interpretable, experts can choose between ensemble methods in full understanding of their functionality.

This study represents a first step in applying LMGP to ensemble climate prediction. While the results demonstrate the potential of evolutionary computation for discovering interpretable combination strategies, several important directions remain to be explored. More extensive analysis of the evolved models could reveal patterns in which ensemble configurations contribute most to accurate predictions under different conditions. Additional physical variables beyond the Niño 3.4 index could be incorporated to create more comprehensive prediction strategies. Nevertheless, these preliminary results already suggest that LMGP can serve as a valuable tool for scientific modeling, offering solutions that balance performance with interpretability. The discovered models not only improve prediction accuracy but also provide insights into which aspects of the ensemble forecast system matter most for different prediction tasks.

4.3 Perspectives

Scientific discovery has always required both exploration of the unknown and understanding of what we find. The work presented in this chapter demonstrates how machine learning can enhance both aspects of this process. Through interpretable models discovered by genetic programming, we showed how computational techniques can reveal new patterns in complex environmental systems while maintaining the clarity needed for scientific insight.

Our investigations of shoreline evolution and ENSO prediction illustrate different facets of this approach. In coastal modeling, genetic programming discovered equations that not only improved prediction accuracy but also suggested new relationships between environmental drivers. The ENSO study showed how machine learning can enhance existing scientific models by discovering more effective ways to combine ensemble predictions. Both cases demonstrate that machine learning need not operate as a black box - with appropriate methods, it can generate discoveries that scientists can analyze and understand.

The systematic exploration enabled by evolutionary algorithms proves particularly valuable for scientific discovery. Where traditional hypothesis testing follows relatively linear paths, computational exploration can systematically investigate vast spaces of possible models. This capability might have accelerated historical discoveries - imagine if Kepler had access to tools that could systematically test different orbital shapes against Brahe's data, perhaps suggesting elliptical orbits as a promising alternative to circles. However, the key insight is that such tools augment rather than replace human scientific reasoning. They expand our capacity to explore and understand complex phenomena, just as microscopes and telescopes expanded our ability to observe the natural world.

The interpretability of discovered models remains crucial for scientific progress. While black-box machine learning models might achieve impressive predictive accuracy, they cannot advance scientific understanding without providing insights into the mechanisms behind their predictions. Our work demonstrates that we need not sacrifice interpretability for performance. Through careful design of search spaces and optimization objectives, we can discover models that are both accurate and understandable.

The urgency of climate change makes this approach particularly relevant today. Better understanding of phenomena like coastal erosion and ENSO can help communities prepare for and adapt to changing environmental conditions. The acceleration of scientific discovery through computational means may prove essential for developing effective responses to climate challenges. Just as the advent of computers transformed fields like space mission planning - not by replacing human insight but by enabling more efficient exploration of possibilities - machine learning can enhance our capacity for scientific discovery when we need it most.

These examples from environmental science point toward a broader transformation in how we conduct scientific research. The next chapter will examine this transformation in detail, proposing a framework for integrating machine learning into the scientific process. By combining the creative power of computational exploration with the rigor of interpretable modeling, we can develop tools that accelerate discovery while maintaining the understanding that science requires.

5 Directions

Scientific research rarely follows a straight path. Through advising six PhD students and leading multiple research projects, my work has evolved from optimizing artificial intelligence to applying it for scientific discovery. This chapter examines that evolution, from early work on automated machine learning through to current efforts in climate science. The journey reflects both personal growth as a researcher and shifting priorities in the face of global challenges.

Previous chapters detailed key themes that emerged through this research: exploration methods that systematically search vast possibility spaces, interpretable models that advance scientific understanding, and frameworks for automated discovery. These themes now converge on a singular focus: accelerating climate science. Where Chapter 2 developed methods for creative exploration and Chapter 3 showed how to maintain understanding of complex systems, Chapter 4 demonstrated their application to pressing environmental challenges. This final chapter charts the course ahead.

The chapter proceeds through three stages of evolution in research direction. First, it examines early work on automated machine learning, particularly through the thesis of Kaitlin Maile on neural architecture optimization. Second, it traces the transition toward climate science, following the work of Mahmoud Al Najar's thesis on shoreline forecasting presented in Chapter 4 and leading to current work on climate modeling.

The chapter concludes with a detailed examination of future directions, centered on a research proposal. This proposed research program aims to accelerate climate science through interpretable optimization of scientific code. By applying the lessons learned from years of research in evolutionary computation and machine learning, it represents both a natural progression of previous work and a focused response to one of our greatest challenges, the climate crisis.

5.1 Accelerating AI Discovery

The first steps under my research direction began with an attempt common to many AI researchers: applying AI to improve AI itself (Hutter, Kotthoff, and Vanschoren 2019). AI requires many choices in its application, from hyperparameters to architectures, and making these choices consumes both human time and computational resources. Through the thesis of Kaitlin Maile (Maile 2023), we explored how AI could help make these decisions.

This research direction exemplifies a core theme that would later drive my interest in scientific discovery: the challenge of exploring vast spaces of possibilities to find effective

solutions. The space of possible neural architectures is immense, growing exponentially with depth and constrained only by our computational resources (Elsken, Metzen, and Hutter 2019). The question of architecture choice has taken on new urgency as AI systems grow larger and consume more energy (Samsi et al. 2023). The following work on automated neural architecture search, including the optimization of both structure and symmetry constraints, points toward methods for creating more efficient AI systems.

5.1.1 Discovering symmetry

Neural networks can be constrained to respect symmetries in data, such as how rotating an image should result in the same classification rotated (Veeling et al. 2018). These constraints, known as equivariances, improve generalization but are typically applied uniformly across an entire network. The challenge lies in determining which symmetry constraints to apply at each layer of the network.

In Maile, Wilson, and Forré (2023), we developed two approaches to automatically discover appropriate symmetry constraints. The first uses an evolutionary algorithm to gradually relax symmetry constraints while preserving network function. The second approach allows each layer to smoothly mix different symmetry constraints, learning the right balance through gradient descent.

We tested these methods on several image classification tasks with varying degrees of inherent symmetry. The algorithms discovered that early layers benefit from stronger symmetry constraints while later layers work better with more flexibility. This matches the intuition that early visual processing should respect basic geometric transformations while higher-level features may need to break these symmetries.

Most notably, networks with appropriate symmetry constraints achieved equal or better performance with fewer parameters. By sharing weights according to geometric symmetries, these networks required less memory and computation. This demonstrates how incorporating knowledge about data structure can lead to more efficient AI systems.

5.1.2 Neurogenesis

Most neural networks start with a fixed size, often larger than necessary to ensure sufficient capacity for learning. This overprovisioning of resources can lead to wasteful computation and energy usage (Luccioni, Jernite, and Strubell 2024). Through the NORTH* algorithm, we developed methods to grow networks dynamically during training, adding neurons only when needed (Maile et al. 2022).

The key insight was to monitor the independence of neural activations. When neurons begin producing redundant outputs, it signals that the network has saturated its current capacity. At this point, new neurons are added in a way that maximizes their potential to contribute novel features. This process continues until the network achieves both its performance goals and efficient use of parameters.

This approach consistently produced networks that were both smaller and more efficient than traditional fixed architectures. Rather than starting with an oversized network and pruning it down, NORTH* builds networks from the ground up, ensuring each neuron serves a purpose. The resulting networks typically used 50-80% fewer parameters while maintaining competitive performance.

This work demonstrates a fundamental principle for efficient AI: start small and grow only as needed. Just as nature builds complex systems through gradual development rather than trimming down oversized structures, neural networks can benefit from careful, targeted growth. This principle becomes increasingly important as we consider the energy costs of training and deploying AI systems at scale.

