Evolving Decomposed Plasticity Rules for Information-Bottlenecked Meta-Learning

Anonymous authors
Paper under double-blind review

Abstract

Artificial neural networks (ANNs) are typically confined to accomplishing pre-defined tasks by learning a set of static parameters. In contrast, biological neural networks (BNNs) can adapt to various new tasks by continually updating their connection weights based on their observations, which is aligned with the paradigm of learning effective learning rules in addition to static parameters, *e.g.*, meta-learning. Among broad classes of biologically inspired learning rules, Hebbian plasticity updates the neural network weights using local signals without the guide of an explicit target function, closely simulating the learning of BNNs. However, typical plastic ANNs using large-scale meta-parameters violate the nature of the genomics bottleneck and deteriorate the generalization capacity. This work proposes a new learning paradigm decomposing those connection-dependent plasticity rules into neuron-dependent rules thus accommodating $O(n^2)$ learnable parameters with only $O(n)$ meta-parameters. The decomposed plasticity, along with different types of neural modulation, are applied to a recursive neural network starting from scratch to adapt to different tasks. Our algorithms are tested in challenging random 2D maze environments, where the agents have to use their past experiences to improve their performance without any explicit objective function and human intervention, namely *learning by interacting*. The results show that rules satisfying the genomics bottleneck adapt to out-of-distribution tasks better than previous model-based and plasticity-based meta-learning with verbose meta-parameters.

1 Introduction

Artificial Neural Networks (ANNs) with a vast number of parameters have achieved great success in various tasks (LeCun et al., 2015). Despite their *innate capability* of accomplishing pre-defined tasks, the *learning potential* and *generalizability* to various tasks at low costs is much questioned. On the other hand, Biological Neural Networks (BNNs) (Hebb, 2005) can easily acquire new skills with a few examples, while their innate ability is relatively weak. Inspired by BNNs that acquire generalizability and learning potential by compromising innate capability, researchers (Soltoggio et al., 2008; 2018) have focused on the natural evolution of BNNs (Figure 1) to build machine intelligence with nested learning loops: an outer loop optimizes meta-parameters that initialize neural networks and make rules of learning; an inner loop further adapts the neural networks to varying tasks by utilizing observations and feedbacks.

More recently, the nested learning loops of BNNs have been used to inspire meta-learning (Zoph & Le, 2017; Finn et al., 2017), especially model-based meta-learning (Santoro et al., 2016; Mishra et al., 2018; Chen et al., 2021). However, these works fail to meet one major hypothesis widely assumed in research on BNNs—*Genomic Bottleneck* (Zador, 2019; Pedersen & Risi, 2021; Koulakov et al., 2021). For higher animals such as human beings, it has been found that the amount of information in their DNA is much less than that of synapses in their nervous system (such as the brain) in orders of magnitude (Zador, 2019). This phenomenon suggests the weak abilities of a newborn but the strong potential for learning for further development, such like human babes. In this way, compared to existing meta-learning algorithms, such as few-shots learning or fine-tuning (Yosinski et al., 2014; Finn et al., 2017), which intensively rely on pre-training over-parameterized learning rules or meta-parameters (genomes) and relatively short inner loops (life cycle), BNNs actually acquire more information within the life cycle than those received from genomes.

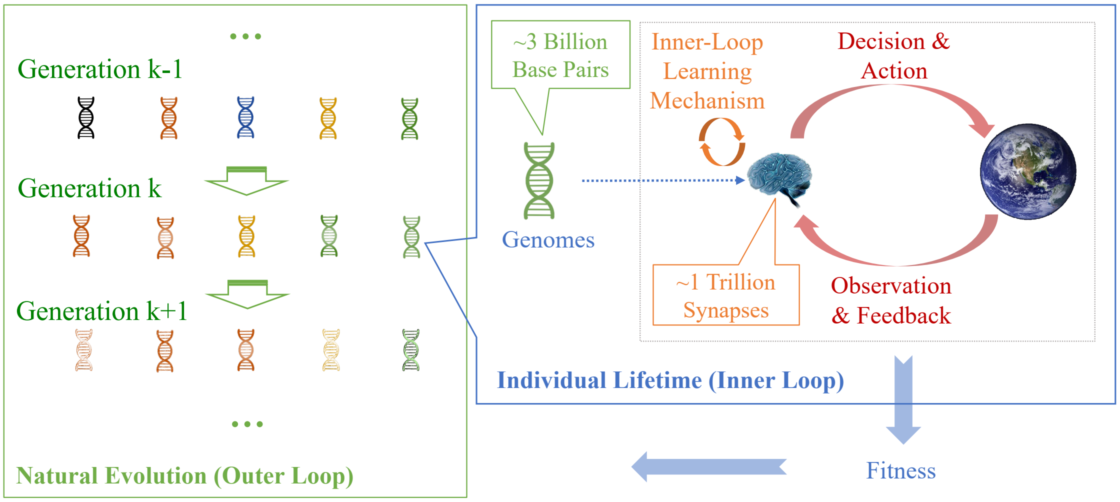


Figure 1: An illustration of the emergence of BNNs: In the outer loop of natural evolution, the genomes are mutated and selected based on the *fitness*. The life cycle of each individual composes the inner loop. The genomes that have a low capacity for information decide the learning mechanisms and initial configurations of the BNNs for each individual. As the individual interacts with the environment through actions and observations, the BNNs are further updated to better adapt to the environment, which is capable of memorizing massive information.

Thus, in this work, we aim to study a novel meta-learning framework that incorporates fewer meta-parameters within the nested learning loops in a BNN-style. To meet the desire of self-adaption, we follow the *Learning by Interacting* (LBI) principle of BNNs — Compared to the learning algorithms of ANNs, where the training loss/objective needs to be specified, and data collection/cleaning is required, human beings (Gottlieb et al., 2013) however learn by making noisy/incomplete observations (including vision, hearing, etc.) and taking actions, via innate biochemical mechanisms such as dopamine (Jalabert et al., 2011). Although the inner loops of BNNs have been used to interpret or even design ANN-based learning algorithms (Averbeck & Costa, 2017) for specific tasks, from an external perspective, in BNNs, reinforcement learning, supervised learning, and unsupervised learning are unified under the LBI principle by regarding features, labels, and feedbacks as part of the observations. The learning process becomes a black-box mechanism encoded in the network. By embedding LBI in the inner loop, the nested learning loops together make the overall populations of genomes adaptable to various tasks of different objectives, generation by generation.

To reduce the number of meta-parameters, this work revisits the canonical Hebbian plasticity rules (Hebb, 1949), which update the connection weights within the forward process by considering the states of the pre-synaptic and post-synaptic neurons. To fit ANNs, Hebbian plasticity rules typically employ 3 to 4 meta-parameters for each connection weight (Najarro & Risi, 2020). In order to lower the number of meta-parameters used for Hebbian plasticity rules, researchers have proposed either decreasing the number of rules used through evolving-and-merging (Pedersen & Risi, 2021) or using proper initial weights (Miconi et al., 2019b;a). Compared to the above works, we assume that the plasticity rule depends on its pre-synaptic neuron type and post-synaptic neuron type separately. As a result, **(1)** instead of assigning a unique plasticity rule for each connection, we correspond plastic rules to neurons and lower the size of meta-parameters from $O(n^2)$ to $O(n)$ (n is the hidden size of neurons); **(2)** since human beings generate reward signals (such as dopamine) by themselves, this work proposes neural modulator signals calculated from neural layers and enables *learning-by-interacting* as dopamine neurons; **(3)** the proposed framework, for the first time, learns plasticity rules that can update an RNN from scratch, leading to better generalizability in more challenging tasks.

