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 the neural connections based on the inputs, which is aligned with the paradigm of learning effective learning rules in addition to static parameters, *e.g.*, meta-learning. Among variant biologically inspired learning rules, Hebbian plasticity updates the neural network weights using local signals without the guide of an explicit target function, thus enabling an agent to learn automatically without human efforts. However, typical plastic ANNs using a large amount of meta-parameters violate the nature of the genomics bottleneck and potentially deteriorate the generalization capacity. This work proposes a new learning paradigm decomposing those connection-dependent plasticity rules into neuron-dependent rules thus accommodating $\Theta(n^2)$ learnable parameters with only $\Theta(n)$ meta-parameters. We also thoroughly study the effect of different neural modulation on plasticity. Our algorithms are tested in challenging random 2D maze environments, where the agents have to use their past experiences to shape the neural connections and improve their performances for the future. Our experiment results validate that: 1. Plasticity can be adopted to continually update a randomly initialized RNN to surpass pre-trained, more sophisticated recurrent models, especially when coming to long-term memorization; 2. Following the genomics bottleneck, the proposed decomposed plasticity can be comparable to or even more effective than canonical plasticity rules in some instances.

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 capability of accomplishing pre-defined tasks, the generalizability to various new tasks at low costs is much questioned. Biological Neural Networks (BNNs) acquire new skills continually within their lifetime through neuronal plasticity (Hebb, 1949), a learning mechanism that shapes the neural connections based on local signals (pre-synaptic and post-synaptic neuronal states) only. In contrast, typical ANNs are trained once for all and can hardly be applied to unseen tasks.

More recently, the emergence of BNNs has been used to inspire meta-learning (Zoph & Le, 2017; Finn et al., 2017). Instead of training a static neural network once for all, meta-learning searches for learning rules, initialization settings, and model architectures that could be generalizable to variant tasks. We hereby make an analogy between Meta-Learning and BNNs (Figure 1): The natural evolution and lifetime learning of BNNs correspond to the nested learning loops (outer loop and inner loop) in Meta-Learning; The *genotype* corresponds to the meta-parameters shaping the innate ability and the learning rules; The *phenotype* corresponds to the neural connections and hidden states that could be updated within the inner loop.

Although the panorama of the inner-loop learning mechanism of human beings is not well known yet, investigations in this area have been used to interpret or even design ANN-based learning algorithms (Niv, 2009; Averbeck & Costa, 2017) for specific tasks. Those specific learning algorithms (including supervised learning, unsupervised learning, and reinforcement learning) can be directly applied to the inner loop in meta-learning (Finn et al., 2017). However, the majority of the previous works over-rely on the expert's

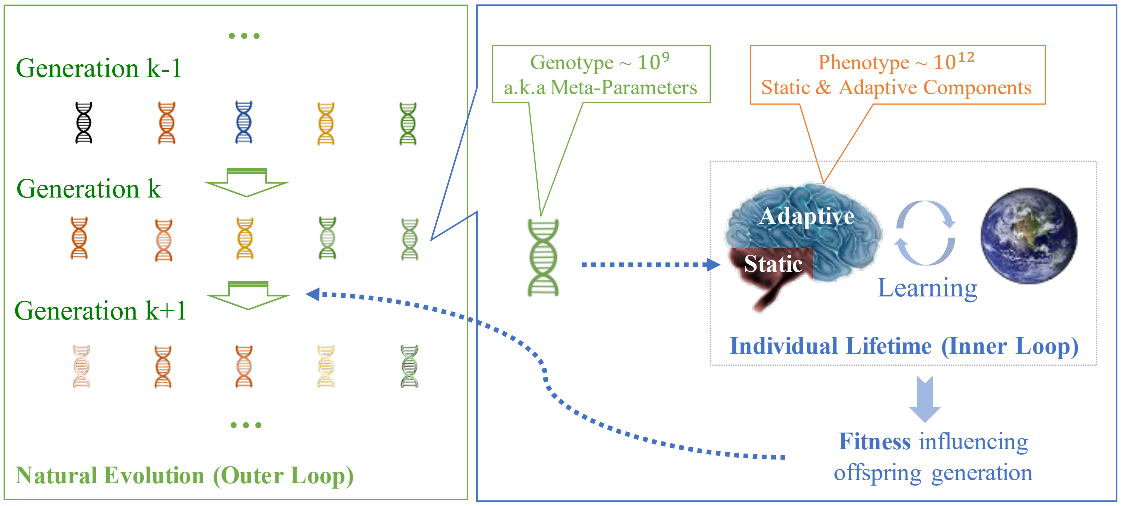


Figure 1: A simple illustration of the emergence of BNNs compared with meta-learning: The *genotype* evolved based on its fitness, which may be a result of the individual development in its life cycle. The *phenotype* is initially dependent on the genotype but updated in its life cycle. We may more specifically divide it into the static components and adaptive components. The static components won't change within its life cycle, including the learning functionalities, initial neural architectures, and static neural connections. The adaptive components are continually updated with learning, including the neuron's internal states and plastic neural connections. The natural evolution and individual life cycle are similar to the nested learning loops of meta-learning, which include the *outer loop* and the *inner loop*. The genotype is similar to meta-parameters, and the phenotype corresponds to the parameters and hidden states in ANNs. Inspired by BNNs, the genotype that has a low capacity for information (e.g., 10^9 base pairs in the DNAs human beings) decides the learning mechanisms and the initialization of the phenotype. The phenotype has a massive capacity for memorizing information (e.g., up to 10^{12} synapses in the human brain).

efforts in objective function design, data cleaning, optimizer selection, etc., thus devastating the potential of automatically generalizing across tasks. Another class of meta-learning does not explicitly seek to understand the learning mechanism; Instead, they implicitly embed different learning algorithms in black-box mechanisms such as recursion (Duan et al., 2016) and Hebbian plasticity (Soltoggio et al., 2008; Najarro & Risi, 2020). Results have shown that forward-only learning mechanisms can be more sample efficient and less noise-sensitive than human-designed gradient-based learners. Unfortunately, they suffer from other defects: Model-based learners can be less effective when the life cycles are longer; Hebbian plasticity-based learners require many more meta-parameters than the neural connections, raising considerable challenges in meta-training.

Considering the analogy between meta-learning and the emergence of BNNs, most previous works fail to meet one significant hypothesis widely assumed in research on BNNs:*Genomic Bottleneck* (Zador, 2019; Pedersen & Risi, 2021; Koulakov et al., 2021). Compared to existing meta-learning algorithms that intensively rely on pre-training many meta-parameters (genotype) (Yosinski et al., 2014; Finn et al., 2017) with less adaptive components (in the phenotype), BNNs actually acquire more information within the life cycle than those inherited from genotype (see Figure 1). In our opinion, the genomics bottleneck preserves relatively high learning potential while keeping the evolutionary process light. Previous investigations further find that reducing the meta-parameters also has a positive correlation with the generalizability of ANNs (Risi & Stanley, 2010; Pedersen & Risi, 2021).

Motivated by the aforementioned considerations, we propose a meta-learning framework with fewer meta-parameters and more adaptive components. The proposal is remarked with the following aspects.

- We revisit the canonical Hebbian plasticity rules (Hebb, 1949; Soltoggio et al., 2008) that employ 3 to 4 meta-parameters for each neural connection and proposed *decomposed plasitcity*. Instead of assigning unique plasticity rules for each neural connection, we assume that the plasticity rules depend on the pre-synaptic neuron and post-synaptic neuron separately. As a result, we introduce a decomposition of the plasticity rules reducing the meta-parameters from $\Theta(n^2)$ to $\Theta(n)$ (n is the hidden size of neurons). For better generality and fewer meta-parameters, we further follow Najarro & Risi (2020) to force the plasticity rules to update neural connections from scratch, i.e., from random initialization, instead of searching for proper initialization of the connection weights.