5.1.3 Energy impact of AI

These works aimed to automate architectural decisions in neural networks, seeking to reduce the human effort in applying AI. However, these methods for creating minimal, efficient networks take on new importance as the energy consumption of AI becomes a pressing concern. Recent studies show that the computational demands of large AI models contribute to a significant carbon footprint (Samsi et al. 2023; Luccioni, Jernite, and Strubell 2024)

The rise of commercial AI products, particularly large language models, has led to rising energy consumption in the AI sector. While a single inference might seem negligible, the cumulative effect of millions of queries to these models creates substantial energy demands. The trend toward larger, more general models rather than specialized ones further compounds this issue. Task-specific models can be orders of magnitude more efficient than their general-purpose counterparts.

Methods like EquiNAS and NORTH* could help address this challenge by optimizing model architectures for specific tasks. Rather than deploying large, general models, we could automatically generate smaller, specialized networks that incorporate relevant symmetries and use only necessary neurons. The equivariance constraints discovered by EquiNAS could reduce parameter counts, while the growing process of NORTH* could ensure networks remain as small as possible.

However, an important question remains: would more efficient AI systems actually reduce overall energy consumption? Historical precedent suggests that making a technology more efficient often leads to increased usage rather than reduced resource consumption, referred to as Jevons' paradox (Jevons 1865). The development of more efficient AI architectures might accelerate AI adoption, potentially increasing rather than decreasing total energy usage. This paradox is one argument of why my future research focuses not on making AI more efficient, but on applying AI directly to climate challenges.

5.1.4 Neural Networks as Code

A second research direction that follows the work of Kaitlin Maile would be to make neural networks more like code—correctable, manually definable, and decomposable into understandable components. Neural networks excel at pattern recognition but remain opaque black boxes, while scientific code offers interpretability but lacks the adaptability to learn from data. Rather than optimizing code like we optimize neural networks, this approach would transform neural networks to work more like software.

For neural networks to work like code, they must first be modular. The self-distillation approach proposed in Zhang, Bao, and Ma (2021) moves in this direction by creating intermediate outputs that can be separately analyzed. Modularity has long been a subject of interest in the evolution and development of neural networks (Gruau 1994; Miller and Wilson 2017), but recent trends favor large, monolithic models with obscure interfaces. By

making neural networks more modular, we could break them into discrete functional units, study each component's decision process, and fix specific parts that exhibit errors without retraining the entire system.

A second necessary advance would be the ability to directly define neural networks through human specification rather than optimization alone. Currently, creating a neural network that implements even a simple function like $f(x) = 0.2 \cdot \sin(x) + x^2$ requires generating training data and optimizing parameters through gradient descent. In contrast, programmers can simply write such functions directly in code. The recent work in Yuksekgonul et al. (2025) moves toward this vision by using text-based feedback to modify AI systems, but still lacks the immediacy and precision of manual definition that is characteristic of software development.

If neural networks could be manipulated like software—with modules that can be composed, inspected, modified, and reused—then interpretability would emerge naturally from the design process. This would allow scientists to combine the learning capabilities of neural networks with the transparency of traditional scientific code. Errors could be corrected through targeted interventions rather than complete retraining, and domain expertise could be directly incorporated into models rather than indirectly through data curation. This direction, while challenging, could lead to AI systems that maintain the performance benefits of neural networks while gaining the reliability, trustworthiness, and intellectual accessibility of well-written scientific code.

5.2 Towards Climate Modeling

While there are interesting directions in using automated discovery methods to improve AI, it is questionable if the pace of AI research would benefit from an even faster pace. Rather, these methods can be put to immediate use in climate science, where a better understanding of the warming world can help us find solutions and mitigate damage. Recent findings show that global warming has already exceeded 1.5°C above pre-industrial levels (McCulloch et al. 2024). Even if carbon emissions ceased immediately, the thermal inertia of our oceans means continued warming for decades to come (Oh et al. 2024). This reality creates an urgent need for better understanding of climate systems, not just for mitigation but for adaptation to changes already locked in.

Some argue that we already understand climate change sufficiently, that the time for research has passed and only action remains. I believe that this view, while understandable given the urgency of the crisis, misses a crucial point: improved scientific understanding represents a form of action. Better models enable more effective responses, whether in coastal protection, agricultural adaptation, or disaster preparedness. The methods developed in previous chapters - systematic exploration and interpretable modeling - find natural application in advancing this understanding.

Climate models face three key limitations that machine learning could help address: uncertainty in current understanding, gaps in prediction capabilities, and difficulty in attribution of human impacts. These challenges align with the strengths of automated discovery methods, suggesting productive directions for future research. The following sections examine these limitations and how interpretable machine learning might help overcome them.

5.2.1 Climate Model Limitations

Current climate models represent remarkable achievements of scientific understanding, simulating complex interactions between the atmosphere, oceans, and land masses. Yet they face persistent uncertainties, particularly in representing phenomena that operate across different temporal and spatial scales. These limitations matter not just for scientific understanding but for society's ability to prepare for and adapt to climate change. Many critical processes, such as Arctic melt, cloud formation, and ocean circulation, remain incompletely understood or poorly represented in current models. As climate change accelerates, these uncertainties become increasingly consequential, as they affect our ability to predict and mitigate its impacts.

The El Niño-Southern Oscillation (ENSO) phenomenon exemplifies these challenges. Despite its profound influence on global weather patterns, from droughts in Australia to flooding in South America, our ability to predict ENSO events remains limited (Timmermann et al. 2018; Ibebuchi and Richman 2024). Significant uncertainties remain about how ENSO patterns might shift as oceans warm and circulation patterns change (Yeh et al. 2009). Understanding these changes becomes increasingly crucial as communities worldwide depend on ENSO forecasts for agricultural planning and disaster preparedness.

Scale presents another fundamental challenge. Climate processes operate across vast ranges of time and space, from microscale cloud formation to global circulation patterns. Current models must either simplify these interactions or demand enormous computational resources (Eyring et al. 2019). These compromises create uncertainties that cascade through predictions, affecting everything from regional rainfall patterns to sea level rise estimates. For example, Arctic melt is a critical process that remains poorly modeled. The melting of glaciers and sea ice involves complex feedback loops between temperature, albedo, and ocean currents, which are not fully captured in current simulations. As the Arctic warms faster than other regions, these uncertainties grow, making it difficult to predict the rate and extent of melting and its global impacts.

The advent of satellite observation and sophisticated sensor networks has generated massive amounts of climate data. Yet integrating this wealth of information into existing models remains challenging (Eyring et al. 2024). Our work on shoreline forecasting demonstrated this gap—while traditional models relied primarily on wave data, incorporating satellite observations of sea level anomalies improved predictions (Al Najar et al. 2023). Similar opportunities exist across climate science, where observational data might reveal patterns that current physics-based models miss or oversimplify. For instance, satellite data on Arctic ice thickness and melt ponds could help refine models of glacial melt (Reil et al. 2024), but integrating such data into existing frameworks remains a work in progress.

These limitations do not diminish the achievements of current climate science. Rather, they point toward opportunities where machine learning techniques, particularly those focused on interpretable modeling and efficient exploration, could enhance our understanding. As we face the need to prepare for climate impacts, addressing these limitations becomes increasingly urgent. The next section examines how improved models could enhance society's preparedness for climate change.

5.2.2 Preparing for the crisis

Climate change has moved beyond a future threat to a present reality requiring immediate adaptation. Communities worldwide face decisions about infrastructure, agriculture, and disaster response with profound long-term consequences. Models serve as essential tools for these decisions, translating scientific understanding into actionable insights for planners and policymakers.

