|  |
| --- |
| Master’s in Science |
|  |
| Equilibrium Point Learning |
| Dowoo Baik |
|  |
| Department of Cybersecurity |
|  |
| School of Cybersecurity |
|  |
| Korea University |
|  |
| November 2023 |

Equilibrium Point Learning

by Dowoo Baik

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

under the supervision of Professor Ji Won Yoon

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

|  |
| --- |
| Department of Cybersecurity |
|  |
| School of Cybersecurity |
|  |
| Korea University |
| . |
| November 2023 |

The thesis of Dowoo Baik has been approved

by the thesis committee in partial fulfillment of the requirements for the degree of Master of Science.

November 2023

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Committee Chair: Ji Won Yoon |
|  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Committee Member: Seong Jun Oh |
|  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Committee Member: Sang Kyun Lee |

Equilibrium Point Learning

by Dowoo Baik

Department of Cybersecurity

under the supervision of Professor Ji Won Yoon

**ABSTRACT**

We present a novel approach, Equilibrium Point Learning (EPL), for training the deep equilibrium model (DEQ). In this method, the equilibrium point of the DEQ serves as the learnable parameters. Notably, the DEQ parameters encapsulate the learning algorithm itself and remain fixed. Consequently, by exploring the parameter space, we can discover a more efficient learning algorithm without relying on conventional techniques such as backpropagation or Q-learning. In this paper, we adopt an evolutionary approach inspired by biological neurons to evolve the DEQ model parameters. Initially, we examine the physical dynamics of neurons at the molecular level and translate them into a dynamical system representation. Subsequently, we formulate a deep implicit layer that is mathematically proven to possess an equilibrium point. The energy function of the implicit layer is defined using a quadratic form augmented with entropy and momentum terms. Given the resemblance between the dynamics of the deep implicit layer and the principles of physics and chemistry, it can effectively capture the biomodel of systems biology and the neural model of spiking neural networks (SNNs). This equivalence enables us to define the implicit layer of the DEQ, allowing for seamless integration with existing artificial neural networks (ANNs). Finally, we employ HyperNEAT to evolve the parameters of the dynamical system. Through our experiments, we observe a consistent improvement in learning efficiency, with each successive generation exhibiting a 0.2% increase in learning speed per generation.

**Keywords:** Deep equilibrium model, Learning Algorithm, Biomodel, HyperNEAT

평형점 학습

백 도 우

정 보 보 호 학 과

지도교수: 윤 지 원

**국문 초록**

우리는 DEQ(Deep Equilibrium Model) 훈련을 위한 새로운 접근법인 평형점 학습(Equilibrium Point Learning, EPL)을 제시한다. 이 방법에서는 DEQ의 평형점이 학습 가능한 매개변수로 사용된다. 눈에 띄게도 DEQ 매개변수는 학습 알고리즘 자체를 포함하며 고정된다. 따라서 매개변수 공간을 탐색함으로써 역전파나 Q-학습과 같은 전통적인 기법에 의존하지 않고 더 효율적인 학습 알고리즘을 발견할 수 있다. 본 논문에서는 생물학적 뉴런에서 영감을 받은 진화적 접근법을 사용하여 DEQ 모델 매개변수를 진화시킨다. 우선, 분자 수준에서 뉴런의 물리적 동역학을 검토하고 동역학 시스템 표현으로 변환한다. 그 후, 수학적으로 평형점을 갖는 것이 증명된 깊은 암시적 계층을 구성한다. 암시적 계층의 에너지 함수는 엔트로피와 모멘텀 항으로 확장된 2차 형태로 정의된다. 깊은 암시적 계층의 동역학과 물리학 및 화학의 원리 사이의 유사성을 감안할 때, 이는 시스템 생물학의 바이오모델과 스파이킹 뉴럴 네트워크(SNNs)의 뉴럴 모델을 효과적으로 포착할 수 있다. 이 등가성은 DEQ의 암시적 계층을 정의하게 해, 기존 인공 뉴럴 네트워크(ANNs)와의 원활한 통합을 가능하게 한다. 마지막으로, 우리는 동역학 시스템의 매개변수를 진화시키기 위해 HyperNEAT를 사용한다. 우리의 실험을 통해, 각 세대마다 학습 속도가 세대당 0.2%씩 지속적으로 개선되는 것을 관찰했다.

**중심어:** 깊은 평형 모델, 학습 알고리즘, 바이오모델, HyperNEAT

**PREFACE**

This thesis embodies the culmination of years of academic exploration and rigorous research. The central research presented within these pages was also submitted to the Asian Conference on Machine Learning (ACML) and was received with positive recognition from experts in the field. The decision to incorporate this research into my graduate thesis was made with the belief that the insights and findings here not only contribute to the academic community but also serve as a testament to my academic journey.

I'd like to express my heartfelt gratitude to my advisor, professors, and colleagues who provided invaluable feedback and guidance throughout the research process. The journey was not without its challenges, but the continued support and mentorship I received made it all worthwhile.

As you delve into this thesis, I hope you find the research enlightening and the discussions thought-provoking. This work is a testament to my passion for machine learning, my commitment to pushing the boundaries of knowledge, and my gratitude to all those who walked this academic path with me.

**ACKNOWLEDMENTS**

Ji Won Yoon was supported by the MSIT(Ministey of science and ICT), Korea under the ITRC (Information Technology Research Center) support program (IITP-2023-RS-2022-00164800) supervised by the IITP (Institute for Information & Communications Technology Planning \& Evaluation)

**TABLE OF CONTENTS**

**ABSTRACT** i

**국문 초록** ii

**PREFACE** iii

**ACKNOWLEDMENTS** iii

**TABLE OF CONTENTS** iv

**LIST OF FIGURES** iv

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. RELATED WORK 1

CHAPTER 3. PROPOSED EPL METHOD 2

3.2 Overview of the EPL Learning 3

3.2 EPL Learning Process 3

3.3 Find Back-Propagation Algorithm Parameter of EPL Learning 4

3.4 Property of EPL Learning 5

3.5 Design Implict Layer 5

CHAPTER 4. EXPERIMENT 8

CHAPTER 5. RESULT 9

CHAPTER 6. CONCLUSION 11

REFERENCES 11

**LIST OF FIGURES**

[Figure 1 Concept of EPL Learning 3](#_Toc118204031)

[Figure 2 Generation Dynamical System by HyperNeat and Example 8](#_Toc118204031)

[Figure 3 Training results for two environments 9](#_Toc118204031)

[Figure 4 Examples of Dynamical systems 10](#_Toc118204031)

[Figure 5 Best Fitness Examples from the 1st, 10th, and 50th Generations 10](#_Toc118204031)

# Introduction

To describe biological neurons mathematically, there have been several academic attempts, including spiking neural networks, computational biology, and systems biology [Pfeiffer and](#_bookmark33) [Pfeil](#_bookmark33) ([2018](#_bookmark33)), [Buesing et al.](#_bookmark18) ([2011](#_bookmark18)), [Bick et al.](#_bookmark17) ([2020](#_bookmark17)). These endeavors aim to identify the function and role of neuronal components, such as gene-protein-reaction (GPR) rules [Di Filippo et al.](#_bookmark24) ([2021](#_bookmark24)). The ultimate goal of these efforts is to reach human intelligence and artificial general intelligence (AGI). In the field of systems biology, biological systems are mathematically analyzed and modeled as dynamical systems, often referred to as biomodels [Chelliah et al.](#_bookmark20) ([2013](#_bookmark20)). These biomodels can be simulated over time using ordinary differential equation (ODE) solvers, and well-crafted models exhibit similarities or comparable values to experimental observations [Hernjak et al.](#_bookmark27) ([2005](#_bookmark27)). However, due to the complexity and vastness of biological systems, many biological functions remain unexplained. In the context of spiking neural networks (SNNs), the problem of nonlinear learning is addressed using

backpropagation learning method, although it has not yet been fully biologically identified [Tavanaei and Maida](#_bookmark35) ([2019](#_bookmark35)).

From the perspective of artificial neural networks (ANNs), we hypothesized that biomodels share similarities with DEQ models. Additionally, simulating a biomodel is akin to the approach taken in neural ordinary differential equations (neural ODEs) [Chen et al.](#_bookmark21) ([2018](#_bookmark21)), which involves computing through an ODE solver in the context of weight-tied feedforward networks with infinitely many layers. However, learning in neurons through this method occurs without the need for separate learning algorithms such as backpropagation or Q- learning. In other words, the calculation of a deep implicit layer as an ODE solver constitutes the entire learning process. This is a capability that has not yet been achieved in ANN architectures. In this paper, we introduce a novel approach to learning the DEQ model, which we refer to as Equilibrium Point Learning (EPL). In EPL, the parameters of the model serve as the learning algorithms themselves, analogous to backpropagation in traditional approaches. We draw parallels to the learning process in biological neurons, where the search for optimal parameters can be seen as a favorable case. EPL encompasses elements of both supervised and unsupervised learning, and it distinguishes between the time for behavior determination and the time for feedback acceptance. This learning method aligns with the principles outlined in studies on neural learning and decision-making [Glimcher](#_bookmark26) ([2011](#_bookmark26)). By leveraging EPL, our objective is to explore the parameter space of the DEQ model and identify the most effective learning algorithm.