- Following Miconi et al. (2018; 2019), we combine Hebbian plasticity with recursion-based learners in a meta-learning framework. But we take a step further by validating that plasticity rules can update the whole recursive neural networks from scratch, including both the recurrent neural connections and the input neural connections. We also propose a method to visualize the plasticity-based learners, showing that plasticity is potentially more capable of long-term memorization than recurrence-based learners.
- Like humans generating dopamine through innate mechanisms, we thoroughly investigate the effects of dopamine neurons modulating the plastic layers in ANNs. We validate that the agent can do better by generating dopamine from post-synaptic neurons.

We select the tasks of 2D random maze navigation to validate our proposal, where the maze architectures, the agent origins, and the goals are randomly generated. The agents can only observe their surrounding grids and have no prior knowledge of the maze and the destination. Compared with the other benchmarks, it is able to generate a nearly infinite number of different tasks, thus effectively testing the agents' generalizability. The agents must preserve both short-term and long-term memory to localize themselves while exploiting shorter paths. Following the genomics bottleneck, we found that decomposed plasticity yields comparable or even better performance than canonical plasticity rules while requiring fewer outer-loop learning steps. We also validate that plasticity can be a better long-term memorization mechanism than recurrence. For instance, our plastic RNNs surpass Meta-LSTM with over 20K meta-parameters in very challenging tasks by using only 1.3K meta-parameters.

2 Related Works

2.1 Deep Meta-Learning

In meta-learning, an agent gains experience in adapting to a distribution of tasks with nested learning loops: The outer learning loop optimize the meta-parameters that may involve initialize settings (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 arcitectures (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; Wang 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, and the exemption of human designed objective function. Our work combines 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). Model-based learners own potential of unifying all different learning paradigms (supervised learning, unsupervised learning, reinforcement learning) within one paradigm. Still, the limitation of those learners becomes evident when the life cycles get long. A reasonable guess is that the limited memory space restricted the learning potential since the adaptive components are typically much sparser than the static components (For the recurrent models, the adaptive components are the hidden states, which is in the order of $\Theta(n)$; The static components are the neural connections, which is $\Theta(n^2)$, with n being the number of hidden units.). In contrast, learning paradigms that update the neural connections have higher learning potential and better asymptotic performances.

2.3 Plastic Artificial Neural Networks

The synaptic plasticity of BNNs is found to depend on the pre-synaptic and post-synaptic neuronal states, which is initially raised by Hebb's rule (Hebb, 1949), known as "neurons that fire together wire together". Hebb's rule allows neural connections to be updated using only local signals, including the pre-synaptic and post-synaptic neuronal states. 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_t and post-synaptic neuron state y_t, the neural connection weight w_t is updated by

$$w_{t+1} = w_t + m_t[A \cdot x_t y_t + B \cdot x_t + C \cdot y_t + D], \tag{1}$$

where m_t is the modulation signal, typically specified by dopamine neurons (Soltoggio et al., 2008), feedback (Frémaux & Gerstner, 2016), or trainable constants (Pedersen & Risi, 2021). A, B, C, D are meta-parameters. As for the learning rules, some also use the *eligibility traces* (Miconi et al., 2019), denoted by

$$\begin{aligned} w_{t+1} &= w_t + m_t e_{t+1}, \\ e_{t+1} &= (1-\eta)e_t + \eta x_t y_t, \end{aligned} \tag{2}$$

Plastic neural layers can either be in a feed-forward layer (Najarro & Risi, 2020) or part of the recurrent layer (Miconi et al., 2018; 2019).

A challenge for plastic ANNs is the requirement for extensive meta-parameters. For instance, connections with n_x input neurons and n_y output neurons requires over $4n_xn_y$ meta-parameters (A, B, C, D), which is even more than the neural connections updated. Rules with less meta-parameters such as eligibility traces has been only validated in cases the neural connections are properly initialized (Miconi et al., 2018; 2019).

2.4 Implementing Genomics Bottleneck

Large-scale deep neural networks typically lack robustness and generalizability (Goodfellow et al., 2014). A potential way to address the challenge is to manipulate a large-scale neural network with relatively simple rules, following the genomics bottleneck in biology. Previous works utilizing genomics bottleneck include encoding forward, backward rules, and neural connections with a number of tied smaller-scale genomics networks (Koulakov et al., 2021), reducing plasticity rules (Pedersen & Risi, 2021), encoding extensive neural network parameters with pattern producing networks (or hyper-networks) (Stanley et al., 2009; Clune et al., 2009; Ha et al., 2016), and even representing plasticity rules with hyper-networks (Risi & Stanley, 2010; 2012). Among those works, Evolving&Merging (Pedersen & Risi, 2021) is more related to our decomposed plasticity. Based on similar motivations of reducing the learning rules, Evolving&Merging tie plasticity rules of variant neural connections based on the similarity and re-evolve the tied rules. However, compared to the decomposed plasticity, it is less biologically plausible, more computationally expensive, and harder to scale up.

3 Algorithms

Problem Settings. We suppose an agent has its behaviors dependent on both static components (including learning functionalities, and static neural connections) and adaptive components (including plastic neural connections and neuronal states, denoted by θ), which is initially decided by a group of meta-parameters (genotype, denoted by ϕ, see Figure 2). Notice that in our cases, the adaptive components θ is always started from scratch, and the meta-parameters ϕ only decides the static components; In other cases the initialization of the adaptive components might also depend on ϕ (Miconi et al., 2018; 2019). In *meta training* or outer-loop learning, the meta-parameters ϕ are optimized across a set of training tasks $T_j \in \mathcal{T}_{tra}$, and then used for initialization. In *meta testing*, ϕ is evaluated over a set of validating/testing tasks $\mathcal{T}_{tst}$. For each step in meta training and meta testing, the individual *life cycle* of the agent (i.e. the inner loop) refers to the process that it interacts with environments through observations i_t and actions a_t, where θ is continually updated and change its behaviors. Specifically, in this paper we mainly consider the meta reinforcement learning problems, where the observation i_t combines current state (s_t), preivous-step action (a_{t-1}), and previous-step feedback (r_{t-1}) (Duan et al., 2016; Mishra et al., 2018; Wang et al., 2018) (In supervised learning i_t combines the features x_t and previous-step label y_{t-1} (Santoro et al., 2016)). The inner loop typically has two phases (Beaulieu et al., 2020): The agent first tentatively explores the environment in *meta-training-training* and learns from the observations; It is latterly evaluated in the *meta-training-testing* phase. In *meta-testing*, similarly, the learned meta-parameters are given *meta-testing-training* and *meta-testing-testing* in order. A *life cycle* marks the summarized length of an agent's inner-loop training and testing phases.

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

$$\Delta W(m, x, y) = m \cdot [W_A \odot (\mathrm{y} \otimes \mathrm{x}) + W_B \odot (\mathbf{1} \otimes \mathrm{x}) + W_C \odot (\mathrm{y} \otimes \mathbf{1}) + W_D], \tag{3}$$

where we use $\odot$ and $\otimes$ to represent "element-wise multiplication" and "outer product" respectively. Here ΔW is the updates for the neural connections, and $W_A, W_B, W_C, W_D \in \mathbb{R}^{n_y \times n_x}$ are the meta-parameters deciding the learning rules. In decomposed plasticity, we introduce a neuron dependent decomposition of those meta-parameters, e.g., $W_A = \mathrm{v}_{Ay} \otimes \mathrm{v}_{Ax},$, thus Equation 3 can be changed to

$$\begin{aligned} \Delta W(m, \mathrm{x}, \mathrm{y}) = m \cdot [&(\mathrm{v}_{Ay} \odot \mathrm{y}) \otimes (\mathrm{v}_{Ax} \odot \mathrm{x}) + \mathrm{v}_{By} \otimes (\mathrm{v}_{Bx} \odot \mathrm{x}) \\ &+ (\mathrm{v}_{Cy} \odot \mathrm{y}) \otimes \mathrm{v}_{Cx} + \mathrm{v}_{Dy} \otimes \mathrm{v}_{Dx}], \end{aligned} \tag{4}$$

where $\mathrm{v}_{*,x} \in \mathbb{R}^{n_x}, \mathrm{v}_{*,y} \in \mathbb{R}^{n_y}$. The decomposed plasticity rule contains $4(n_x + n_y)$ parameters in all. For large n_x and n_y, it is orders of magnitude smaller than the scale of the neural connections ($n_x \times n_y$).