Different methods are compared in 2D random maze environments, where the maze architecture, the agent origin, and the goal are randomly generated for each task. Besides generalizing across the pre-defined distributions, we also test the ability of the learned meta-parameters to generalize to out-of-distribution tasks. We prove that by following the genomics bottleneck, the proposed framework surpasses the performance of the previous Meta-RNNs (5000+ meta-parameters) and plasticity-based learners (20000+ meta-parameters) with only 1300+ meta-parameters. Moreover, we show that the proposed methods can generalize better in out-of-distribution tasks, which sheds light on open-ended learning research.

2 Related Works

2.1 Deep Meta-Learning

In meta-learning, an agent gains experience in adapting to a distribution of tasks with two nested learning loops: The outer learning loop optimize the meta-parameters that may involve initial parameters (Finn et al., 2017; Song et al., 2019), learning rules (Li & Malik, 2016; Oh et al., 2020; Najarro & Risi, 2020; Pedersen & Risi, 2021), and model structures (Zoph & Le, 2017; Liu et al., 2018; Real et al., 2019); The inner learning loops adapt the model to specific tasks by utilizing the meta-parameters. Based on the genre of inner-loop learners, those methods can be roughly classified into *gradient-based* (Finn et al., 2017; Song et al., 2019), *model-based* (Santoro et al., 2016; Duan et al., 2016; Mishra et al., 2018), and *metric-based* (Koch et al., 2015) methods (Huisman et al., 2021). In addition, the *Plasticity-based* (Soltoggio et al., 2008; 2018; Najarro & Risi, 2020) methods updates the connection weights of neural networks in the inner loop, but not through gradients. A key advantage of plasticity and model-based learning is the capability of learning by forward-only process, and the exemption of human designed objective function, which closely simulates the learning of BNNs. Our work can be classified under both model-based and plasticity-based meta-learning.

2.2 Model-based Meta-Learning

Models with memories (including recurrence and self-attention) are capable of adapting to various tasks by continually updating their memory through forwarding (Hochreiter & Schmidhuber, 1997). Those models are found to be effective in automatically discovering supervised learning rules (Santoro et al., 2016), even complex reinforcement learning rules (Duan et al., 2016; Mishra et al., 2018). Similar capabilities are found in large-scale language models (Brown et al., 2020). We see hopes of unifying all different learning paradigms (supervised learning, unsupervised learning, reinforcement learning) within one model. Still, the limitation of those models becomes evident when the input sequences get long. A reasonable guess is that the limited memory space restricted the learning potential since the memories are typically much sparser than the parameters (For the recurrent models, memories are in the order of $O(n)$ while the parameters are $O(n^2)$, with n being the number of hidden units; For self-attention models, the inference cost is proportional to $O(m^2)$ as m is the memory length). In contrast, learning paradigms that update parameters embrace higher learning potential and better asymptotic performances.

2.3 Plastic Artificial Neural Networks

The synaptic plasticity of BNNs is found to be related to the pattern of pre-synaptic and post-synaptic neurons, which is initially raised by Hebb's rule (Hebb, 1949), known as "neurons that fire together wire together". For ANNs, those rules are found ineffective without proper modulation and meta-parameters. For instance, in the $\alpha ABCD$ plasticity rule (Soltoggio et al., 2008), given the pre-synaptic neuron state x and post-synaptic neuron state y, the connection weight w is updated by

$$\delta w = m[A \cdot xy + B \cdot x + C \cdot y + D], \tag{1}$$

where m is the modulation signal, typically specified by dopamine neurons (Soltoggio et al., 2008), reward (Frémaux & Gerstner, 2016), or trainable scalars (Pedersen & Risi, 2021). As for the learning rules, some also use the *eligibility traces* e to replace the four terms in the brackets (Miconi et al., 2019b), denoted by

$$\delta w \propto e = (1-\eta)e + \eta xy. \tag{2}$$

Plastic neural layers can either be in a feed-forward layer (Najarro & Risi, 2020) or part of the recurrent layer (Miconi et al., 2018; 2019a). The initial parameters can either start from scratch (Najarro & Risi, 2020; Pedersen & Risi, 2021) or be treated as part of the meta-parameters (Miconi et al., 2019b;a).

A challenge for Plastic Artificial Neural Networks (PANN) is the requirement for verbose meta-parameters. Connections with n input neurons and n output neurons costs over $4n^2$ hyper-parameters (A, B, C, D), which is even more than the connection weights to be updated (n^2).

2.4 Implementing Genomics Bottleneck

Most large-scale deep neural networks perform poorly in task generalization, especially in out-of-distribution (OOD) tasks. The bias can grow very large by introducing human-imperceptible minor disturbances to the inputs (Goodfellow et al., 2014). A potential way to pursue robustness is to manipulate a large-scale neural network with relatively simple rules, following the genomics bottleneck of BNNs. Previous works utilizing genomics bottleneck include encoding learning and forwarding rules with a number of tied tiny genomics networks (Koulakov et al., 2021) or fewer learning rules (Pedersen & Risi, 2021), and encoding extensive neural-network parameters with smaller genomics networks (or hyper-networks) (Clune et al., 2009; Ha et al., 2016; Kirsch & Schmidhuber, 2021). Among those works, merging plasticity rules of different connections and re-evolving the tied rules (Pedersen & Risi, 2021) is more related to our proposed decomposed plasticity. However, compared with (Pedersen & Risi, 2021) that seeks to tie meta-parameters among the connections irregularly, our solution is more biologically plausible and easier to scale up.

3 Algorithms

Problem Settings. We consider an agent (learner) that is dependent on meta-parameters (genomes) θ_{Gene}. It adapts to a distribution of tasks $T_j \in \mathcal{T}$ by interacting through observations i_t and actions a_t. Specifically, in this paper we mainly consider the settings of reinforcement learning, where the observation i_t include the current state (s_t), preivous-step action (a_{t-1}), and previous-step feedback (r_{t-1}) (In supervised learning we can use the feature x_t and previous-step label y_t as observation) (Mishra et al., 2018). The agent first makes some observations by tentatively exploring the environment (this stage can be referred to as *meta-training-training*, see (Beaulieu et al., 2020)), where its parameters or memories are updated. It then improves its performance in the following steps by utilizing those learned memories/parameters, where the fitness is evaluated (*meta-training-testing*). In *meta-testing*, similarly, the learned meta-parameters are given *meta-testing-training* and *meta-testing-testing* in order. A *life cycle* marks the length of an agent's inner-loop training and testing process. The goal of *meta-training* is to optimize the meta parameters or genomes (θ_{Gene}) such that the agent achieves higher fitness in *meta-training-testing*.

Decomposed Plasticity. Considering a plastic layer with pre-synaptic (input) neurons states $\mathrm{x} \in \mathbb{R}^{n_x}$ and post-synaptic (output) neurons states $\mathrm{y} \in \mathbb{R}^{n_y}$, we can rewrite Equation 1 in the matrix form of

$$\begin{aligned} W_{t+1} = W_t + m_t(&W_A \odot (\mathrm{y}_t \otimes \mathrm{x}_t) + W_B \odot (\mathbf{1} \otimes \mathrm{x}_t) \\ &+ W_C \odot (\mathrm{y}_t \otimes \mathbf{1}^{\mathrm{T}}) + W_D), \end{aligned} \tag{3}$$

where we use $\odot$ and $\otimes$ to represent "element-wise multiplication" and "outer product" respectively. Here $W_0 \in \mathbb{R}^{n_y \times n_x}$ is initialized from scratch, and $W_A, W_B, W_C, W_D \in \mathbb{R}^{n_y \times n_x}$ are the meta-parameters of learning rules. To satisfy the genomics bottleneck, we introduce a neuron dependent decomposition of those meta-parameters, which is denoted by

$$W_A = \mathrm{v}_{A,y} \otimes \mathrm{v}_{A,x}, \tag{4}$$

where $\mathrm{v}_{A,x} \in \mathbb{R}^{n_x}, \mathrm{v}_{A,y} \in \mathbb{R}^{n_y}$. We perform similar decomposition for W_B, W_C and W_D, which gives us $4(n_x + n_y)$ rules in all. It is orders of magnitude smaller than the scale of the connection weights $n_x \times n_y$.