Our work on shoreline forecasting illustrates this practical dimension. Coastal communities cannot wait for perfect understanding of ocean dynamics before taking action. They need reliable predictions to guide decisions about seawalls, managed retreat, or natural solutions like mangrove restoration. The interpretable models we developed not only improved prediction accuracy but also helped explain which factors drive coastal change, leading the way to more informed adaptation strategies.

Similarly, enhanced ENSO prediction could transform how regions prepare for climate variability. When El Niño events strengthen, some regions face increased risk of flooding while others confront severe drought (Power et al. 2013). Better forecasting, particularly at longer time scales, would allow communities to adjust agricultural practices, prepare emergency responses, or manage water resources more effectively. These preparations become increasingly critical as climate change potentially alters the intensity and frequency of ENSO events (Cai et al. 2014).

Recent advances in weather forecasting through machine learning, exemplified by models like GraphCast, demonstrate the potential for improved prediction of extreme events (Lam et al. 2023). As climate change increases the frequency and intensity of such events, the ability to provide accurate, timely forecasts becomes crucial for public safety. However, these models currently operate as black boxes, limiting their utility for understanding how climate change affects weather patterns. This highlights the need for interpretable approaches that can both predict and explain changing weather dynamics.

The gap between scientific understanding and practical action often stems from uncertainty about local impacts. While global climate projections have grown more robust, translating these into local implications remains challenging. Interfaces between different systems are often poorly understood, as they fall between expertise of specialists in specific systems. Machine learning methods, particularly those maintaining interpretability, could help bridge this gap. By discovering patterns in local data while respecting physical constraints, they could provide the specific insights needed for effective adaptation planning.

5.2.3 Understanding our impact

Understanding which climate changes stem from human activity and which represent natural variation presents a fundamental scientific challenge. This distinction matters not just for attribution but for evaluating the effectiveness of mitigation efforts. Models serve as essential tools for this analysis, helping separate anthropogenic signals from natural climate variability.

Consider agricultural practices, where human decisions directly affect local and regional climate patterns. Studies of cover crops demonstrate how farming choices influence both carbon sequestration and local temperature regulation (Kaye and Quemada 2017). Models

helped quantify these effects, showing that cover crops can mitigate warming by approximately 100-150g CO2 equivalent per square meter annually. Such precise understanding enables evidence-based policy decisions.

As climate impacts intensify, we increasingly see examples of unintended consequences from human interventions in natural systems. These range from the effects of large-scale deforestation on regional rainfall patterns to the impact of urban development on local temperature extremes. Understanding these dynamics becomes crucial as communities implement adaptation measures that may themselves affect climate systems.

The scientific community now faces an additional challenge: evaluating proposed deliberate interventions in the climate system. Despite the risks and uncertainties involved, some researchers have begun modeling various geoengineering approaches (Kravitz et al. 2011). These studies reveal the complexity of climate response to intervention, with models showing that even successful global temperature reduction could have uneven regional impacts (Kravitz et al. 2013).

The existence of these proposals makes robust modeling capabilities more crucial than ever. Not to promote such interventions, but to understand their potential consequences and risks. Current models already show that proposed interventions like stratospheric aerosol injection could have widely varying regional effects (Visioni et al. 2021). This underscores the need for models that can better predict the full range of potential impacts before any large-scale intervention is considered.

The role of climate modeling thus extends beyond prediction to responsibility. As pressure for climate action grows, models must help evaluate both intended and unintended consequences of human decisions. This requires not just accurate predictions but interpretable results that can inform policy discussions and public understanding. The stakes of climate intervention demand nothing less than complete transparency in our modeling approaches.

5.3 Accelerating Climate Science

The previous sections establish both the urgency of climate modeling and the potential for machine learning to accelerate scientific understanding. I have formulated my research objectives in this directions through an ERC Starting Grant proposal, entitled Accelerating Science with Automatic and Interpretable Model Improvement (AIMI). Rather than applying machine learning directly to raw climate data, AIMI focuses on improving the scientific models themselves through interpretable optimization techniques. While the AIMI project is only a proposal, I detail it here as it lays out a concrete vision for applying AI to climate science.

This approach builds on key themes developed throughout this manuscript: exploration methods for discovering novel solutions, interpretable modeling techniques that advance scientific understanding, and automated discovery processes that accelerate research. However, where previous work used a variety of standard benchmarks to measure and demonstrate methodological progress, AIMI redirects these techniques toward the more urgent challenge of understanding climate change.

The project divides this challenge into two complementary objectives, shown in Figure 5.1. The first develops novel methods for optimizing scientific code, building on

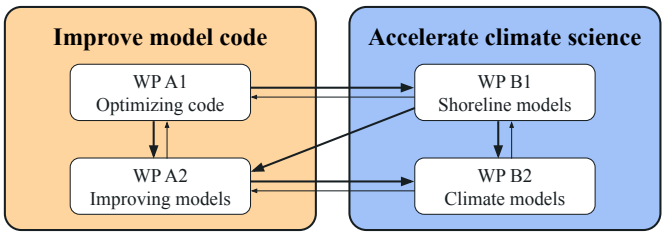


Figure 5.1: The two objectives and four theses of the AIMI project. Arrows indicate the expected flow of information and collaboration between theses.

recent advances in large language models and evolutionary algorithms. The second applies these methods to accelerate climate science, focusing on two critical areas: shoreline forecasting and ENSO prediction. This structure reflects a key insight from my previous research: the need to develop methods in parallel with their application, allowing each to inform the other.

The scope of AIMI represents an intentional focus of my research agenda. Where previous work explored diverse applications of evolutionary computation and machine learning, this project commits fully to climate science. This focus stems from both the urgency of the climate crisis and the recognition that meaningful progress requires sustained, coordinated effort. The following sections detail how AIMI's approach to code optimization and model improvement could accelerate our understanding of climate systems.

5.3.1 Improve model code

The first component of AIMI centers on developing methods to automatically improve scientific code. This work builds on recent advances in Large Language Models (LLMs) that can generate and modify computer programs, as demonstrated in Chapter 4. Rather than replacing existing scientific models, these methods aim to enhance them by discovering more efficient or accurate implementations.

The first proposed thesis in this direction explores how LLMs can generate and optimize code based on scientific data. While LLMs have shown an impressive ability to write code (Li et al. 2022), their output often lacks the precision required for scientific computing. By combining LLMs with evolutionary search techniques, we can generate code that both performs well on observational data and respects physical constraints. This approach draws from my previous work on genetic programming while leveraging the advanced capabilities of modern language models.

The second proposed thesis addresses a complementary challenge: integrating physical knowledge into code optimization. Scientific models must respect fundamental laws of physics, conservation principles, and domain-specific constraints. This work develops methods for embedding these constraints into the optimization process, ensuring that generated code remains physically valid while improving performance. The techniques build on advances in physics-informed machine learning but apply them to direct code optimization rather than neural networks.

Supporting these theses, a research engineer position focuses on developing robust software tools for scientific code optimization. This role ensures that theoretical advances

translate into practical tools that scientists can use. The goal is to create an open-source framework that makes automated code improvement accessible to researchers across scientific domains.

While these methods apply broadly to scientific computing, their development is guided by the needs of climate modeling. The complexity of climate models, with their intricate physical relationships and computational demands, provides an ideal testing ground for code optimization techniques. Success in this domain would demonstrate the potential for automated discovery to accelerate scientific understanding more broadly.

5.3.2 Accelerate climate science

The second component of AIMI applies code optimization techniques to two critical areas of climate science: coastal dynamics and large-scale climate patterns. These applications build directly on my previous work while targeting areas where improved models could impact our understanding of climate change.