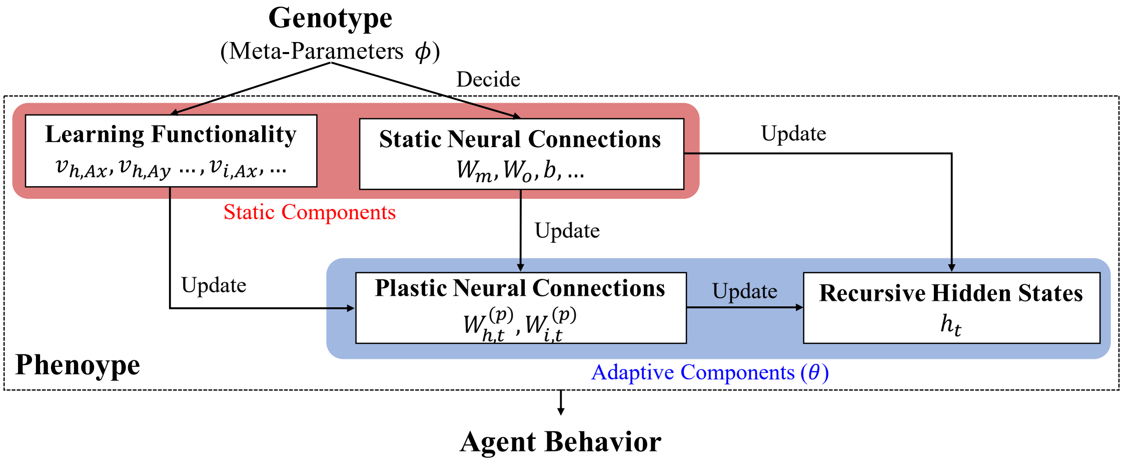


Figure 2: An overview of the modulated plastic RNNs. The static components include the learning rules and static neural connections decided by genotype, which won't change within the inner loop. The adaptive components include the plastic neural connections and hidden neuronal states continually shaped by learning rules and other components.

Modulated Plastic RNN. Given a sequence of observations $i_1, ..., i_t, ...$, a plastic RNN updates the hidden states h_t with the following equation:

$$\begin{aligned} h_{t+1} &= tanh(W^{(p)}_{h,t} h_t + W^{(p)}_{i,t} i_t + b), && (5) \\ a_t &= f(W_o h_{t+1}), && (6) \\ W^{(p)}_{h,t+1} &= W^{(p)}_{h,t} + \Delta W(m_{h,t}, h_t, h_{t+1}) && (7) \\ W^{(p)}_{i,t+1} &= W^{(p)}_{i,t} + \Delta W(m_{i,t}, i_t, h_{t+1}). && (8) \end{aligned}$$

We use superscript (p) to represent plastic neural connections. 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 $W^{(p)}_{i,t}$. This further reduces our meta-parameters.

We consider two types of modulation: 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:

$$\text{Pre-synaptic Dopamine Neuron } (\textbf{PreDN}): \; m_{h,t}, m_{i,t} = \sigma(W_m[i_t, h_t]) \quad (9)$$

$$\text{Post-synaptic Dopamine Neuron } (\textbf{PostDN}): \; m_{h,t}, m_{i,t} = \sigma(W_m h_{t+1}) \quad (10)$$

The proposed plasticity can be implemented in both recurrent NNs and forward-only NNs. An overview of plastic RNN is shown in Figure 2. Parameters that decide the learning functionality (plasticity rules) and the static neural connections are regarded as meta-parameters (ϕ). The variables contained by plastic neural connections and recursive hidden states are regarded as adaptive components in phenotype (θ). Figure 2 can fit various model-based and plasticity-based learning algorithms. Based on this setting, the genomics bottleneck indicates that $|\theta| \gg |\phi|$.

Outer-Loop Evolution. Given task $T_j \in \mathcal{T}$, by continually executing the inner loop including *meta-training-training* and *meta-training-testing*, we acquire the fitness of the genotype (meta-parameters) ϕ at the end of its life cycle, denoted as $Fit(\phi, T_j)$. By following Evolution Strategies (ES) (Salimans et al., 2017a) the genotype shall be updated by

$$\phi^{k+1} = \phi^k + \alpha \frac{1}{g} \sum_{i=1}^{g} Fit(\phi^{k,i}, T_k)(\phi^{k,i} - \phi^k). \quad (11)$$

The superscript k and i represent the kth generation and the ith individual in that generation. The subscript τ marks the length of an individual life cycle. The population $\phi^{k+1,i}$ is sampled around ϕ^{k+1} with the covariance matrix $C = \sigma^2 \mathbb{I}$. For high-dimensional meta-parameters, selecting proper hyperparameters (e.g. the covariance matrix C) is non-trivial. Improper selection could end up in inefficient optimization and local optimums. To address this challenge, CMA-ES greatly improves optimization efficiency and robustness by automatically adapting covariance matrix using evolution path (Hansen & Ostermeier, 2001). However, it is at a price of increasing the per-generation computational complexity from $\Theta(|\phi|)$ to $\Theta(|\phi|^2)$, which is infeasible for large-scale ANNs. Alternatively, we use seq-CMA-ES (Ros & Hansen, 2008) where the covariance matrix degenerates to $\Theta(|\phi|)$ by preserving the diagonal elements of C only, which is both affordable and empirically more efficient compared with ES.

The proposed model can also be optimized with a gradient-based optimizer following Miconi et al. (2018). In cases of supervised training, ES is typically less efficient than gradient-based optimization. However, for meta-RL with sparse rewards, ES could be more efficient than gradient-based optimizers (Salimans et al., 2017b). Moreover, considering the models obeying genomics bottleneck ($|\theta| \gg |\phi|$) and long life cycles ($\tau \gg 1$), ES-type optimizers could be a potentially more economical choice in both CPU/GPU memory and computation consumption.

Biological Plausibility. The decomposed plasticity gets inspiration from neuronal differentiation (Morrison, 2001) in biological systems. Instead of assuming each neural connection has unique learning rules, it might be more biologically plausible to propose that the learning rules are related to pre-synaptic and post-synaptic neurons separately. Although there are other ways to make the learning rules more compact, e.g., hypernetworks (Risi & Stanley, 2012), and Evolving&Merging (Pedersen & Risi, 2021), the decomposed plasticity is relatively straightforward and easier to be implemented by tensor operations. The settings of neural modulation are also related to the inspiration of biological systems. It is found that dopamine neurons respond not only to direct sensory stimuli but also to reward-expecting cases (Schultz, 1997), which is likely to suggest that the backward PostDN is more biologically plausible.

Summary. We formalize the inner-loop learning and meta-training process in Algorithm 1 and Algorithm 2. The framework [1] are also applicable to the other model-based and plasticity-based meta-learning. Notice that in meta-RL, a life cycle includes multiple episodes. The agent must gain experience from the earlier episodes (meta-training-training) to perform well in later episodes (meta-training-testing). Thus, we use the variable w_z to tune the importance of each episode for evaluating the fitness. An explanation of these training settings can be found in appendix A.2 .