Modulated Plastic RNN. Interaction between the agent and the environment yields a sequence of observations $i_1, ..., i_t, ...$, a plastic RNN updates the hidden states h_t with the following equation:

$$\begin{aligned} h_{t+1} &= tanh(W^{(p)}_{h,t} h_t + W^{(p)}_{i,t} i_t + b), \\ a_t &= f(W_o h_{t+1}). \end{aligned} \tag{5}$$

We use superscript (p) to represent plastic connection weights. Different from previous works of plastic RNN or plastic LSTM that only implement plasticity in $W^{(p)}_{h,t}$, we apply decomposed plasticity for both $W^{(p)}_{h,t}$ and $W^{(p)}_{i,t}$ by regarding h_t and i_t as input neurons respectively. This further reduces our meta-parameters. Notice that this formulation is somewhat similar to hyper-RNN (Ha et al., 2016), but there is a fundamental difference in that decomposed plasticity updates the connections from scratch while hyper-RNN decides the connections from another RNN.

We consider two types of modulation regarding m_t: The pre-synaptic dopamine neuron generates the modulation by a non-plastic layer processing the pre-synaptic hidden states; The post-synaptic dopamine neuron generates the modulation by processing the post-synaptic hidden states, as follows:

$$\text{Pre-synaptic Dopamine Neuron } (\textbf{PreDN}): m_{h,t}, m_{i,t} = \sigma(W_m [i_t, h_t]) \tag{6}$$

$$\text{Post-synaptic Dopamine Neuron } (\textbf{PostDN}): m_{h,t}, m_{i,t} = \sigma(W_m h_{t+1}) \tag{7}$$

The proposed plasticity can be implemented in both recurrent NNs and forward-only NNs. Notice that $\theta_{\text{Gene}} = \{W_m, W_o, b, \mathrm{v}_{h,A,x}, \mathrm{v}_{h,B,x}, ..., \mathrm{v}_{h,A,y}, ..., \mathrm{v}_{i,A,x}, \mathrm{v}_{i,B,x}, ...\}$ are the meta-parameters optimized by meta-training but kept static within the inner loop. For convenience we directly refer to those meta-parameters with *genomes*. On the other hand, $\theta_{\text{Mem}} = \{W^{(p)}_{h,t}, W^{(p)}_{i,t}, h_t\}$ starts from scratch but is continually updated in the inner loop, which can be regarded as *memories*. We use N_{Gene} and N_{Mem} to denote the number of parameters in genomics and memories respectively.

Outer-Loop Evolution. Given task $T_j \in \mathcal{T}$, by continuously executing the inner loop including *meta-training-training* and *meta-training-testing*, we acquire the fitness of each individual at the end of its life cycle. By following Evolution Strategies (ES) (Salimans et al., 2017) the genomes θ_{Gene} shall be updated by

$$\begin{aligned} Fit(\theta_{\text{Gene}}, T) &= Fitness(i_1, a_1, i_2, a_2, ..., i_\tau, a_\tau), \\ \theta^{k+1}_{\text{Gene}} &= \theta^{k}_{\text{Gene}} + \alpha \frac{1}{n} \sum_{i=1}^{n} Fit(\theta^{k,i}_{\text{Gene}}, T_k)(\theta^{k,i}_{\text{Gene}} - \theta^{k}_{\text{Gene}}). \end{aligned} \tag{8}$$

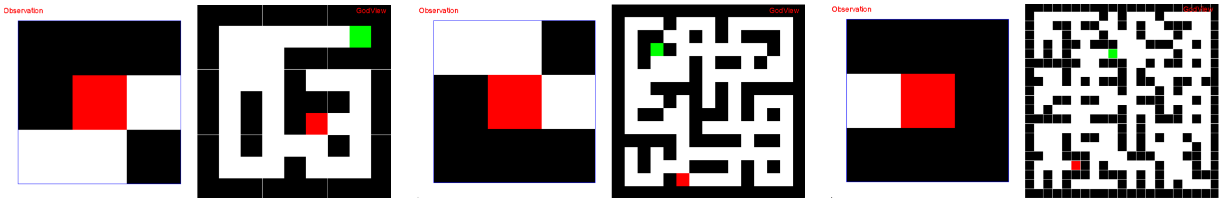
The superscript k and i represent the kth generation and the ith individual in that generation respectively. The subscript τ marks the length of an individual life cycle. For computational efficiency, in our work, we actually use seq-CMA-ES (Ros & Hansen, 2008) with the fitness normalized across the generation, which is empirically more efficient than Equation 8.

4 Experiments

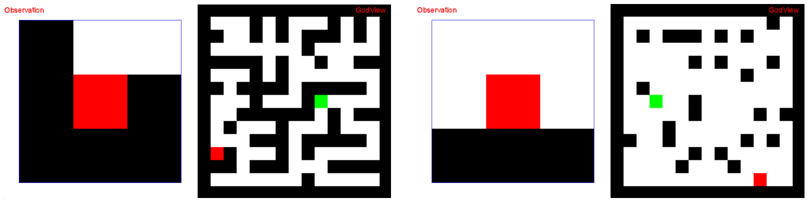
4.1 Experiment Settings

We validate the proposed method in MetaMaze2D (Wang, 2021), an open-source maze simulator that can generate maze architectures, start positions, and goals at random. The observation i_t is composed of three parts: the 3×3 observed grids, the previous-step action, and the previous-step reward. The maze structures, their positions, and the goals are hidden from the agents. Our settings give 15-dimensions input and 5-dimensions output in all. The output action includes 4 dimensions deciding the probability of taking a step in its four directions (east, west, south, north) and one additional dimension deciding whether it will take a softmax sampling policy or an argmax policy. On top of the plastic layers, we add a non-plastic output layer that processes the hidden units to 5-dimensional output. The agents acquire the reward of 1.0 by reaching the goal and -0.01 in other cases. Each rollout (episode) terminates when reaching the goal, or at the maximum of 200 steps. We apply two different settings for the experiments: For *Short Life Cycle* each individual has

3-rollouts lifetime, with the first 2 rollouts being the training phase the last one being the testing phase; For *Long Life Cycle*, we extend the life cycle to 8 rollouts, with the first 2 rollouts being the training phase, the last 6 rollouts being the testing phase.



(a) Standard 9×9 Maze (b) Standard 15×15 Maze (c) Standard 21×21 Maze



(d) Crowded 15×15 Maze (e) Spacious 15×15 Maze

Figure 2: Cases of standard, crowed, and spacious mazes: The red squares mark the current positions of the agents; The green squares mark the goals; The left parts show the observed state (s_t) of the agents.

For meta-training, individuals acquire their fitnesses by repeatedly running $n_l = 4$ life-cycles for $n_t = 8$ different tasks, which is $n_t \times n_l = 32$ life-cycles in all. Each generation requires evaluating $n_{pool} = 360$ genomes, which are distributed to 360 CPUs (400 Intel(R) Xeon(R) CPU E5-2650). The variance of the noises in Seq-CMA-ES is initially set to be 0.01 and updated in training. The fitness is calculated by $Fit = \sum_{z \in [0, l_r)} w_z \cdot R_z$, with l_r being the number of rollouts in a life cycle and R_z being the reward of zth rollout. For *Short Life Cycle*, $l_r = 3$, and we set $w_0, w_1 = 0.0$ and $w_2 = 1.0$; For *Long Life Cycle* $l_r = 8$, we set $w_0, w_1 = 0.0$, and $w_z = 0.8^{7-z}$ for $z > 1$. We show the pseudo code of meta-training and meta-testing in Algorithm 2 [1]. Every 100 generations we add a validating phase by testing the current genome in 1024 pre-sampled mazes (*validating tasks*). By reducing n_t, n_l, or n_{pool} we observed obvious drop in asymptotic performance. Scaling up those settings will stabilize the training but lead to an obvious increase in time and computation costs. To accelerate the meta-training process, we set $n_t = 2$ and $n_t = 4$ for the first 4,000 and the second 4,000 generations, after which we set $n_t = 8$. Meta training includes 15,000 to 30,000 generations (For our proposed methods, we basically use 15,000 generations; However, for some baseline methods, we found that 15,000 is hard to acquire satisfying results, so we extend the meta-training to up to 30,000 generations.), among which we pick those with the highest validating scores for meta testing.