The first proposed thesis extends our work on shoreline forecasting from Chapter 4 to regional scales. Where previous models operated at single coastal points, this work aims to capture how shorelines evolve across entire coastlines. This scaling presents both computational and physical challenges - shorelines don't evolve in isolation but influence each other through sediment transport and wave patterns. By combining satellite data with optimized physical models, we seek to understand these broader coastal dynamics.

The second proposed thesis focuses on improving predictions of the El Niño-Southern Oscillation (ENSO). Working with the Japan Agency for Marine-Earth Science and Technology (Japan Agency for Marine-Earth Science and Technology (JAMSTEC)), this research aims to enhance their ENSO prediction system, building on the work presented in Chapter 4. Going beyond the ensemble predictions, this thesis is intended to directly improve the physical code of the ENSO models. This work could improve our ability to forecast ENSO events, improving understanding of climate events worldwide.

A postdoctoral researcher bridges these two efforts, focusing on how local and global climate patterns interact. This position examines how large-scale phenomena like ENSO influence local processes like coastal erosion. Understanding these cross-scale interactions becomes increasingly important as climate change alters both global circulation patterns and local weather extremes.

These applications target phenomena where better predictions could directly inform adaptation strategies. Improved shoreline models could help coastal communities plan for sea-level rise, while better ENSO forecasts could enhance agricultural and disaster preparedness. The interpretability of our approach ensures that discoveries not only improve predictions but advance scientific understanding of these critical climate systems.

5.3.3 Perspectives

The AIMI project represents a shift in my approach to research direction. Through previous advising experiences, I learned valuable lessons about guiding research while maintaining individual creativity. These experiences shape how AIMI structures its multiple theses towards a common goal.

One key insight from past projects concerns the complexity of truly interdisciplinary research. The thesis of Mahmoud Al Najar, bridging computer science and coastal dynamics, revealed both the potential and challenges of such work. Research communities, means of publication, and nomenclature vary between fields. Expecting a PhD student to understand these differences and deliver a successful thesis in them is a tall demand. Rather than expecting researchers to fully bridge disciplines within the scope of a single thesis, AIMI pairs theses across domains. Each researcher maintains firm grounding in their primary field while collaborating closely with counterparts in complementary areas.

The project also marks a departure from my previous approach to research direction. Earlier work often followed diverse interests, allowing PhD students freedom to explore their motivations. While this approach led to interesting discoveries, such as Paul Templier's work on quality-diversity to Kaitlin Maile's insights into neurogenesis, I believe that large-scale interdisciplinary projects like AIMI demand more focused effort. AIMI's structured approach draws from my experience with the industrial CIFRE theses of Paul-Antoine le Tolguenec and Estelle Chigot, where clear objectives have guided research without limiting creativity.

A concrete example of this is in the creation and maintenance of scientific code. While each of the theses that I have advised has produced open-source code libraries, they have been distinct. I did not seek to dedicate resources to their maintenance or adoption. That does not limit their adoption; Quality with Just Enough Diversity (JEDi) (Templier, Grillotti, et al. 2024) is implemented in the popular QDAX framework (Chalumeau et al. 2024), for example, enabling its future use. The work of Kaitlin Maile has been reimplemented for later study (Douka et al. 2025), as another example. However, the AIMI project intends for four PhD students to share common methodology around diverse applications. A standard, maintained, and accessible code base will be a necessary part of that and motivated the inclusion of a research engineer in the project.

Perhaps most importantly, AIMI reflects a commitment to building scientific communities rather than just developing algorithms. The interaction between computer scientists and climate researchers requires careful cultivation of shared understanding and trust. My role shifts from purely technical direction to fostering these connections, ensuring that computational advances truly serve scientific needs.

The methods and insights presented in this manuscript point toward a broader transformation in how we approach scientific discovery. There is a need for approaches like convergence research (Sharp and Hockfield 2017)— collaborative, transdisciplinary approaches that unite scientists, engineers, policymakers, and communities. By combining the strengths of machine learning, climate science, and other fields, we can accelerate discovery, improve predictive models, and develop actionable strategies to mitigate and adapt to climate change. This is the future of scientific inquiry and the key to addressing the most urgent challenges of our era.

Bibliography

Al najar, Mahmoud. 2023. "Modelling coastal evolution with machine learning." 2023ESAE0060. PhD diss. http://www.theses.fr/2023ESAE0060.

Al Najar, Mahmoud, Rafael Almar, Erwin WJ Bergsma, Jean-Marc Delvit, and Dennis G Wilson. 2023. "Improving a Shoreline Forecasting Model with Symbolic Regression." In *Tackling Climate Change with Machine Learning, ICLR 2023.*

Al Najar, Mahmoud, Rafael Almar, and Dennis G. Wilson. 2025. "Interpretable Machine Learning for Shoreline Forecasting." *In preparation.*

Almar, Rafael, Julien Boucharel, Marcan Graffin, Gregoire Ondoa Abessolo, Gregoire Thoumyre, Fabrice Papa, Roshanka Ranasinghe, Jennifer Montano, Erwin WJ Bergsma, Mohamed Wassim Baba, et al. 2023. "Influence of El Niño on the variability of global shoreline position." *Nature communications* 14 (1): 3133.

Almar, Rafael, Roshanka Ranasinghe, Nadia Sénéchal, Philippe Bonneton, Dano Roelvink, Karin R Bryan, Vincent Marieu, and Jean-Paul Parisot. 2012. "Video-based detection of shorelines at complex meso–macro tidal beaches." Publisher: The Coastal Education and Research Foundation 1656 Cypress Row Drive, West ... *Journal of Coastal Research* 28 (5): 1040–1048.

Angelov, Plamen P, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. 2021. "Explainable artificial intelligence: an analytical review." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11 (5): e1424.

Atrey, Akanksha, Kaleigh Clary, and David Jensen. 2020. "Exploratory Not Explanatory: Counterfactual Analysis of Saliency Maps for Deep Reinforcement Learning." In *International Conference on Learning Representations.*

Audibert, Jean-Yves, Rémi Munos, and Csaba Szepesvári. 2009. "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits." *Theoretical Computer Science* 410 (19): 1876–1902.

Bellemare, M. G., Y. Naddaf, J. Veness, and M. Bowling. 2013. "The Arcade Learning Environment: An Evaluation Platform for General Agents." *Journal of Artificial Intelligence Research* 47 (June): 253–279.

Berndt, Jon. 2004. "JSBSim: An open source flight dynamics model in C++." In *AIAA Modeling and Simulation Technologies Conference and Exhibit,* 4923.

Beuzen, Tom, and Kristen Splinter. 2020. "Machine learning and coastal processes." In *Sandy beach morphodynamics,* 689–710. Elsevier.

Bi, Kaifeng, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2022. "Pangu-Weather: A 3d High-Resolution Model for Fast and Accurate Global Weather Forecast." arXiv: 2211.02556.

Bodnar, Cristian, Ben Day, and Pietro Lió. 2020. "Proximal distilled evolutionary reinforcement learning." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 34:3283–3290. 04.

Bonou, Frédéric, Donatus B. Angnuureng, Zacharie Sohou, Rafael Almar, Gael Alory, and Yves du Penhoat. 2018. "Shoreline and Beach Cusps Dynamics at the Low Tide Terraced Grand Popo Beach, Bénin (West Africa): A Statistical Approach." *Journal of Coastal Research,* no. 81 (10081) (September): 138–144. ISSN: 0749-0208. https://doi.org/10.2112/SI81-018.1. eprint: https://meridian.allenpress.com/jcr/article-pdf/doi/10.2112/SI81-018.1/1295303/si81-018_1.pdf. https://doi.org/10.2112/SI81-018.1.