Algorithm 1 Inner-Loop Learning

```
Input $\phi$, $\mathcal{T}$.
for $T \in \mathcal{T}$ do
    Reset $\theta$ to scratch.
    for $z = 0, 1, 2, ...$ until the end of a life cycle do
        for $t = 0, 1, 2, ...$ until the end of an episode do
            Observe $s_t$, set $i_t = [s_t, a_{t-1}, r_{t-1}]$.
            Update $\theta$ using $\phi$ and Equation 4,5,6,7,8, acquire output $a_t$.
            Execute $a_t$, receive $r_t$.
        $R_z = \sum_t r_t$
    $Fit(\phi, T) = \sum_z w_z \cdot R_z$.
Return $Fit(\theta_{\text{Gene}}, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} Fit(\phi, T)$.
```

Algorithm 2 Meta-Training and Meta-Testing

```
Pre-sample $\mathcal{T}_{\text{valid}}$ and $\mathcal{T}_{\text{tst}}$.
Randomly sample $g$ initial genotypes $\phi^{0,i}$, $i = 1, ..., g$.
for Generations $k = 0, 1, 2, ...$ until convergence do
    Randomly sample training tasks $\mathcal{T}_{\text{tra}}$.
    for $i = 1, 2, ..., g$ do
        Acquire average fitness $Fit(\phi^{k,i}, \mathcal{T}_{\text{tra}})$ by calling Algorithm 1
    Apply Seq-CMA-ES to acquire the next generation centroid $\phi^{k+1}$ and population $\phi^{k+1,i}$
    Acquire $Fit(\phi^k, \mathcal{T}_{\text{valid}})$ by Algorithm 1, record $\phi^*$ acquiring the best fitness.
Return $\phi^*$, $Fit(\phi^*, \mathcal{T}_{\text{tst}})$.
```

4 Experiments

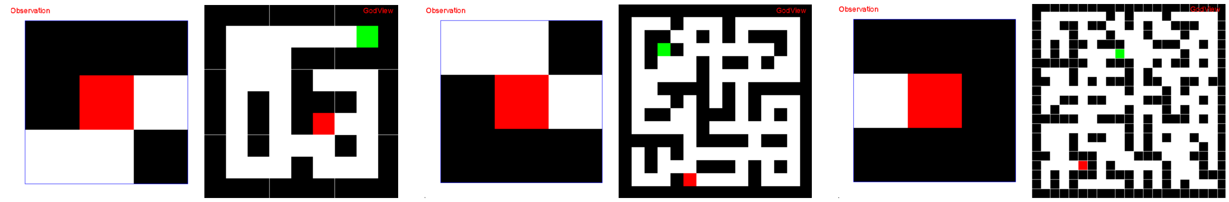
4.1 Experiment Settings

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 (s_t), the previous-step action (a_{t-1}), and the previous-step reward (r_{t-1}). The maze structures, their positions, and the goals are hidden from the agents. Our settings have 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

---

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

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. Pitfalls are encountered if we further replace the output layer with plastic neural connections. The agents acquire the reward of 1.0 by reaching the goal and -0.01 in other cases. Each episode terminates when reaching the goal, or at the maximum of 200 steps. A life cycle has totally 8 episodes.



(a) 9×9 (b) 15×15 (c) 21×21

Figure 3: Cases of mazes of different scales including the observed states (s_t) and the god view. The red squares mark the current positions of the agents; The green squares mark the goals.

For meta-training, each generation includes $g = 360$ genotypes evaluated on $|\mathcal{T}_{\text{tra}}| = 12$ tasks. The genotypes are distributed to 360 CPUs (400 Intel(R) Xeon(R) CPU E5-2650) to execute inner loops. The variance of the noises in Seq-CMA-ES is initially set to be 0.01. Every 100 generations we add a validating phase by evaluating the current genotype in $|\mathcal{T}_{\text{valid}}| = 1024$ (*validating tasks*). By reducing g or $|\mathcal{T}_{\text{tra}}|$ we observed obvious drop in performances. Scaling up those settings will stabilize the training but lead to increase of time and computation costs. Meta training goes for at least 15,000 generations, among which we pick those with the highest validating scores for meta-testing.

The testing tasks include 9×9 mazes (Figure 3 (a)), 15×15 mazes (Figure 3 (b)), and 21×21 mazes (Figure 3 (c)) sampled in advance. There are $|\mathcal{T}_{\text{tst}}| = 2048$ tasks for each level of mazes. We run meta-training of the compared method twice, from each of which we pick those with the highest fitness in validating tasks. We report the meta-testing results on the two groups of selected meta-parameters. The convergence curves of meta-training is left to Appendices A.3.2.

We include the following methods into comparison:

- **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 input.
- **Meta-RNN**: Employing RNN to encode the observation sequence, the parameters of RNN are treated as meta-parameters. We evaluate the hidden sizes of 8 (Meta-RNN-XS), 16 (Meta-RNN-S), and 64 (Meta-RNN).
- **Meta-LSTM**: Employing LSTM to encode the observation sequence, the parameters of LSTM are treated as meta-parameters. We evaluate the hidden sizes of 8 (Meta-LSTM-XS), 16 (Meta-LSTM-S), and 64 (Meta-LSTM).
- **PRNN**: Applying the $\alpha ABCD$ plasticity rule (Equation 3) to the PRNN. We also evaluate the hidden sizes of 8 (PRNN-XS), 16, (PRNN-S) and 64 (PRNN).
- **DecPDNN**: Applying the decomposed plasticity to the first two layers of Meta-DNN.
- **DecPRNN**: Applying the decomposed plasticity to PRNN (Equation 4). We evaluate the hidden sizes of 32 (DecPRNN-S) and 64 (DecPRNN).
- **Eligibility Traces** : Applying the eligibility-traces plasticity (Equation 2) to PRNN, but only to the recursive connections ($W^{(p)}_{h,t}$), the input connections ($W^{(p)}_{i,t}$) are not included. Following

Backpropamine (Miconi et al., 2019), the initial parameters of the connection weights are not from scratch, but decided by meta-parameters.

- **Eligibility Traces (From Scratch)**: Start the plastic neural connections from scratch in Miconi et al. (2019).
- **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, same as the scale of meta-parameters in DecPRNN.

Notice the plastic neural networks may be further combined with different types of neural modulation, including non-modulation, PreDN (Equation 9), and PostDN (Equation 10). We compare different modulations in DecPRNN. For all the other plasticity-based methods, we apply PostDN, which is proven to be state-of-the-art. To clearer state the impact of different meta-parameters and adaptive components, we show the number of meta-parameters in genotypes and the number of the variables in adaptive components of all the compared methods in Figure 4.

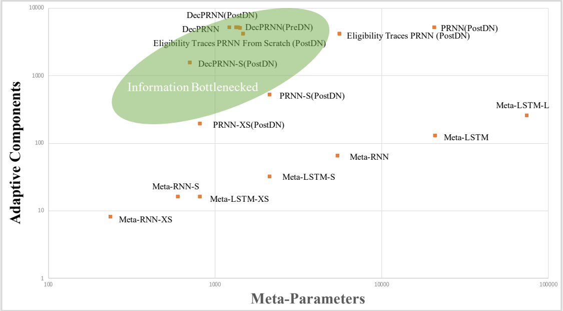


Figure 4: Over-viewing the number of variables in static and adaptive components of variant methods.

4.2 Experiment Results

4.2.1 Best-Rollout Performances

We first show the best-rollout performance of the compared methods, which is calculated by picking the rollout with the highest average rewards among their 8-rollout life cycles (Figure 4). Besides the rewards, we also show the *failure rate*, which marks the ratio of mazes among $\mathcal{T}_{tst}$ where the agent fails to reach the goal within 200 steps. The best-rollout performances are regarded as an indicator of the agents' learning potential. We evaluated all the compared methods in 9 × 9 and 15 × 15 mazes and only the most representative and competitive methods in 21 × 21. Here we omit all the results of DNN and Meta-DNN (see appendix A.3.1) as showing those bars might cover up the other differences.