The testing tasks include 2048 9×9 mazes (Figure 2 (a)), 2048 15×15 mazes (Figure 2 (b)), 2048 21×21 mazes (Figure 2 (c)) sampled in advance, universally used for all the methods for comparison. To further investigate the capability of generalizing to out-of-distribution tasks, we sample 2048 15×15(*crowded*) mazes, and 2048 15×15(*spacious*) mazes in addition, where the mazes are either more crowded with obstacles (Figure 2 (d)) or more spacious (Figure 2 (e)). MetaMaze2D allows developers to specify density of obstacles. To maintain reasonable difficulty level, in 9×9 mazes we set the obstacle density to 0.30, in 15×15 and 21×21 mazes we set the density to 0.35. For 15×15 (crowded) it is 0.45 and for 15×15 (spacious) it is 0.15 (Figure 2). To investigate the generalizability, our experiments also include *Standard Meta Testing* and *Out-Of-Distribution (OOD) Meta Testing*, depending on whether or not the tasks used for meta-training and those used for meta-testing are identically distributed.

We add the following methods into comparison:

---

[1] source code available at **anonymous**

Algorithm 1 Inner-Loop Learning

```
Input $\theta_{\text{Gene}}$ (defined in Section 3), $\mathcal{T}$.
for $T \in \mathcal{T}$ do
    Randomly reset $\theta_{\text{Mem}}$ (defined in Section 3).
    for $z = 0, 1, 2, ..., l_r - 1$ do
        for $t = 0, 1, 2, ...$ until the end of the episode do
            Observe $s_t$, set $i_t = [s_t, a_{t-1}, r_{t-1}]$.
            Update $\theta_{\text{Mem}}$ using $\theta_{\text{Gene}}$ and Equation 3, 5, 6, 7, acquire output $a_t$.
            Execute $a_t$, receive $r_t$.
        $R_z = \sum_t r_t$
    $Fit(\theta_{\text{Gene}}, T) = \sum_z w_z \cdot R_z$.
Return $Fit(\theta_{\text{Gene}}, \mathcal{T}) = \frac{1}{n_t} \sum_{T \in \mathcal{T}} Fit(\theta_{\text{Gene}}, T)$.
```

Algorithm 2 Meta-Training and Meta-Testing

```
Pre-sample $\mathcal{T}_{\text{valid}}$ and $\mathcal{T}_{\text{tst}}$.
Randomly sample $n_{pool}$ initial genomes $\theta_{\text{Gene}}^{0,i}$ (defined in Section 3).
for Generations $k = 0, 1, 2, ...$ until convergence do
    Randomly sample training tasks $\mathcal{T}_{\text{tra}} = \{T_1, ..., T_{n_t}\}$.
    for $i = 1, 2, ..., n_{pool}$ (distributed to $n_{pool}$ CPUs) do
        Acquire average $Fit(\theta_{\text{Gene}}^{k,i}, \mathcal{T}_{\text{tra}})$ by repeating Algorithm 1 for $n_l$ runs.
    Apply Seq-CMA-ES to acquire the next generation centroid $\theta_{\text{Gene}}^{k+1}$ and population $\theta_{\text{Gene}}^{k+1,i}$
    Acquire $Fit(\theta_{\text{Gene}}^{k}, \mathcal{T}_{\text{valid}})$ by Algorithm 1, record $\theta_{\text{Gene}}^{*}$ acquiring the best score.
Return $Fit(\theta_{\text{Gene}}^{k}, \mathcal{T}_{\text{tst}})$ by Algorithm 1.
```

- **DNN**: Evolving the parameter of a forward-only NN with two hidden fully connected layers (both with a hidden size of 64) and one output layer. Two different settings are applied: In DNN, we only use the current state as input; In Meta-DNN, we concatenate the state and the previous-step action and feedback as the observation.

- **Meta-RNN**: Employing RNN to encode the observation sequence, the parameters of RNN are treated as meta-parameters. We evaluate Meta-RNNs with hidden sizes of 32 and 64.

- **Meta-LSTM**: Employing LSTM (with hidden units of 64) to encode the observation sequence, the parameters of LSTM are treated as meta-parameters.

- **PRNN**: Applying the original $\alpha ABCD$ plasticity rule to the PRNN.

- **DecPDNN**: Applying the decomposed plasticity to the first two layers of Meta-DNN.

- **DecPRNN**: Applying the decomposed plasticity to PRNN (Equation 5).

- **PRNN (ET)** : Applying the eligibility-traces plasticity (Equation 2) to PRNN, but only to the recursive connections ($W_h^{(p)}$), the input connections ($W_i^{(p)}$) are not included. Following Backpropamine (Miconi et al., 2019a), the initial parameters of the connection weights are not from scratch, but involved in meta-parameters.

- **PRNN (Scratch-ET)**: In PRNN (ET), start the connection weights from scratch.

- **Evolving&Merging**: Implementing evolving and merging (Yaman et al., 2021) in PRNN, where we start training with the $\alpha ABCD$ rules and reduce those rules using K-Means clustering and re-train the tied rules. But unlike the original proposal that evolves and merges multiple times, we merge and re-evolve for only one time, reducing the 20224 rules to 1144 rules, equal to the size of genomes in DecPRNN.

Those synaptic plasticities may be further combined with different types of neural modulation, including non-modulation, PreDN (Equation 6), and PostDN (Equation 7).

4.2 Experiment Results with Short Life Cycles

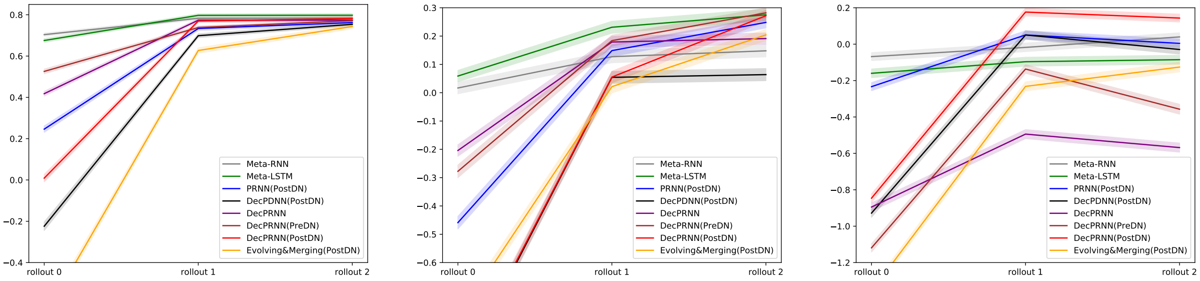
4.2.1 Standard Meta Testing

In standard meta-testing, meta-training and meta-testing are with identically distributed tasks. We show the results on 9 × 9 and 15 × 15 mazes in Table 1, including the mean and variance of the evaluation score. We also list the scale of genomes (N_{Gene}) and memories (N_{Mem}, defined in Section 3) for the agents. For clarity, we select some of the competitive groups and show their per-rollout rewards in Figure 3(a) and (b).