Brafman, Ronen I, and Moshe Tennenholtz. 2002. "R-max-a general polynomial time algorithm for near-optimal reinforcement learning." *Journal of Machine Learning Research* 3 (Oct): 213–231.

Brameier, Markus, Wolfgang Banzhaf, and Wolfgang Banzhaf. 2007. *Linear genetic programming.* Vol. 1. Springer.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language models are few-shot learners." In *Proceedings of the 34th International Conference on Neural Information Processing Systems.* NIPS '20. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9781713829546.

Burda, Yuri, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. "Exploration by random network distillation." In *International Conference on Learning Representations.*

Cai, Shengze, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. 2021. "Physics-informed neural networks (PINNs) for fluid mechanics: A review." *Acta Mechanica Sinica* 37 (12): 1727–1738.

Cai, Wenju, Simon Borlace, Matthieu Lengaigne, Peter Van Rensch, Mat Collins, Gabriel Vecchi, Axel Timmermann, Agus Santoso, Michael J McPhaden, Lixin Wu, et al. 2014. "Increasing frequency of extreme El Niño events due to greenhouse warming." *Nature climate change* 4 (2): 111–116.

Cava, William La, Patryk Orzechowski, Bogdan Burlacu, Fabricio Olivetti de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. 2021. "Contemporary Symbolic Regression Methods and Their Relative Performance." In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1).* https://openreview.net/forum?id=xVQMrDLyGst.

Chalumeau, Felix, Bryan Lim, Raphael Boige, Maxime Allard, Luca Grillotti, Manon Flageat, Valentin Macé, Guillaume Richard, Arthur Flajolet, Thomas Pierrot, et al. 2024. "Qdax: A library for quality-diversity and population-based algorithms with hardware acceleration." *Journal of Machine Learning Research* 25 (108): 1–16.

Chen, Angelica, David Dohan, and David So. 2024. "EvoPrompting: language models for code-level neural architecture search." *Advances in Neural Information Processing Systems* 36.

Compton, Kate. 2016. "So you want to build a generator." *Published online: http://www. galaxykate. com/builda-generatorkcompton. pdf (Last checked: December 7, 2016).*

Conti, Edoardo, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. 2018. "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents." *Advances in neural information processing systems* 31.

Cortacero, Kévin, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, et al. 2023. "Evolutionary Design of Explainable Algorithms for Biomedical Image Segmentation." *Nature Communications,* no. 1, 7112.

Cully, Antoine, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015a. "Robots that can adapt like animals." *Nature* 521 (7553): 503–507.

Cully, Antoine, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015b. "Robots That Can Adapt like Animals." *Nature* 521, no. 7553 (7553): 503–507. ISSN: 1476-4687. https://doi.org/10.1038/nature14422.

Cully, Antoine, and Yiannis Demiris. 2017. "Quality and diversity optimization: A unifying modular framework." *IEEE Transactions on Evolutionary Computation* 22 (2): 245–259.

Cuomo, Salvatore, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. 2022. "Scientific machine learning through physics–informed neural networks: Where we are and what's next." *Journal of Scientific Computing* 92 (3): 88.

Davidson, MA, KD Splinter, and IL Turner. 2013. "A simple equilibrium model for predicting shoreline change." Publisher: Elsevier, *Coastal Engineering* 73:191–202.

Dawkins, Richard. 2016. *The selfish gene.* Oxford university press.

De Jong, Kenneth. 2017. "Evolutionary computation: a unified approach." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion,* 373–388.

De La Torre, Camilo, Kévin Cortacero, Sylvain Cussat-Blanc, and Dennis G Wilson. 2024. "Multimodal Adaptive Graph Evolution." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion.*

De La Torre, Camilo, Giorgia Nadizar, Yuri Lavinas, Hervé Luga, Dennis G Wilson, and Sylvain Cussat-Blanc. 2025. "Evolution of Inherently Interpretable Visual Control Policies." In *Under review.*

De La Torre, Camilo, Giorgia Nadizar, Yuri Lavinas, Robin Schwob, Camille Franchet, Hervé Luga, Dennis G Wilson, and Sylvain Cussat-Blanc. 2025. "Evolved and Transparent Pipelines for Biomedical Image Classification." In *European Conference on Genetic Programming.* Springer.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6 (2): 182–197.

Doi, Takeshi, Swadhin K Behera, and Toshio Yamagata. 2016. "Improved Seasonal Prediction Using the S INTEX-F2 Coupled Model." *Journal of Advances in Modeling Earth Systems* 8 (4): 1847–1867.

Doi, Takeshi, Swadhin K Behera, and Toshio Yamagata. 2019. "Merits of a 108-Member Ensemble System in ENSO and IOD Predictions." *Journal of Climate* 32 (3): 957–972.

Douka, Stella, Manon Verbockhaven, Théo Rudkiewicz, Stéphane Rivaud, François P Landes, Sylvain Chevallier, and Guillaume Charpiat. 2025. "Growth strategies for arbitrary DAG neural architectures." *arXiv preprint arXiv:2501.12690.*

Downing, Keith L. 2015. *Intelligence emerging: adaptivity and search in evolving neural systems.* MIT Press.

Earle, Sam, Julian Togelius, and Lisa B Soros. 2021. "Video games as a testbed for open-ended phenomena." In *2021 IEEE Conference on Games (CoG),* 1–9. IEEE.

Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter. 2019. "Neural architecture search: A survey." *Journal of Machine Learning Research* 20 (55): 1–21.

Eyring, Veronika, William D Collins, Pierre Gentine, Elizabeth A Barnes, Marcelo Barreiro, Tom Beucler, Marc Bocquet, Christopher S Bretherton, Hannah M Christensen, Katherine Dagon, et al. 2024. "Pushing the frontiers in climate modelling and analysis with machine learning." *Nature Climate Change,* 1–13.

Eyring, Veronika, Peter M Cox, Gregory M Flato, Peter J Gleckler, Gab Abramowitz, Peter Caldwell, William D Collins, Bettina K Gier, Alex D Hall, Forrest M Hoffman, et al. 2019. "Taking climate model evaluation to the next level." *Nature Climate Change* 9 (2): 102–110.

Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. "Diversity is All You Need: Learning Skills without a Reward Function." In *International Conference on Learning Representations.*

Ferro, Christopher AT, and David B Stephenson. 2011. "Extremal dependence indices: Improved verification measures for deterministic forecasts of rare binary events." *Weather and Forecasting* 26 (5): 699–713.

Fogel, LJ, AJ Owens, and MJ Walsh. 1965. "Intelligent decision-making through a simulation of evolution." *Simulation* 5 (4): 267–279.

Fontaine, Matthew, and Stefanos Nikolaidis. 2023. "Covariance matrix adaptation map-annealing." In *Proceedings of the Genetic and Evolutionary Computation Conference,* 456–465.

Fontaine, Matthew C, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. 2020. "Covariance matrix adaptation for the rapid illumination of behavior space." In *Proceedings of the 2020 genetic and evolutionary computation conference,* 94–102.

Freeman, C Daniel, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. 2021. "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation." *arXiv preprint arXiv:2106.13281.*

Funahashi, Ken-Ichi. 1989. "On the approximate realization of continuous mappings by neural networks." *Neural networks* 2 (3): 183–192.

Gould, Harvey, Jan Tobochnik, Dawn C Meredith, Steven E Koonin, Susan R McKay, and Wolfgang Christian. 1996. "An introduction to computer simulation methods: applications to physical systems." *Computers in Physics* 10 (4): 349–349.

Gruau, Frederic. 1994. "Automatic definition of modular neural networks." *Adaptive behavior* 3 (2): 151–183.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In *International conference on machine learning,* 1861–1870. PMLR.