Biological neurons have previously explored the parameter space using genetic algorithms, aiming to identify dynamical systems that are conducive to achieving survival goals within their physical environment. It is this sequence of evolutionary processes that serves as our primary focus. In light of this, we introduce a dynamical system that can capture certain aspects of neuronal behavior, such as the Hodgkin-Huxley (HH) model in spiking neural networks (SNNs). Our approach centers around biological and chemical phenomena, seeking to depict the physical system and energy dynamics from a molecular perspective within the context of a dynamical system. By adopting this perspective, we can effectively represent the neuron models found in systems biology. Additionally, we reinterpret the dynamical system as a means of solving a convex optimization problem. Through this reinterpretation, we devise a deep implicit layer that is mathematically proven to possess an equilibrium point.

We further developed the deep implicit layer using a Neat algorithm, which is a type of artificial neural network evolution method. Specifically, we employed the HyperNeat algorithm, which is well-suited for dynamic systems that necessitate the creation of large-scale networks and the repetition of network patterns.

# Related Work

**Deep Equilibrium model** The DEQ model [Bai et al.](#_bookmark16) ([2019](#_bookmark16)) focuses on a weight-tied deep implicit layer, which involves finding the equilibrium point through an infinite layer via root finding. In contrast to conventional deep implicit layers, the DEQ model enables analytical backpropagation through the equilibrium point using implicit differentiation. This approach requires only a single layer of information, resulting in lower memory usage. In

this paper, we adopt the same architecture as the DEQ model and compare our learning methods with it.

**Training of Spiking Neural Network** The Spiking Neural Network (SNN) is a neural network model that draws inspiration from the spiking behavior observed in biological neurons during information processing. SNN encompasses various neuron models, including the Hodgkin-Huxley (HH) model [Cronin and Boutelle](#_bookmark22) ([1987](#_bookmark22)), which closely resembles computational biology by simulating sodium and potassium channels, and the Izhikevich model [Izhikevich](#_bookmark30) ([2003](#_bookmark30)), which exhibits similar behavior with lower computational complexity. In this paper, our focus lies in designing an implicit layer that can accommodate these dynamical system models. SNN incorporates parallel and biologically plausible learning rules such as Spike-Timing-Dependent Plasticity (STDP) and the Bienenstock-Cooper- Munro (BCM) theory, along with extensive training methods like backpropagation. In a previous study [Huang and Huang](#_bookmark29) ([2016](#_bookmark29)), researchers evolved the best-fit model constants by controlling the neuron model constants. Neurons are known to learn through Hebbian learning, an unsupervised learning paradigm exemplified by STDP [Caporale and Dan](#_bookmark19) ([2008](#_bookmark19)). More recently, a method was proposed in [Xiao et al.](#_bookmark38) ([2021](#_bookmark38)) that applies implicit differentiation of DEQ to backpropagation in the equilibrium state. This suggests a way to incorporate DEQ’s implicit differentiation into the training process.

**Training of Biological Neural Network** In the study by [Glimcher](#_bookmark26) ([2011](#_bookmark26)), it is demon- strated that nerves are activated by both an activation signal and a feedback signal mediated by dopamine. As the learning progresses, the feedback signal gradually diminishes due to the ability to predict the reward signal.

[Taylor and Ivry](#_bookmark36) ([2014](#_bookmark36)) present research on reinforcement learning algorithms in the cerebellum. If there exists a dynamical system that is intricately represented by biomodels such as purkinje cells, granule cells, and star cells in the cerebellum, this suggests that the dynamical system may implicitly incorporate a Q-learning algorithm.

**HyperNeat** NEAT [Stanley and Miikkulainen](#_bookmark34) ([2002](#_bookmark34)) is a genetic algorithm that evolves artificial neural networks (ANNs). It aims to strike a balance between fitness and diversity, allowing for the evolution of species within the network. HyperNEAT (Stanley et al., 2009) utilizes a method of network creation known as a Connection Pattern Producing Network (CPPN), which is evolved by NEAT. HyperNEAT employs an indirect encoding approach that leverages the geometric properties of the network. This enables the design of regular patterns within the network structure.

# Proposed EPL Method

In this section, we first provide an overview of the Equilibrium Point Learning (EPL) approach and introduce its core concepts. We then provide a detailed explanation of the EPL method in comparison with the learning methodology of DEQ. Additionally, we briefly discuss the training of model parameters in EPL learning and explain how to align it with the back-propagation algorithm, which is well known as a learning method in ANN. Next, we introduce the properties of the implicit layer required in EPL learning. Finally, we introduce how to design an implicit layer that convergence (a prerequisite for EPL learning) of the model is mathematically proven. It can also accommodate the biomodel in system biology

## Overview of the EPL Learning

Equilibrium Point Learning (EPL) works as schematically shown in Fig. [1](#_bookmark0). The key is to ex- pect all learning to take place while overwriting ***z*** to ***z***∗. In the context of traditional artificial neural networks (ANN), The role of the learning parameter shifts from θ to z, and the model’s output inherently includes the parameter learning amount, with this role attributed to the fixed parameter θ. Therefore, exploring the parameter space of θ implies searching for a more efficient learning algorithm.

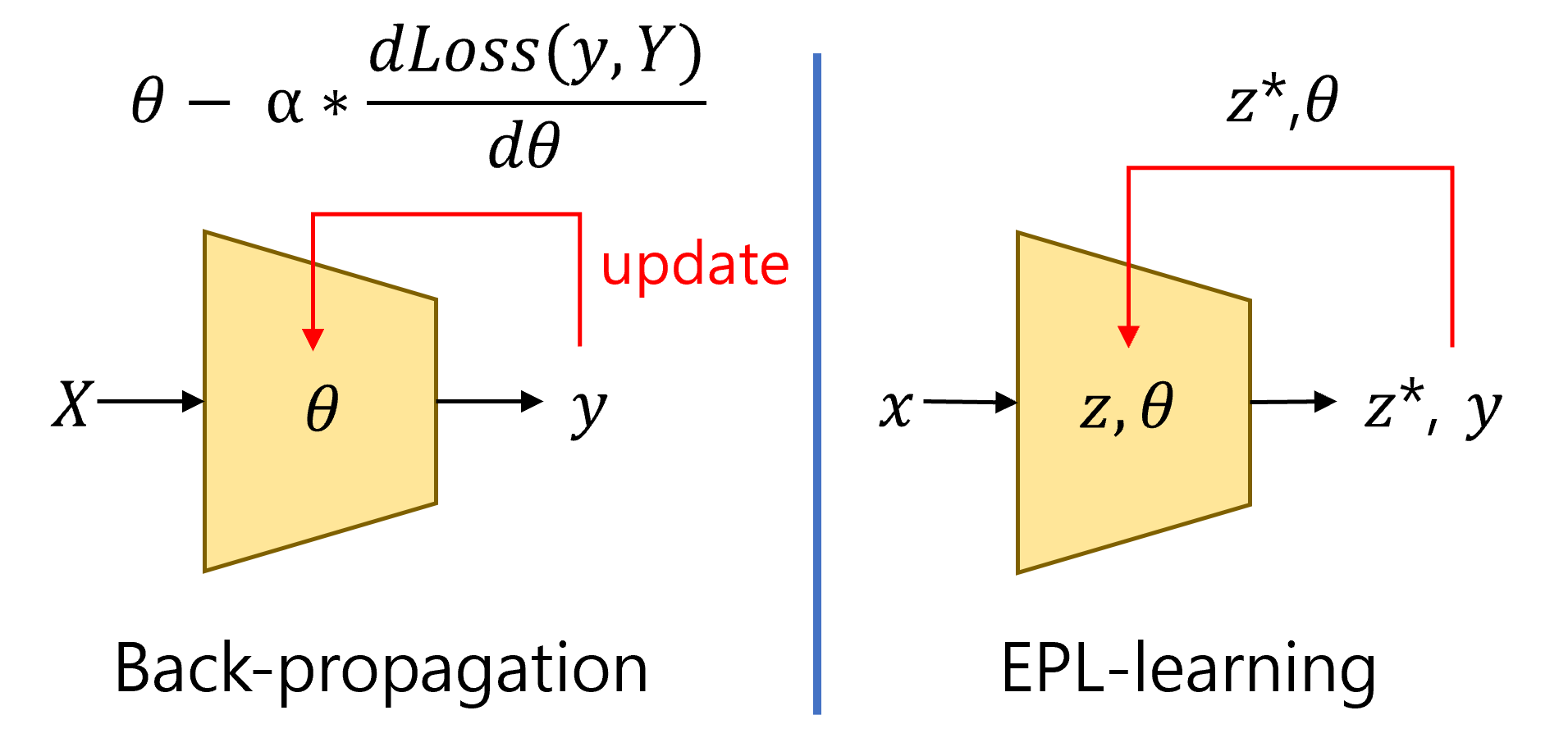


Figure 1: Concept of EPL Learning

## EPL Learning Process

**Algorithm 1** Compare Learning Algorithm from DEQ

1: **procedure** Train DEQ f*θ*(***z***, ***x***)

2: **while** epoch **do**

3: ***z****post* ***x***

←

4: **while *z****post* ***z****pre* > ε **do**

| − |

5: ***z****pre* ***z****post*

←

6: ***z****post* = f*θ* (***z****pre*, ***x***)