Table 1: Evaluating all compared methods in standard meta-testing with a 3-rollouts life cycles

	N_{Gene}	N_{Mem}	Maze 9 × 9	Maze 15 × 15
DNN	5,125	0	-0.933 ± 0.024	-1.751 ± 0.016
Meta-DNN	5,509	0	0.495 ± 0.013	-0.507 ± 0.024
Meta-RNN(Hidden=32)	1,701	32	0.651 ± 0.009	-0.197 ± 0.023
Meta-RNN(Hidden=64)	5,445	64	$\mathbf{0.784} \pm 0.008$	0.148 ± 0.021
Meta-LSTM(Hidden=64)	20,805	128	$\mathbf{0.798} \pm 0.008$	$\mathbf{0.275} \pm 0.020$
PRNN	20,613	5,120	0.714 ± 0.011	0.080 ± 0.020
PRNN(PreDN)	20,694	5,120	0.770 ± 0.010	0.249 ± 0.019
PRNN(PostDN)	20,743	5,120	0.762 ± 0.009	0.248 ± 0.020
DecPDNN	1,281	5,056	0.586 ± 0.014	-0.246 ± 0.025
DecPDNN(PreDN)	1,362	5,056	0.697 ± 0.011	-0.002 ± 0.022
DecPDNN(PostDN)	1,411	5,056	0.754 ± 0.008	0.064 ± 0.021
DecPRNN	1,217	5,120	0.774 ± 0.008	0.191 ± 0.020
DecPRNN(PreDN)	1,298	5,120	0.772 ± 0.006	$\mathbf{0.282} \pm 0.017$
DecPRNN(PostDN)	1,347	5,120	$\mathbf{0.782} \pm 0.008$	$\mathbf{0.271} \pm 0.019$
PRNN(ET)	5,511	4,160	0.757 ± 0.008	0.234 ± 0.018
PRNN(Scratch-ET)	1,481	4,160	0.618 ± 0.016	-0.829 ± 0.021
Evolving&Merging(PostDN)	1,347	5,120	0.743 ± 0.011	0.204 ± 0.020
Oracle			0.908 ± 0.001	0.820 ± 0.001
Random			-1.308 ± 0.031	-1.934 ± 0.010

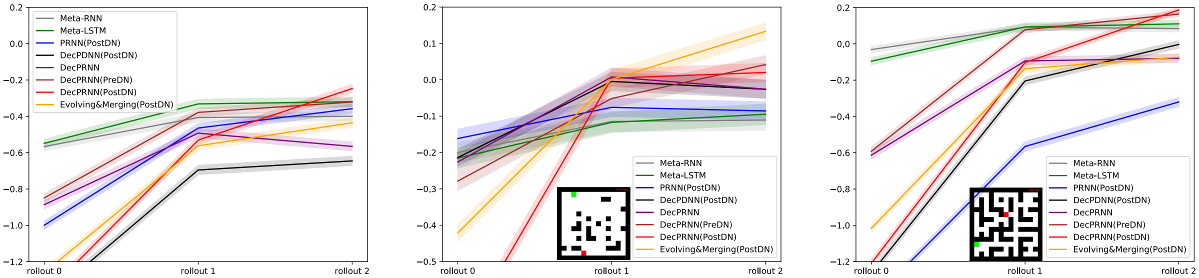
We make several remarks from the above results: First, by comparing DecPRNN with PRNN, despite using only 1/15 of the size of genomes of $\alpha ABCD$ rules, the decomposed plasticity rules perform comparably or even surprisingly better. A possible explanation is that the $\alpha ABCD$ rules are more vulnerable to noise and local optimum in meta-training. DecPRNN and modulated DecPRNN also achieve better performance than Meta-RNN (Hidden=32/64), which has fewer memories but more meta-parameters. PostDN modulated DecPRNN are even comparable to the performances of LSTM, which has 15 times more meta-parameters. Second, by comparing Hebbian plasticity-based methods (including PRNN, DecPDNN, DecPRNN) with the other methods with lower memories, we observe relatively higher performance in more complex 15 × 15 mazes than simpler 9 × 9 mazes, implying larger memories facilitating longer life span. Third, models obeying genomics bottlenecks (including DecPDNN and DecPRNN) with lower N_{Gene} and higher N_{Mem} start with weaker performance but exhibits the higher learning potential, while others begin from a relatively higher level (including PRNN, Meta-RNN, Meta-LSTM) but the growth prospects are lower. This is consistent with the phenomenon that BNNs have a low start point but high potential. Compared with Evolving&Merging(PostDN), DecPRNN(PostDN) achieves even better performance in both types of mazes given an equal number of genomes, which is more encouraging considering Evolving&Merging is more costly in meta-training. Moreover, synaptic plasticity with too sparse meta-parameters can not do well enough to learn an RNN structure from scratch (PRNN(Scratch-ET) has the least plasticity rules with only one meta-parameter) but has to count on proper initialization of the connection weights (PRNN(ET)). It meets



(a) 9 × 9

(b) 15 × 15

(c) 9 × 9 → 15 × 15



(d) 15 × 15 → 21 × 21

(e) 15 × 15(*spacious*)

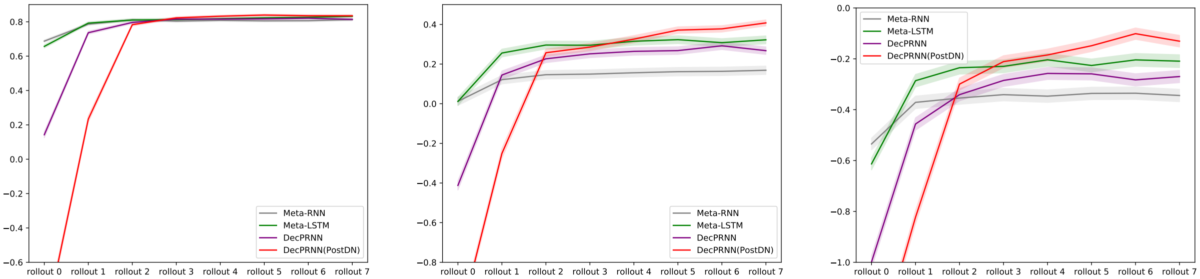
(f) 15 × 15(*crowded*)

Figure 3: Per-rollout performances of selected methods with 3-rollouts life cycles in standard meta-testing ((a), (b)) and OOD meta-testing ((c), (d), (e), (f)).

expectations since there is "no free lunch" for learning. Synaptic plasticity is only effective given "reasonably complex" rules. It is also helpful to compare different model structures (DecPRNN, DecPDNN) and neural modulations (PreDN, PostDN). Although DecPDNNs exhibit the ability to adapt to some extent, the combination of recursion and plasticity obviously outperforms DecPDNNs, especially in more complex 15×15 tasks. As for neural modulations, they brought solid improvements in DecPDNN, PRNN, and DecPRNN. PostDN modulation produces better results than PreDN in DecPDNN, but the comparison between different types of modulation in cases of PRNN and DecPRNN is a draw.