Ham, Yoo-Geun, Jeong-Hwan Kim, and Jing-Jia Luo. 2019. "Deep learning for multi-year ENSO forecasts." *Nature* 573 (7775): 568–572.

Hansen, Nikolaus, and Andreas Ostermeier. 2001. "Completely derandomized self-adaptation in evolution strategies." *Evolutionary computation* 9 (2): 159–195.

Harari, Yuval Noah. 2014. *Sapiens: A brief history of humankind.* Random House.

Harding, Simon, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. 2012. "MT-CGP: Mixed Type Cartesian Genetic Programming." *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12,* 751.

Hein, Daniel, Steffen Udluft, and Thomas A Runkler. 2018. "Interpretable policies for reinforcement learning by genetic programming." *Engineering Applications of Artificial Intelligence* 76:158–169.

Hemberg, Erik, Stephen Moskal, and Una-May O'Reilly. 2024. "Evolving code with a large language model." *Genetic Programming and Evolvable Machines* 25 (2): 21.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan.

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated machine learning: methods, systems, challenges.* Springer Nature.

Ibaceta, Raimundo, Kristen D Splinter, Mitchell D Harley, and Ian L Turner. 2022. "Improving multi-decadal coastal shoreline change predictions by including model parameter non-stationarity." *Frontiers in Marine Science* 9:1012041.

Ibebuchi, Chibuike Chiedozie, and Michael B Richman. 2024. "Deep learning with autoencoders and LSTM for ENSO forecasting." *Climate Dynamics,* 1–15.

Jevons, W.S. 1865. *The Coal Question: An Enquiry Concerning the Progress of the Nation, and the Probable Exhaustion of Our Coal-mines.* Making Of The Modern World. Part 2. Macmillan. ISBN: 9781789876468. https://books.google.fr/books?id=gAAKAAAAIAAJ.

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. 2021. "Highly Accurate Protein Structure Prediction with AlphaFold." *nature* 596 (7873): 583–589.

Kaye, Jason P, and Miguel Quemada. 2017. "Using cover crops to mitigate and adapt to climate change. A review." *Agronomy for sustainable development* 37:1–17.

Kearns, Michael, and Satinder Singh. 2002. "Near-optimal reinforcement learning in polynomial time." *Machine learning* 49:209–232.

Kepler, Johannes, and William H Donahue. 1992. "New astronomy." *Cambridge [England]; New York.*

Khadka, Shauharda, and Kagan Tumer. 2018. "Evolution-guided policy gradient in reinforcement learning." *Advances in Neural Information Processing Systems* 31.

Koza, John R. 1994. "Genetic programming as a means for programming computers by natural selection." *Statistics and computing* 4:87–112.

Koza, John R, and James P Rice. 1992. "Automatic programming of robots using genetic programming." In *AAAI,* 92:194–207.

Kravitz, Ben, Ken Caldeira, Olivier Boucher, Alan Robock, Philip J Rasch, Kari Alterskjaer, Diana Bou Karam, Jason NS Cole, Charles L Curry, James M Haywood, et al. 2013. "Climate model response from the geoengineering model intercomparison project (GeoMIP)." *Journal of Geophysical Research: Atmospheres* 118 (15): 8320–8332.

Kravitz, Ben, Alan Robock, Olivier Boucher, Hauke Schmidt, Karl E Taylor, Georgiy Stenchikov, and Michael Schulz. 2011. "The geoengineering model intercomparison project (GeoMIP)." *Atmospheric Science Letters* 12 (2): 162–167.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25.

Krizhevsky, Alex, Ilya Sutskever, Geoffrey E Geoffret E Hinton, IIya Sulskever, and Geoffrey E Geoffret E Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information and Processing Systems (NIPS),* 1–9. ISSN: 10495258.

Lam, Remi, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. 2023. "Learning skillful medium-range global weather forecasting." *Science* 382 (6677): 1416–1421.

Lamont, Byron B, Tianhua He, and Zhaogui Yan. 2019. "Evolutionary history of fire-stimulated resprouting, flowering, seed release and germination." *Biological Reviews* 94 (3): 903–928.

Landajuela, Mikel, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. 2021. "Discovering symbolic policies with deep reinforcement learning." In *International Conference on Machine Learning,* 5979–5989. PMLR.

Le Tolguenec, Paul-Antoine, Emmanuel Rachelson, Yann Besse, Florent Teichteil-Koenigsbuch, Nicolas Schneider, Hélène Waeselynck, and Dennis Wilson. 2024. "Exploration-Driven Reinforcement Learning for Avionic System Fault Detection (Experience Paper)." In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis,* 920–931. ISSTA 2024. Vienna, Austria: Association for Computing Machinery. ISBN: 9798400706127. https://doi.org/10.1145/3650212.3680331.

Le Tolguenec, Paul-Antoine, Emmanuel Rachelson, Yann Besse, and Dennis G Wilson. 2022. "Curiosity Creates Diversity in Policy Search." *ACM Transactions on Evolutionary Learning and Optimization.*

Le Tolguenec, Paul-Antoine, Florent Teichteil-Koenigsbuch, Yann Besse, Dennis G Wilson, and Emmanuel Rachelson. 2024. "Exploration by Learning Diverse Skills through Successor State Measures." In *The Thirty-eighth Annual Conference on Neural Information Processing Systems.*

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep learning." *nature* 521 (7553): 436–444.

Lee, H., K. Calvin, D. Dasgupta, G. Krinner, A. Mukherji, P. Thorne, C. Trisos, et al. 2023. "Synthesis Report of the IPCC Sixth Assessment Report (AR6)."

Lee, Ritchie, Mykel J Kochenderfer, Ole J Mengshoel, Guillaume P Brat, and Michael P Owen. 2015. "Adaptive stress testing of airborne collision avoidance systems." In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC),* 6C2–1. IEEE.

Lehman, Joel, Jeff Clune, and Dusan Misevic. 2018. "The surprising creativity of digital evolution." In *Artificial Life Conference Proceedings,* 55–56. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...

Lehman, Joel, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J Bentley, Samuel Bernard, Guillaume Beslon, David M Bryson, et al. 2020. "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities." *Artificial life* 26 (2): 274–306.

Lehman, Joel, and Kenneth O Stanley. 2011a. "Abandoning objectives: Evolution through the search for novelty alone." *Evolutionary computation* 19 (2): 189–223.

Lehman, Joel, and Kenneth O Stanley. 2011b. "Evolving a diversity of virtual creatures through novelty search and local competition." In *Proceedings of the 13th annual conference on Genetic and evolutionary computation,* 211–218.

Li, Cai, Yong Zhang, Jianwen Li, Lesheng Kong, Haofu Hu, Hailin Pan, Luohao Xu, Yuan Deng, Qiye Li, Lijun Jin, et al. 2014. "Two Antarctic penguin genomes reveal insights into their evolutionary history and molecular changes related to the Antarctic environment." *GigaScience* 3:1–15.

Li, Yujia, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, et al. 2022. "Competition-Level Code Generation with AlphaCode." *Science* (December 9, 2022). https://doi.org/10.1126/science.abq1158.

Lipton, Zachary C. 2018. "The Mythos of Model Interpretability." *ACM Queue* 16 (3): 31–57. Accessed November 9, 2024. https://doi.org/10.1145/3236386.3241340.

Liu, Zimo, Jingya Wang, Shaogang Gong, Huchuan Lu, and Dacheng Tao. 2019. "Deep reinforcement active learning for human-in-the-loop person re-identification." In *Proceedings of the IEEE/CVF international conference on computer vision,* 6122–6131.

