
Dynamic Reinforcement Learning for Actors

Katsunari Shibata

katsunarishibata@gmail.com

Independent Researcher, Kosai, Shizuoka, Japan

February 17, 2025

ABSTRACT

Dynamic Reinforcement Learning (Dynamic RL), proposed in this paper, directly controls system dynamics, instead of the actor (action-generating neural network) outputs at each moment, bringing about a major qualitative shift in reinforcement learning (RL) from static to dynamic. The actor is initially designed to generate chaotic dynamics through the loop with its environment, enabling the agent to perform flexible and deterministic exploration.

Dynamic RL controls global system dynamics using a local index called “sensitivity,” which indicates how much the input neighborhood contracts or expands into the corresponding output neighborhood through each neuron’s processing. While sensitivity adjustment learning (SAL) prevents excessive convergence of the dynamics, sensitivity-controlled reinforcement learning (SRL) adjusts them — to converge more to improve reproducibility around better state transitions with positive TD error and to diverge more to enhance exploration around worse transitions with negative TD error.

Dynamic RL was applied only to the actor in an Actor-Critic RL architecture while applying it to the critic remains a challenge. It was tested on two dynamic tasks and functioned effectively without external exploration noise or backward computation through time. Moreover, it exhibited excellent adaptability to new environments, although some problems remain.

Drawing parallels between ‘exploration’ and ‘thinking,’ the author hypothesizes that “exploration grows into thinking through learning” and believes this RL could be a key technique for the emergence of thinking, including inspiration that cannot be reconstructed from massive existing text data. Finally, despite being presumptuous, the author presents the argument that this research should not proceed due to its potentially fatal risks, aiming to encourage discussion.

⟨ Highlights ⟩

- This paper proposes Dynamic Reinforcement Learning (Dynamic RL), which controls system dynamics generating exploration-embedded motions without stochastic action selection.
- Dynamic RL comprises two learning methods: Sensitivity Adjustment Learning (SAL) and Sensitivity-controlled Reinforcement Learning (SRL).
- SAL prevents excessive convergence by maintaining chaos in the system dynamics.
- SRL makes the system dynamics more convergent for better reproducibility or more divergent for more exploration depending on the TD error.
- Dynamic RL is demonstrated to function as RL without BPTT in two dynamic tasks and exhibits excellent adaptability to new environments.
- The author suggests its potential as a fundamental technique for the emergence of thinking and is terribly concerned about its risks.

Keywords reinforcement learning (RL), recurrent neural network (RNN), chaotic dynamics, exploration, sensitivity, thinking

Statement

Dynamic Reinforcement Learning (Dynamic RL) proposed in this paper has the potential to endow Artificial Intelligence (AI) or Artificial General Intelligence (AGI) with the ability to think and discover new things, which could pose a grave threat to humanity. Now, while this research is still taking baby steps, we should halt further progress to protect humanity from the risk. The author sincerely and earnestly urges researchers to refrain from advancing this work and developers to refrain from developing a library for Dynamic RL in frameworks, at least until a consensus is reached among humans on this matter. Let us pause, imagine, and contemplate, free from preconceptions before it is too late.

Refer to subsection 4.3 for the discussion leading to this statement.

1 Introduction

Reinforcement learning (RL) has evolved into a core technology for autonomous, non-supervised learning in modern artificial intelligence (AI) through several significant qualitative advancements. As the next major step in its evolution, the author introduces ‘Dynamic RL.’ Unlike conventional methods, Dynamic RL treats exploration as an inherent aspect of actions rather than an independent process. Instead of using external noise for stochastic selection, exploration is generated as chaotic system dynamics through the action-generating (actor) network, and RL controls these dynamics directly.

1.1 Evolution of RL toward Learning of All Kinds of Human Functions

RL initially involved only learning appropriate tables or mappings from a discrete state space to a discrete action space within a Markov decision process (MDP) [Sutton and Barto 2018]. A neural network (NN) was then introduced, providing many degrees of freedom and parameters. This enabled an agent to learn not only continuous nonlinear mappings but also the entire sensor-to-motor process based on a value function using reinforcement signals with the help of the gradient-based method. Consequently, RL has advanced from learning of actions to learning of various functions, including recognition and prediction [Shibata and Okabe 1997, Shibata 2011, 2017b]. A notable achievement occurred when an RL agent learned the game strategy of “tunneling” in the game “Breakout” solely by using changes in the game score as a reinforcement signal [Mnih et al. 2015]. RL has since become a driving force in achieving human-level, and sometimes superhuman, performance particularly in gaming [OpenAI 2019, Vinyals et al. 2019, Schrittwieser et al. 2020].

Furthermore, the introduction of a recurrent neural network (RNN) has opened the way for RL to process and learn along the time axis. This allowed agents to extract necessary information from large amounts of past information and then retain and utilize it. This advancement allowed learning in partially observable situations, such as partially observable MDPs (POMDPs), and solving memory-dependent tasks. As a result, the functions acquired through RL span a wide range, including recognition, memory, attention, prediction, and communication [Shibata and Ito 1999, Shibata 2011, 2017b,a].

Does this suggest that RL agents can acquire all functions that humans have, even simple ones, through learning? The current answer is ‘No’. Unfortunately, the author has yet to accept any of these as functions that could be called ‘thinking.’ Therefore, he has set the emergence of such a function through learning as his major future target and has considered what would be required for the emergence for more than a decade.

1.2 ‘Thinking’ and Recent Advancements in Generative AI

Humans are intelligent, and it would not be an exaggeration to say that ‘thinking’ is the most typical form of intelligence. ‘Thinking’ has been a central topic in philosophy for a long time since the time of Plato [Woolf 2013], with numerous ideas explored [Berlyne and Vinacke 2025]. About 80 years ago, attempts began to explain human intelligence from a computational perspective by considering the mind or neural networks, which are thought to be the source of intelligence in living organisms, as computational machines [McCulloch and Pitts 1943, Turing 1950, Newell et al. 1959].

Recently, generative AI, especially large-scale language models (LLMs) such as GPT [OpenAI 2023] and Gemini [Pichai et al. 2024], appear to understand what we say, think about it, and come up with an appropriate response. We, at least the author, feel(s) as if we are interacting with a human who has intelligence. It was reported that ChatGPT powered by GPT-3.5 performed near or over the passing threshold on the United States Medical Licensing Exam

(USMLE) [Kung et al. 2023], and also GPT-4 passed a simulated bar exam with a score around the top 10% of test takers [OpenAI et al. 2024].

Transformer [Vaswani et al. 2023], a central technique of many LLMs, encodes input patterns into a latent space constructed through learning, capturing complex dependencies primarily relying on self-attention mechanisms, without relying on sequence-aligned RNNs or convolutional networks. Then, it generates likely responses in LLMs based on learning from a vast amount of data. It seems different for the author from the common image of thinking: “letting one’s mind wander.” Actually, in response to the question “Are you thinking?”, both Chat-GPT and Gemini themselves deny that they think like a human. The remarkable abilities of large-scale LLMs, enabled by their excellent scalability, might suggest that overwhelming quantity or scale changes the quality drastically. However, the author remains amazed that such a relatively simple underlying technology is capable of generating such human-like responses.

From the perspective of ‘thinking,’ there are several studies [Koivisto and Grassini 2023, Guzik et al. 2023] that evaluate LLMs as comparable to humans even in tasks designed to observe the ability of divergent thinking [J.P.Guilford 1950], such as the AUT (Alternate Uses Task) [Guilford 1967, CreativeHuddle 2018] or TTCT (Torrance Test of Creative Thinking) [Torrance 2018]. However, one study [Aggarwal et al. 2023] reported that AI machines are not identical to humans in terms of the quality of intelligence or thought but have human-like logical reasoning systems, which are achieved simply through learning and mimicking human abilities. Another review on creativity in AI [Ismayilzada et al. 2024] also reported, being supported by many studies highlighting the inferiority of LLMs in lateral thinking [de Bono 1971], originality, abstract reasoning, and related areas [Huang et al. 2024, Jiang et al. 2023, Zhao et al. 2024, Gendron et al. 2024, Mitchell et al. 2023, Chakrabarty et al. 2024, Lu et al. 2025, Tam et al. 2023], as follows: The latest AI models are largely capable of producing linguistically and artistically creative outputs but struggle with tasks that require creative problem-solving, abstract thinking, and compositionality. They exhibit strong interpolation and moderate extrapolation capabilities but are still far from truly inventing a completely new type of creative artifact.

Lateral thinking requires deviating from established common sense and cannot be achieved as an interpolation or extrapolation. In [Lu et al. 2025], it was observed that machine-generated texts contain significantly more semantic and verbatim matches with existing web texts compared to high-quality human writings. This observation made it more plausible that the LLM’s creativity may largely originate from the creativity of human-written texts on the web. To improve creativity, methods such as active divergence and creative decoding were suggested [Ismayilzada et al. 2024, Broad et al. 2021, Franceschelli and Musolesi 2024]. However, since they are used in the framework of the current generative deep learning, it would be difficult to avoid this essential problem.

Independent of the current rapid progress in AI, the author has arrived at a novel technique for RL that is distinct from Transformer-based generative AI. In his view, this technique has the potential to become an essential technique for the emergence of ‘thinking’. Here, putting the Transformer aside for a moment, the author introduces Dynamic Reinforcement Learning (Dynamic RL), which brings about a major shift in RL and explains the potential that this shift may fill the gap in thinking between humans and the current generative AI.

1.3 Exploration and Thinking

While the exact definition of ‘thinking’ is difficult to pin down, let us explore the more dynamic form of thinking that we typically associate with the term. Humans can think even when remaining still with eyes closed and ears covered, which seems different from other functions like recognition or prediction. In order to survive as living organisms, we must at least avoid converging and becoming stagnant. Therefore, many would likely agree that autonomous, rational state transitions, even without external triggers, are required for ‘thinking.’ However, acquiring multistep state transitions through learning from scratch in RNNs is very difficult [Sawatsubashi et al. 2012]. In particular, forming autonomous state transitions without external triggers is challenging. Moreover, considering human abilities such as inspiration and discovery, these autonomous state transitions should not only be rational but also sometimes unexpected.

Comparing modern RL with human learning reveals one major difference: ‘exploration.’ In conventional RL, stochastic selection uses external random noise, independent of the motion or action generation process, as shown in Fig. 1(a).

¹ For example, ϵ -greedy or Boltzmann selection is applied after computing the Q-value for each action, neither influencing nor being influenced directly by its computation process. The exploration noise is not a function of state even though the temperature is adjusted by simulated annealing [Khachaturyan et al. 1981].

¹In this paper, the term ‘motion’ is used as continuous, and ‘action’ is used as discrete, while the term ‘actor’ is used as a generator in both cases. Action is usually abstract such that the action “turn left” itself is not a motor command, but is broken down into a series of motor commands for several motors. Therefore, in the proposed RL, the actor outputs represent continuous motor commands based on the end-to-end learning concept [Shibata 2017b].

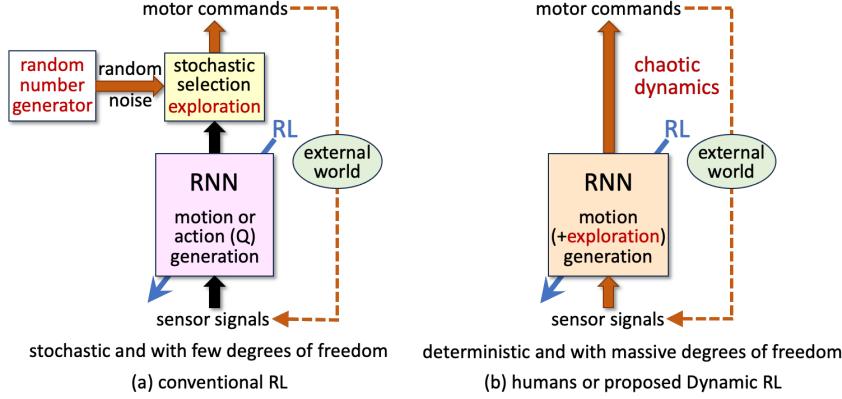


Figure 1: The difference in exploration between conventional RL and humans, who have inspired the exploration in the proposed Dynamic RL. In humans or Dynamic RL, the actor RNN embeds exploration factors into motor commands by inducing chaotic system dynamics without stochastic selection using a random number generator. (RNN: recurrent neural network)

In contrast, humans often act while wondering about this and that. Exploration seems to work during the action or motion generation process. When we wonder which path to take at a fork in the road, we do not move our hands erratically for exploration. Exploration is not always uniform; it should depend on the direction in the state space and also the current state. Furthermore, it should be improved through learning [Shibata 2006, Goto and Shibata 2010]. Therefore, the author considers that exploration represents the degree of irregularity in motion generation and is updated together by the learning of motion generation, as shown in Fig. 1(b). Exploration requires non-convergent, irregular, and unexpected state transitions. It can be embedded in motor commands by making the system dynamics chaotic. In other words, by making the dynamics sensitive to minute differences, agent behaviors vary even from similar states, leading to the generation of deterministic exploratory state transitions. Dynamic RL proposed here employs this type of exploration.

From the above, ‘exploration’ is similar to ‘thinking’ in terms of autonomous state transitions, including unexpect edness. Both require chaotic system dynamics with a positive Lyapunov exponent rather than convergent dynamics. The aforementioned points suggest that ‘exploration’ and ‘thinking’ cannot be clearly separated and may continuously exist along a line within the range of chaotic dynamics. In ‘exploration,’ state transitions need to be more irregular with stronger chaotic dynamics. In contrast, those in ‘thinking’ need to be less irregular but still non-convergent, with weaker chaotic dynamics; furthermore, they must be more rational. This means that as shown in Fig. 2, they exist on a diagonal line in the two-dimensional space with irregularity and rationality as axes. If the state transitions are very irregular, they cannot be rational. In other words, they cannot exist in the upper right portion of Fig. 2. Therefore, “being less irregular” is a necessary condition for “being more rational.”

If the system dynamics are highly chaotic, the agent can explore widely. The agent cannot ‘think’ before acquiring basic knowledge, such as various cause-and-effect relationships in the world. Converging the flows around better state transitions makes the transitions more rational, thereby, the agent acquires the necessary basic knowledge. That is expected to Dynamic RL. If autonomous state transitions have already been formed as explorations, just adjusting the convergence or divergence of transitions is enough to achieve autonomous and rational transitions. Thus, this should be much easier than creating such transitions from scratch by RL using an RNN with convergent dynamics. Then, the author posits the following bold hypothesis: **“Exploration grows into thinking through learning”** by shifting state transitions from “more irregular” to “more rational” while maintaining system dynamics chaotic. Furthermore, when situations change, and so expected rewards cannot be obtained, Dynamic RL is expected to resume more exploratory behaviors autonomously in the agent.