7: **end while**

8: ***z***∗ ***z****post*

←

9: compute ℓ(***z***∗, ***y***)

10: θ ← θ − α∇ℓ

11: **end while**

13: **procedure** Train EPL f*θ*(***x***)

14: ***z****post* θ[***x***0]

←

15: **while *z****post* ***z****pre* > ε **do**

| − |

16: ***z****pre* ***z****post*

←

17: ***z****post* = f*θ* (***z****pre*)

18: **end while**

19: ***z***∗ ← ***z****post* ▷ ***z****i*∗*nit*

20: **while** epoch **do**

21: ***z****post* ← ***z***∗ + ***I****z* · ***x***

22: ***y***^ = emptylist

| − |

23: **while *z****post* ***z****pre* < ε **do**

*post*

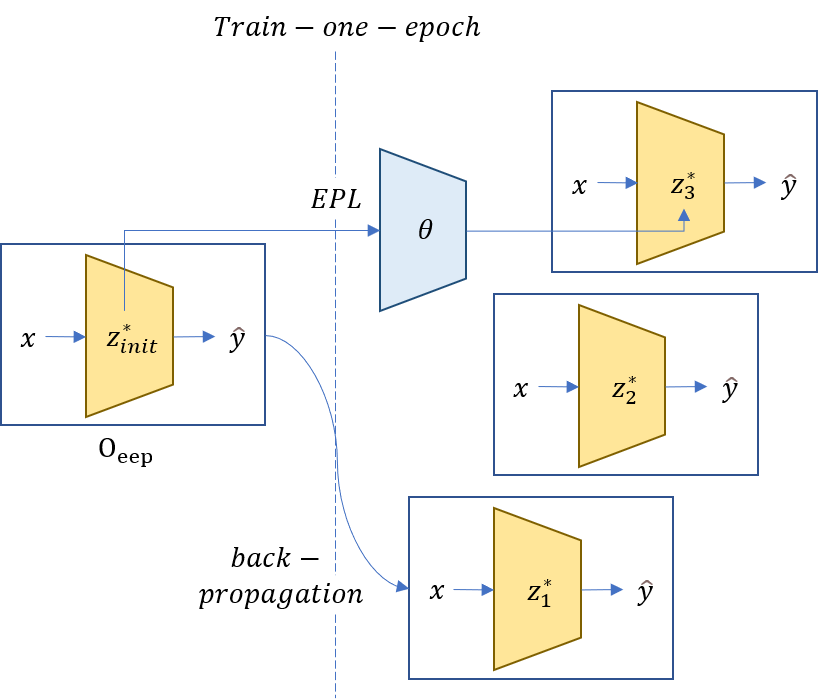
12: **end procedure**

24:

***y***^.append(O*z* · ***z*** )

←

25: ***z****pre* ***z****post*

26: ***z****post* = f*θ* (***z****pre*)

27: **end while**

28: ***z***∗ ← ***z****post* ▷ ***z****s*∗*elf* −*learn* 29: ℓ ℓ(Active(***y***), ***y***) ▷ ℓ(O*eep*, ***y***) 30: ***z****post* ***z***∗ + ***F****z* ℓ

^

← ·

←

31: **while *z****post* ***z****pre* > ε **do**

| − |

32: ***z****pre* ***z****post*

←

33: ***z****post* = f*θ* (***z****pre*)

34: **end while**

35: ***z***∗ ← ***z****post* ▷ ***z****s*∗*upervised learn*

−

36: **end while**

37: **end procedure**

In the learning process of the DEQ, equilibrium point ***z***∗ ∈ R*t*×1 is the output of input value ***x*** ∈ R*t*×1. Then, calculate the loss of ***z***∗ and ***y*** ∈ R*t*×1, and update the parameter of the model that in the direction of the loss decrease. In comparison, EPL training presented in this paper, at first, find the first equilibrium point ***z****i*∗*nit* ∈ R*n*×1 from the given initial value ***x***0 ∈ R*n*×1 in model parameter, like Eq. ([9](#_bookmark9)). This becomes the first pre-learned parameter. Next, using the input constant matrix ***I****z* ∈ R*n*×*t*, n > t, the vector ***z***∗ + ***I****z* · ***x*** is the next input for the layer. This is a key difference between DEQ and EPL methods. When ***z*** reaches the new equilibrium point ***z****s*∗*elf* −*learn*, it is the self-learning process of the equilibrium point. Until equilibrium point ***z****s*∗*elf* −*learn* is reached, we collect ***O****z* · ***z****post*, that the inner value of output constant matrix ***O****z* ∈ R*r*×*n* and ***z***. Then, output becomes the value taken by its activation function (like the average or the number of times the threshold has been

exceeded). We name the output for these inputs as *Output of Exploration Equilibrium Point*,

that is, O*eep*. In addition, the function that represents the newly reached equilibrium point is named *Function of New Equilibrium Point*, f*nep* (Eq. ([1](#_bookmark1))).

∀***z***, ***z***0 ← ***z***, ***z****i*+1 = f*θ*(***z****i*)

find ∗ ∈ N s.t |f*θ*(***z***∗) − ***z***∗| ≤ ϵ,

Let O*eep*(***z***) = Active((***O****z* · ***z****i*)[0:∗]), f*nep*(***z***) = ***z***∗

(1)

These output values can be an action in the environment and obtain a corresponding fitness score, or a feedback (loss) value ℓ = ℓ(O*eep*, ***y***) ∈ R1×1 can be obtained by calculating the distance from the expected output value ***y***. Then, use the feedback constant F*z* ∈ R*n*×1, so that vector ***z***∗ + F*z* · ℓ, which is a linear combination of the equilibrium point ***z****s*∗*elf learn* and the feedback, is the next input of the layer. ***z*** reaches another new equilibrium point ***z****s*∗*upervised learn* , which becomes the supervision-learning process of the equilibrium point. One learning ends when a new equilibrium point is reached.

−

−

## Find Back-Propagation Algorithm Parameter of EPL Learning

If EPL learning occurs (if the loss value is reduced), this means that Eq. ([2](#_bookmark2)). must be satisfied.

ℓ(O*eep*(***z****p*∗*re*−*train* + ***I****z* · ***x***), ***y***) > ℓ(O*eep*(***z****p*∗*ost*−*train* + ***I****z* · ***x***), ***y***) (2)

If EPL learning expects to learn, such as back-propagation algorithm, in which all steps of epoch flow exactly to a gradient that reduces loss, the conditions to be satisfied can be written as Eq. ([3](#_bookmark3)).

−∆***z***∗ = ***z***∗ − f

*nep*

(f*nep*

(***z***∗ + ***I****z*

· ***x***) + ***F****z*

ℓ) = α( ∂ℓ(O*eep*(***z***∗ + ***I****z* · ***x***), ***y***))*T* (3)

∂***z***∗

·

This means that if the purpose of the model parameter is to find the back-propagation algorithm, if ***I****z*, ***O****z* is small, it is to reduce the cosine-similarity like this Eq. ([4](#_bookmark4)).

Let ***A*** = −∆***z***∗ ≈ −∂f*nep*(***z***∗ + ***z***)(***I***

∂***z***

*z*

· ***x*** + ***F****z*

ℓ), ***B*** = ( ∂ℓ(O*eep*(***z***∗ + ***I****z* · ***x***), ***y***))*T*

∂***z***∗

·

***A*** · ***B***

(4)

L(θ, back − propagation − algorithm) = |***A***| |***B***|

In this paper, we explore θ spaces in which numerous training methods exist, including back-propagation algorithm.

## Property of EPL Learning

EPL learning occurs by repeatedly passing through implicit layers infinitely. During this process, if even a single value of the vector starts to diverge, it becomes impossible to compute the output of the implicit layer. Therefore, the implicit layer requires a *Non- diverging* property. In this paper, we can achieve this with the following strategy:

* + 1. We divide the implicit layer into two stages: first passing through the local implicit layer and then the external implicit layer.
    2. For every input space, there exists a function that maps it to a single scalar energy.
    3. The possible input space of the implicit layer cannot have infinite negative energy.
    4. The output of the local implicit layer always has lower energy than the input.
    5. At this time, the amount of reduction is proportional to the magnitude of the vector’s components. In other words, if even one value is infinite, the amount of energy reduction becomes infinite.
    6. The external implicit layer always flows finite energy with an upper limit.

## Design Implict Layer

We propose a dynamical system *d****z*** = F*θ*(***z***), ***z*** = (***x***, ***p***), ***x*** ∈ R*n*×1, ***p*** ∈ R*n*×1, compatible

*dt*

*>*0

with biomodel of system biology. This can be treated as implicit layer f*θ* follows: *d****z*** =

*dt*

F*θ*(***z***) ⇔ ***z****i*+1 = ***z****i* +dt · F*θ*(***z****i*) = f*θ*(***z****i*) ⇒ ***z***∗ = ***z***∗ +dt · F*θ*(***z***∗) ⇔ F*θ*(***z***∗) = 0 We prove mathematically convergence of the dynamical system. In addition, we propose a reduced

dynamical system that can significantly reduce the computation of the implicit layer.