Lorenz, Edward N. 1982. "Atmospheric predictability experiments with a large numerical model." *Tellus* 34 (6): 505–513.

Lu, Yulong, and Jianfeng Lu. 2020. "A universal approximation theorem of deep neural networks for expressing probability distributions." *Advances in neural information processing systems* 33:3094–3105.

Luccioni, Sasha, Yacine Jernite, and Emma Strubell. 2024. "Power hungry processing: Watts driving the cost of AI deployment?" In *The 2024 ACM Conference on Fairness, Accountability, and Transparency,* 85–99.

Ma, Yecheng Jason, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. "Eureka: Human-Level Reward Design via Coding Large Language Models." In *The Twelfth International Conference on Learning Representations.* https://openreview.net/forum?id=IEduRUO55F.

Maile, Kaitlin. 2023. "Dynamic architectural optimization of artificial neural networks." 2023TOUC0008. PhD diss. http://www.theses.fr/2023TOUC0008/document.

Maile, Kaitlin, Emmanuel Rachelson, Hervé Luga, and Dennis G Wilson. 2022. "When, Where, and How to Add New Neurons to ANNs." In *International Conference on Automated Machine Learning,* 18–1. PMLR.

Maile, Kaitlin, Dennis G Wilson, and Patrick Forré. 2023. "Equivariance-Aware Architectural Optimization of Neural Networks." In *The Eleventh International Conference on Learning Representations.*

McCulloch, Malcolm T, Amos Winter, Clark E Sherman, and Julie A Trotter. 2024. "300 years of sclerosponge thermometry shows global warming has exceeded 1.5 C." *Nature Climate Change* 14 (2): 171–177.

Medvet, Eric, and Giorgia Nadizar. 2023. "GP for Continuous Control: Teacher or Learner? The Case of Simulated Modular Soft Robots." *Genetic Programming Theory and Practice XX.*

Meng, Kevin, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. "Locating and editing factual associations in gpt." *Advances in neural information processing systems* 35:17359–17372.

Merchant, Amil, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. 2023. "Scaling deep learning for materials discovery." *Nature* 624 (7990): 80–85.

Meyerson, Elliot, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. 2024. "Language model crossover: Variation through few-shot prompting."

Miller, Julian F. 2011. "Cartesian genetic programming." In *Cartesian Genetic Programming,* 17–34. Springer.

Miller, Julian F, and Dennis G Wilson. 2017. "A Developmental Artificial Neural Network Model for Solving Multiple Problems." In *Proceedings of the Genetic and Evolutionary Computation Conference Companion,* 69–70. ACM.

Miller, Julian Francis. 2020. "Cartesian genetic programming: its status and future." *Genetic Programming and Evolvable Machines* 21 (1): 129–168.

Mitchell, Melanie. 1998. *An introduction to genetic algorithms.* MIT press.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. 2015. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (February): 529–533. ISSN: 0028-0836.

Montaño, Jennifer, Giovanni Coco, Jose AA Antolínez, Tomas Beuzen, Karin R Bryan, Laura Cagigal, Bruno Castelle, Mark A Davidson, Evan B Goldstein, Raimundo Ibaceta, et al. 2020. "Blind testing of shoreline evolution models." *Scientific reports* 10 (1): 1–10.

Monteleoni, Claire, Gavin A Schmidt, Shailesh Saroha, and Eva Asplund. 2011. "Tracking climate models." *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4 (4): 372–392.

Mooney, Christopher Z. 1997. *Monte carlo simulation.* 116. Sage.

Mouret, Jean-Baptiste, and Jeff Clune. 2015. "Illuminating search spaces by mapping elites." *arXiv preprint arXiv:1504.04909.*

Nadizar, Giorgia, Eric Medvet, and Dennis G Wilson. 2024a. "Naturally Interpretable Control Policies via Graph-Based Genetic Programming." In *European Conference on Genetic Programming (Part of EvoStar),* 73–89. Springer.

Nadizar, Giorgia, Eric Medvet, and Dennis G Wilson. 2024b. "Searching for a Diversity of Interpretable Graph Control Policies." In *Proceedings of the Genetic and Evolutionary Computation Conference.*

Nearing, Grey, Deborah Cohen, Vusumuzi Dube, Martin Gauch, Oren Gilon, Shaun Harrigan, Avinatan Hassidim, Daniel Klotz, Frederik Kratzert, Asher Metzger, et al. 2024. "Global prediction of extreme floods in ungauged watersheds." *Nature* 627 (8004): 559–563.

Nilsson, Olle, and Antoine Cully. 2021. "Policy gradient assisted map-elites." In *Proceedings of the Genetic and Evolutionary Computation Conference,* 866–875.

North, Gerald R. 1975. "Theory of energy-balance climate models." *J. Atmos. Sci* 32 (11): 2033–2043.

Norton, John D. 2012. "Chasing the light: Einstein's most famous thought experiment." In *Thought experiments in science, philosophy, and the arts,* 123–140. Routledge.

Oh, Ji-Hoon, Jong-Seong Kug, Soon-Il An, Fei-Fei Jin, Michael J McPhaden, and Jongsoo Shin. 2024. "Emergent climate change patterns originating from deep ocean warming in climate mitigation scenarios." *Nature Climate Change* 14 (3): 260–266.

Ortega, Juan, Noor Shaker, Julian Togelius, and Georgios N Yannakakis. 2013. "Imitating human playing styles in super mario bros." *Entertainment Computing* 4 (2): 93–104.

Paolo, Giuseppe, Alexandre Coninx, Stéphane Doncieux, and Alban Laflaquière. 2021. "Sparse Reward Exploration via Novelty Search and Emitters." In *Proceedings of the Genetic and Evolutionary Computation Conference,* 154–162.

Parnas, David Lorge. 1972. "On the criteria to be used in decomposing systems into modules." *Communications of the ACM* 15 (12): 1053–1058.

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. "Curiosity-driven exploration by self-supervised prediction." In *International conference on machine learning,* 2778–2787. PMLR.

Petsiuk, Vitali, Abir Das, and Kate Saenko. 2018. "RISE: Randomized Input Sampling for Explanation of Black-box Models." In *Proceedings of the British Machine Vision Conference (BMVC).*

Power, Scott, François Delage, Christine Chung, Greg Kociuba, and Kevin Keay. 2013. "Robust twenty-first-century projections of El Niño and related precipitation variability." *Nature* 502 (7472): 541–545.

Pugh, Justin K, Lisa B Soros, and Kenneth O Stanley. 2016. "Quality diversity: A new frontier for evolutionary computation." *Frontiers in Robotics and AI* 3:202845.

Rechenberg, Ingo. 1978. "Evolutionsstrategien." In *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977,* 83–114. Springer.

Reil, Marlena, Gunnar Spreen, Marcus Huntemann, Lena Buth, and Dennis G Wilson. 2024. "Machine Learning for the Detection of Arctic Melt Ponds from Infrared Imagery." In *Tackling Climate Change with Machine Learning, ICLR 2024.*

Reimann, Lena, Athanasios T Vafeidis, and Lars E Honsel. 2023. "Population development as a driver of coastal risk: current trends and future pathways." *Cambridge Prisms: Coastal Futures* 1:e14.

Richardson, Lewis F, and Oliver M Ashford. 1993. *The Collected Papers of Lewis Fry Richardson: Volume 1.* Vol. 1. CUP Archive.

Riu, Guillaume, Mahmoud Al Najar, Gregoire Thoumyre, Rafael Almar, and Dennis G Wilson. 2023. "Global Coastline Evolution Forecasting from Satellite Imagery Using Deep Learning." In *NeurIPS 2023 Workshop on Tackling Climate Change with Machine Learning.*

Romera-Paredes, Bernardino, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. "Mathematical discoveries from program search with large language models." *Nature* 625 (7995): 468–475.