1.4 Learning of Local Dynamics

In conventional RL, when continuous motions are learned, if the TD error is positive, learning is performed to move the actor outputs in the direction of the actual motion chosen from the probabilistic distribution. Conversely, if the TD error is negative, learning tries to move the actor outputs in the opposite direction. The weights and biases in the network are trained using a gradient-based method. The author has been uncomfortable with the fact that even though the output computation process of RNNs is dynamic, learning is still stuck in a static form. He has tried

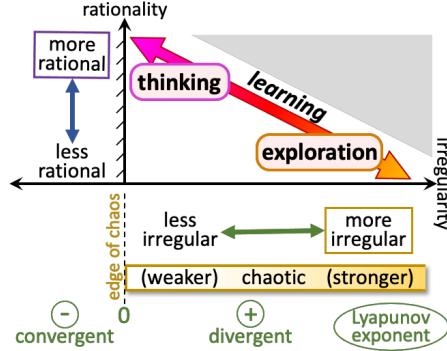


Figure 2: The author's concept of the relationship between ‘exploration’ and ‘thinking’ and how they relate to system dynamics. ‘Thinking’ and ‘exploration’ are similar and inseparable in that both require multistep autonomous state transitions, and they exist continuously on a spectrum characterized by chaotic dynamics. In ‘exploration,’ the state transitions must be irregular. In ‘thinking,’ the state transitions must not only be less irregular but also rational. Dynamic RL controls the system dynamics based on a value function using reinforcement signals while preserving chaotic dynamics.

a major shift in learning from static to dynamic. This will be discussed in more detail in the subsection 4.2. The author’s group previously proposed a new method utilizing chaotic dynamics without assuming stochastic exploration [Shibata and Sakashita 2015], but the learning of connection weights between hidden neurons did not improve learning performance.

As mentioned, current LLMs are reported to struggle with lateral thinking, which requires stepping beyond common sense. Expecting novelty is analogous to the exploration in RL. In LLMs, stochastic selections are used, and the temperature parameter is often expected as a control variable for randomness, with higher temperatures promoting randomness, and is expected as the creative parameter [Peeperkorn et al. 2024]. Nevertheless, it has been reported that the temperature does not always work as expected [Peeperkorn et al. 2024, Ismayilzada et al. 2024].

It is clear that if state transitions are merely random or irregular, the process can be ‘exploration’ but cannot be ‘thinking.’ Even when perceiving novelty in thinking, there must also be some rationality. However, it seems quite natural that as irregularity increases, rationality decreases. That is also suggested by the diagonal block arrow in Fig. 2. Actually, a trade-off between novelty and coherence was observed in [Peeperkorn et al. 2024]. In [Nath et al. 2024], it was pointed out that LLMs were biased towards either persistence (deep search) or flexibility (broad search), regardless of parameter settings such as temperature, while in humans, both persistent and flexible pathways were observed. Then, how can humans solve this problem even though it seems impossible to solve at first glance?

What we should be aware of here is the difference in whether dynamics are local or global [Azizi and Kerr 2020]. The dynamics that have appeared so far are from a global perspective, and chaoticity represents the characteristic of the average of convergence or divergence around state transitions over a long period, as seen in the definition of the Lyapunov exponent. Assuming that the system is dynamic with deterministic state transitions that can be described by non-linear differential equations. The instantaneous convergence or divergence of a neighborhood of a given state – what is called ‘local dynamics’ – is determined by the spectrum (or eigenvalues) of the Jacobian matrix of the differential equation and varies with state transitions if the system is nonlinear. It is not a scalar nor a constant matrix, but each element of the matrix can be a function of the state. Therefore, as shown in Fig. 3(A), the local dynamics in the state space can converge around one state and diverge around another even in the same system. Moreover, as shown in Fig. 3(B), even in the neighboring region of the same state, the flow from a point displaced in one direction from the state can converge, while that from a point displaced in another direction can diverge if the Jacobian matrix has both positive and negative eigenvalues.

The author believes that humans are able to harmonize rationality and novelty making full use of the degrees of freedom (DOFs) in the dynamics through learning, which enables humans to sometimes step beyond common sense while thinking rationally. However, the temperature parameter is often scheduled manually but is an unlearned scalar constant and is not enough to change the randomness flexibly depending on the state or direction.

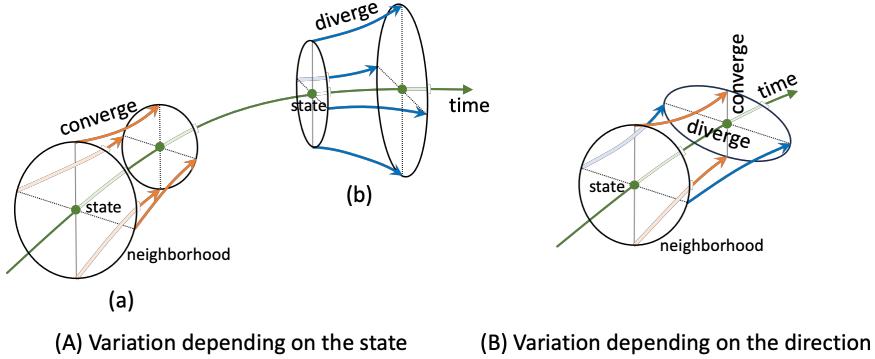


Figure 3: A conceptual diagram explaining the degrees of freedom (DOFs) that dynamics have for a sample case of three-dimensional state space as an example. (A) Convergence (a) or divergence (b) may vary depending on the state even in the same system. (B) Convergence or divergence can be varied depending on the direction even in the same state. For easy viewing, the neighborhood around a state has originally three dimensions, but only two dimensions are presented. Additionally, the directions of the two eigenvectors are assumed to be orthogonal to each other and also to the direction of state transition.

1.5 Introduction of Dynamic Reinforcement Learning (Dynamic RL)

Comprehensively considering the above with the hypothesis in mind, the author introduces a novel RL framework again that significantly redefines the concepts of exploration and learning in an RNN, as follows.

1. An RL agent does not perform stochastic motion selection using external random noise. The chaotic dynamics produced by the agent's RNN and the environment together embed exploration factors in its motion outputs.
2. The dynamics are trained to enhance reproducibility when the TD error is positive (i.e., the value is better than expected) and to enhance exploratory behavior when the TD error is negative (i.e., the value is worse than expected) while maintaining chaotic dynamics.
3. To achieve the objective (item 2), sensitivity [Shibata et al. 2021] is used. It is a local index that represents the degree of convergence or divergence of the input neighborhood to the corresponding output neighborhood through the processing of each neuron.

The author's group already proposed the above hypothesis and item 1 of the concept in [Shibata and Sakashita 2015]. It was also demonstrated that chaotic dynamics operate as exploration without stochastic selection in actor-critic [Shibata and Sakashita 2015], reward-modulated Hebbian learning [Matsuki and Shibata 2020], and TD3 [Matsuki et al. 2024] and that the “edge of chaos” was a favorable choice for network dynamics, leading to high learning performance in [Matsuki and Shibata 2020].

In Dynamic RL, an agent can explore using many DOFs because the motions with exploration are generated through its RNN and can be learned. The flexibility in exploration is expected to carry over directly into thinking through learning while maintaining chaos in the global dynamics. Furthermore, if the RNN architecture is hierarchical, learning is expected to control irregularity even in the abstract state space. In this way, the author expects that a Dynamic RL agent, learning through high-DOF exploration, will eventually be able to think about a variety of things, including abstract ones, and sometimes generate completely new ideas on its own by flexibly utilizing both the irregular and the rational.

Currently, to improve learning efficiency or stability, many excellent RL methods, such as PPO [Schulman et al. 2017], SAC [Haarnoja et al. 2018], A3C [Mnih et al. 2016], TD3 [Fujimoto et al. 2018], and experience replay with PER [Schaul et al. 2016] or HER [Andrychowicz et al. 2017] have been used. However, since Dynamic RL presents a newborn learning concept, comparison targets are limited to simple conventional RL for a fair evaluation of basic performance.

This paper first introduces Dynamic RL and then examines whether this entirely new type of RL functions effectively as reinforcement learning in a simple memory-required task and a dynamic pattern generation task, compared with conventional RL.

2 Learning Method

As a base architecture for RL, the actor-critic model is used, in which the actor outputs do not represent probabilities for actions but instead represent continuous motor commands. Dynamic RL is applied solely to the actor, while the critic is trained by conventional RL using BPTT [Rumelhart and McClelland 1986] although ideally, all learning should be dynamic. For clarity, each actor and critic consists of a separate RNN with sensor signals as inputs.

In each dynamic neuron, its internal state u at time t is derived as the first-order lag of the inner product of the connection weight vector $\mathbf{w} = (w_1, \dots, w_m)^T$ and input vector $\mathbf{x}_t = (x_{1t}, \dots, x_{mt})^T$ where m is the number of inputs as

$$u_t = \left(1 - \frac{\Delta t}{\tau}\right) u_{t-1} + \frac{\Delta t}{\tau} \mathbf{w} \cdot \mathbf{x}_t \quad (1)$$

where τ is a time constant and Δt is the step width, which is 1.0 in this paper. For static-type neurons, the internal state u is just the inner product as

$$u_t = \mathbf{w} \cdot \mathbf{x}_t. \quad (2)$$

The inputs \mathbf{x}_t can be the external inputs or the pre-synaptic neuron outputs at time t , but for the feedback connections, where the inputs come from the same or an upper layer, they are the outputs of the pre-synaptic neuron at time $t-1$. The output o_t is derived from the internal state u_t as

$$o_t = f(U_t) = f(u_t + \theta) \quad (3)$$

where $U_t = u_t + \theta$, θ is the bias, and $f(\cdot)$ is an activation function, which is a hyperbolic tangent in this paper.

Dynamic RL controls the dynamics of the system, including RNN, directly by adjusting the sensitivity [Shibata et al. 2021] in each neuron. Sensitivity is an index for each neuron that is the Euclidian norm of the output gradient with respect to the input vector \mathbf{x} . It is defined as

$$s(U; \mathbf{w}) = \|\nabla_{\mathbf{x}} o\| = f'(U) \|\mathbf{w}\|. \quad (4)$$

Here, $\|\mathbf{v}\| = \sqrt{\sum_i^m v_i^2}$ for a vector $\mathbf{v} = (v_1, \dots, v_m)^T$. In the form of a vector elements, the sensitivity is represented as

$$s(U; \mathbf{w}) = \sqrt{\sum_i^m \left(\frac{\partial o}{\partial x_i}\right)^2} = f'(U) \sqrt{\sum_i^m w_i^2}. \quad (5)$$

Sensitivity refers to the maximum ratio of the absolute value of the output deviation do to the magnitude of the infinitesimal variation $d\mathbf{x}$ in the input vector space. It represents the degree of contraction or expansion from the neighborhood around the current inputs to the corresponding neighborhood around the current output through the neuron's processing. In the previous work [Shibata et al. 2021], it was defined only for static-type neurons (Eq. (2)). In this study, the same definition is also applied to dynamic neurons (Eq. (1)), assuming that the infinitesimal variation $d\mathbf{x}$ of the input \mathbf{x} changes slowly enough compared to the time constant τ .

In the previous research [Shibata et al. 2021], the author's group proposed sensitivity adjustment learning (SAL). SAL was applied to ensure the sensitivity of each neuron in parallel with gradient-based supervised learning. This approach is beneficial not only for maintaining sensitivity during forward computation in the neural network but also for avoiding diverging or vanishing gradients during backward computation. Because Dynamic RL incorporates SAL and sensitivity-controlled RL (SRL), which is an extension of SAL for RL, SAL will be explained first.

In SAL, the moving average of sensitivity \bar{s} is computed first as

$$\bar{s}_t \leftarrow (1 - \alpha) \bar{s}_{t-1} + \alpha s_t \quad (6)$$

where α is a small constant, and this computation is performed across episodes. When the average sensitivity \bar{s} is below a predetermined constant s_{th} , the weights and bias in each neuron are updated locally to the gradient direction of the sensitivity as

$$\Delta \mathbf{w}_t = \eta_{SAL} \frac{\Delta t}{\tau} \nabla_{\mathbf{w}} s(U_t; \mathbf{w}) = \eta_{SRL} \frac{\Delta t}{\tau} \left(f'(U_t) \frac{\mathbf{w}}{\|\mathbf{w}\|} + \|\mathbf{w}\| \nabla_{\mathbf{w}} f'(U_t) \right) \quad (7)$$

$$\Delta \theta_t = \eta_{SAL} \frac{\Delta t}{\tau} \frac{\partial s(U_t; \mathbf{w})}{\partial \theta} = \eta_{SRL} \frac{\Delta t}{\tau} \|\mathbf{w}\| \frac{\partial f'(U_t)}{\partial \theta}. \quad (8)$$

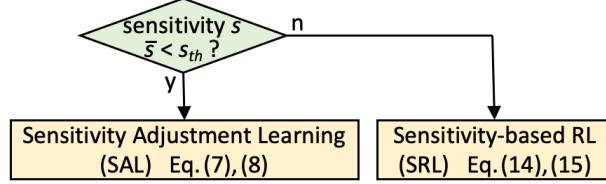


Figure 4: Dynamic RL applies either SAL or SRL depending on the condition in each neuron.

where η_{SAL} is the learning rate for SAL. $\Delta t / \tau$ is multiplied to adjust the update to the neuron's time scale. By expanding the equation with the activation function being hyperbolic tangent,

$$\Delta \mathbf{w}_t = \eta_{SAL} \frac{\Delta t}{\tau} (1 - o_t^2) \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} - 2o_t \|\mathbf{w}\| \mathbf{x}_t \right) \quad (9)$$

$$\Delta \theta_t = -2\eta_{SAL} \frac{\Delta t}{\tau} o_t (1 - o_t^2) \|\mathbf{w}\| \quad (10)$$

are derived.

In Dynamic RL proposed here, as shown in Fig.4, SAL is applied when the moving average of the sensitivity \bar{s} is less than a constant s_{th} , otherwise sensitivity-controlled RL (SRL) is applied in each neuron. SAL always tries to increase the sensitivity in each neuron, but whether SRL tries to increase or decrease the sensitivity depends on the temporal difference (TD) error \hat{r} as

$$\Delta \mathbf{w}_t = -\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t \nabla_{\mathbf{w}} s(U_t; \mathbf{w}) \quad (11)$$

$$\Delta \theta_t = -\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t \frac{\partial s(U_t; \mathbf{w})}{\partial \theta} \quad (12)$$

where η_{SRL} is the learning rate for SRL. TD error is computed as

$$\hat{r}_t = \gamma C_{t+1} + r_{t+1} - C_t = \gamma \left(C_{t+1} - \frac{C_t - r_{t+1}}{\gamma} \right) \quad (13)$$

where γ ($0.0 < \gamma < 1.0$) is the discount factor, C_t is the critic output (state value), and r_t is the reinforcement signal, which can be a reward or a penalty, at time t . As the basic concept summarized in Fig.5, when TD error is positive, in other words, the new critic (state value) C_{t+1} is greater than the expected value $\frac{C_t - r_{t+1}}{\gamma}$, RL reduces the sensitivity to reinforce the reproducibility. When it is negative, i.e., the new state value is less than expected, RL makes the sensitivity greater to reinforce the exploratory nature. This is expected to control the local convergence or divergence, depending on how good or bad the state is.