4.2.2 Out-of-Distribution (OOD) Meta-Testing

We further investigate the performances of selected models in OOD environments as follows: 1. Applying those meta-trained in 9×9 mazes to random 15×15 mazes ($9 \times 9 \rightarrow 15 \times 15$, Figure 3(c)); 2. Applying those meta-trained in 15×15 mazes to random 21×21 mazes (Figure 3(d)); 3. Applying those meta-trained in standard 15×15 mazes to 15×15 (spacious) mazes (Figure 3(e)); 4. Applying those meta-trained in standard 15×15 mazes to 15×15 (crowded) mazes (Figure 3(f)). As a result, several Hebbian plasticity-based methods (especially the proposed DecPRNN (PostDN)) substantially outperform Meta-LSTM and Meta-RNN in the last rollout. Considering that those methods can not surpass Meta-LSTM and Meta-RNN in standard tests (e.g., modulated DecPRNNs and Evolving&Merging in Figure 3(a) and (b)), they apparently exhibit better generalizability in OOD tasks. Moreover, PRNN with verbose $\alpha ABCD$ rules performs poorly in those tasks, too, indicating that generalizability to OOD environments is more related to the genomics bottleneck rather than the Hebbian plasticity. Additional discovery is that neural modulation is also helpful in this part, especially PostDN modulation. It substantially surpasses the non-modulated and PreDN-modulated models. Moreover, in both $9 \times 9 \rightarrow 15 \times 15$ and $15 \times 15 \rightarrow 21 \times 21$ tests, the non-modulated and PreDN-modulated DecPRNN models fail to continually improve the performance within the life span (especially when comparing the score of the last two rollouts) compared with PostDN-modulated models. This is possibly due to the decline of the learning potential as the agent experiences a longer life cycle than it experiences in meta-training. Interestingly, human beings also experience similar cognitive decline when getting old; a reasonable guess is the explosive growth of human life cycle length in recent years goes far beyond our average life cycle length in evolutionary history.



(a) 9×9

(b) 15×15

(c) 21×21

Figure 4: Per-rollout performances of selected methods for 8-rollouts life cycles.

Table 2: Performances of the best rollout with 8-rollouts life cycles

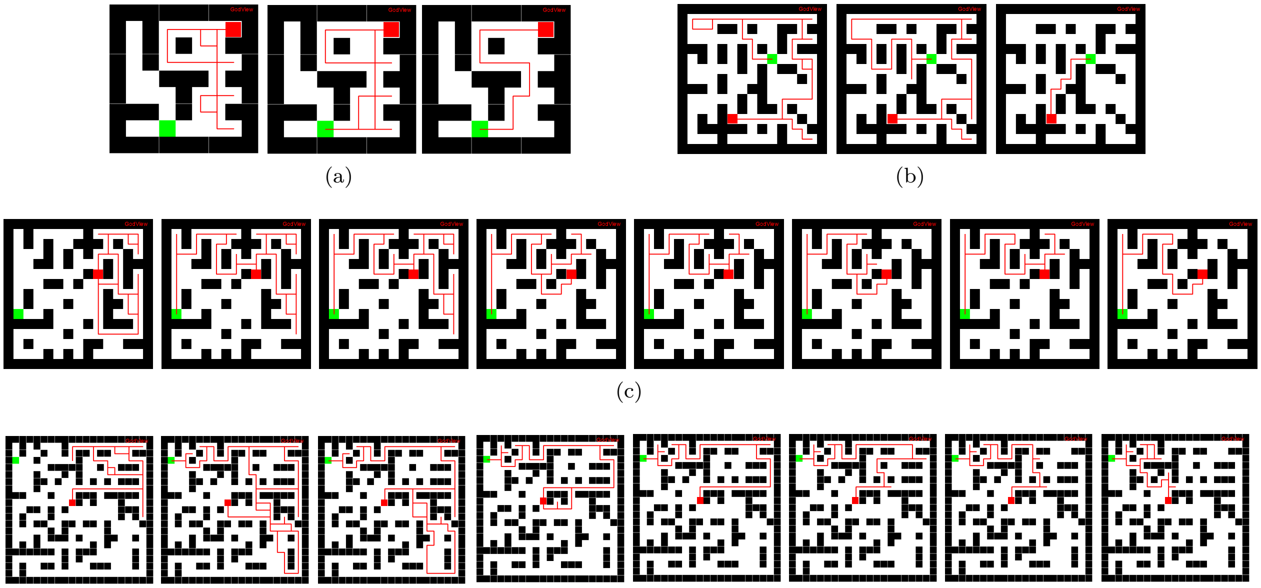
	Maze 9×9	Maze 15×15	Maze 21×21
Meta-RNN(Hidden=64)	0.812 ± 0.006	0.169 ± 0.021	-0.336 ± 0.025
Meta-LSTM	0.832 ± 0.006	0.323 ± 0.020	-0.204 ± 0.025
DecPRNN	0.821 ± 0.006	0.292 ± 0.019	-0.257 ± 0.024
DecPRNN(PostDN)	$\mathbf{0.839} \pm 0.006$	$\mathbf{0.408} \pm 0.015$	$\mathbf{-0.101} \pm 0.022$

4.3 Experiment Results with Long Life Cycles

To investigate the performance of decomposed plasticity in the long term, we further meta-train Meta-RNN, Meta-LSTM, DecPRNN, and DecPRNN(PostDN) in standard 9×9, 15×15, and 21×21 mazes by extending the lifetime to 8 rollouts. The per-rollout performance is shown in Figure 4. In Table 2 we list the highest fitness in the testing phase. As expected, the performance of all compared methods improves over the 3-rollouts runs. However, DecPRNN(PostDN) surpasses those which yield SOTA performance in 3-rollouts life cycles. Moreover, the advantage of DecPRNN(PostDN) is becoming more prominent when coming to larger-scale mazes. It further affirms the superiority of plasticity-based methods and the importance of the capacity of memories regarding a long inner loop. In Section A.1 we further show that plasticity captures the long-term information while hidden states is responsible the short-term memory.

4.4 Case Analysis

To get intuitive understanding of the inner-loop learning, we randomly sample several mazes and test the trained DecPRNN(PostDN) on them, and show the trajectory of the agents of each rollout in Figure 5. Although the agents fail to find the globally shortest paths in large part, probably attributed to the limited rollouts for exploration, we observe substantial improvements in the trajectories in the life cycle for most cases. More interestingly, there seem to be both behaviors of *exploitation* to maintain high performance in the current rollout and *exploration* to reveal better routes for the following rollouts. For instance, it tends to explore new directions in case its previous rollout is not successful enough (e.g., the first two rollouts of Figure 5(d)) and take the shortcut discovered in the previous rollout.



(a) (b)

(c)

(d)

Figure 5: Example trajectories of the DecPRNN (PostDN) agents in each rollout in different mazes (9×9, 15×15, and 21×21) for agents with the life cycle of 3 rollouts ((a),(b)) and 8 rollouts ((c), (d)). The red square marks the start position, the green square marks the goal, and the red lines denote the agents' trajectories.

5 Discussions

5.1 Biological Plausibility

Plasticity rules decided by the types of pre-synaptic and post-synaptic neurons separately can be related to neuronal differentiation (Morrison, 2001) in biological systems, where different neurons (such as motor and sensory neurons) have different characteristics as well as learning rules. Based on this intuition, our proposal further reduces the meta-parameters by allowing synapses in or out of the same neuron to share part of the rules. Our experiments have further shown that modulation processed from post-synaptic neurons performs better than pre-synaptic neurons, especially in OOD tasks. It is also in line with the phenomenon that dopamine neurons respond not only to direct sensory stimuli but also to reward-expecting cases (Schultz, 1997), implying that the dopamine neurons are more possibly affected by hidden neurons than direct sensory inputs.

5.2 Limitations

In contrast to its high potential, the meta-training expenses are relatively high compared with gradient-based supervised and reinforcement learning methods, which is attributed to both the nested learning loops (The computational efficiency of the meta-training of our method is superior to others such as Meta-RNN, and PRNN, see Appendix A.2, it is still more costly than traditional supervised/reinforcement learning by an order of magnitude) and evolution strategies (ES). ES is less efficient than gradient-based optimization in typical cases, given an explicit objective function. However, considering the desirable larger memories (N_{Mem}), sparser meta-parameters (N_{Gene}), and longer life cycles (τ) in LBI, ES is potentially more economic choice in both CPU/GPU memory and computation consumption compared with back-propagation. It is also prospective to further scale up the life cycle length, the hidden size, the depth of layers, and the maze scale in order to further validate its capability of scaling up. We are looking forward to simulating inner loops as long as the human life cycle, genomes (N_{Gene}) and memories (N_{Mem}) of human scale, which has the potential of revealing human-like general intelligence. Unfortunately, it is prohibitively expensive at the current stage.