Ros, Raymond, and Nikolaus Hansen. 2008. "A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity." In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature—PPSN X-Volume 5199,* 296–305.

Rudin, Cynthia. 2019. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead." *Nature machine intelligence* 1 (5): 206–215.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1986. "Learning representations by back-propagating errors." *nature* 323 (6088): 533–536.

Sagan, Carl. 1981. *Cosmos.* Random House.

Samsi, Siddharth, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. "From words to watts: Benchmarking the energy costs of large language model inference." In *2023 IEEE High Performance Extreme Computing Conference (HPEC),* 1–9. IEEE.

Schmidhuber, Jürgen. 1991. "Curious model-building control systems." In *Neural Networks, 1991. 1991 IEEE International Joint Conference on,* 1458–1463. IEEE.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347.*

Seirawan, Yasser, Herbert A Simon, and Toshinori Munakata. 1997. "The implications of kasparov vs. deep blue." *Communications of the ACM* 40 (8): 21–25.

Sengupta, Ushnish, Matt Amos, Scott Hosking, Carl Edward Rasmussen, Matthew Juniper, and Paul Young. 2020. "Ensembling geophysical models with Bayesian neural networks." *Advances in Neural Information Processing Systems* 33:1205–1217.

Sharp, Phil, and Susan Hockfield. 2017. "Convergence: the future of health." *Science* 355 (6325): 589–589.

Sigaud, Olivier. 2023. "Combining evolution and deep reinforcement learning for policy search: A survey." *ACM Transactions on Evolutionary Learning* 3 (3): 1–20.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. 2016. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529 (7587): 484–489. ISSN: 0028-0836.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. "Mastering the game of go without human knowledge." *nature* 550 (7676): 354–359.

Singla, Sahil, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. 2021. "Understanding failures of deep networks via robust feature extraction." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 12853–12862.

Soros, LB, Nick Cheney, and Kenneth O Stanley. 2016. "How the strictness of the minimal criterion impacts open-ended evolution." In *Artificial Life Conference Proceedings,* 208–215. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...

Splinter, Kristen D, Ian L Turner, Mark A Davidson, Patrick Barnard, Bruno Castelle, and Joan Oltman-Shay. 2014. "A generalized equilibrium model for predicting daily to interannual shoreline response." Publisher: Wiley Online Library, *Journal of Geophysical Research: Earth Surface* 119 (9): 1936–1958.

Sterling, Peter, and Simon Laughlin. 2015. *Principles of neural design.* MIT Press.

Sudhakaran, Shyam, Miguel González-Duque, Matthias Freiberger, Claire Glanois, Elias Najarro, and Sebastian Risi. 2024. "Mariogpt: Open-ended text2level generation through large language models." *Advances in Neural Information Processing Systems* 36.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

Taylor, Tim, Mark Bedau, Alastair Channon, David Ackley, Wolfgang Banzhaf, Guillaume Beslon, Emily Dolson, Tom Froese, Simon Hickinbotham, Takashi Ikegami, et al. 2016. "Open-ended evolution: Perspectives from the OEE workshop in York." *Artificial life* 22 (3): 408–423.

Temple, Kenneth L, and Arthur R Colmer. 1951. "The autotrophic oxidation of iron by a new bacterium: Thiobacillus ferrooxidans." *Journal of bacteriology* 62 (5): 605–611.

Templier, Paul, Luca Grillotti, Emmanuel Rachelson, Dennis G Wilson, and Antoine Cully. 2024. "Quality with Just Enough Diversity in Evolutionary Policy Search." In *Proceedings of the Genetic and Evolutionary Computation Conference.*

Templier, Paul, Emmanuel Rachelson, Antoine Cully, and Dennis G Wilson. 2024. "Genetic Drift Regularization: on preventing Actor Injection from breaking Evolution Strategies." In *IEEE Congress on Evolutionary Computation.*

Timmermann, Axel, Soon-Il An, Jong-Seong Kug, Fei-Fei Jin, Wenju Cai, Antonietta Capotondi, Kim M Cobb, Matthieu Lengaigne, Michael J McPhaden, Malte F Stuecker, et al. 2018. "El Niño–southern oscillation complexity." *Nature* 559 (7715): 535–545.

Tjoa, Erico, and Cuntai Guan. 2020. "A survey on explainable artificial intelligence (xai): Toward medical xai." *IEEE transactions on neural networks and learning systems* 32 (11): 4793–4813.

Trenberth, Kevin E, and David P Stepaniak. 2001. "Indices of el Niño evolution." *Journal of climate* 14 (8): 1697–1701.

Turing, Alan M. 1950. "Computing machinery and intelligence." *Mind* 59 (236): 433–460.

Valizadegan, Hamed, Miguel JS Martinho, Laurent S Wilkens, Jon M Jenkins, Jeffrey C Smith, Douglas A Caldwell, Joseph D Twicken, Pedro CL Gerum, Nikash Walia, Kaylie Hausknecht, et al. 2022. "ExoMiner: A highly accurate and explainable deep learning classifier that validates 301 new exoplanets." *The Astrophysical Journal* 926 (2): 120.

Veeling, Bastiaan S., Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. 2018. "Rotation Equivariant CNNs for Digital Pathology." In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018,* edited by Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, 210–218. Springer International Publishing. https://doi.org/10.1007/978-3-030-00934-2_24.

Verma, Abhinav, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. "Programmatically interpretable reinforcement learning." In *International Conference on Machine Learning,* 5045–5054. PMLR.

Virgolin, Marco, Andrea De Lorenzo, Eric Medvet, and Francesca Randone. 2020. "Learning a formula of interpretability to learn interpretable formulas." In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II 16,* 79–93. Springer.

Visioni, Daniele, Douglas G MacMartin, Ben Kravitz, Olivier Boucher, Andy Jones, Thibaut Lurton, Michou Martine, Michael J Mills, Pierre Nabat, Ulrike Niemeier, et al. 2021. "Identifying the sources of uncertainty in climate model simulations of solar radiation modification with the G6sulfur and G6solar Geoengineering Model Intercomparison Project (GeoMIP) simulations." *Atmospheric Chemistry and Physics* 21 (13): 10039–10063.

Wang, Hanchen, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. 2023. "Scientific discovery in the age of artificial intelligence." *Nature* 620 (7972): 47–60.

Williams, Christopher, and Carl Rasmussen. 1995. "Gaussian processes for regression." *Advances in neural information processing systems* 8.

Wilson, Dennis G, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. "Evolving Simple Programs for Playing Atari Games." In *Proceedings of the Genetic and Evolutionary Computation Conference,* 229–236. ACM.

Yeh, Sang-Wook, Jong-Seong Kug, Boris Dewitte, Min-Ho Kwon, Ben P Kirtman, and Fei-Fei Jin. 2009. "El Niño in a changing climate." *Nature* 461 (7263): 511–514.

Yuksekgonul, Mert, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. "Optimizing generative AI by backpropagating language model feedback." *Nature* 639 (8055): 609–616.

Zeiler, Matthew D., and Rob Fergus. 2014. "Visualizing and Understanding Convolutional Networks." In *Computer Vision ECCV 2014,* 818–833. Springer International Publishing.

Zhang, Linfeng, Chenglong Bao, and Kaisheng Ma. 2021. "Self-distillation: Towards efficient and compact neural networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (8): 4388–4403.

Zheng, Zhihao, and Pengyu Hong. 2018. "Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks." *Advances in neural information processing systems* 31.

Zhou, Ryan, and Ting Hu. 2023. "Evolutionary approaches to explainable machine learning," 487–506.