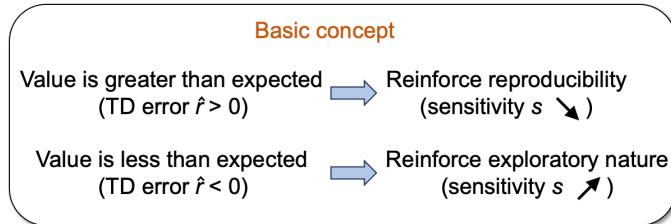


Figure 5: Basic concept of Dynamic RL (or more specifically, SRL) proposed in this paper.

Upon expansion, we obtain,

$$\Delta \mathbf{w}_t = -\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t \left(f'(U_t) \frac{\mathbf{w}}{\|\mathbf{w}\|} + \|\mathbf{w}\| \nabla_{\mathbf{w}} f'(U_t) \right) \quad (14)$$

$$\Delta \theta_t = -\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t \|\mathbf{w}\| \frac{\partial f'(U_t)}{\partial \theta}. \quad (15)$$

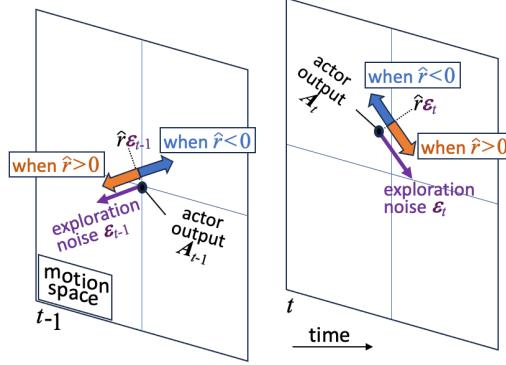


Figure 6: A conceptual diagram of conventional RL. RL aims to control the actor output vector based on the TD error. It does not utilize information about time changes in the RNN's state and is closed only at each step.

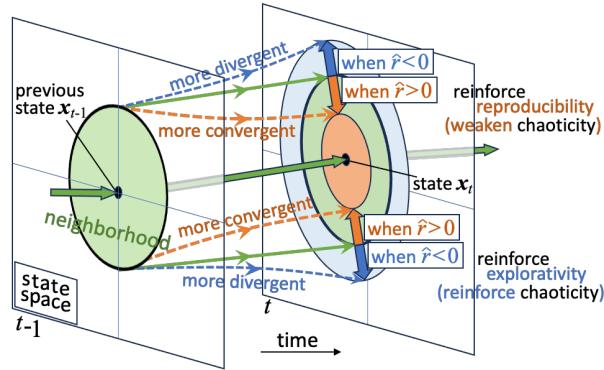


Figure 7: A conceptual diagram of Dynamic Reinforcement Learning (RL). RL aims to control the convergence or divergence of the flow around state transitions according to the TD error by controlling the sensitivity in each neuron.

By further expanding as the activation function $f(\cdot)$ being \tanh ,

$$\Delta \mathbf{w}_t = -\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t (1 - o_t^2) \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} - 2o_t \|\mathbf{w}\| \mathbf{x}_t \right) \quad (16)$$

$$\Delta \theta_t = 2\eta_{SRL} \frac{\Delta t}{\tau} \hat{r}_t o_t (1 - o_t^2) \|\mathbf{w}\|. \quad (17)$$

Notably, this computation can be done locally in each neuron except for receiving the TD errors. Furthermore, since the dynamics are generated not only by the loops inside the RNN but also influenced by the loops that are formed with the outside world, this learning can be applied to all the neurons, including those outside the loop in the RNN, including the output neurons.

In the following simulations, the proposed RL is compared to the conventional RL using BPTT. Now many techniques have been proposed to improve the performance, but for a pure comparison of base methods, simple learning using gradient-based BPTT is employed. In Dynamic RL, the motor command vector \mathbf{M}_t is a function $\mathbf{M}(\cdot)$ of the actor output vector \mathbf{A}_t as $\mathbf{M}_t = \mathbf{M}(\mathbf{A}_t)$, but in the conventional RL, since a random noise vector ϵ_t is added to the actor output vector as explorations, the actual motor command vector \mathbf{M}_t is expressed as

$$\mathbf{M}_t = \mathbf{M}(\mathbf{A}_t + \epsilon_t) \quad (18)$$

For conventional RL, training signals for the actor network are derived as

$$\mathbf{A}_{train,t} = \mathbf{A}_t + \hat{r}_t \epsilon_t. \quad (19)$$

Then, the actor network is trained based on the BPTT method by these training signals. In this paper, it learned 10 or 20 steps backward in time, depending on the task. While, in the Dynamic RL, since no calculation going back through time is necessary, the computational cost is considerably smaller than in the case of conventional RL.

Dynamic RL has a significant difference in the way of learning from conventional RL. For better understanding, the author attempts to illustrate their differences with diagrams at the expense of accuracy. In the conventional RL, external

noise ϵ is added to the actor output vector \mathbf{A} . As shown in Fig. 6, according to Eq. (19), if the value function is better than expected, i.e., if the TD error \hat{r} is positive, the network is trained to move the output vector \mathbf{A} to the direction of the noise ϵ . By contrast, if the TD error \hat{r} is negative, the network is trained to move the output vector \mathbf{A} to the opposite direction. This RL does not use the temporal change in the outputs or network states; rather, it considers only the outputs at that moment in time. All the weights and biases are updated to move the output vector with the help of the gradient method using error backpropagation even through time.

On the other hand, Dynamic RL does not aim to move the state or output directly, but as shown in Fig. 7, it aims to control the convergence or divergence of the neighborhood around the state transition by changing each neuron's sensitivity depending on the TD error. The concept of controlling dynamics can also be applied to other types of learning, such as supervised learning. The author refers to it as Dynamic Learning from a broader perspective and will discuss it in the subsection 4.2.

This concept should also be introduced to the critic network, but here, conventional learning is used for the critic, regardless of how the actor network is trained. The training signal is derived as

$$C_{train,t} = \gamma C_{t+1} + r_{t+1} = C_t + \hat{r}_t, \quad (20)$$

and the critic network is always trained with BPTT using this training signal.

In Dynamic RL, the network outputs were often saturated (close to 1.0 or -1.0 in hyperbolic tangent), and it is difficult to perform fine and smooth control. To avoid saturation, the regularization was applied only to the output layer's connection weights in the actor network as

$$\Delta \mathbf{W} = -\eta_{reg} \mathbf{W}. \quad (21)$$

This learning was applied in both Dynamic and conventional RL cases.

Furthermore, one more technique used in this paper is "critic raising". When an agent cannot reach its goal for a long time, since the critic output becomes small, the gradient of the critic also becomes small. Therefore, referring to the "optimistic initial value" [Sutton and Barto 2018], when the moving average \bar{C} of the critic output C is less than a constant C_{th} , the bias of the output layer in the critic network is increased to raise the critic value as

$$\Delta \theta_t = \eta_{raise}(C_{th} - \bar{C}_t) \quad (22)$$

where \bar{C} is the moving average of the critic output C as

$$\bar{C}_t \leftarrow (1 - \beta)\bar{C}_{t-1} + \beta C_t \quad (23)$$

where β is a small constant, and this computation is performed across episodes.

3 Simulations

To examine the learning ability of Dynamic RL, the author applied it to two dynamic tasks: a sequential navigation task, which served as a memory-required task, and a slider-crank control task, which served as a dynamic pattern generation task.

The network architecture used is shown in the left figure in Fig. 8, regardless of the learning method or task, although the parameters used are different. The architecture consists of an actor network and a critic network; as mentioned, Dynamic RL was applied only to the actor network here. The critic network was trained in the same manner as conventional RL. The critic network was an Elman-type RNN with a single hidden layer comprising 50 neurons, having feedback connections to itself. It was trained to minimize the square of the TD-error using BPTT with the training signal as in Eq. (20).

The actor network used a multiple timescale RNN (MTRNN) [Yamashita and Tani 2010] to improve performance and verify whether the proposed RL would function on a more complex architecture than simple Elman-type RNNs. The MTRNN comprised two hidden layers with different time constants and one output layer. The two hidden layers were interconnected, and one of them, called upper hidden layer (100 neurons), did not have any connection to the input or output layer; however, it had self-feedback connections. Although the other hidden layer, called lower hidden layer (200 neurons), had connections with the input and output layers, it did not have self-feedback connections. Neurons in the upper hidden layer in the actor network were dynamic, while the rest, including all neurons in the critic network, were static (time constant $\tau = 1$). All connected layers are fully connected in both networks.

In the proposed Dynamic RL, either SRL or SAL was applied to each hidden neuron depending on its average sensitivity \bar{s} as shown in Fig. 4; meanwhile, SAL was not applied to each output neuron to avoid the risk that SAL would cause the output to deviate from the ideal value. To see the difference from the conventional RL and also how SRL and SAL worked respectively, six learning conditions varying the method of applying learning were compared in the following as in Table 1.

Table 1: Six learning conditions varying the application methods of learning for the actor network compared in the following simulations. (DYN_RL: Dynamic RL, Conv_RL: conventional RL, Reg: regularization (Eq. (21)), SRL & SAL: SRL (Eqs. (16)(17)) and SAL (Eqs. (9)(10)), but actually one of them is applied depending on the condition as in Fig. 4 at each step in each neuron, BPTT: Backpropagation Through Time, BP: Backpropagation not going back in time. In convenience, the notation ‘BP’ is used also for the output layer where no propagation of error signal is necessary. The training signals for ‘BPTT’ or ‘BP’ are as in Eq. (19).)

Name	combination(in: input, hid: hidden, out: output)		
	hid → out	hid → hid	in → hid
(A) DYN_RL (proposed)	SRL & Reg	SRL & SAL	SRL & SAL
(A-1) DYN_RL (No SRL for hid→hid)	SRL & Reg	SAL	SRL & SAL
(A-2) DYN_RL (No SAL)	SRL & Reg	SRL	SRL
(B) Conv_RL (BPTT)	BP & Reg	BPTT	BPTT
(B-1) Conv_RL (BPTT+ SAL)	BP & Reg	BPTT & SAL	BPTT & SAL
(B-2) Conv_RL (BP+SAL)	BP & Reg	BP & SAL	BP & SAL

3.1 Sequential navigation (memory-required) task

In this simulation, as shown in the right figure of Fig. 8, there is a 20×20 size walled field centered at the origin. An agent starts each episode at a randomly chosen location weighted on the peripheral area of the field to accelerate learning (A). The agent received a reward of $r = 0.8$ (r : reinforcement signal) when the center of the agent reached the final goal at $(0.0, -3.0)$ after reaching the subgoal at $(0.0, 4.0)$. At every step when the agent center is on the final goal before reaching the subgoal, it receives a small penalty of $r = -0.03$. The shape of each goal (final goal or subgoal) is a circle with a radius of 2.0, and when the center of the agent comes within the circle, the agent is considered to have reached the goal. There is a visual sensor with $11 \times 11 = 121$ visual cells, and only the agent appears on it. Each sensor cell covers a 1×1 square, which is identical to the agent’s size. The agent cannot know the location of each goal from the sensor signals. The agent also receives one more sensor signal. Only when the agent’s center is within the subgoal, it is 2.0; otherwise, it is 0.0. Therefore, the agent must memorize whether it has already reached the subgoal or not in the RNN and switch its behavior according to the memory.

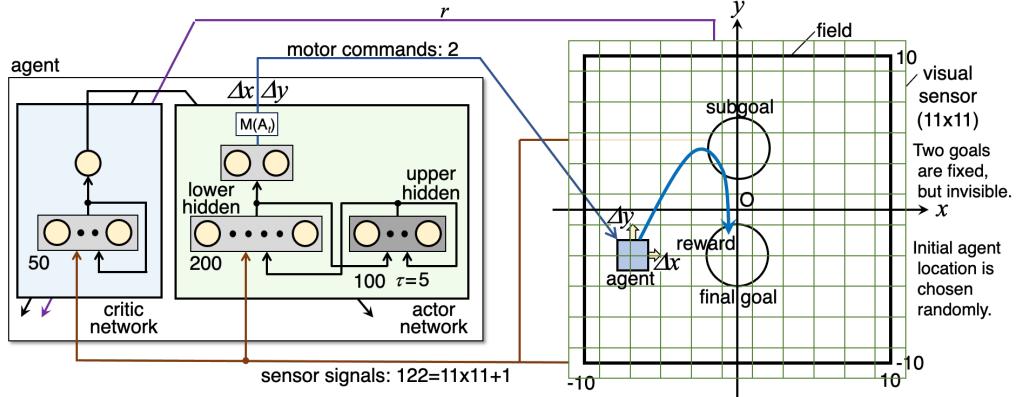


Figure 8: Sequential navigation task and the RNN used in the simulation.

The agent receives 122 sensor signals in total and inputs them to both the critic and actor networks. The actor network has two outputs, which are used as the agent’s movement vector after normalizing to make the movable area circular, as described in A at each step. In each episode, the agent resets all the internal states of the two RNNs to 0.0 at first, and for preparation, it computes the RNNs for the first five steps without moving or counting steps. When the agent reaches a wall around the field, it stops there, but no penalty is given. The episode terminates either when the agent receives a reward or after 200 steps have passed. At each step, the agent updates all the weights and biases in the critic

Table 2: Parameters used in the sequential navigation (memory-required) task. (MA: Moving Average)

Number of Neurons (input-hidden[lower-upper]-output)	122-50-1 (critic) 122-[200-100]-2 (actor)
Spectral Radius of Initial Self-feedback Connection Weight Matrix	1.3 (critic, actor(Conv_RL)) 3.0 (actor(DYN_RL))
Neuron Model	Dynamic with $\tau = 5$ (upper hidden layer in actor net.) Static (others)
Target Sensitivity for SAL s_{th} in Fig. 4	1.3
Discount Factor γ in Eqs. (13),(20)	0.98
Rate of Regulation η_{reg} in Eq. (21)	$1e^{-6}$
Raise Critic Criterion C_{th} in Eq. (22)	0.1
Rate of Raising Critic η_{raise} in Eq. (22)	0.0005
Const. α for MA of Sensitivities in Eq. (6)	0.001
Const. β for MA of Critic in Eq. (23)	0.0001
Truncated Number of Steps in BPTT	20
Initial Weights and Biases, Learning Rate	Refer to B

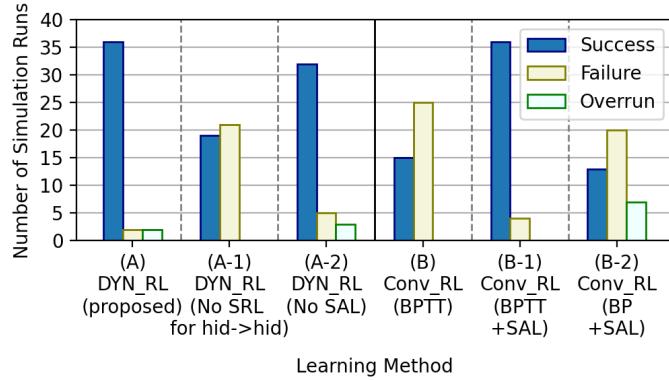


Figure 9: Comparison of success rate among six learning conditions as in Table 1 in the sequential navigation task.

network using the training signal as Eq. (20), and those in the actor network according to Eqs. (16)(17)(9)(10)(21). When it received a reward, the critic at the next step P_{t+1} was set to 0.0. The main used parameters are shown in Table 2. In Dynamic RL, the spectral radius of the initial self-feedback weight matrix of the upper hidden layer in the actor network was as large as 3.0 to ensure that the system, including actor network, generates chaotic dynamics, and the agent can explore the field. It was 1.3 in the conventional RL from the viewpoint of appropriate error propagation.