* + 1. Energy

The energy function of the dynamical system (DS) is defined based on the actual physical quantity. Consider a node by grouping the same states (component, composition state, momentum, direction of motion, position) of molecules. Each node in DS has a mass and momentum, expressed as ***x*** and ***p*** . DS represents the sum of four energies: H(***x***, ***p***) =

−S*entrophy* + H*enthalpy* + U*electric* + E*kinetic*. From a physical and chemical point of view, physical energy in a system with a *constant temperature* can be expressed. The presented

form of the energy function Eq. ([5](#_bookmark5)). is consequently an extension of entropy and physical quantity in quadratic function [Nocedal and Wright](#_bookmark32) ([2006](#_bookmark32)). [Low et al.](#_bookmark31) ([1973](#_bookmark31)) This allows us to explain the laws and models of biology and chemistry. (***x****T* : Transpose of vector, ***x*** ◦ ***y*** : Hadamard product)

H(***x***, ***p***) = ***x****T* ln( ***x*** ) + 1 ***x****T* ***V x*** + 1 (***p*** ◦ ***p*** ◦ ***x***) (5)

***x***∗ 2

∗***x***

2***m****T*

## Entrophy & Enthalpy −S

*entropy*

(***x***, ***p***) + H

*enthalpy*

(***x***, ***p***) = ***x****T* ln( ***x*** ) = ***x****T* ln(***x***) + ***a****T* ***x***

The probability that a single molecule exists in that state, by Arrhenius equation, is inversely

proportional to its energy state, the enthalpy ***x***∗ ∝ e−***E****x* . In this paper, we replace the energy constant with ***a*** ∈ R*n*×1. The molecular chemical reaction network (CRN) can be

described as a flow that reduces the defined Gibbs free energy H − S. However, when electric Potential Energy and kinetic Energy (the energy of the other two energy terms U, E) change, It means that the energy state of the molecule ( *∂H* ) changes, which also changes the probability of a molecule’s presence. This explains the change in the activation rate according to the voltage of the neuron’s voltage-gated sodium channels [de Lera Ruiz and](#_bookmark23) [Kraus](#_bookmark23) ([2015](#_bookmark23)).

*∂****x***

2

**Electric Potential Energy** U (***x***, ***p***) =  1 ***q****T* 1 ***q*** = 1 ***x****T* ***V x***, ***V*** = *diag*(***q****c*) ∈

*electric*

4*πϵ*0

*x* ***R***

*x*

2

2*πϵ*0***R***

R*n*×*n* Electrical potential energy is related to the movement that creates an electric

current. The movement of electrons perpendicular to the gradient of the energy function, the movement of the hamiltonian flow, and the movement of the longitudinal velocity at which electrons collide with protons as they move, are current and resistance [Vilasi](#_bookmark37) ([2001](#_bookmark37)).

**Kinetic Energy** E*kinetic*(***x***, ***p***) = 1 *T* (***p*** ◦ ***p*** ◦ ***x***) This is the momentum energy of the molecules. ***m*** ∈ R*n*×1 corresponds to the mass of the molecule. Electrical potential energy is converted to kinetic energy due to hamiltonian flow.

2***m***

*>*0

* + 1. Flow

We define symbol ***M*** ∈ R*n*×*c* that is the elemental matrix of ***x***, and ***M* ⊥** ∈ R*n*×*e* that is

≥0

stoichiometric metrics. They are orthogonal (***M*** *T* ***M* ⊥** define a total of four types of flows.

= 0*c*×*e*) to each other. Using this, we

**Total Flow** The total flow is as per the equation below (r(***x***) := relu(***x***)), This means dynamical system.

*∂****x*** *∂****x***

*d****x*** = ***M* ⊥** h−***k*** ◦ (e *∂H r*(***M* ⊥)** − e *∂H r*(−***M* ⊥**))*T* + ***v*** ◦ ( *∂H* ***M* ⊥**)*T* ◦ (eln(***x***)*T* |***M* ⊥**|)*T* i + ***h*** ◦ ***x*** ◦ (e− *∂H* − 1)*T*

*dt ∂p*

*∂****x***

(6)

*d****p*** = −***M* ⊥** h***v*** ◦ ( *∂H* ***M* ⊥**)*T* ◦ (eln(***x***)*T* |***M* ⊥**|)*T* i − ***c*** ⊙ ( *∂H* )*T*

*dt*

*∂****x***

*∂****p***

The gradient ( *∂H* ∈ R1×*n*, *∂H* ∈ R1×*n*) at point (***x***, ***p***) is calculated as Eq. ([7](#_bookmark7)).

*∂****x*** *∂****p***

∂H = (ln(***x***) + 1)*T* + ***a****T* + ***x****T* ***V*** + ***p*** ◦ ***p*** *T* ∂H = ( ***p*** ◦ ***x*** )*T* (7)

∂***x***

2***m***

∂***p***

***m***

**Chemical Flow** *d****x*** = −***M* ⊥** h−***k*** ◦ (e *∂H r*(***M* ⊥)** − e *∂H r*(−***M* ⊥**))*T* [i](#_bookmark28) , ***k*** ∈ R*e*×1

*dt*

*∂****x***

*∂****x***

≥0

This flow can explain chemical reaction and diffusio [Hochberg and Rib´o](#_bookmark28) ([2018](#_bookmark28)).

## Hamiltonian Flow

h i h i

*d****x*** = ***M* ⊥ *v*** ◦ ( *∂H* ***M* ⊥**)*T* ◦ (eln(***x***)*T* |***M* ⊥**|)*T* , *d****p*** = −***M* ⊥ *v*** ◦ ( *∂H* ***M* ⊥**)*T* ◦ (eln(***x***)*T* |***M* ⊥**|)*T*

*dt*

*∂p*

*dt*

*∂****x***

***v*** ∈ R*e*×1 This flow can account for the current. With hamiltonian flow alone, it’s like a current without resistance. This preserves the energy of the entire system. It’s a term that’s never considered in the problem of finding the solution of the convex optimization flow, but in real biology, this flow causes the flow of electrons. It’s important to play an algorithmic role in going to an increasingly learned fixed point, not just looking for fixed points.

*>*0

**Collision Flow** *d****p*** = −***c*** ⊙ ( *∂H* )*T* , ***c*** ∈ R*e*×1 It means that the momentum decreases due

*dt*

*∂****p***

*>*0

to the collision of molecules. Unlike hamiltonian flow, these flows reduce the energy of the

system. In the actual physical phenomenon, just as heat energy is emitted from resistance, it is transformed into the translational or rotational motion of molecules, the thermal energy. But these temperatures quickly spread around. Therefore, it can reduce the amount of computation by ignoring this term.

**External Homeostasis Flow *h*** ◦ (eln(***x***)−( *∂H* )*T* − ***x***), ***h*** ∈ R*n*×1 Living things need

*∂****x***

≥

0

endless sources of energy from the outside to match the homeostasis of ATP-ADP energy

sources. To accommodate this, we treat that ***x***[*i*:*n*], part of ***x***, is connected to the external system. Their flow is expressed by the constant h. In other words, it satisfies ***h***[0:*i*−1] = 0, and ***h***[*i*:*n*] indicates the speed of the external flow. Unlike the three flows above, only substances that are not connected to the external system satisfy the law of conservation of

mass. *d* (***M*** *T* ***x***) = 0

*dt h*

* + 1. Convex Optimization Problem

All defined flows can be explained by the direction of decreasing the proposed system’s energy function H(***x***, ***p***) by proving to Appendix [A](#_bookmark39). Also, ***x*** has a convex domain that satisfies both the law of conservation of mass : = *d* (***M*** *T* ***x***) = 0 and positive of mass. So,

*dt h*

their flow can be reinterpreted as the solving convex optimization problem Eq. ([8](#_bookmark8)).

min

***x****,****p***

H(***x***, ***p***)

(8)

subject to ***x*** > 0, ***M*** *T* ***x*** = ***C****h*

*h*

By solving the finite convex set, the energy satisfies the bounded condition. Through energy reduction and bounded conditions, mathematically satisfied that it always converges to the equilibrium point.

* + 1. Reduced Dynamical System

We will solve the system defined above with ODE solver to find the equilibrium point. However, the computational difficulty is high due to frequent log and exp functions. Therefore, we present a reduced dynamical system (RDS), which is a scale-down model of the defined dynamical system.

RDS := F***θ***(***x***, ***p***), ***θ*** = [***x***0, ***M*** , ***M*** ⊥, ***S***, ***D***, ***m****c*, ***q****c*, ***a***, ***k***, ***v***, ***c***, ***h***] (9)

Give each node its genetic and spatial information. This will become a substrate in HyperNeat as the system evolves in the future. Most log and exp calculations can be converted to multiplication functions by grouping the spaces of nodes.