Also, more decent environments with diverse sensory inputs, including image and language, and diverse tasks are a must for this prospect. Although much effort has also been put into this line (Chevalier-Boisvert et al., 2018; Yu et al., 2018; Silver et al., 2021), there is still a long way to go. Moreover, as we show that plasticity captures longer-term memories compared with the hidden states of recursive models (see Appendix A.1), it is not yet clear whether the decomposed plasticity is free from catastrophic forgetting when the tasks are continually switched within the agent's lifetime, which will be left to the future work.

5.3 Scaling Up With Genomics Bottleneck

To characterize the efficiency of learning with fewer genomes / meta-parameters, we can define *bottleneck factor* (BF) by $N_{\text{Mem}}/N_{\text{Gene}}$. For instance, given layers with $n_x = n_y = n \gg 1$, the BF of naive RNN is approximately $1/2n$; for $\alpha ABCD$ plastic layers, the BF is close to $1/4$; for decomposed plastic layers, the BF is close to $n/9$. Our results have shown to some extent that higher BF can lead to higher generalizability and learning potential simultaneously. Currently, typical large-scale deep models work with relatively low BF. They have been powerful in pre-defined tasks but suffered from high customization costs and the inability to generalize to variant scenarios. It is desirable and promising to design models with higher BF and with N_{Mem} in a large scale. Those models are not necessarily capable of everything initially but capable of learning to accomplish variant tasks by LBI with the high potential of shaping its memories. They learn not by the arduous efforts of AI experts, but by exploring the environments and interacting with human beings or even the other AI agents through a natural interface (e.g., natural language) on their own, just like human beings learn from environments and their teachers.

References

Bruno B Averbeck and Vincent D Costa. Motivational neural circuits underlying reinforcement learning. *Nature Neuroscience*, 20(4):505–512, 2017.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. In *ECAI 2020*, pp. 992–1001. IOS Press, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Jeff Clune, Benjamin E Beckmann, Charles Ofria, and Robert T Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *2009 iEEE congress on evolutionary computation*, pp. 2764–2771. IEEE, 2009.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in neural circuits*, 9:85, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences*, 17(11):585–593, 2013.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Donald Olding Hebb. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Mike Huisman, Jan N Van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541, 2021.

Marion Jalabert, Romain Bourdy, Julien Courtin, Pierre Veinante, Olivier J Manzoni, Michel Barrot, and François Georges. Neuronal circuits underlying acute morphine action on dopamine neurons. *Proceedings of the national academy of sciences*, 108(39):16446–16450, 2011.

Louis Kirsch and Jürgen Schmidhuber. Meta learning backpropagation and improving it. *Advances in Neural Information Processing Systems*, 34, 2021.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pp. 0. Lille, 2015.

Alexei Koulakov, Sergey Shuvaev, and Anthony Zador. Encoding innate ability through a genomic bottleneck. *bioRxiv*, 2021.

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

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pp. 3559–3568. PMLR, 2018.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL `https://openreview.net/forum?id=r1lrAiA5Ym`.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2019b.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

Sean J Morrison. Neuronal differentiation: proneural genes inhibit gliogenesis. *Current Biology*, 11(9): R349–R351, 2001.

Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *arXiv preprint arXiv:2007.02686*, 2020.

Junhyuk Oh, Matteo Hessel, Wojciech M Czarnecki, Zhongwen Xu, Hado P van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, 33:1060–1070, 2020.

Joachim Winther Pedersen and Sebastian Risi. Evolving and merging hebbian learning rules: increasing generalization by decreasing the number of rules. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 892–900, 2021.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In *International conference on parallel problem solving from nature*, pp. 296–305. Springer, 2008.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850. PMLR, 2016.

Wolfram Schultz. Dopamine neurons and their role in reward mechanisms. *Current opinion in neurobiology*, 7(2):191–197, 1997.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

Andrea Soltoggio, John A Bullinaria, Claudio Mattiussi, Peter Dürr, and Dario Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th international conference on artificial life (Alife XI)*, pp. 569–576. MIT Press, 2008.

Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. In *International Conference on Learning Representations*, 2019.

Fan Wang. MetaMaze: Efficient 3D Navigation Simulator Benchmarking Meta-learning. `https://github.com/PaddlePaddle/MetaGym/tree/master/metagym/metamaze`, 2021.

Anil Yaman, Giovanni Iacca, Decebal Constantin Mocanu, Matt Coler, George Fletcher, and Mykola Pechenizkiy. Evolving plasticity for autonomous learning under changing environmental conditions. *Evolutionary computation*, 29(3):391–414, 2021.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Haonan Yu, Haichao Zhang, and Wei Xu. Interactive grounded language acquisition and generalization in a 2d world. In *International Conference on Learning Representations*, 2018.

Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

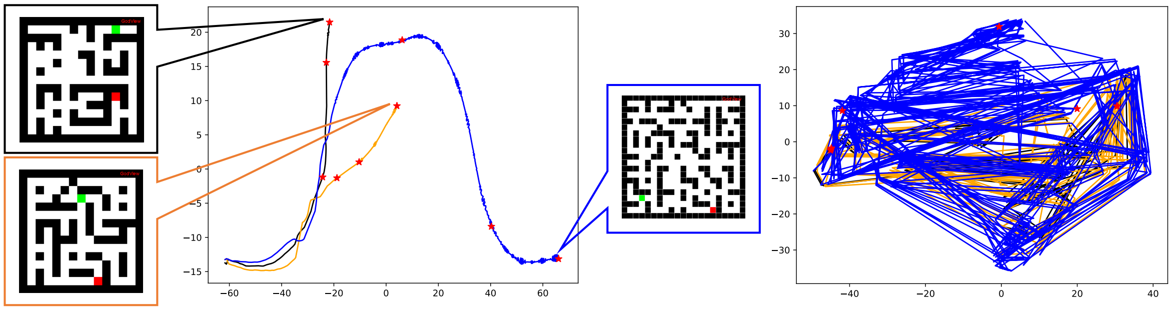
A Appendix

A.1 Inner Loop Visualization

We present the development of the memories within the agents' life cycles, including hidden states (h_t, c_t) and connection weights ($W_{h,t}^{(p)}$). We run t-SNE visualization to map those tensors to 2-D space and show their temporal trajectories in Figure 6. Comparing Figure 6(a) and (b), we see that the trajectory of the connection weights behaves differently from hidden states in DecPRNN(PostDN): The connection weights seem to keep the long term information related to the overall task configurations since they slowly diverge depending on different tasks; The hidden states seem to keep the short term memory only since they vibrate fast but stay in a static region. In Figure 6(c) to (h) we also plot the connection weights and hidden states from the other models including DecPRNN, PRNN(PostDN), DecPDNN(PostDN), Meta-LSTM and Meta-RNN. Conclusion verifies some of our proposals: Trajectories of the connection weights for PRNN (PostDN) (Figure 6(d)) are more spiking and noisy compared to DecPRNN (Figure 6(c)) and DecPRNN(PostDN) (Figure 6(a)), implying the verbose meta-parameters leading to instability in updates; Similar problems are found in the trajectory of DecPDNN(PostDN) connection weights (Figure 6(e)), a reasonable guess is that because of the absence of short term memory, DecPDNN(PostDN) has to keep necessary short-term information in its connection weights, which is blended with those long-term information. Moreover, the hidden states (as well as cell states) of Meta-LSTM seem to catch both long-term and short-term information at the same time, resulting in both short-term vibrations and long-term migrations in both its hidden states and cell states (Figure 6(f) and (g)). Similar phenomenons are found in Meta-RNN (Figure 6(h)), but with even fewer long-term behaviors. Those curves substantially validate that the plasticity is more capable of capturing long-term memories than hidden states in recursion, which make those agents more qualified long-term learners than model-based-only learners.