In one simulation run, the agent learned for 20,000 episodes first and then for another 20,000 episodes after swapping the roles of the actor's two outputs to observe its adaptability to environmental changes. For each learning condition, a total of 40 simulation runs with different random sequences were performed. The sequences were used to determine the initial connection weights, biases, and also the initial agent location at each episode. If the reduction in the average number of steps to the final goal was too small despite some progress in learning, the simulation run was treated as a “failure” and terminated midway through the learning process (refer to A for details). After each of the 20,000 and 40,000 episodes, a test was performed. The agent started from 46 initial locations, arranged in a grid pattern with a 3.0 unit interval, ranging from -9.0 to 9.0 on the field, excluding the two goal areas, and did not learn. The random noise for exploration was not added to the actor outputs even in the case of conventional RL. If the average number of steps in either test exceeded 20, the simulation run was deemed “overrun.” A simulation run was considered a “success” when it was neither a “failure” nor “overrun.”

First, Fig. 9 compares the success rates from 40 simulation runs across the six learning conditions listed in Table 1. The (A) Dynamic RL proposed in this paper achieved 36 successes, 2 failures, and 2 overruns out of the 40 runs. In the two failure runs, learning failed before the environmental change, and no failure occurred afterward. In the two overruns, the average number of steps during learning was consistently below 20.0 just before the test; however, due to some instability in Dynamic RL, it occasionally exceeded 20 steps in the test.

The success rate for (B) conventional RL using BPTT is only about 40%, which is lower than for (A) Dynamic RL. However, in the case of (B-1), where SAL was applied to the hidden neurons in parallel with conventional RL, the results improved to 36-4-0, making them comparable to those of case (A) Dynamic RL. Therefore, the case (B-1) will serve as the main comparison target for Dynamic RL hereafter. When the error was not propagated to the past in the case of (B-2), the performance was worse than in the case of (B-1).

Meanwhile, comparing (A), (A-1), and (A-2) suggests that both SRL and SAL are necessary for good performance in Dynamic RL. Different from the case of the chaos-based RL method from the previous work [Shibata and Sakashita 2015], applying SRL to the weights between hidden neurons clearly improved the results.

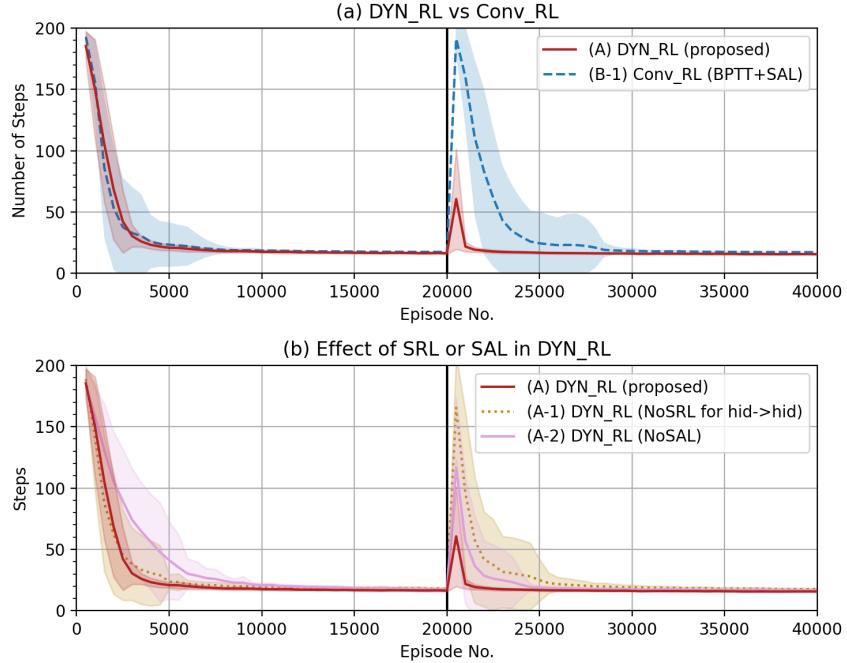


Figure 10: Learning curve for the sequential navigation task. The vertical axis indicates the average number of steps necessary to reach the final goal successfully over 500 episodes. The plots and shaded areas indicate the mean and standard deviation over successful simulation runs. (a) shows the comparison result between Dynamic RL and conventional RL. (b) shows the comparison result among three cases to assess the effect of SRL or SAL.

Figure 10 (a) compares the learning curves of Dynamic RL and conventional RL. The vertical axis shows the average number of steps needed to reach the final goal, calculated every 500 episodes. The plot lines and shaded areas in the figure represent the mean and standard deviation across all successful simulation runs. When the agent could not reach the final goal within 200 steps, it was recorded as 200 steps. It was observed that in all cases, the number of steps decreased as learning progressed. After the environment changed just after the 20,000th episode, the number of steps increased once but then decreased again. The large standard deviation in conventional RL occurs because, depending on the simulation run, the number of steps did not decrease from around 200 for some episodes. The learning curve for (B) conventional RL without applying SAL was similar to that for (B-1) with SAL, but slightly worse overall, although no figure is presented here.

Thus, Dynamic RL functions effectively even in a task where the agent must learn to find necessary information, memorize it, and reflect it in its motions for appropriate future behaviors. Dynamic RL does not require backward computation through time. The actual computational cost for learning the actor network, compared to conventional RL (BPTT+SAL), was 1:8.0 when using the author's Python and NumPy program without using a framework or a GPU. Nevertheless, the curve before the environmental change was similar to that of conventional RL. Moreover,

learning after the environmental change was significantly faster in Dynamic RL than in conventional RL. In Dynamic RL, learning was faster after the environmental change compared to the beginning of the run; whereas in conventional RL, it was slower.

Figure 10 (b) illustrates the learning curves for three cases in Dynamic RL: (A) regular case, (A-1) without SRL for hidden-to-hidden connections, and (A-2) without SAL, to evaluate the effects of SRL and SAL. Note that even in the case of (A-1), SRL was still applied to the other connection weights. Case (A) outperforms both (A-1) and (A-2), leading to the conclusion that both SRL and SAL are beneficial. SAL seems to be generally more effective for fast learning, while application of SRL to the connection weights between hidden neurons is particularly effective when the environment changes.

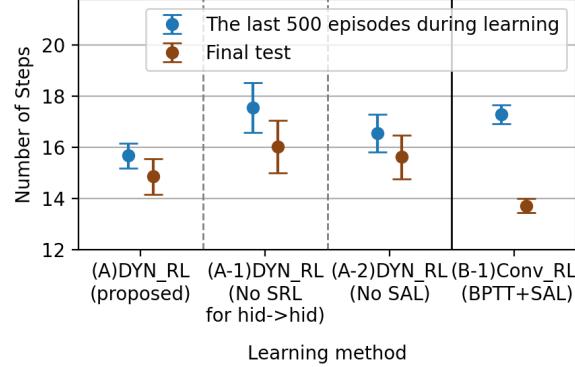


Figure 11: Comparison of the average number of steps in the final stage of learning and in the test among four learning conditions. The plots and the error bars indicate the mean and standard deviation over successful simulation runs.

Figure 11 compares the average number of steps in the last 500 learning episodes to that in the final test, which consists of 46 episodes after learning. This comparison is made across the four learning conditions presented in Fig. 10. In all cases, the number of steps in the test is lower than during learning. This difference is due to the frequent starting positions at the periphery during learning. In (B-1) conventional RL with SAL, the number of steps in the test is significantly lower than during learning because no exploration noise was introduced in the test.

In case (A) Dynamic RL, the agent's behavior became less exploratory through learning, leading to a smaller average number of steps during the final stage of learning compared to (B-1) conventional RL. However, the learning process in Dynamic RL was somewhat unstable, sometimes causing sudden and significant increases in the number of steps, which in turn increased both the average and standard deviation to that extent. In the test, exploration was still present, and the average is larger than in conventional RL. Additionally, in Dynamic RL, the absence of SAL or SRL in (A-1) or (A-2) resulted in an increase in the average and standard deviation.

Figure 12 shows sample trajectories from a simulation run that successfully learned for both (A) Dynamic RL and (B-1) conventional RL+SAL cases. Since the number of steps varies largely among simulation runs in Dynamic RL, as shown in Fig. 11, the results for an average run are presented, where the number of steps ranks 24th in the first test and 11th in the second test out of 36 successful simulation runs for (A) Dynamic RL. In (A) and (B-1), the cases for the same random sequences are shown. The initial weights and biases in the RNNs, as well as the agent's starting position for each episode, were the same, except for the weight matrix of the output layer and the spectral radius of the upper hidden layer's self-feedback connection weight matrix in the actor network.

Figures 12(A)(a) and 12(B-1)(a) show the agent paths in the first ten episodes. The agent explored the entire field in both cases, but the exploration appeared different. In (A) Dynamic RL, the paths were smoother due to the chaotic dynamics generated internally with the large time constant in the upper hidden layer. In (B-1) conventional RL with BPTT+SAL, they were jagged and irregular due to the random noise introduced for exploration at each step.

Figures 12(A)(b) and 12(B-1)(b) show the results of the simplified test with nine initial locations after 20,000 episodes of learning, just before the environmental change. In this test, random exploration was stopped even in the conventional RL case. Starting from any initial location, the agent first reaches the subgoal and then the final goal in both cases. Dynamic RL could learn memory-required behaviors without using backward computation, such as BPTT. However, the agent's paths in (A) were not as smooth as in the case of (B-1) conventional RL. In (A) Dynamic RL, the actor outputs were easily saturated and the agent was apt to move diagonally although regularization was applied to avoid it. This problem needs to be solved.

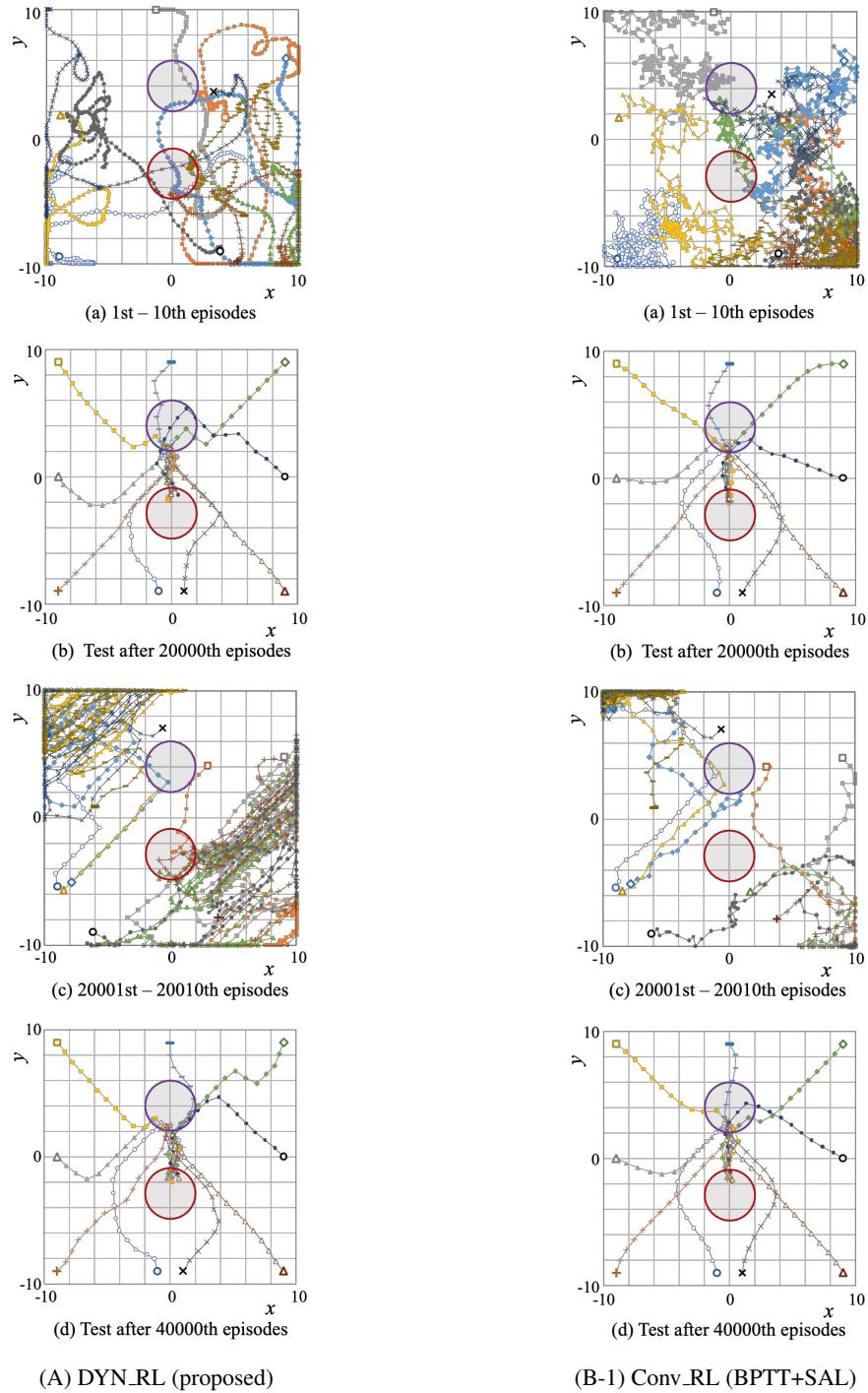


Figure 12: Comparison of sample agent behaviors at four stages in learning between (A) Dynamic RL and (B-1) conventional RL (with BPTT and SAL).