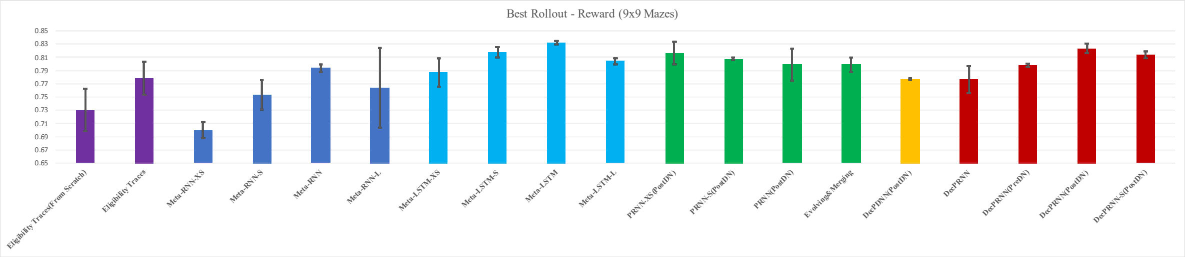
Plasticity-based vs. Model-based Learners. Generally, we show that plasticity-based learners perform better than model-based learners (Meta-RNN, Meta-LSTM), especially in more complex cases of larger mazes. For instance, in 9 × 9 mazes, the best model-based learner (Meta-LSTM) is slightly better than the best plasticity-based learner (DecPRNN(PostDN)). However, they can not compare to plasticity-based learners in 15 × 15 mazes and 21 × 21 mazes. Among model-based learners, Meta-LSTMs are better than Meta-RNNs, which is also more obvious in complex cases. For both Meta-LSTMs and Meta-RNNs, a hidden unit size below 16 yields a clear decline in the performance, while a larger scale generally results in better performances. Considering that larger mazes means longer life cycle (see appendix A.3.3), our results show that the plasticity-based learners are more powerful in long-term memorization. Notice that model-based learners generally have smaller-scale adaptive components (see Figure 4 or Table 1) but larger-scale meta-parameters compared with plasticity-based learners. We believe that the scale of adaptive components also plays an important role in memorization.

Plasticity Rules Comparison. We validate that eligibility traces with PostDN can improve the Meta-RNNs, which is consistent with Miconi et al. (2019). However, eligibility traces suffer from pitfall when the connection weight is started from scratch. It is not surprising as the eligibility traces have only 1 meta-parameter for learning rules. In contrast, the $\alpha ABCD$ rule (PRNN), Evolving&Merging, and decomposed plasticity rule can do reasonably well in variant tasks. For standard hidden unit size (64), decomposed plasticity can do the best among all plasticity rules, even though theoretically, the upper bound of $\alpha ABCD$ rule should be higher. It might be related to the meta-training process since a larger meta-parameter space raises higher challenge for ES. Another possible cause can be larger meta-parameters requires larger $|\mathcal{T}_{tra}|$ to avoid overfitting specific task distributions. The Evolving&Merging can not do as well as decomposed plasticity in 9×9 and 15×15 mazes. Considering it has the same scale of meta-parameters as decomposed plasticity, and it requires at least two stages of training (meta-train PRNN in the first stage, then switch to merged rules in the second stage), the Evolving&Merging seems less attractive than decomposed plasticity. Another surprising fact is that PRNN-S with only 16 hidden units can do very well in 9×9 and 15×15 mazes and reasonably well in 21×21 mazes. It has slightly larger-scale meta-parameters compared with DecPRNN (2119 vs. 1347). The scale of its adaptive components is much smaller than PRNN or DecPRNN (512 vs. 5120) but still much larger than model-based learners. It seems that the reduction of meta-parameters plays a crucial role here. However, as we tested DecPRNN-S ($|\theta| = 1536$, $|\phi| = 707$), and even smaller PRNN-XS ($|\theta| = 192$, $|\phi| = 809$), pitfalls are encountered. Based on these findings, and considering eligibility traces' performance and model-based learners' performance, we may conclude that relatively smaller-scale meta-parameters and larger-scale adaptive variables are helpful but with boundaries.

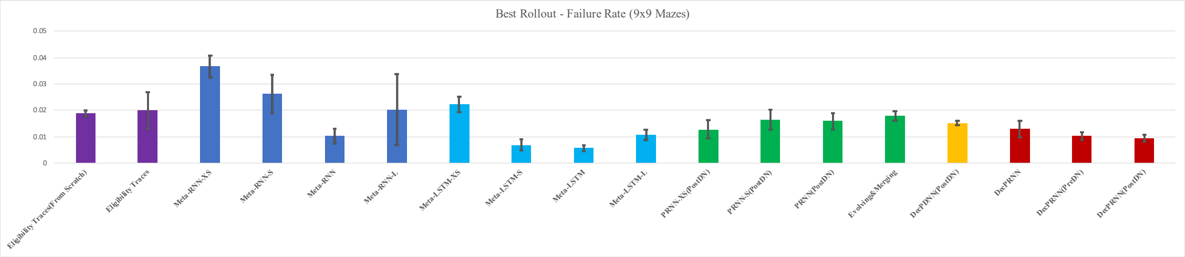
Effect of Neural Modulations. We also study the effect of different modulations in DecPRNN, including non-modulation, PreDN, and PostDN. The results prove that PostDN > PreDN > non-modulation for DecPRNN, validating the effect of neural modulation. The results also validate the superiority of biologically plausible PostDN, suggesting that the modulation signal should flow "backward" to tune the learning procedure.

4.2.2 Inner-Loop Visualization

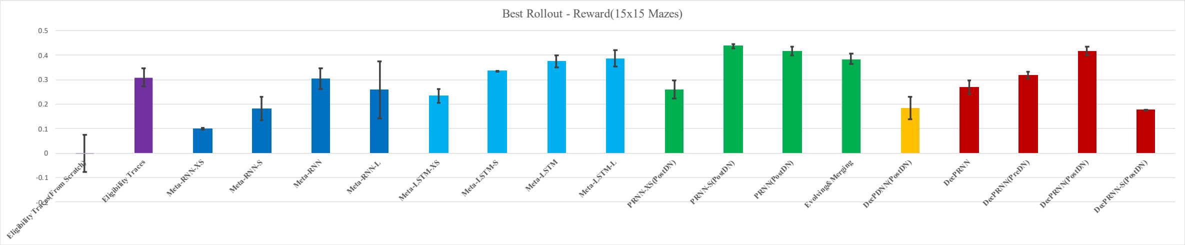
We plot the per-rollout rewards and failure rates within the agent's life cycles of several competitive methods in Figure 5. The performances of model-based and plasticity-based learners diverge in their initial performances