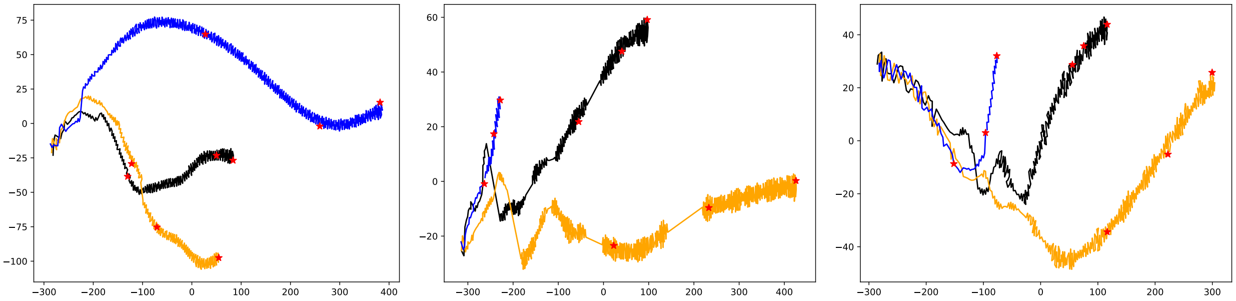
A.2 Convergence of Meta-Training

To show the cost of meta-training, we selected DecPRNN(PostDN), PRNN(PostDN), and Meta-RNN for comparison by plotting the mean and variance of the validating score against the evolved generations and wall time in the meta-training process with 3-rollouts life cycle (Figure 7). We see that PRNN(PostDN) and DecPRNN(PostDN) nearly overlap with each other regarding the evolved generations (Figure 7(a),(b)), but considering the wall time cost, DecPRNN(PostDN) is more efficient due to the lower I/O cost and computation cost per step. Although Meta-RNN has the advantage of the lowest wall time cost per generation, the wall time and generations cost for convergence is higher than the other two.



(a) Connection weights ($W_{h,t}^{(p)}$) in DecPRNN(PostDN)

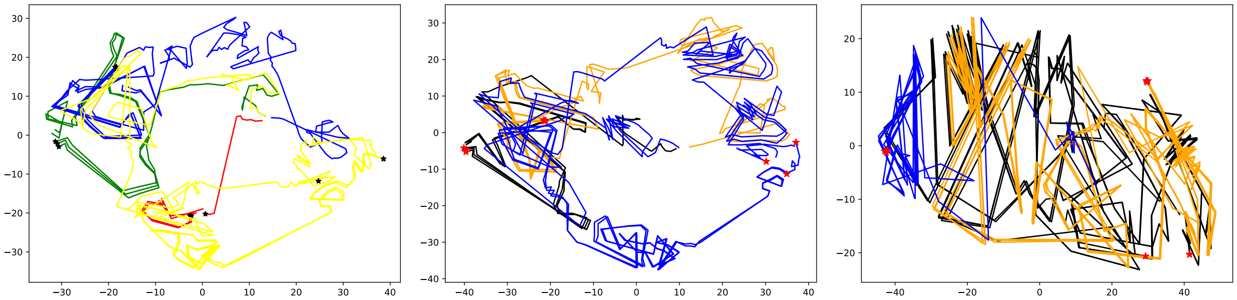
(b) Hidden states (h_t) in DecPRNN(PostDN)



(c) Connection weights in DecPRNN

(d) Connection weights in PRNN(PostDN)

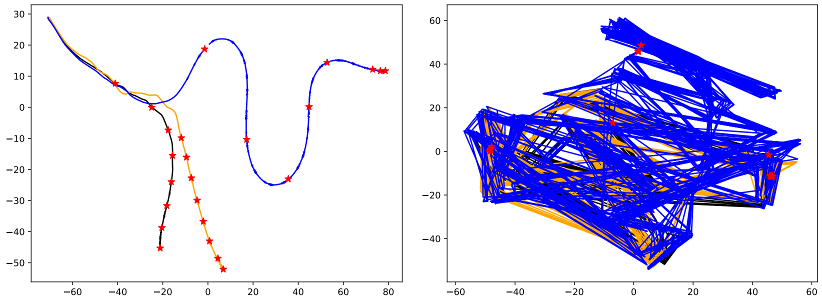
(e) Connection weights (of the second layer) in DecPDNN(PostDN)



(f) Hidden states in Meta-LSTM

(g) Cell states in Meta-LSTM

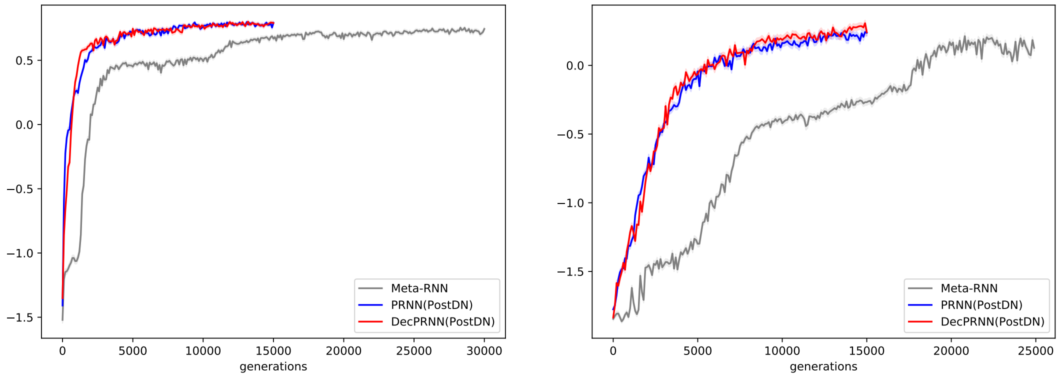
(h) Hidden states in Meta-RNN



(i) $W_{h,t}^{(p)}$ in DecPRNN(PostDN) (8-rollouts life cycle)

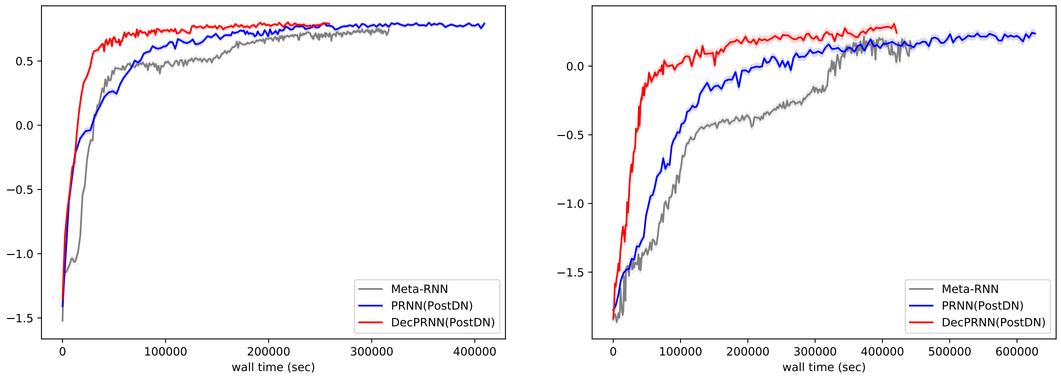
(j) h_t in DecPRNN(PostDN) (8-rollouts life cycle)

Figure 6: t-SNE visualization of the transformation of the connection weights ($W_{h,t}^{(p)}$) and hidden states (h_t) in various methods. Each trajectory corresponds to the trajectory of the vector in a unique maze shown in (a). The red ★ marks the end of a rollout.



(a) Generations (9 × 9)

(b) Generations (15 × 15)



(c) Wall time (9 × 9)

(d) Wall time (15 × 15)

Figure 7: Plotting the mean and variance of the validating score against the evolved generations (or wall clock time) in the meta-training process for selected methods with life cycle=3 rollouts.