**Gene of Entity** For all ***x****i* ∈ X, the function f*g* : X ›→ G = {***g***1, ***g***2, ..., ***g****m*} , ***g****i* ∈ R*g*×1 maps to the genetic space G. For all ***g****i*, the function f*c* : G ›→ C = {***c***1, ***c***2, ..., ***c****k*} , ***c****i* ∈ R*c*×1 maps to the component space C. Each component has a mass and a charge, which is expressed as ***m****c* ∈ R*c*×1 and ***q****c* ∈ R*c*×1. That is, the mass and charge of ***x****i* are f*c*(f*g*(***x****i*))*T* · ***m****c* and

*T*≥0

f*c*(f*g*(***x****i*)) · ***q****c*. For any gene ***g****i*, ***g****j*, ***g****k*, Whether a chemical reaction ***g****i* + ***g****j* ⇒ ***g****k* is satisfied

can be determined as a reaction function f*r*, that f*r*(***g****i*, ***g****j*) = ***g****k* and f*c*(***g****i*) +f*c*(***g****j*) = f*c*(***g****k*)

. In RDS, only two reactants (two may be the same) are allowed to create one product. The Na+ channel of the HH model can operate as a function of the active element only when three active factors are combined. Such a multi-molecular product and the like can be expressed by the markov model chemical reaction network, such as [Fink and Noble](#_bookmark25) ([2009](#_bookmark25)).

**Space of Entity** Every ***x****i* ∈ X has their own space ***si***, a function that maps it is defined as follows: f*s* : X = {***x***1, ***x***2, ..., ***x****n*} ›→ S = {***s***1, ***s***2, ..., ***s****s*} . Let a space matrix ***S*** ∈ R*n*×*s*

≥0

as follows: ***S***

:= (0 if f*s*(***x****i*)

***s****j* . Let a distance matrix ***D*** ∈ R*s*×*s*

*j*

, This is the

matrix that determines the distance between spaces. For computational gain, we make the

*ij*

1 if f (***x*** ) = ***s***

≥0

*s*

*i*

inverse of these distance matrix  1

***D***

sparse matrix. In other words, it makes the distance

of the neighboring spaces infinite, leaving them out of the calculate electrical potential energy. With the newly defined gen-space information, the electric potential matrix ***V*** can be calculated as follows: ***q*** = diag((***Mq****c*)*T* )***S***, ***V*** = ***q***( 1 )***q****T*

***D***

**Reduce Flow and Calculate** Depending on whether the space of the components is the same or different, an allowable flow is determined. For two elements of the same space, only the chemical flow of nodes capable of chemical reaction is acceptable In other spaces, diffusion flow and hamiltonian flow of nodes with the same genetic information are possible. Through these constraints, the calculation volume can be significantly reduced for three reasons.

1. ***D*** becomes sparse matrix, so ***V*** also becomes sparse matrix.
2. In the chemical reaction, the flow in the same space occurs, so electric potential energy is

not changed. ∵ *dUelectric* ∝ *d****x****T* ***q*** ∝ ***M*** ⊥ diag((***Mq*** )*T* )***S*** ∝ ***M*** ⊥ ***M*** = 0

*d*(*chemical react*)

*d*(*chemical react*)

*ch*

*c*

*ch*

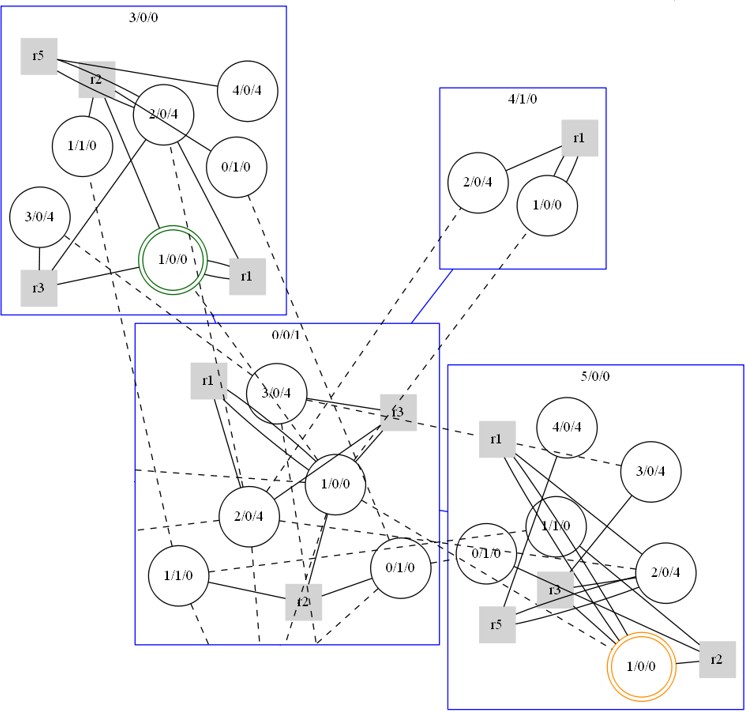
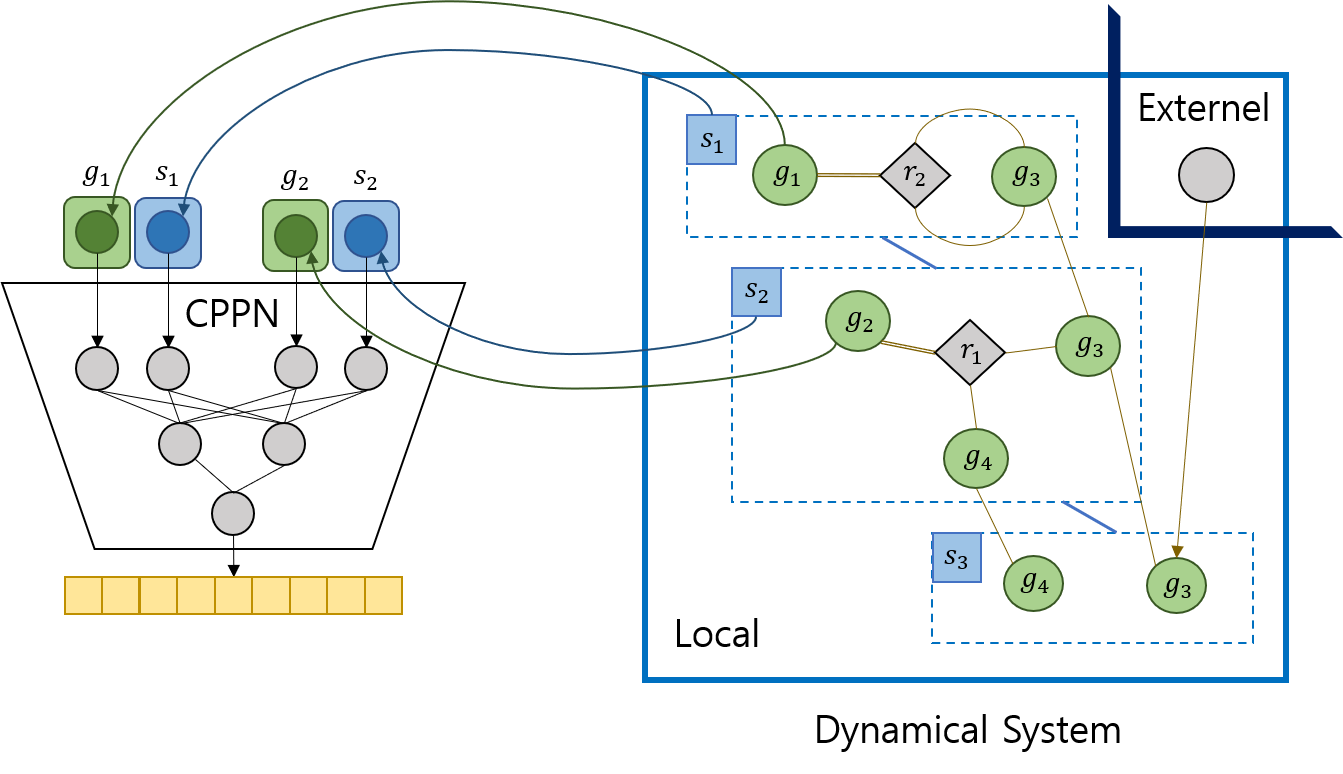
*ch*×*s*

1. Since the chemical flow has two reactants, one product, log and expterm can be replaced

by one multiplication. (ex) In the reaction of a + b ⇔ c, the existing one is eln(*a*)+ln(*b*) − eln(*c*)

If so, it can be calculated by replacing it with a ∗ b − c)

# Experiment



* 1. Using space and gene information to HyperNeat substrate (b) Example of DS

Figure 2: Generation Dynamical System by HyperNeat and Example

When biological cells undergo expression, the configuration of the dynamical system is influenced by the concentration of surrounding substances, and it exhibits symmetry within the same cell type. To leverage this characteristic, we employ HyperNeat (Fig. [2(*a*)subfigure](#_bookmark10)) for evolving the dynamical system, which enables the design of regular patterns such as symmetry, repetition, and variation. By selectively disabling specific components of the CPPN input, we define the necessary functions.