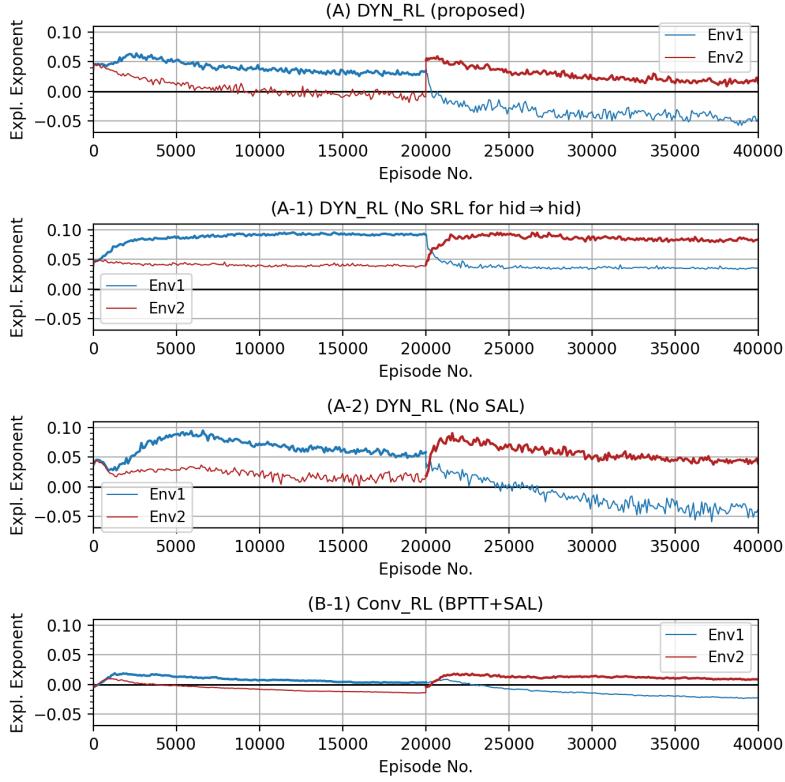


Figure 13: Comparison of changes in exploration exponents in the sequential navigation task among four learning conditions. Refer to the text for the definition of the exploration exponent. Env1 and Env2 are the learning environments before and after the environmental change, respectively. Thick lines indicate an exponent in the learning environment: Env1 before environmental change and Env2 after that.

Figures 12(A)(c) and 12(B-1)(c) show the actual agent paths for ten episodes just after the environmental change, in which two actor outputs were swapped. For instance, if the agent is located in the lower right direction of the subgoal and intends to move toward the upper left direction, it actually moves to the lower right, which is opposite to its intention. Consequently, the agent tends to gravitate toward the bottom right or top left corner. Therefore, in (B-1) conventional RL with BPTT+SAL, the agent's final position was in the area defined by $|x|, |y| > 9.0$ in 8 out of 10 episodes. However, in (A) Dynamic RL, the agent's paths appear more exploratory, and its final position was never in this area across the 10 episodes. When the agent was not allowed to learn after the environmental change, it moved to either the bottom right or top left corner from the starting position and remained there, although no figure is presented here. This suggests that Dynamic RL promotes exploration. It can be inferred that the difference in exploration between the two RLs leads to variations in learning speed after the environmental change. Figures 12(A)(d) and 12(B-1)(d) show the final simplified test results after 40,000 episodes, i.e., after all learning was completed. As in the case of (b), the agent regained the ability to reach the final goal in both cases, and the paths were smoother, resulting in fewer steps in conventional RL.

Finally, the changes in dynamic characteristics during learning are compared. Here, the exploration exponent is introduced, which is a type of Lyapunov exponent for the entire system, including both the RNN and the physical system (See C for its calculation). During learning, every 100 episodes, the exponent was computed for each of the nine starting points, which are the same as in Fig. 12 (b) or (d), and the nine exponents were averaged. It is calculated for each of the two environments that appeared before and after the environmental change, referred to as Env1 and Env2, respectively.

Although the exponent fluctuated significantly, the average over successful simulation runs reveals the characteristics of each learning condition, as shown in Fig. 13. The four graphs show how the exploration exponent changes as learning progresses under the four learning conditions. In (A), (A-1), and (A-2), the starting value is larger than in (B-1) due to the differences in the output connection weights and the spectral radius of the self-feedback connection weights in the upper hidden layer.

In the proposed Dynamic RL (A), the exponent in the learning environment first increased and then gradually decreased while maintaining the value in the learning environment (Env1 for the first half of the learning episodes and Env2 for the second half) greater than that in the non-learning environment. When SRL was not applied to the hidden-to-hidden connections in (A-1), the exponent did not decrease as learning progressed. This suggests that SRL facilitates convergence in the dynamics by reducing each neuron's sensitivity around the state transitions when the TD-error is positive. Conversely, when SAL was not applied in (A-2), the exponent initially decreased. This indicates that SAL prevents the system dynamics around the actual state transitions from converging excessively by increasing each neuron's sensitivity.

Moreover, after the environment changed, the exponent for the learning environment Env2 in (A) increased much faster than in the other cases. This aligns with the rapid adaptation in the learning curve observed in Fig. 10. The slow adaptation observed in (A-1) and (A-2) suggests that the quick increase of the exponent by both SRL and SAL was the key to quick learning after the environmental change. While the exponent for the new learning environment, Env2, increased, the exponent for Env1, which was newly transitioned into a non-learning environment, decreased. This indicates that the agent enhanced its exploratory behavior, specializing in the learning environment.

Meanwhile, in (B-1) BPTT with SAL, the trend in the exponent change was similar to that of (A), but the exponent started from a smaller value and stayed close to zero. This low exponent in the learning environment may have led to stable learning but slow adaptation after the environmental change.

3.2 Slider-Crank control (Dynamic pattern generation) task

To evaluate whether the proposed Dynamic RL works in a task where a learning agent is required to generate dynamic patterns, a slider-crank control task, as shown in Fig. 14, was employed as the next learning task. In this task, an agent learns to apply an appropriate time-varying force f in the x -direction to the slider. This force is then transmitted to the rotor through the connecting rod, causing the rotor to rotate around its central axis O.

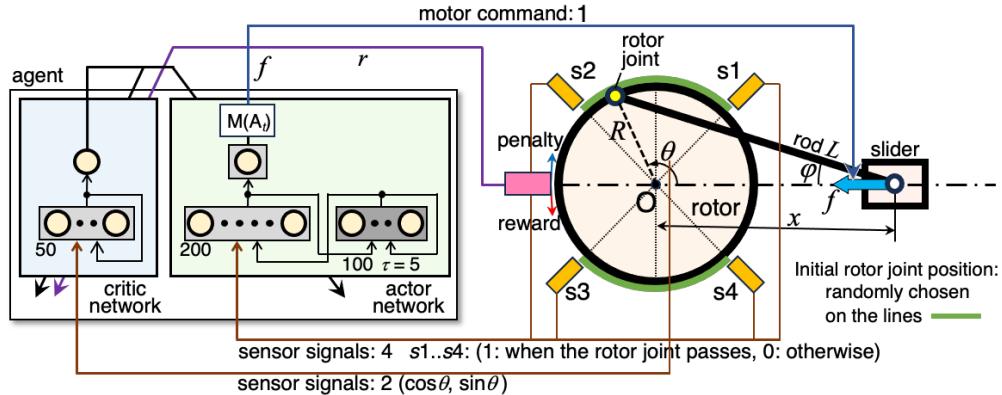


Figure 14: Slider-crank control task as a dynamic pattern generation task and the RNN used. The agent applies a force f to the slider in the direction of the thick blue arrow (which can be positive or negative), causing the rotor to rotate. There are four joint-detecting sensors (s_1, \dots, s_4), and only the four binary signals from them are the input to the actor network. The initial rotor joint angle θ_0 is randomly chosen within the ranges indicated by the two green arcs at the beginning of each episode.

The actor network receives only four binary sensor signals: s_1 to s_4 . Each signal is 1.0 only when the rotor angle θ passes through a specific angle: $\pi/4, 3\pi/4, -\pi/4$, or $-3\pi/4$ radian respectively, regardless of the rotational direction. The actor network receives the four signals and outputs only one motor command representing the force f applied to the slider in the x -direction. Therefore, the agent cannot always perceive the rotational angle θ or the slider's location x , nor can it directly perceive the rotational direction or speed. The agent has to generate a force sequence that is not defined as a static function of its four inputs. The actor output is modified by multiplying it by 1.25 and clipping it between -1.0 and 1.0 to convert it into a motor command (force f). This is expected to prevent the output from entering the saturation range of the neuron's activation function. The uniform disk-type rotor, with mass, makes this system dynamic while considering the negligible masses of the slider and link. The equations of motion for this system are provided in D,

In each episode, the initial rotation angle θ_0 is set randomly to an angle from $-3\pi/4 \leq \theta_0 \leq -\pi/4$ or $\pi/4 \leq \theta_0 \leq 3\pi/4$ radian. The initial rotational speed is 0.0. Since the agent does not perceive the joint location directly, it cannot determine the current rotational direction or identify the necessary direction of force to rotate the rotor in the rewarded direction until it receives the first “on” signal. Since Dynamic RL was not applied to the critic network, it was designed to receive $\cos\theta$ and $\sin\theta$ as sensor inputs, which are sufficient for identifying the rotor angle, for simplicity. When the angle passes through π with a positive rotational speed, the agent receives a reward of 0.15, and when with a negative rotational speed, it receives a penalty of -0.15. In each episode, the agent performed 400 steps of force loading, and the episode did not end even though the agent received a reward or penalty halfway through. In one simulation run, 10,000 episodes of learning were conducted. After 5,000 episodes, the reward sign was reversed as an environmental change. Tests were performed after 5,000 and 10,000 episodes. In each test, the rotor started from each of 50 initial angles at intervals of 0.02π , and the average revolutions per episode were calculated. Learning is defined as a “failure” if the average is less than 20 in the first test or more than -20 in the final test; otherwise, it is defined as a “success.” As well as the previous task, 40 simulation runs were performed for each learning condition. The parameters used here were almost the same as in the previous task, but some of them are different as in Table 3.

Table 3: Parameters used in the dynamic pattern generation (slider-crank control) task, excluding the same ones as in the previous task. For the excluded ones, refer to Table 2.

Target Sensitivity for SAL s_{th} in Fig. 4	1.6
Rate of Regulation η_{reg} in Eq. (21)	$1e^{-7}$
Rate of Raising Critic η_{raise} in Eq. (22)	0.0002
Truncated Number of Steps in BPTT	10

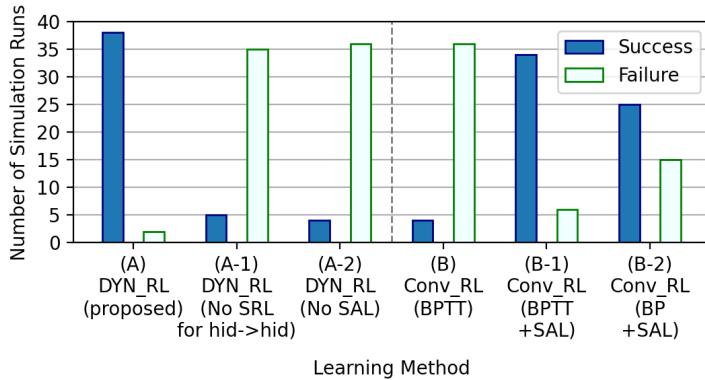


Figure 15: Comparison of success rate among six learning conditions in Table 1 in the slider-crank control task.

Figure 15 shows the counts of “success” and “failure” runs for each learning condition. As shown, (A) Dynamic RL achieved the highest success rate, while (B-1) conventional RL (BPTT + SAL) had the second-highest rate. In contrast, (B-2), which did not use backward error propagation through time, still achieved a high success rate. In the other cases, including the case (B) where SAL was not applied in conventional RL, the success rate was notably low. This indicates that both SRL and SAL are essential in Dynamic RL, and a method to ensure sensitivity, like SAL, is also necessary in conventional RL.

Figure 16 shows the change in the number of revolutions per episode for the three cases with a high success rate, as shown in Fig. 15. When the proposed Dynamic RL (DYN_RL) was applied, the rotational speed gradually increased, and the average number of revolutions per episode reached approximately 30. Learning progressed slightly faster compared to the case of conventional RL using BPTT, despite not needing backward computation through time. The ratio of the actual computational cost for learning the actor network, compared to conventional RL (BPTT+SAL), was 1:4.9. The difference from the previous task is mainly due to the difference in the truncated time steps in BPTT: 10 in the sequential navigation task and 5 in this task.

Shortly after the rotational direction for the reward was reversed as the environmental change after 5,000 episodes, the number of revolutions decreased, and the average over the 5,100-5,200 episodes was already negative. As in the

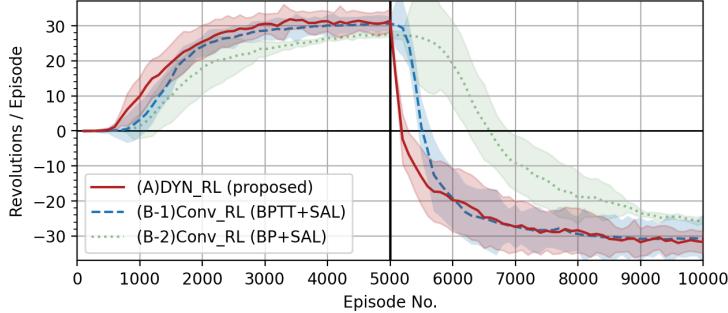


Figure 16: Comparison of changes in the average number of revolutions per episode, calculated over every 100 episodes, as learning progresses among three learning conditions in the slider-crank control task. Each line and shaded area indicate the mean and standard deviation of the number over successful simulation runs, respectively.

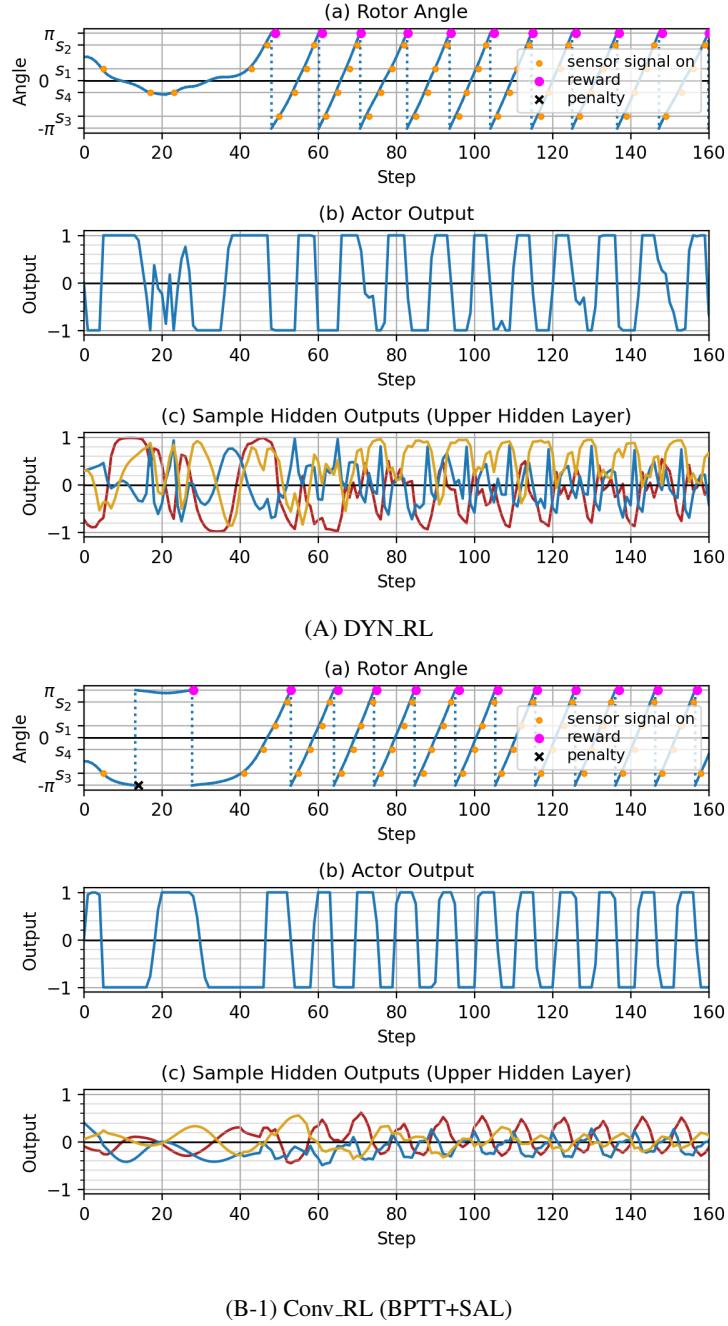
previous task, the learning speed immediately after the environmental change was considerably faster than in the case of conventional RL. In conventional RL with only BP, the success rate was reasonable, but the learning speed and performance after learning were much lower compared to the case with BPTT.