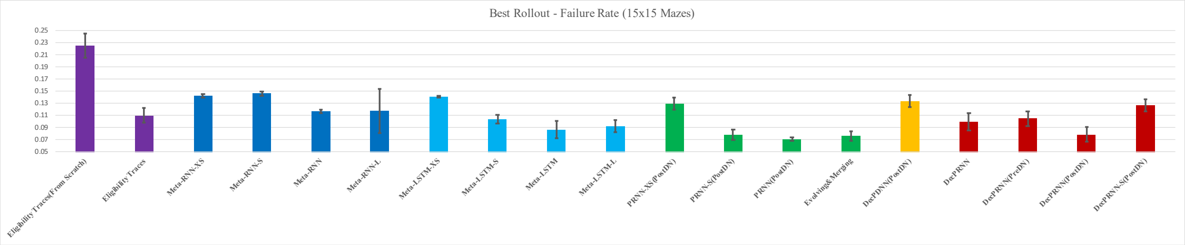
(a) 9×9 best-rollout rewards



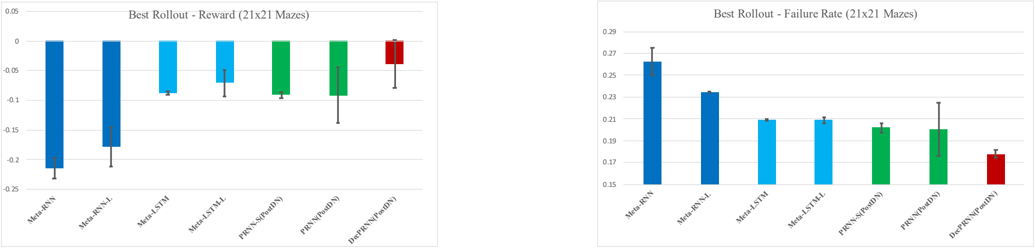
(b) 9×9 best-rollout failure rates



(c) 15×15 best-rollout rewards

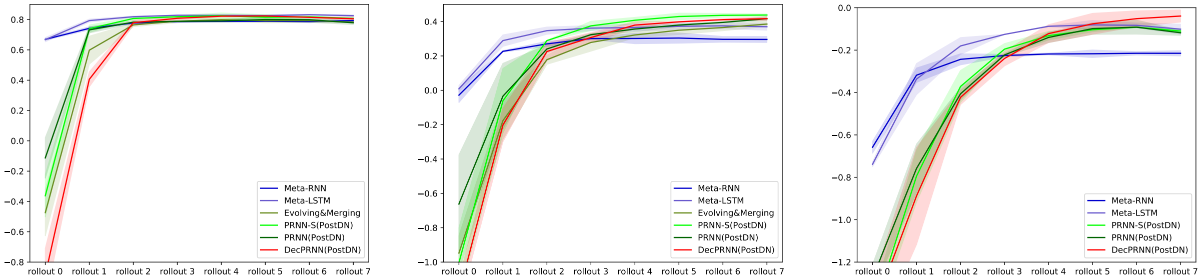


(d) 15 × 15 best-rollout failure rates

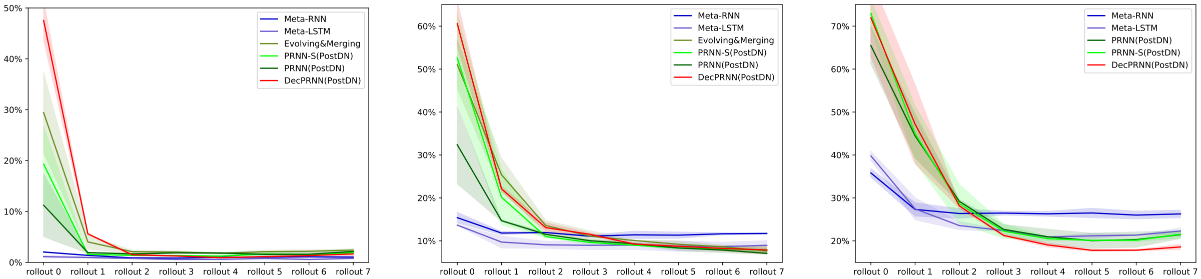


(e) 21 × 21 best-rollout rewards (f) 21 × 21 best-rollout failure rates

Figure 4: Best-rollout performances (including the rewards and failure rates) of all compared methods.



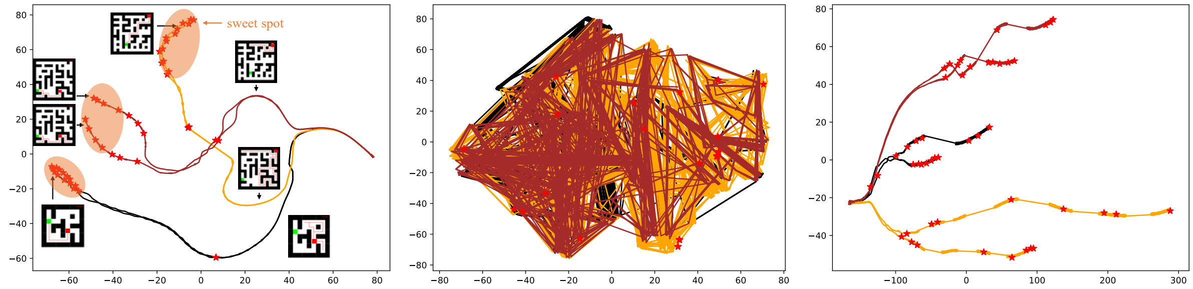
(a) 9 × 9 rewards (b) 15 × 15 rewards (c) 21 × 21 rewards



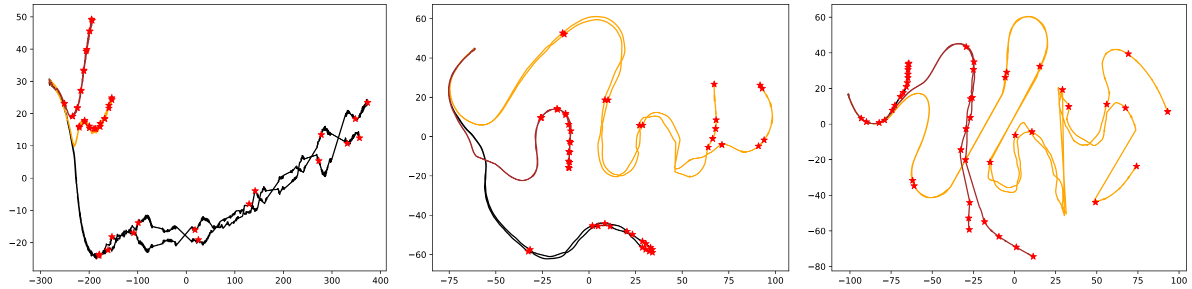
(d) 9 × 9 failure rate (e) 15 × 15 failure rate (f) 21 × 21 failure rate

Figure 5: Per-rollout performances of selected methods for 8-rollouts life cycles.

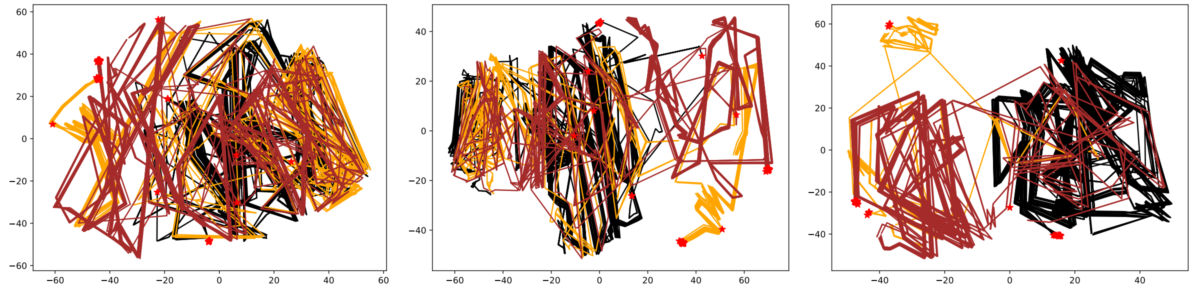
(0th rollout) and asymptotic performances (7th rollout): The model-based learners typically start at a good level but stop growing at 3rd rollouts and suffer from a relatively low ceiling. In contrast, the plasticity-based learner starts from scratch but keeps growing through its life cycle to eventually a higher level. Also, we observe the different levels of growth in different mazes. For simpler 9 × 9 mazes, most methods stop growing after their 3rd rollouts. For complex 21 × 21 mazes, we can see signs of growing even at the last rollouts regarding the best plasticity-based learners.