**Algorithm 2** Expression Algorithm

1: **procedure** Expression by CPPN(***g****i*, ***s****i*, ***g****i*, ***s****i*)

2: ***g****i* exist ← tanh−1(CPPN(***g****i*, 0, ***g****i*, 0))[0] > 0 ▷ Generation gene map 3: ***s****i* exist ← tanh−1(CPPN(0, ***s****i*, 0, ***s****i*))[0] > 0 ▷ Generation Space map 4: node (***g****i*, ***s****i*) exist ← tanh−1(CPPN(***g****i*, ***s****i*, ***g****i*, ***s****i*))[0] > 0 ▷ Generation node 5: reaction function f*r* ← abs(tanh−1(CPPN(***g****i*, 0, ***g****j*, 0)[0])

6: node (***g****i*, ***s****i*) connected externel nodes ← tanh−1(CPPN(***g****i*, ***s****i*, ***g****i*, ***s****i*))[1] > 0

7: space (***s****i*), (***s****j*) is neighborhood ← tanh−1(CPPN(0, ***s****i*, 0, ***s****j*))[1] > 0

8: charge of (***c****i*) ← tanh−1(CPPN(***c****i*, 0, 0, 0))[2]

9: mass of (***c****i*) ← abs(tanh−1(CPPN(0, 0, ***c****i*, 0))[2])

10: k or non-exist(< 0) of chemical reaction edge ← abs(tanh−1(CPPN(***g****i*, ***s****i*, ***g****j*, ***s****i*))[2])

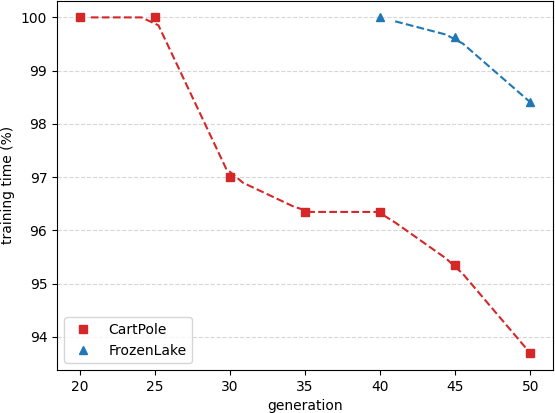
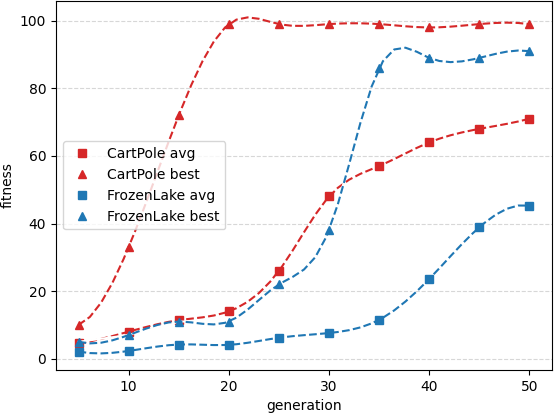
11: v or non-exist(< 0) of hamiltonian edge ← abs(tanh−1(CPPN(***g****i*, ***s****i*, 0, ***s****j*))[2])

12: k or non-exist(< 0) of diffusion edge ← abs(tanh−1(CPPN(0, ***s****i*, ***g****i*, ***s****j*))[2])

13: **end procedure**

The dynamical system (DS) is shown in Fig. [2(*b*)subfigure](#_bookmark11). It consists of a blue square region with nodes as circles. Input nodes are green, the output node is orange, and nodes with chemical reactions are gray squares labeled with the reaction type. Connections exist between nearby spaces. Diffusion flow uses solid lines and Hamiltonian flow uses dotted lines, but the external flow isn’t visually shown; it’s stored as a model parameter.

# Result

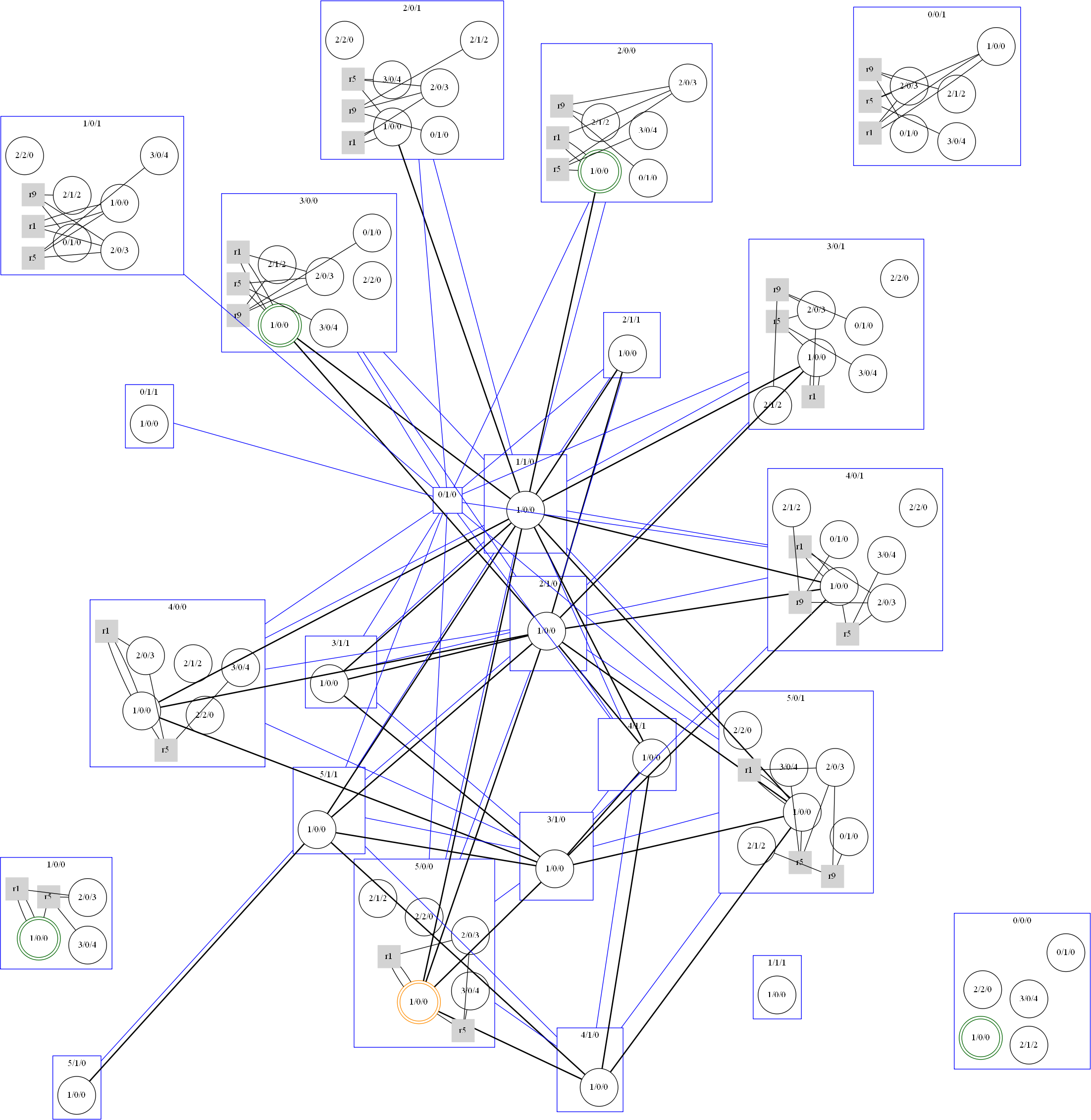
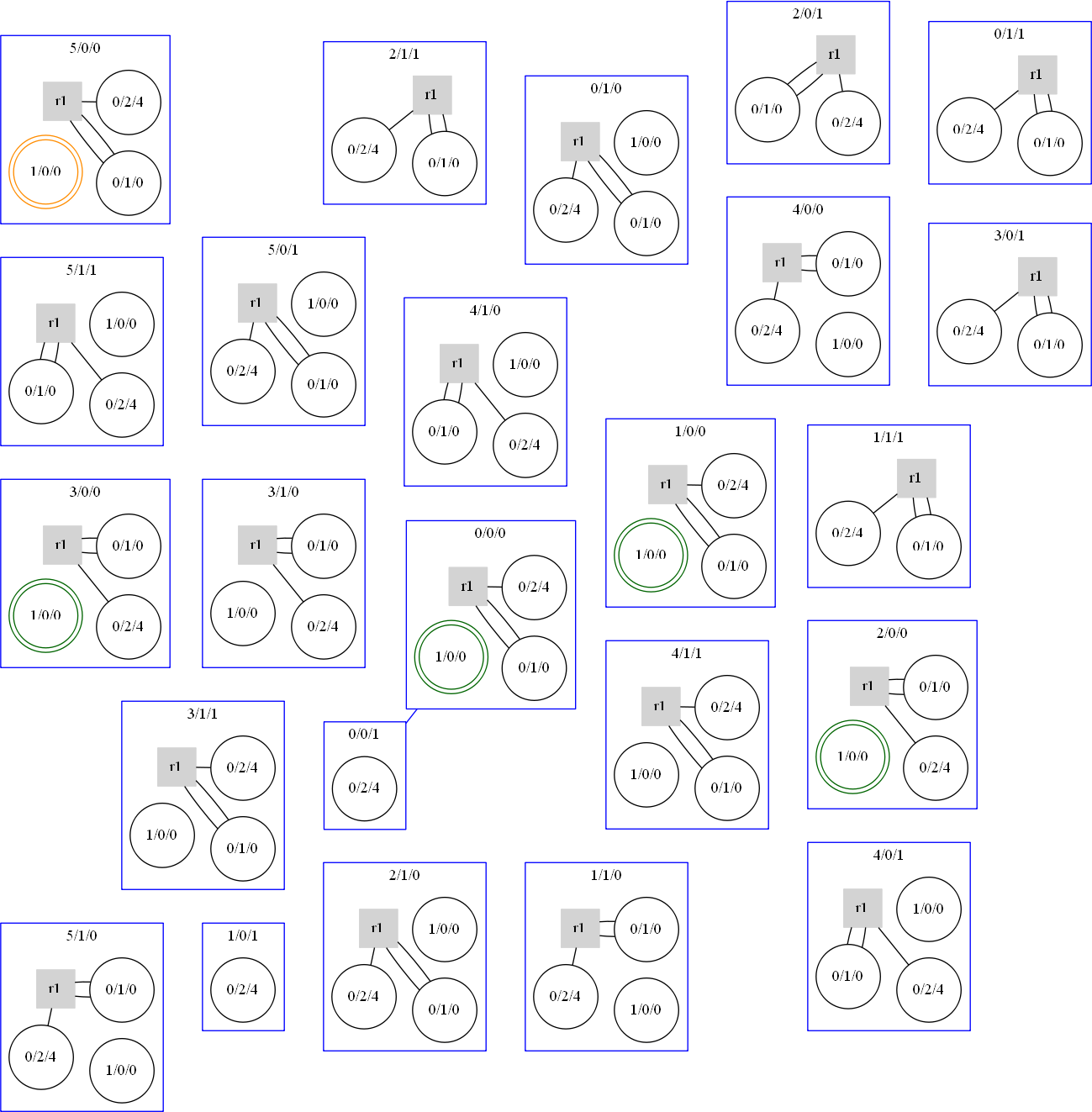