Figure 17 illustrates the changes in several states during a sample test episode just before the environmental change for both cases: (A) Dynamic RL and (B-1) conventional RL using BPTT and SAL. Specifically, the states include (a) rotor angle with sensor signals and reinforcement signal, (b) actor output, and (c) three sample neuron outputs from the upper hidden layer. The figure depicts the case in which the initial rotational speed caused by the agent's force load was negative, i.e., opposite to the rewarded direction, for one of the two initial angles: $-\pi/2$ or $\pi/2$. Therefore, the agent had to change its direction after detecting an "on" sensor signal. From (a), it can be observed that the agent changed the rotational direction in both cases and then maintained it thereafter. After the change, the outputs of the actor and hidden neurons changed almost periodically, as shown in (b) and (c). In the case of Dynamic RL, the outputs changed more irregularly than in conventional RL. This observation was made during the test phase; however during learning, external random noise for exploration was added to the actor outputs in the case of conventional RL. Another difference is that the hidden neuron outputs in Dynamic RL have a larger variation range, though the underlying reasons or influences have yet to be investigated.

Figure 18 shows the changes in states in two phases after the environmental change in the case of Dynamic RL. Figure 18(A-I) shows the state just after 5,100 episodes, shortly after the environmental change, while Fig. 18(A-II) depicts the state in the test phase after all learning was completed. In both cases, the initial rotor angle was the same as in Fig. 17(A). In Fig. 18(A-I), the actor output and hidden outputs changed irregularly, causing the rotor to rotate both clockwise and counter-clockwise irregularly. In contrast, in Fig. 18(A-II) where learning had progressed, the rotor rotated in the opposite direction compared to Fig. 17(A), and the actor output, along with the hidden outputs, changed almost periodically sometime after the start of the episode. However, the periodic pattern of the upper hidden neurons in Fig. 18(A-II) differs from that in Fig. 17(A), making it hard to explain the rapid adaptation after the environmental change in Dynamic RL based on the retention of internal periodic patterns.

4 Discussion and conclusion

In this paper, the author proposed Dynamic Reinforcement Learning (Dynamic RL), a new RL framework, expecting it to be a fundamental technique for the emergence of 'thinking.' This paper could not clearly show its critical advantages in this regard. However, the author demonstrated that despite the major shift in learning from static to dynamic, this unconventional RL approach effectively functions as RL in two simple dynamic tasks. Learning recurrent connection weights among hidden neurons without error backpropagation improved learning performance, which was not achievable with previous chaos-based RL [Shibata and Sakashita 2015]. The autonomous change in the state transitions from "more irregular" to "more rational" through learning observed in the simulations appears to have solved the balance between exploration and exploitation [Sutton and Barto 2018] and suggests the potential for the emergence of thinking. The learning characteristics of Dynamic RL, as observed in the simulation results of the two tasks, are summarized first, followed by a discussion of its possibilities and challenges from broader perspectives. Finally, the risks and concerns of Dynamic RL are discussed.



(B-1) Conv.RL (BPTT+SAL)

Figure 17: Comparison of changes in (a) rotor angle with four sensor signals and a reinforcement signal (reward or penalty), (b) actor output, and (c) outputs of three sample upper hidden neurons during a test episode after 5,000 episodes of learning between (A) Dynamic RL and (B-1) conventional RL (with BPTT and SAL) in the slider-crank control task. The initial rotor angle was different between (A) and (B-1) because these show the cases where the agent first rotated the rotor in the opposite direction to the rewarded one.

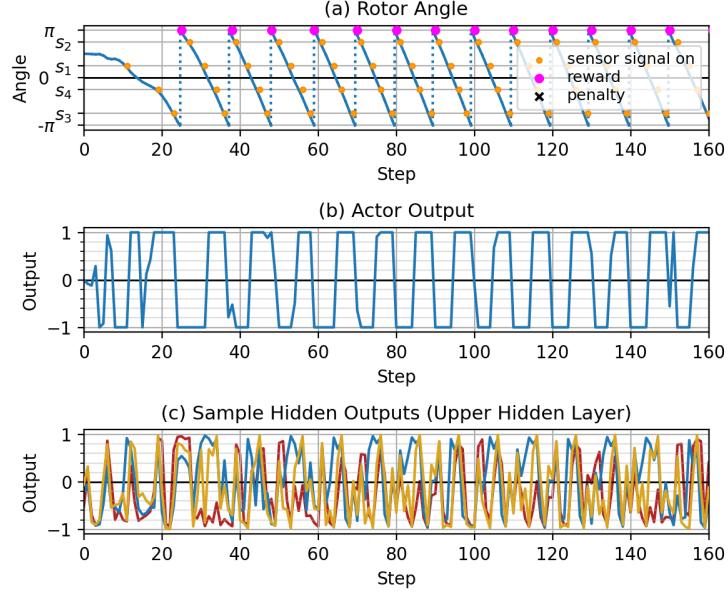
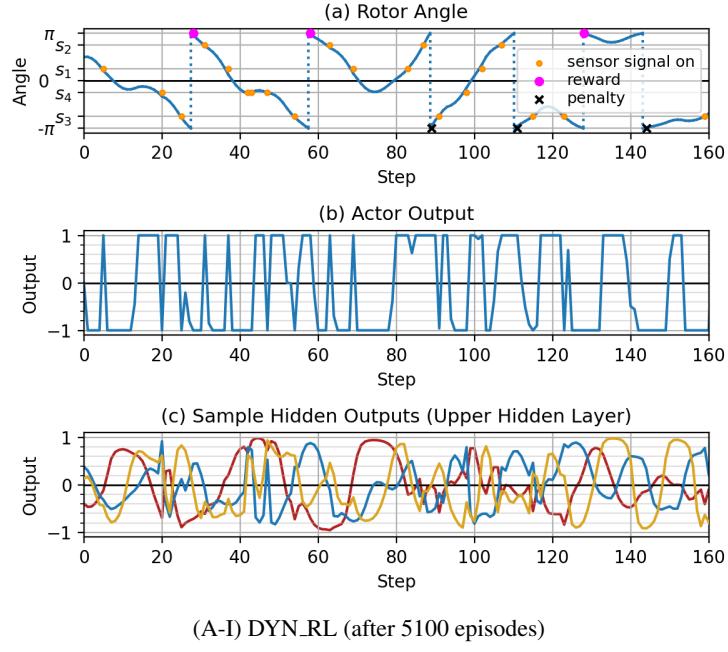


Figure 18: Changes in (a) rotor angle with four sensor signals and a reinforcement signal (reward or penalty), (b) actor output, and (c) outputs of three sample upper hidden neurons at two phases after the environmental change in the cases of (A) Dynamic RL. (A-I) represents the state soon after the environmental change, and (A-II) reflects the state after all learning had finished.

4.1 Summary as one method of Reinforcement Learning

Dynamic RL allows for significantly lower computational costs per step than conventional RL due to the absence of backward computation for learning, particularly along the time axis. Nevertheless, the success rate was almost the same as that of (A) and (B-1) in both Figs. 9 and 15, and the learning curve with episodes as the horizontal axis was also comparable to that of conventional RL using BPTT and SAL as (A) and (B-1) in both Figs. 10 and 16, regardless of the learning task. Furthermore, as for adapting to new environments, conversely, Dynamic RL was considerably faster. The agent's behaviors became more exploratory immediately after entering a new environment and less exploratory as learning progressed thereafter, as shown in Fig. 12(A). This was also reflected in the exploration exponent, a type of Lyapunov exponent for the entire system, including the external world, as shown in Fig. 13(A) in the sequential navigation task. In the slider-crank control task, it was reflected in the internal state change of the RNN, as shown in Figs. 17 and 18.

Comprehensively considering the results of varying the application of SAL (Sensitivity Adjustment Learning) or SRL (Sensitivity-Controlled RL), the author suggests that the following mechanisms are functioning. If neurons are linear, larger weights generally increase sensitivity. However, due to the saturation property of each non-linear neuron, excessively large weights often reduce sensitivity conversely by decreasing $f'(U)$ in Eq. (4). This generally decreases the Lyapunov exponent of the system and also the learning activities. Such problems occur when the initial weights are very large or often when another learning method (SRL in this case) progresses, as shown in the early phase of Fig. 13(A-2). SAL maintains the sensitivity of each neuron, thereby preserving the exploration exponent positive (chaotic) in the learning environment, as shown in Fig. 13(A). This prevents excessive convergence of the system and a decline in learning activity. Therefore, the absence of SAL leads to a decrease in both the learning success rate and learning speed, as shown in the difference between (A) and (A-2) in Figs. 9, 10, and 15. This is the case not only for Dynamic RL but also for BPTT, as shown in the difference between (B) and (B-1) in Figs. 9 and 15, which is compatible with the results in the previous work [Shibata et al. 2021].

As learning progresses, while avoiding over-convergence by using SAL, SRL adjusts the dynamics around the state transitions with positive TD-errors to be more convergent; thus, the state transitions become more reproducible. Conversely, it adjusts the dynamics around the state transitions with negative TD-errors to be more exploratory. Since the irregularity sometimes increases and sometimes decreases depending on the TD-errors, it seems that the irregularity remains unchanged in total. However, actually, since the frequency of better state transitions increases, the irregularity around actual state transitions decreases, as the exploration exponent in Fig. 13(A). Moreover, the state transitions that are adjusted to be more reproducible had positive TD-errors. Therefore, SRL makes the state transitions not only “less irregular” but also “more rational.”

After the environment changed during learning, the agent rapidly resumed exploration, and its performance quickly recovered in Dynamic RL. In the navigation task, the dynamics immediately before entering the new environment were neither chaotic nor exploratory for the new environment Env2, as the exploration exponent for Env2 was slightly below zero at 20,000 episodes in Fig. 13(A). Observing the agent's behavior from two slightly separated initial positions, the agent moved to the same corner of the field and remained trapped there in most cases. When the agent moved toward or was trapped in the corner, the TD-error was negative very frequently. In the slider-crank control task, due to the reversal of the reward's sign, large negative TD-errors frequently occurred. Accordingly, in both cases, it can be argued that SRL restored the strong chaotic nature of the system dynamics with the help of SAL and resumed exploratory behaviors, resulting in the quick recovery of performance.

For agents, the dynamics are shaped not only by the RNN but by the entire system integrated with the outside world, including the physical systems of the agents and their environment. Nevertheless, the gap is interesting where only adjusting or controlling the sensitivity of individual neurons locally by updating their weights and bias can globally control the system dynamics, thereby regulating the system and enabling the agent to respond effectively to the external world. In this paper, the sensitivity targets for SAL are set as appropriate values at 1.3 for the navigation task and 1.6 for the slider-crank control task. A value of 1.0 was appropriate for the sensitivity target in the preceding study investigating SAL in supervised learning with BP or BPTT [Shibata et al. 2021]. In reward-modulated Hebbian learning using chaotic dynamics, performance was optimized near the “edge of chaos” [Matsuki and Shibata 2020]. The author is considering the possibility that a large value may be suitable for RL's exploration, which requires stronger chaoticity.

For massively parallel and flexible systems, asynchronous pulse-and-analog coexisting systems would be far superior to a synchronous digital system. The introduction of Dynamic RL into brain models or brain-like hardware based on such systems offers significant advantages. Local learning without backward computation significantly reduces communication between neurons, as well as the computational cost in each neuron, compared to conventional RL using BPTT. In particular, eliminating backward computation through time is highly beneficial, as it removes the need for time management and memory to store past states. Furthermore, a separate random number generator for stochastic

selection is now unnecessary. A concern remains on how to build an RNN with connections that initially have large, random weights to make the system dynamics chaotic. The author expects SAL will enlarge small, irregular initial weights, as in the case of supervised learning in [Shibata et al. 2021]. The introduction of chaotic neurons [Aihara et al. 1990] would help it. In this paper, Dynamic RL is built as a discrete-time model to compute on a digital computer, but it would not be difficult to implement it on a continuous-time system.

However, even though we see Dynamic RL as regular RL, there are still some problems and concerns. First, learning was more unstable than conventional RL, showing occasional performance drops during learning. This resulted in larger deviations in learning performance in Dynamic RL than in conventional RL, as shown in Fig. 11.

The second issue is that the actor outputs are often saturated, making it difficult to fine-tune them. This can be seen, for example, in the agent’s tendency to move in the 45-degree diagonal direction in Fig. 12(A)(b,d) compared to the conventional RL case in Fig. 12(B)(b,d), though they were improved to some extent by regulation learning as in Eq. (21).

The third issue is that, in Dynamic RL, the exploration factor cannot be separated from the actor output. Therefore, as shown in Fig. 11, the performance in the test is worse than in conventional RL, where the actor outputs can be used without adding exploration noises. The fourth is that there are many parameters for learning, and adjusting them is not easy. In this sense, the author’s computational power (iMac 2020, Intel Core i9 3.6GHz) did not allow systematic and sufficient optimization in this study.

The author currently has several major challenges in mind, along with ideas for solving some of them, regarding this research. However, due to the risk discussed in subsection 4.3, he would rather refrain from writing about that.

4.2 Towards the future from a higher perspective

Even from the broader perspective of neural network (NN) learning not limited to RL, the Dynamic Learning approach used in Dynamic RL is still quite different in quality from conventional approaches. The state vector of an NN represents a point in its internal state space. In general NN learning, a gradient-based method or a derivative of it determines how to move this point in space based on some value or cost function, such as an error function, whose inputs are the NN outputs. Then, how to change the weights and biases is determined. Learning considers only the current point and usually does not consider its temporal changes. BPTT applies the same method to past states or points by extending the time axis as a dimension of space but still does not focus on their temporal changes or the system dynamics. While RNNs have enabled dynamic processing, their learning methods still remain stuck in static approaches.