(a) $W_{h,t}^{(p)}$ - DecPRNN(PostDN) (b) h_t - DecPRNN(PostDN) (c) $W_{h,t}^{(p)}$ - DecPRNN



(d) $W_t^{(p)}$ - DecPDNN (e) $W_{h,t}^{(p)}$ - PRNN(PostDN) (f) $W_{h,t}^{(p)}$ - Evolving&Merging

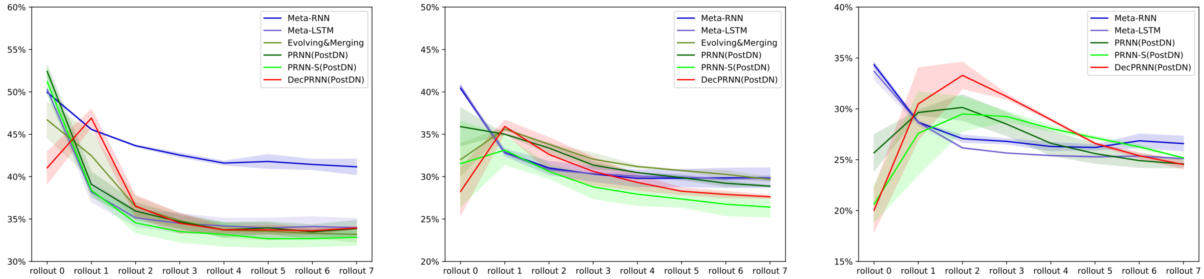


(g) h_t - Meta-LSTM (h) c_t (cell states) - Meta-LSTM (i) h_t - Meta-RNN

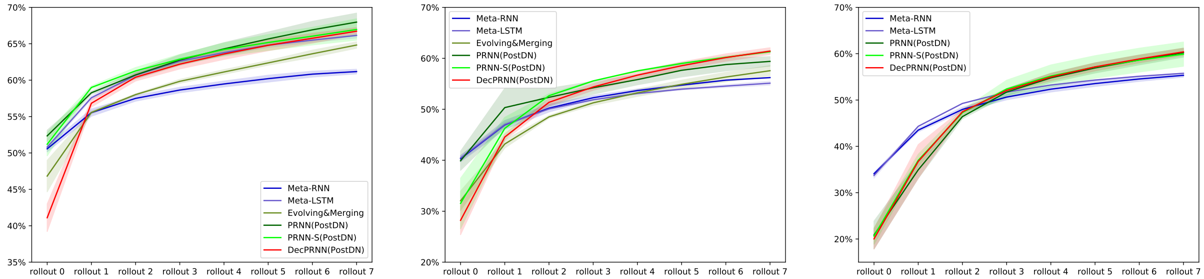
Figure 6: t-SNE visualization of the transformation of the connection weights ($W_{h,t}^{(p)}$) and hidden states (h_t) in various methods. Each trajectory corresponds to the trajectory of the vector in a unique maze shown in (a). The red ★ marks the end of a rollout.

Visualizing Adaptive Components We present the development of the adaptive components within the agents' life cycles, including hidden states (h_t, c_t) and the plastic neural connection weights ($W_{h,t}^{(p)}$). We first sample 3 different tasks. For each task, we run the life cycle of different methods twice, which yields 6 trajectories of adaptive variables for each method. Since those tensors are in relatively high dimensions, we run t-SNE visualization to map them to 2-D space and show their temporal traces in Figure 6. In Figure 6(a), we also show the correspondence of the traces to the actual trajectories in the maze. Comparing Figure 6(a) and (b), we see that the plastic neural connection weights behave differently from hidden states. The connection weights seem to keep the long-term information related to the overall task configurations. The hidden states seem to keep the short-term information only. It becomes clearer if we compare Figure 6(a,b) with Figure 6(g,h,f): In Meta-RNN and Meta-LSTM, the hidden states seem to catch both long-term and short-term information at the same time, resulting in both short-term vibrations and long-term migrations. It could possibly be used to explain why plasticity-based learners do better than model-based learners in longer sequences and why combining recursion and plasticity can yield better performances compared with

DecPDNN(PostDN). The recurrent neural states might be less capable of preserving long-term memorization. It becomes even more challenging as both short-term and long-term information must be preserved in the same vector space. In contrast, in plastic RNN, we conjecture that the slowly changing plastic neural connections keep the long-term information while leaving the short-term information to recurrence. One more point worth mentioning is the "sweet spots" of the plastic neural connections for different tasks, which can be easily discriminated from Figure 6(a). For non-modulated DecPRNN (Figure 6(c)) and DecPDNN(PostDN) (Figure 6(d)) it seems more difficult to find those "sweet spots", implying that their inner-loop learning might have not converged. Also, we observe some spikes in the traces of non-modulated DecPRNN, indicating that the learning process is noisier without neural modulation.



(a) 9 × 9 per-rollout Exploration (b) 15 × 15 per-rollout Exploration (c) 21 × 21 per-rollout exploration



(d) 9 × 9 accumulated exploration (e) 15 × 15 accumulated exploration (f) 21 × 21 accumulated exploration

Figure 7: The coverage rate of the visited grids showing the exploration of different agents in variant mazes.

Analyses on Explorations.It is well known that reinforcement learning depends on exploration and exploitation simultaneously. We are then interested in investigating whether the inner loop has learned to balance exploration and exploitation. We visualize the exploration of the agents by using the *coverage rate* of the mazes, which is the unique grids the agent visited divided by all the reachable grids in that maze. We plot both the per-rollout coverage rate, and the accumulated coverage rate (by counting the uniquely visited grids since the beginning of its life cycle) of meta-testing in Figure 7. There are several interesting points worth mentioning. First, we see that all model-based and plasticity-based learners learn to explore more at the beginning (meta-testing-training stage) and gradually reduce the exploration. Second, some plasticity-based methods, especially those obeying genomics bottleneck (Evolving&Merging and DecPRNN(PostDN)), are more deterministic in their first rollouts but then explore more in the 2nd and 3rd rollout. It is also consistent with Figure 6(a), where we show the traces of neural connections in DecPRNN(PostDN) for different tasks highly overlapped in the very beginning. A reasonable guess to this strange behavior is that the agent is experiencing some " warm-up stage" before actually learning about the maze since its initial neural connections can not support any complex behavior. Third, we found that by summing up the per-rollout explorations, the plasticity-based learners do not have the largest exploration every rollout but have the highest accumulated exploration. It could possibly mean that plasticity-based learners are more efficient in designing cross-episode exploration strategies.

5 Limitations and Future Prospects

Currently, typical large-scale deep models work with mostly static components and few adaptive components. They have been powerful in pre-defined tasks but suffered from high customization costs and the inability to generalize to variant scenarios with little human efforts. In this paper, we suggest designing models with relatively larger-scale adaptive components by getting inspiration from plasticity and genomics bottleneck. Those models are expected to be not necessarily capable of everything initially but capable of learning to accomplish very complex tasks by interacting with the environments and learning automatically.

There are several limitations to this work. First, although our environments have probably been the most challenging 2D maze tasks that have been addressed until now, it is still far too simple compared with many tasks in reality. It is also to be validated in image-related studies. We believe building diverse and close-to-reality simulators is essential for the future development of AI. Recently much effort has been put into this topic (Chevalier-Boisvert et al., 2018; Yu et al., 2018), but there is still a long way to go. Second, we believe our outer-loop optimizer (the Seq-CMA-ES) is a good choice at the current stage, but it still severely limits the model design. Plasticity rules will be more prospective if combined with more sophisticated models, but the meta-training turns out to be the bottleneck. Meanwhile, the genomics bottleneck can alleviate this problem since it means fewer meta-parameters and thus less burden on meta-training, but it is not without limits. To build valuable AIs for reality might still cost millions of meta-parameters calculated on inner loops of millions of steps and maintaining hundreds of millions of adaptive variables. Third, we have mainly tested our settings in a meta-learning framework. An essential setting will be enabling one single agent to adapt to non-stationary tasks continually during its lifetime without forgetting those old ones. This should be a direction for future investigation.

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.

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.

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

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

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

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.

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 *International Conference on Learning Representations*, 2019.

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.

Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009.

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.

Sebastian Risi and Kenneth O Stanley. Indirectly encoding neural plasticity as a pattern of local rules. In *International Conference on Simulation of Adaptive Behavior*, pp. 533–543. Springer, 2010.

Sebastian Risi and Kenneth O Stanley. A unified approach to evolving plasticity and neural geometry. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2012.

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*, 2017a.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017b. URL `http://arxiv.org/abs/1703.03864`.

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.

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.

Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

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

Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

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 Details of The Model Structures

All the compared methods have a static output layer mapping the hidden units to 5-dimension actions. The difference lies in the rest part that maps the inputs to the hidden units. We list the meta-parameters and adaptive components of all the compared methods in Table 1.