* 1. Average(avg) and best result per generation (b) Reduced training time (%)

Figure 3: Training results for two environments

We conducted experiments using the CartPole-v1 and FrozenLake-v0 environments from OpenAI. A total of 100 populations were evolved over 50 generations, utilizing an RTX 2070 GPU. The entire process took approximately 87 hours. We evaluated the fitness of the generated dynamical systems (DS) after 10 epochs of learning.

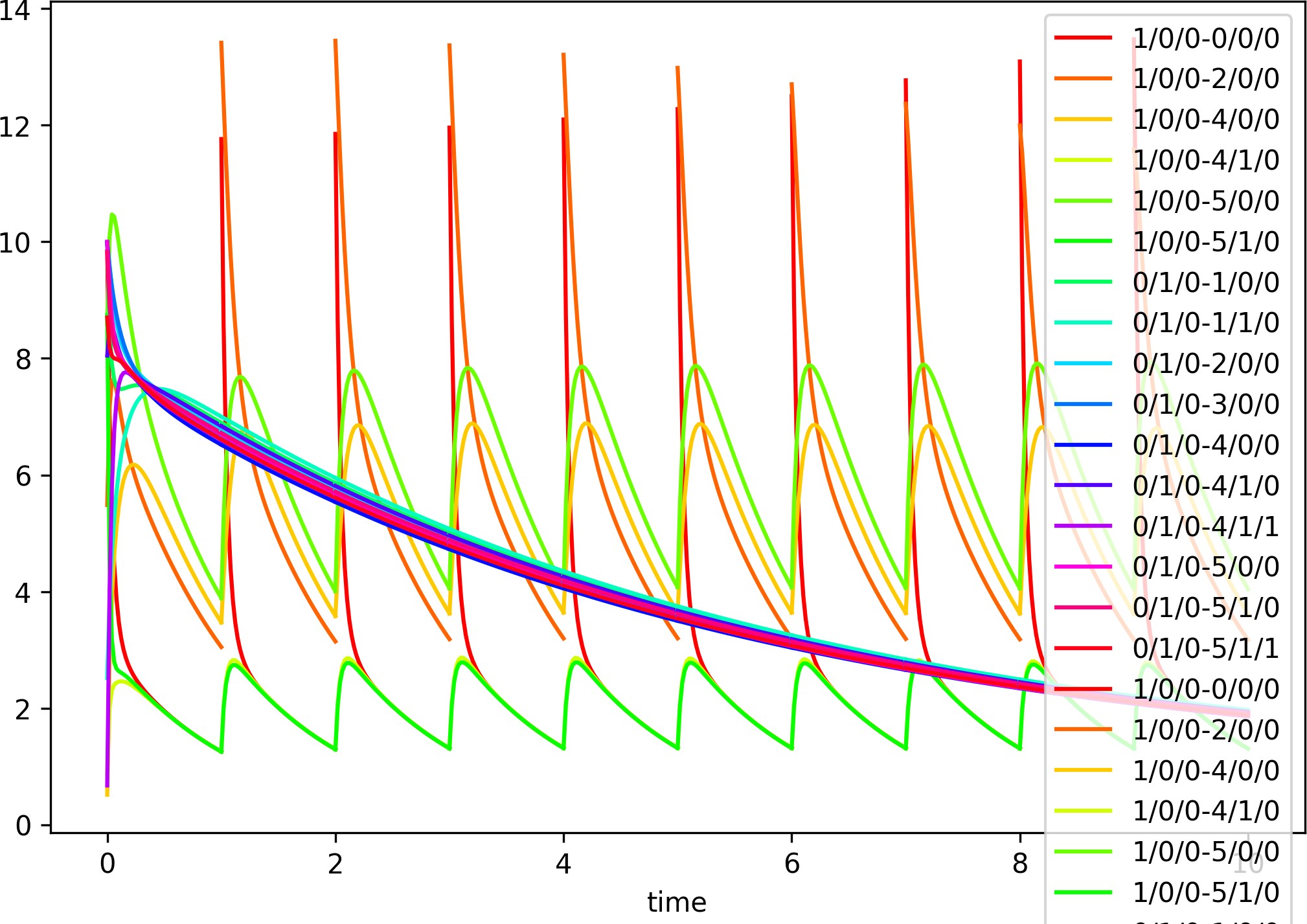
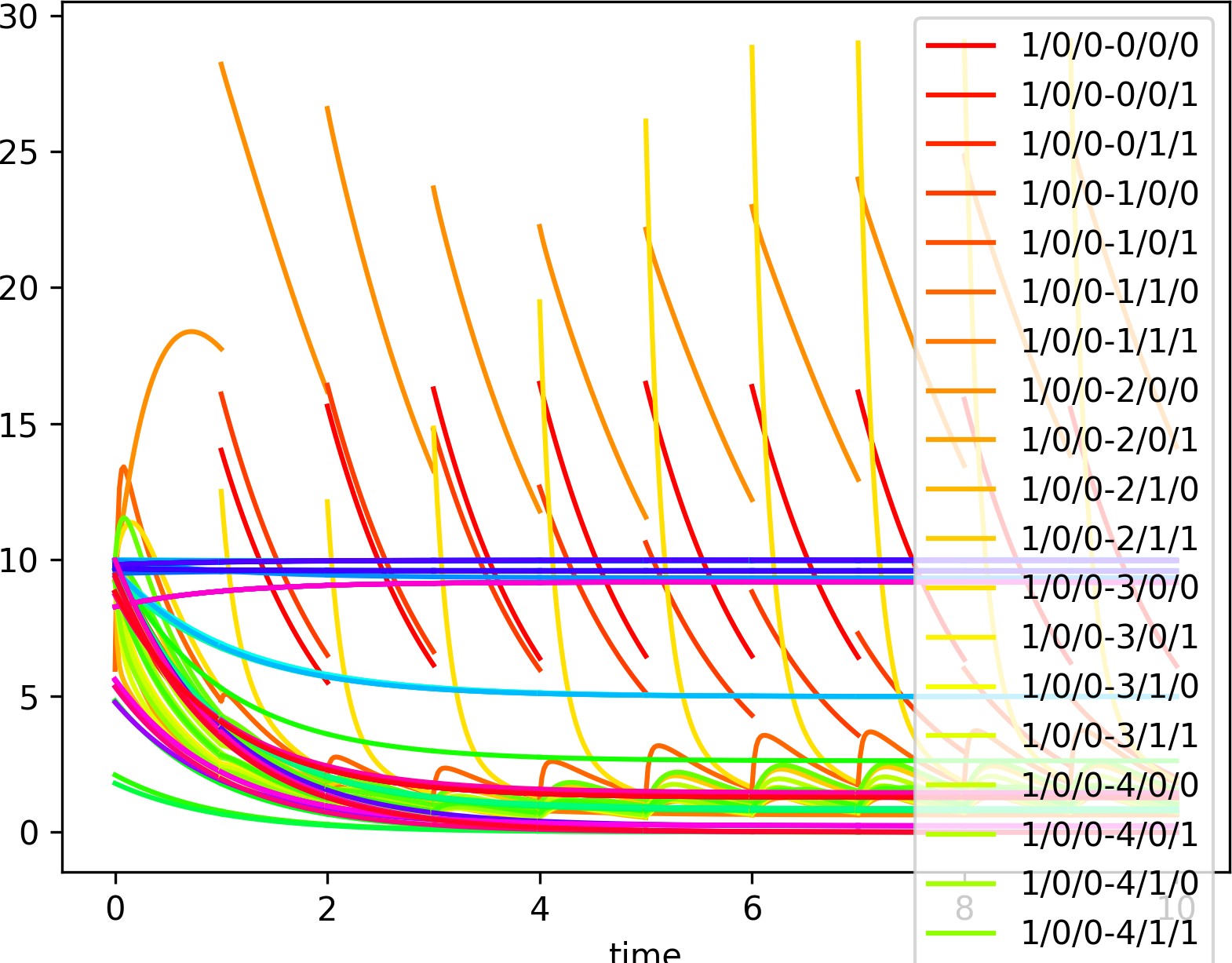
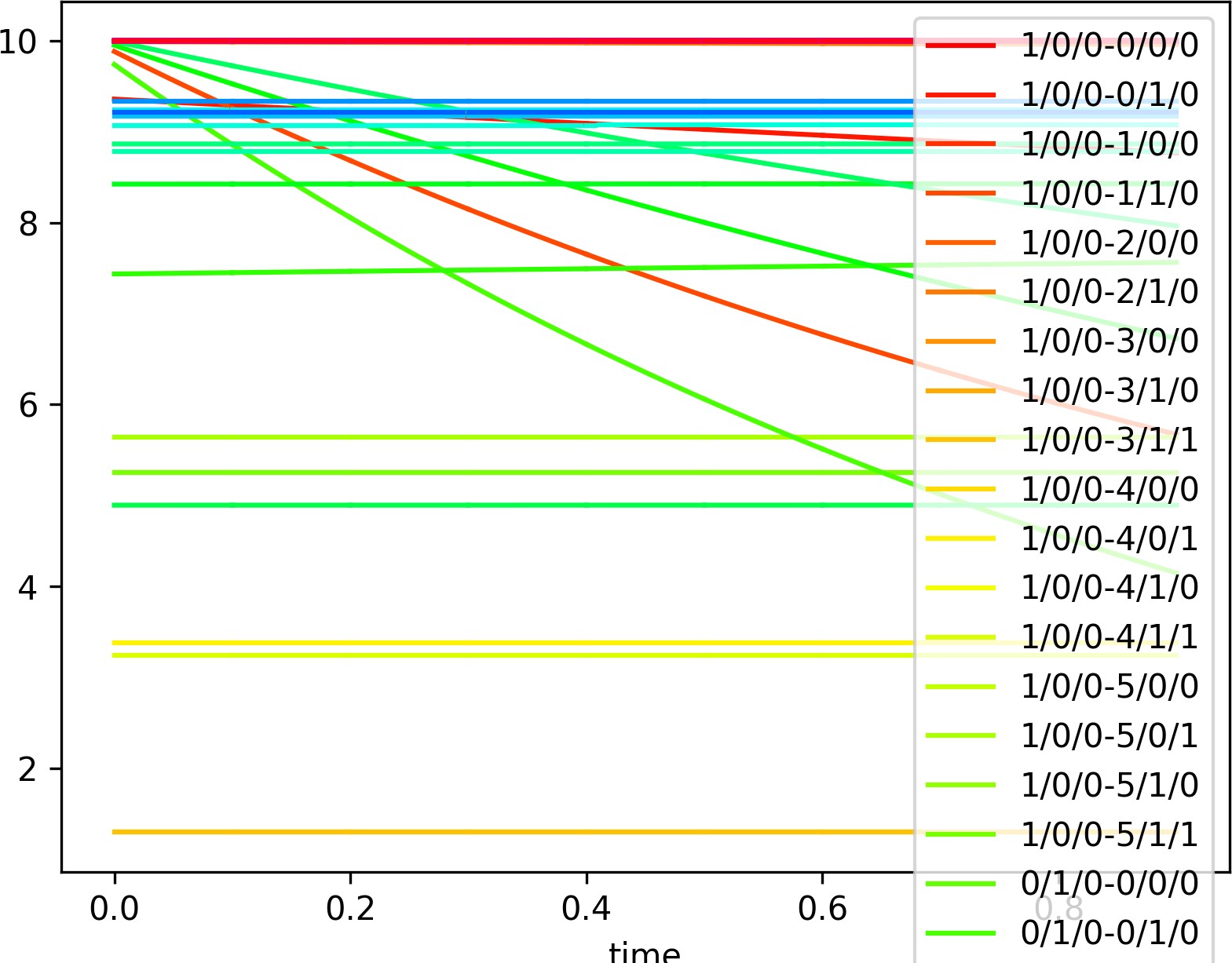
The results showed that the CartPole environment began to learn effectively around generations 10-20, while FrozenLake showed significant progress around generations 30-40. Furthermore, once the goal (including environment reward, learning speed, and memory usage) was achieved, we observed an average reduction of 0.2 in learning time per generation.