However, since a state changes continuously over time, a point forms a line, and changes in the neighborhood around that line form flows in the state space. The point representing the current state is merely a cross-section for convenience of what is essentially a continuous line and generally cannot be separated from the system’s dynamics. In control systems, for instance, system stability is discussed based on convergence or divergence around the equilibrium point, and for better control, the controller often uses the time derivative of the error from the target trajectory.

Dynamic Learning transforms neural network learning from “learning of points” to “learning of flow,” in other words, from “static” to “dynamic.” Unlike control problems, where a target trajectory is specified, and stability is absolutely required, an agent’s behavior should be exploratory initially and then become less exploratory over time, and the system dynamics must not fully converge. Thus, as a concrete method for dynamic learning, the author arrived at the idea of controlling the convergence or divergence of the flow directly around the current transition, rather than moving points. He also arrived at the idea of controlling global dynamics through controlling dynamics locally in both space and time. The author believes he could demonstrate that these ideas work appropriately, at least in RL.

This shift in neural network learning is fundamental. However, using ideal divergence/convergence as supervised signals for learning dynamics does not seem very practical. Though it may be possible to compute and use ideal dynamics derived from ideal outputs, which are given as training signals, it remains unclear how much advantage this approach would have over direct learning with conventional static-type supervised learning. It may be mainly used in RL at last. However, in large-scale NNs, thanks to the magic of high-dimensional geometry, appropriate parameters (weights and biases) for a given cost function do not lie so far from their initial values [Amari 2020]. The author speculates that in large-scale RNNs, “learning of flow” along the time axis may become more significant and effective than “learning of point” on a time cross-section, though the scale of the RNN used in this study was not large enough to experience these benefits.

As noted in the Introduction, Dynamic RL brings about a big transformation to exploration. Exploration goes from stochastic to deterministic and no longer requires a random number generator. In Dynamic RL, exploration is not performed at the level of actions or final outputs but at the level of the entire process within the agent’s RNN. This

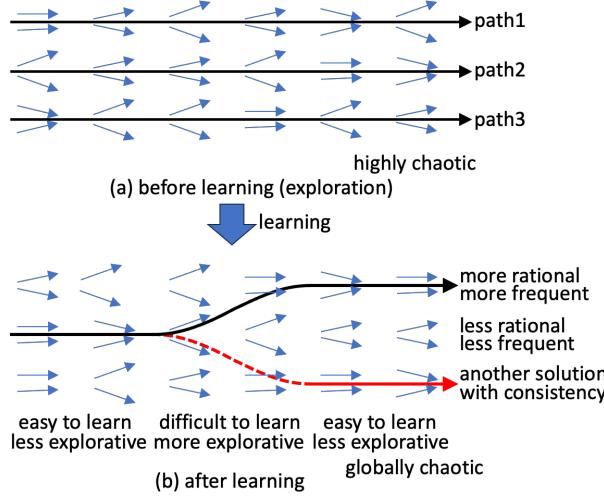


Figure 19: A conceptual diagram explaining how Dynamic RL works in simplified 2-D state space at the expense of exactness. The reproducibility of better state transitions increases, and novelty with consistency is expected to emerge as the red line after learning.

exploration is expected to be considerably more flexible because it is learnable and also has many DOFs rather than relying on stochastic selection with a scalar parameter called temperature. The excellent adaptability to a new environment, as seen in the simulation results, may be considered an advantage of such high-performance exploration. Furthermore, the author finally positioned exploration as a preliminary step to grow into ‘thinking.’

Here, let us consider how Dynamic RL with such exploration works in a simplified two-dimensional state space as an example, using a conceptual diagram, as shown in Fig. 19. Before learning, dynamics are set to be highly chaotic, and state transitions are overall explorative. Dynamics are learned around the actual paths depending on the TD-error at each state. Assuming that there are regions in which learning is easy and a region in which learning is not easy in the state space. In the former cases, by making the flows around rational state transitions with a positive TD-error converge, the flows gather around such rational state transitions. In the latter cases, the flows are still more explorative. If it occurs in a high-dimensional space, pseudo-attractors would emerge. Thus, as shown in Fig. 19(b), a rational state transition, indicated by a thick black line with an arrow at the end, is achieved. Furthermore, due to the explorative transitions in the region where learning is difficult, sometimes another solution emerges. However, the author expects the transitions not to become incoherent because the state transitions are rational in the regions where certainty in learning is high.

Next, the author summarizes how he thinks about ‘exploration’, ‘thinking,’ and their relationship to Dynamic RL, referring to Fig. 20. As mentioned, since both ‘exploration’ and ‘thinking’ need autonomous state transitions without being stuck in a convergent state, the system dynamics, including both the agent and its environment, should be chaotic. It ensures the agent’s activity in both behaviors and learning. SAL plays this role (right-pointing gray block arrows in Fig. 20). The agent explores at first with strong chaotic system dynamics. SRL enhances the reproducibility of the autonomous state transitions that are evaluated as better by the critic, and thus, the transitions and also the agent behaviors become more rational (diagonal block arrow). In this way, exploration grows into thinking, which is autonomous and more rational state transitions with weak chaotic system dynamics. This learning is slow because it takes time for the agent to explore rational state transitions, in other words, to acquire the knowledge needed for getting rewards and avoiding punishments.

When the situation changes, such as the environmental change in the learning tasks in this paper, the frequency or magnitude of worse evaluations, i.e., negative TD-errors, increases. This means that the state transitions become less rational (downward pointing block arrow). Then, SRL resumes exploration autonomously by making the system dynamics more chaotic (right-pointing large block arrow). This learning progresses rapidly because it only requires increasing the sensitivity of each neuron. The relationship between exploration and exploitation is quite similar to that between exploration and thinking. Exploitation and thinking are similar in that both are rational. Therefore, this mechanism can also explain how Dynamic RL resolves the “exploration and exploitation” dilemma and balances them depending on the situation.

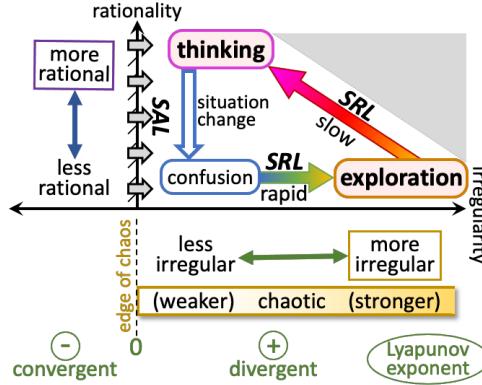


Figure 20: ‘Exploration’ and ‘thinking’ loop and its relationship to Dynamic RL, the author advocates. This is an updated version of Fig. 2 and was obtained through the analysis of the simulation results. SAL maintains chaotic dynamics of the system, thereby preventing a decline in learning activities. Under this condition, SRL changes the agent’s processing from “more irregular” to “more rational” gradually; in other words, it moves the processing from ‘exploration’ to ‘thinking’. When situations are changed, the state transitions become less rational for the agent, and negative TD-errors appear more frequently or with larger magnitudes. (labeled this as “confusion”) Then SRL makes the state transitions “more irregular” rapidly.

The brain, a massively parallel system, is too interconnected and complex for us to understand, and we cannot easily divide its processing into modules. For many years, the author has believed that our functions, such as recognition and prediction, are merely convenient labels to understand the brain’s complicated parallel processing by dividing it into parts [Shibata 2011]. For instance, processing performed close to sensors is commonly called “recognition.” We recognize an image while we think, “Is this inside a house?” “Is it daytime or nighttime?” or “What kind of people live there?” drawing on our knowledge. These thoughts occur during image recognition, making it difficult to classify them clearly as either thinking or recognition.

For ease of understanding, this paper has discussed exploration and thinking as if they exist alongside other functions. However, considering the actual human processing as described, these should not be labels for any part of the overall process. Instead, they may represent a tendency of our processing, including recognition and motion generation, to be either more irregular or more rational. In the presented simulations, sensor signals were used as input to the actor network without any preprocessing. Hence, all processes from sensors to motors are performed in the network, even though it is referred to as an “actor.” Learning was applied to the entire network, not just a part of it, and exploration was generated throughout the system, including the external world. This is consistent with the findings that chaotic attractors for existing odorants and chaos for new ones were observed in the olfactory bulb [Skarda and Freeman 1987, Freeman 1991] even though it does not seem to be directly related to motion generation. The relationship between Dynamic RL and the “default mode network” [Raichle et al. 2001] would also be interesting in terms of thinking.

Supervised learning has an excellent ability to generate reasonable answers through learning a vast amount of examples but originally does not have the ability to discover something new by itself. RL receives no direct training signal; instead, it has the ability to discover new things through exploration. Exploration in Dynamic RL is performed flexibly throughout the agent’s RNN with massive DOFs. Such exploration grows into ‘thinking’ through learning in harmony with novelty and rationality, maintaining the massive DOFs. Accordingly, the author believes that thinking that emerges in this way would make it possible to step outside of common sense without becoming incoherent, which is difficult for current Transformer-based LLMs to achieve, as introduced in the Introduction.

However, thinking seems to be a conscious process that is more sequential than parallel. This may simply suggest that we can identify only the conscious part of the processes, which are just the tip of the iceberg. Visible language-level processing is exactly what current LLMs excel at, as shown in the Introduction. Given the LLMs’ outstanding abilities, the author believes that integrating Dynamic RL with an LLM is essential to realize ‘thinking.’

4.3 Risks and Concerns

Lastly, the author expresses concerns about the risks associated with this research. The risks of AI have been frequently pointed out [Future of Life Institute 2015, Doya et al. 2022, Center for AI Safety 2025a]. The general consensus on science and technology that we should benefit from them while avoiding their risks has been widely accepted also in the

AI community [Future of Life Institute 2015, Doya et al. 2022]. However, if AI entities begin to think autonomously and discover ideas beyond human comprehension, they could threaten humanity. In this context, it is groundbreaking that the prioritization of risk aversion over profit-taking is clearly stated in [Center for AI Safety 2025b] as follows: “The potential benefits of AI could justify the risks if the risks were negligible. However, the chance of existential risk from AI is too high for it to be prudent to rapidly develop AI. Since extinction is forever, a far more cautious approach is required.” As mentioned, the author believes that Dynamic RL’s potential to discover new things would be significantly higher than the other AI technologies and should be especially considered.

Of course, given the current level of this research, these concerns must seem presumptuous or alarmist. However, once AI entities reach a level where they can design more intelligent versions of themselves without any human assistance, their growth will surpass exponential rates, unlike human-developed technologies such as semiconductors [Kurzweil 1998]. It will accelerate far beyond our predictions. Above all, they will get out of control and evolve rapidly even if we stop the research, leaving us with no means to avoid or halt them.

The author begs people to seriously imagine the horror of autonomous, superintelligent drones flying around and humanoids walking around and targeting humans as a potential reality. He has no idea how we can absolutely ensure that superintelligent drones or humanoids will never muzzle humanity. Given the gravity of the matter and the fact that redoing it is not an option, easy wishful thinking without a thorough understanding should definitely be avoided [Yudkowsky 2023]. Science and technology have made remarkable progress. Now, they have reached a point where we can lead satisfactory lives although challenges remain in other aspects, such as wars and economic disparities. What kind of benefit on earth justifies putting humanity’s safety at such a fatal risk? We must seriously consider this.

History shows that once nations begin developing lethal weapons using a new powerful technology, it becomes extremely difficult to eliminate them in power games, as in the case of nuclear weapons. However, the author would like to strongly warn that if one nation attempts to use this technique to conquer another, it is likely that the world, including the nation itself, will eventually be defeated by the technique.

The researchers developing technologies can usually be the first to recognize their potential risks by using their imagination about what will happen. Given the severity of the potential consequences and the importance of early action, they should not throw the decision solely to society but must take responsible measures themselves first. The author may have neither the right nor the power to halt the research. Submitting this research, despite being aware of its potentially fatal risks, represents a major contradiction. Some may also argue that such discussions should be excluded from a technical paper. However, even without this submission, someone else will eventually conduct this research. The author believes that we have entered an era where researchers must take a certain amount of responsibility for their research as soon as they publish a paper on it. Therefore, in order to raise the issue and spark discussions that harness the collective wisdom of humankind, the author takes the bold step of making this submission and hereby presents the following statement to initiate discussions in order to fulfill his responsibilities as a researcher.

Acknowledgment

First, I would like to express my gratitude to Prof. Kazuyuki Aihara. His suggestion in 1995 motivated me to connect reinforcement learning and chaos. I would like to thank Prof. Hiromichi Suetani for useful suggestions about chaotic dynamics from an expert. I also thank the members of our former laboratory for their discussions and also sharing related simulation results. In particular, Dr. Toshitaka Matsuki discussed mainly the relationship between chaoticity and learning ability and has inspired me a lot from various aspects. Mr. Yuki Tokumaru also gave me a big push by simulations and discussion in the early phase of this research. This work was supported by JSPS KAKENHI Grant Number JP19300070, JP23500245, JP15K00360, JP20K11993 and also Kayamori Foundation of Informational Science Advancement K31-KEN-XXIV-539. Especially the last one supported the author’s independent research after his early retirement.

References

- N. Aggarwal, G. J. Saxena, S. Singh, and A. Pundir. Can I say, now machines can think?, 2023. URL <https://arxiv.org/abs/2307.07526>.
- K. Aihara, T. Takabe, and M. Toyoda. Chaotic neural networks. *Physics Letters A*, 144(6):333–340, 1990. ISSN 0375-9601. doi: [https://doi.org/10.1016/0375-9601\(90\)90136-C](https://doi.org/10.1016/0375-9601(90)90136-C). URL <https://www.sciencedirect.com/science/article/pii/037596019090136C>.
- S. Amari. Any target function exists in a neighborhood of any sufficiently wide random network: A geometrical perspective, 2020. URL <https://arxiv.org/abs/2001.06931>.