* + 1. Dynamical system example in gen 1 (b) Dynamical system example in gen 10

Figure 4: Examples of Dynamical systems from the 1st and 10th generations

The initial generations 1-4 consisted of simple repetitive structures where input and output nodes were not even connected (Fig. [4(*a*)subfigure](#_bookmark12)). From generations 5-9, individuals that exhibited a positive impact between their input and output nodes were favored for survival. In generations 10-20, more diverse chemical reaction patterns emerged, and N:N multiple connections were identified (Fig. [4(*b*)subfigure](#_bookmark13)).



* + - 1. Gen 1 (b) Gen 10 (c) Gen 50

Figure 5: Best Fitness Examples from the 1st, 10th, and 50th Generations

Surprisingly, the output responses began to resemble the spiking neuron model (Fig. [5(*b*)subfigure](#_bookmark14)). By generations 40-50, spaces started clustering, forming units resembling neurons. Additionally, the output responses became more sensitive to the input, resembling an adaptive leaky integrate-and-fire (LIF) model similar to that of spiking neural networks (SNN) (see Fig. [5(*c*)subfigure](#_bookmark15) for an example at generation 50).

# Conclusion

We have examined the biological properties and learning mechanisms of neurons and proposed a novel learning approach for artificial neural networks (ANNs). In this approach, the parameters of the model inherently represent the essence of the learning algorithm itself. Consequently, we conducted an evolutionary search for model parameters to discover alternative training methods that do not rely on traditional algorithms like back-propagation. Through this evolutionary process, we identified dynamical systems that are evolutionarily selected, indicating the presence of learnable implicit layers. By improving the efficacy of the evolutionary algorithm, we can accelerate the ODE solver’s ability to discover more effective learning strategies. Ultimately, our aim is to uncover solutions to previously unsolvable problems using conventional learning methods.

# References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Christian Bick, Marc Goodfellow, Carlo R Laing, and Erik A Martens. Understanding the dynamics of biological and neural oscillator networks through exact mean-field reductions: a review. *The Journal of Mathematical Neuroscience*, 10(1):1–43, 2020.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, 7(11):e1002211, 2011.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu.*

*Rev. Neurosci.*, 31:25–46, 2008.

Vijayalakshmi Chelliah, Camille Laibe, and Nicolas Le Nov`ere. Biomodels database: a repository of mathematical models of biological processes. In *In Silico Systems Biology*, pages 189–199. Springer, 2013.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Jane Cronin and Jane Cronin Boutelle. *Mathematical aspects of Hodgkin-Huxley neural theory*.

Number 7. Cambridge University Press, 1987.

Manuel de Lera Ruiz and Richard L Kraus. Voltage-gated sodium channels: structure, function, pharmacology, and clinical indications. *Journal of medicinal chemistry*, 58(18):7093–7118, 2015.

Marzia Di Filippo, Chiara Damiani, and Dario Pescini. Gpruler: Metabolic gene-protein-reaction rules automatic reconstruction. *PLoS computational biology*, 17(11):e1009550, 2021.

Martin Fink and Denis Noble. Markov models for ion channels: versatility versus identifiability and speed. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1896):2161–2179, 2009.

Paul W Glimcher. Understanding dopamine and reinforcement learning: the dopamine reward prediction error hypothesis. *Proceedings of the National Academy of Sciences*, 108(Supplement 3): 15647–15654, 2011.

Nicholas Hernjak, Boris M Slepchenko, Kathleen Fernald, Charles C Fink, Dale Fortin, Ion I Moraru, James Watras, and Leslie M Loew. Modeling and analysis of calcium signaling events leading to long-term depression in cerebellar purkinje cells. *Biophysical journal*, 89(6):3790–3806, 2005.

David Hochberg and Josep M Ribo´. Stoichiometric network analysis of entropy production in chemical reactions. *Physical Chemistry Chemical Physics*, 20(36):23726–23739, 2018.

Jianxin Huang and Zhehuang Huang. A parameters optimization of synergetic neural network based on differential evolution algorithm. *International Journal of Hybrid Information Technology*, 9(4): 359–366, 2016.

Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14 (6):1569–1572, 2003.

Philip S Low, Jeffrey L Bada, and George N Somero. Temperature adaptation of enzymes: roles of the free energy, the enthalpy, and the entropy of activation. *Proceedings of the National Academy of Sciences*, 70(2):430–432, 1973.

Jorge Nocedal and Stephen J Wright. Quadratic programming. *Numerical optimization*, pages 448–492, 2006.

Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: opportunities and challenges.

*Frontiers in neuroscience*, page 774, 2018.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies.

*Evolutionary computation*, 10(2):99–127, 2002.

Amirhossein Tavanaei and Anthony Maida. Bp-stdp: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing*, 330:39–47, 2019.

Jordan A Taylor and Richard B Ivry. Cerebellar and prefrontal cortex contributions to adaptation, strategies, and reinforcement learning. *Progress in brain research*, 210:217–253, 2014.

Gaetano Vilasi. *Hamiltonian dynamics*. World Scientific, 2001.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, 34, 20