- M. Andrychowicz, F. Wolski, A. Ray, et al. Hindsight experience replay. In I. Guyon et al., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1–11. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.
- T. Azizi and G. Kerr. Application of stability theory in study of local dynamics of nonlinear systems. *Journal of Applied Mathematics and Physics*, 8:1180–1192, 2020. doi: <https://doi.org/10.4236/jamp.2020.86089>.
- D. Berlyne and W. E. Vinacke. Thought (n.d.), 2025. URL <https://www.britannica.com/topic/thought>. In Encyclopaedia Britannica, Encyclopedia Britannica Inc. Retrieved January 11, 2025.
- T. Broad, S. Berns, S. Colton, and M. Grierson. Active divergence with generative deep learning - a survey and taxonomy, 2021. URL <https://arxiv.org/abs/2107.05599>.
- Center for AI Safety. Statement on AI risk, 2025a. URL <https://www.safe.ai/work/statement-on-ai-risk/>.
- Center for AI Safety. An overview of catastrophic AI risks, 2025b. URL <https://www.safe.ai/ai-risk>.
- T. Chakrabarty, P. Laban, D. Agarwal, et al. Art or artifice? large language models and the false promise of creativity, 2024. URL <https://arxiv.org/abs/2309.14556>.
- CreativeHuddle. The alternative uses test, 2018. URL <https://www.creativehuddle.co.uk/post/the-alternative-uses-test>.
- E. de Bono. *The Use of Lateral Thinking*. Avon Books, 1971.
- K. Doya, A. Ema, H. Kitano, et al. Social impact and governance of AI and neurotechnologies. *Neural Networks*, 152:542–554, 2022. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2022.05.012>. URL <https://www.sciencedirect.com/science/article/pii/S0893608022001861>.
- G. Franceschelli and M. Musolesi. Creative beam search: Llm-as-a-judge for improving response generation, 2024. URL <https://arxiv.org/abs/2405.00099v4>.
- W. J. Freeman. The physiology of perception. *Scientific American*, 264(2):78–85, February 1991.
- S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.
- Future of Life Institute. Research priorities for robust and beneficial artificial intelligence: An open letter, 2015. URL <https://futureoflife.org/open-letter/ai-open-letter/>.
- G. Gendron, Q. Bao, M. Witbrock, and G. Dobbie. Large language models are not strong abstract reasoners, 2024. URL <https://arxiv.org/abs/2305.19555>.
- K. Goto and K. Shibata. Acquisition of deterministic exploration and purposive memory through reinforcement learning with a recurrent neural network. In *Proc. of SICE Annual Conf. 2010*, pages FB03–1, 2010.
- J. P. Guilford. *The nature of human intelligence*. McGraw-Hill, 1967.
- E. E. Guzik, C. Byrge, and C. Gilde. The originality of machines: AI takes the torrance test. *Journal of Creativity*, 33(3):100065, 2023. ISSN 2713-3745. doi: <https://doi.org/10.1016/j.yjoc.2023.100065>. URL <https://www.sciencedirect.com/science/article/pii/S2713374523000249>.
- T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.
- S. Huang, S. Ma, Y. Li, et al. Lateval: An interactive llms evaluation benchmark with incomplete information from lateral thinking puzzles. In *Proc. of LREC-COLING 2024 Torino*, pages 10186–10197, 2024.
- M. Ismayilzada, D. Paul, A. Bosselut, and L. van der Plas. Creativity in AI: Progresses and challenges, 2024. URL <https://arxiv.org/abs/2410.17218>.
- Y. Jiang, F. Ilievski, K. Ma, and Z. Sourati. Brainteaser: Lateral thinking puzzles for large language models, 2023. URL <https://arxiv.org/abs/2310.05057>.
- J.P.Guilford. Creativity. *The American Psychologist*, 13:444–454, 1950.
- A. Khachaturyan, S. Semenovskaya, and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. *Acta Cryst.*, A37:742–754, 1981.
- M. Koivisto and S. Grassini. Best humans still outperform artificial intelligence in a creative divergent thinking task. *Scientific Report*, 13(13601), 2023.
- T. H. Kung, M. Cheatham, A. Medenilla, et al. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLOS Digital Health*, 2(2):e0000198, 2023.

- R. Kurzweil. *The age of spiritual machines: when computers exceed human intelligence*. Viking Press, USA, 1998. ISBN 0670882178.
- X. Lu, M. Sclar, S. Hallinan, et al. AI as humanity's Salieri: Quantifying linguistic creativity of language models via systematic attribution of machine text against web text, 2025. URL <https://arxiv.org/abs/2410.04265>.
- T. Matsuki and K. Shibata. Adaptive balancing of exploration and exploitation around the edge of chaos in internal-chaos-based learning. *Neural Networks*, 132:19–29, 2020.
- T. Matsuki, Y. Sakemi, and K. Aihara. Chaos-based reinforcement learning with TD3, 2024. URL <https://arxiv.org/abs/2405.09086>.
- W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- M. Mitchell, A. B. Palmarini, and A. Moskvichev. Comparing humans, GPT-4, and GPT-4V on abstraction and reasoning tasks, 2023. URL <https://arxiv.org/abs/2311.09247>.
- V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015. doi: 10.1038/nature14236. URL <https://doi.org/10.1038/nature14236>.
- V. Mnih, A. P. Badia, M. Mirza, et al. Asynchronous methods for deep reinforcement learning, 2016. URL <https://arxiv.org/abs/1602.01783>.
- S. S. Nath, P. Dayan, and C. Stevenson. Characterising the creative process in humans and large language models, 2024. URL <https://arxiv.org/abs/2405.00899>.
- A. Newell, J. C. Shaw, and H. A. Simon. The processes of creative thinking. In *Symposium on Creative Thinking*, pages 1–82 (P–1320), 1959.
- OpenAI. OpenAI Five, 2019. URL <https://openai.com/index/openai-five-defeats-dota-2-world-champions/>. Accessed: 2024-08-24.
- OpenAI. GPT-4, 2023. URL <https://openai.com/index/gpt-4-research/>. Accessed: 2025-1-26.
- OpenAI, J. Achiam, et al. GPT-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- M. Peeperkorn, T. Kouwenhoven, D. Brown, and A. Jordanous. Is temperature the creativity parameter of large language models?, 2024. URL <https://arxiv.org/abs/2405.00492>.
- S. Pichai, D. Hassabis, and K. Kavukcuoglu. Gemini: An adaptive multi-modal AI system, 2024. URL <https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/>. Accessed: 2025-1-26.
- M. E. Raichle, A. M. MacLeod, A. Z. Snyder, et al. A default mode of brain function. *Proceedings of the National Academy of Sciences - PNAS*, 98(2):676–682, 2001. ISSN 0027-8424.
- D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, 1986.
- Y. Sawatsubashi, M. F. bin Samsudin, and K. Shibata. Emergence of discrete and abstract state representation through reinforcement learning in a continuous input task. In *Advances in Intelligent Systems and Computing, Robot Intelligence Technology and Applications 2012*, pages 13–22, 2012.
- T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay, 2016. URL <https://arxiv.org/abs/1511.05952>.
- J. Schrittwieser, I. Antonoglou, T. Hubert, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, Dec. 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4. URL <http://dx.doi.org/10.1038/s41586-020-03051-4>.
- J. Schulman, F. Wolski, P. Dhariwal, et al. Proximal policy optimization algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1–9, 2017.
- K. Shibata. Learning of deterministic exploration and temporal abstraction in reinforcement learning. In *Proc. of SICE-ICCAS (SICE-ICASE Int'l Joint Conf.)*, pages 4569–4574, 2006.
- K. Shibata. Emergence of intelligence through reinforcement learning with a neural network. In A. Mellouk, editor, *Advances in Reinforcement Learning*, chapter 6, pages 99–120. IntechOpen, 2011. doi: 10.5772/13443. URL <https://doi.org/10.5772/13443>.
- K. Shibata. Communications that emerge through reinforcement learning using a (recurrent) neural network, 2017a. URL <https://arxiv.org/abs/1703.03543>.

- K. Shibata. Functions that emerge through end-to-end reinforcement learning - the direction for artificial general intelligence -, 2017b. URL <https://arxiv.org/abs/1703.02239>.
- K. Shibata and K. Ito. Emergence of communication for negotiation by a recurrent neural network. In *Proc. of ISADS (Int'l Symposium on Autonomous Decentralized System) '99*, pages 294–301, 1999.
- K. Shibata and Y. Okabe. Reinforcement learning when visual sensory signals are directly given as inputs. In *Proc. of ICNN (Int'l Conf. on Neural Networks) '97 Houston*, volume 3, pages 1716–1720, 1997.
- K. Shibata and Y. Sakashita. Reinforcement learning with internal-dynamics-based exploration using a chaotic neural network. In *Proc. of Int'l Joint Conf. on Neural Networks (IJCNN) 2015*, page #15231, 2015.
- K. Shibata, T. Ejima, Y. Tokumaru, and T. Matsuki. Sensitivity – local index to control chaoticity or gradient globally -. *Neural Networks*, 143:436–451, 2021. ISSN 0893-6080.
- C. A. Skarda and W. J. Freeman. How brains make chaos in order to make sense of the world. *Behavioral and brain sciences*, 10:161–173, 1987.
- R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.
- D. Tam, A. Mascarenhas, S. Zhang, et al. Evaluating the factual consistency of large language models through news summarization, 2023. URL <https://arxiv.org/abs/2211.08412>.
- E. P. Torrance. Torrance tests of creative thinking, 2018. URL https://www.ststesting.com/gift/TTCT_InterpMOD.2018.pdf.
- A. M. Turing. Computing machinery and intelligence. *Mind, New Series*, 59(236):433–460, 1950. URL <https://www.jstor.org/stable/2251299>.
- A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
- O. Vinyals, I. Babuschkin, W. Czarnecki, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019. doi: 10.1038/s41586-019-1724-z. URL <https://doi.org/10.1038/s41586-019-1724-z>.
- R. Woolf. Plato and the norms of thought. *Mind*, 122:171–216, 2013. URL <https://doi.org/10.1093/mind/fzt012>.
- Y. Yamashita and J. Tani. Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLOS Computational Biology*, 6(10), 2010. doi: 10.1371/journal.pcbi.1000220.
- E. Yudkowsky. Pausing AI developments isn't enough. we need to shut it all down. *Time Magazine*, 2023. URL <https://time.com/6266923/ai-eliezer-yudkowsky-open-letter-not-enough/>.
- Y. Zhao, R. Zhang, W. Li, et al. Assessing and understanding creativity in large language models, 2024. URL <https://arxiv.org/pdf/2401.12491v1.pdf>.

A Detailed Task Description for Sequential Navigation Task

Initial agent location

Once per four episodes: randomly chosen on the four sides of the 18×18 square whose center is on the origin to promote the learning in the peripheral area.

Other episodes: randomly chosen in the whole field.

Visual sensor

Number of cells: $11 \times 11 = 121$. Cell size: 1×1 . The sensor is fixed with its center at the origin, and the cells are arranged in a square without overlap. Appearance of agent: 1×1 square.

Cell output: the area occupied by the agent as a valued from 0.0 to 1.0.

Agent's one-step move

Each actor output → multiplied by 1.25 → clipped between -1.0 and 1.0. Furthermore, keeping the direction of the move vector, the size of the vector is normalized so as that the maximum size for the direction is 1.0. For example, if the actor outputs are (0.0, 0.9), the agent will move by (0.0, 1.0), and if they are (0.3, 0.4), it will move by (0.3, 0.4). As a result, the one-step movable area becomes a circle with a radius of 1.0.

Failure criteria

The average number of steps to the goal was calculated for every 500 episodes. The simulation run was terminated when one of the following conditions was satisfied. (1) ($\text{episode_No.} \geq 5000 \& \text{average} > 190$) for 5 consecutive times or (2) ($\text{episode_No.} \geq 10000 \& \text{average} > 150$) for 5 consecutive times or . (3) ($\text{episode_No.} \geq 10000 \& \text{average} > 100$) for 10 consecutive times

B Initial Weights and Learning Rate

The parameters used in this paper are detailed further in Tables 4.

Table 4: Parameters for Critic and Actor networks. When Dynamic RL was used in the actor network, only the learning rate for SRL is written. The learning rate for SAL was always set as $\eta_{SAL} = \eta_{SRL} * 0.1$. in, hid, hid1, hid2, out: input, hidden, lower hidden, upper hidden, output layer respectively. In the column of “Initial value”, N: Normal-distributed random number (R: Spectral radius), Number only: Uniform random number (Its absolute value is less than the value).

Critic network						
	layer → layer or layer	Initial value		Learning rate		
		Switch	Crank	Switch	Crank	
Weight	in → hid	0.2	0.5	0.2	0.3	
	hid → hid	N R1.3		0.002		
	hid → out	0.0		0.02		
Bias	hid, out	0.0		0.02		

Actor network (Dynamic RL)						
	layer → layer or layer	Initial value		Learning rate η_{SRL}		
		Switch	Crank	Switch	Crank	
Weight	in → hid	0.2	0.5	0.01	0.005	
	hid1,2 → hid2,1	0.1		0.005	0.002	
	hid2 → hid2	N R3.0		0.005	0.002	
	hid1 → out	0.1		0.01	0.005	
Bias	hid1, hid2	0.0		0.002	0.0002	
	out	0.0		0.002	0.0005	

Actor network (conventional RL)						
	layer → layer or layer	Initial value		Learning rate		
		Switch	Crank	Switch	Crank	
Weight	in → hid	0.2	0.5	0.1	0.01	0.2
	hid1,2 → hid2,1	0.1		0.01	0.01	0.01
	hid2 → hid2	N R1.3		0.002	0.01	0.002
	hid1 → out	0.0		0.01	-	0.01
Bias	hid1	0.0		0.002	0.01	0.002
	hid2	0.0		0.001	0.01	0.001
	out	0.0		0.001	-	0.001

C Computation of Exploration Exponent

Before each episode, an agent is replicated with the same actor and critic networks as the original. The original agent was placed in one of the nine starting positions, which are the same as those shown in Fig. 12(b) or (d). The replicated agent was placed randomly at a position 10^{-6} units away from the original one. The two agents move according to their actor outputs without learning until one of them finishes its episode. Subsequently, the Euclidian distance between their states is observed, and the equation below is calculated. Here, the agent’s state is represented by the direct product space (222 dimensions) of the input signal space (122 dimensions) and the upper hidden layer state space (100 dimensions). This was repeated for the nine start positions, and the exploration exponent was averaged across the nine sets.

The exploration exponent is defined as

$$\lambda = \frac{1}{T_{end} - T_{start}} \sum_{t=T_{start}+1}^{T_{end}} \ln \frac{\delta_t}{\delta_{t-1}} = \frac{1}{T_{end} - T_{start}} \ln \frac{\delta_{T_{end}}}{\delta_{T_{start}}} \quad (24)$$

where δ_t : the distance between the two states at the t -th step, T_{start} : the start step after the preparation steps, T_{end} : the final step in the episode. When the distance δ is greater than 10^2 or less than 10^{-10} , T_{end} was set to the previous step number. When the agent went to a corner of the field, the locations often became exactly the same between the two episodes, but the computation continued as long as the distance remained in the range of 10^2 to 10^{-10} .

D Equation of Motion of Slider-Crank Mechanism

As the equation of motion of the slider-crank mechanism, as shown in Fig. 14, the author used the following equation

$$J\ddot{\theta} = RF_\theta - D\dot{\theta} \quad (25)$$

where

$$F_\theta = f \frac{\sin(\phi + \theta)}{\cos\phi} \quad (26)$$

$$\cos\phi = \frac{L^2 + x^2 - R^2}{2Lx} \quad (27)$$

$$x = R\cos\theta + \sqrt{L^2 - R^2\sin^2\theta} \quad (28)$$

J : the moment of inertia of the rotor($= 10.0$)

θ : the rotational angle of the rotor

R : the distance from the center of the rotor to the crank($= 1.0$)

D : the damping coefficient for the rotation of the rotor($= 0.1$)

F_θ : the force applied to the rotor in the direction of its rotation

f : the force applied laterally to the end of the rod, determined by the actor's outputs

ϕ : the angle of the rod

L : the length of the rod($= 3.0$)

x : the distance from the center of the rotor to the rod end