Table 1: Model Structures of Compared methods

	Hidden Units	Meta Parameters ($\|\theta\|$)	Meta Parameters ($\|\phi\|$)
DNN	64+64	0	5,125
Meta-DNN	64+64	0	5,509
Meta-RNN-XS	8	8	237
Meta-RNN-S	16	16	597
Meta-RNN	64	64	5,445
Meta-RNN-L	128	128	19,077
Meta-LSTM-XS	8	16	813
Meta-LSTM-S	16	32	2,133
Meta-LSTM	64	128	20,805
Meta-LSTM-L	128	256	74,373
Evolving&Merging	64	5,120	1,347
Eligibility Traces	64	4,160	5577
Eligibility Traces(From Scratch)	64	4,160	1481
PRNN-XS(PostDN)	8	192	809
PRNN-S(PostDN)	16	512	2,119
PRNN(PostDN)	64	5,120	20,743
DecPDNN(PostDN)	64+64	5,056	1,411
DecPRNN	64	5,120	1,217
DecPRNN(PreDN)	64	5,120	1,379
DecPRNN-S(PostDN)	32	1,536	707
DecPRNN(PostDN)	64	5,120	1,347

A.2 Meta-Training Settings

To calculate the fitness of an agent regarding its life cycle of 8 rollouts, we apply $Fit(\phi, T) = \sum_z w_z \cdot R_z$, with w_z be

$$w_z = \begin{cases} 0 & z < 2 \\ 0.80^{(\tau - z - 1)} & else \end{cases}$$

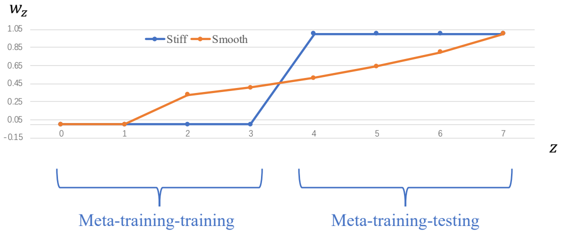


Figure 8: “stiff” and “smooth” training and testing phases in meta-learning.

Here $\tau = 8$ denotes the episodes in the life cycle. This formula is slightly different from canonical meta-supervised-learning settings, where $w_z = 0$ for meta-training-training (or support set) and $w_z = 1$ for meta-training-testing (or query set). The canonical training and testing phase settings guide the meta-training to improve the performance on the query set only. Though this "stiff" setting can be extended to meta-RL Finn et al. (2017), it is actually more difficult to decide the boundary between meta-training-training and meta-training-testing in meta-RL since we want the agent to improve its performance gradually through its lifetime. Thus, we propose this "smooth" setting for training and testing phases in the inner loop (Figure 8). In our cases, we found it to be more effective than "stiff" settings. For the outer-loop optimizer (seq-CMA-ES), we use an initial step size of 0.01, and the covariance $C = \mathbb{I}$ for all the compared methods.

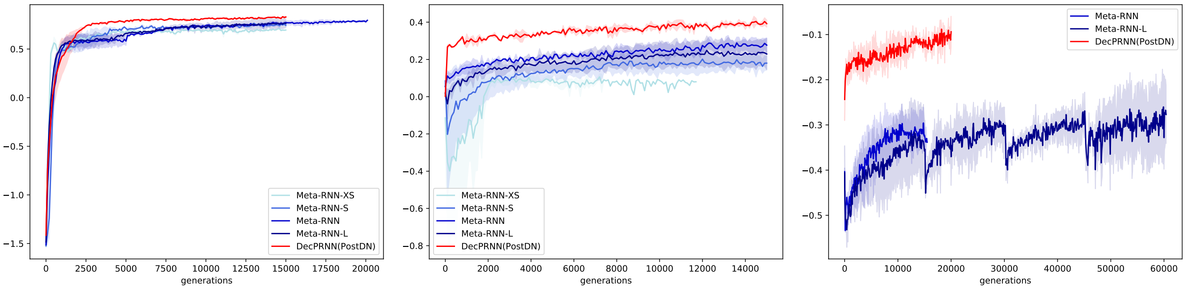
A.3 Additional Results

A.3.1 Performances of DNN and Meta-DNN

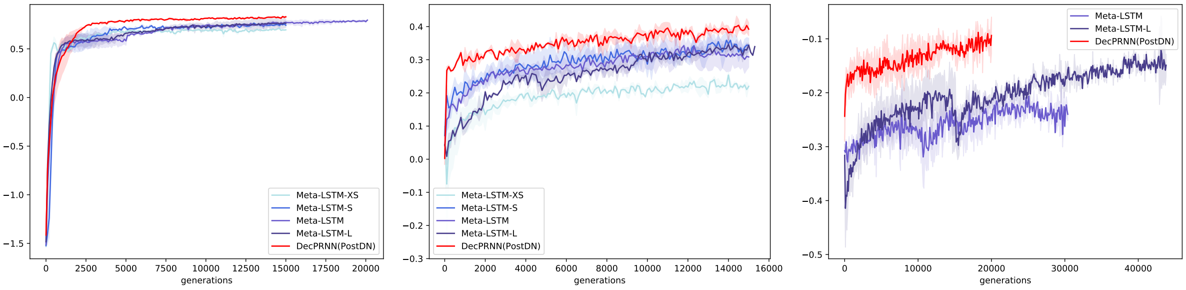
Table 2: Best-Rollout performance (rewards) of DNN and Meta-DNN, along the performances of random policy and oracles.

	Maze 9×9	Maze 15×15	Maze 21×21
DNN	-0.860 ± 0.038	-1.603 ± 0.025	
Meta-DNN	0.480 ± 0.070	-0.506 ± 0.018	
Random	-1.308 ± 0.031	-1.934 ± 0.010	-1.976 ± 0.005
Oracle	0.908	0.820	0.781

A.3.2 Convergences of Meta-Training

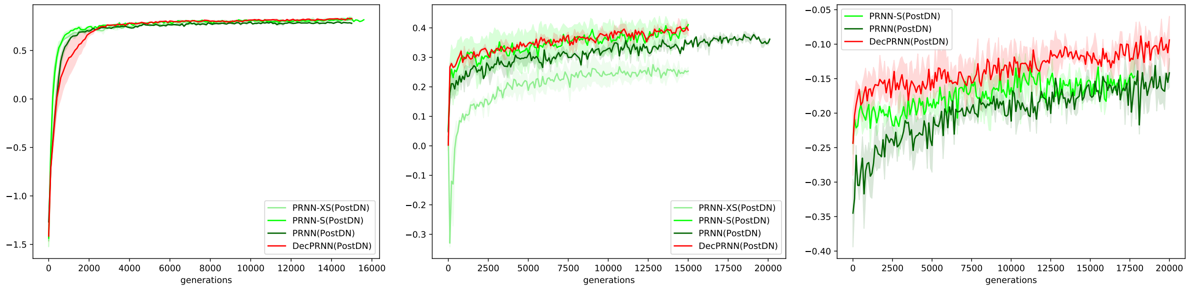


(a) 9×9 (b) 15×15 (c) 21×21

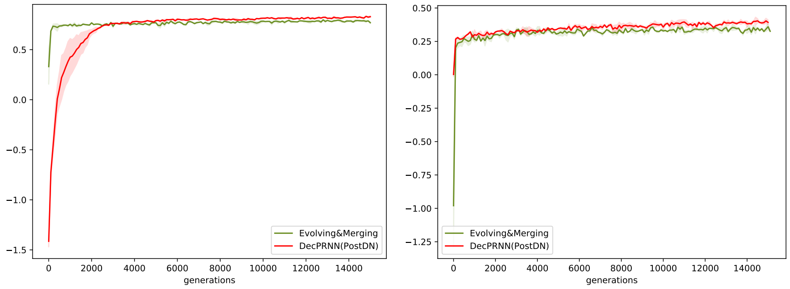


(d) 9×9 (e) 15×15 (f) 21×21

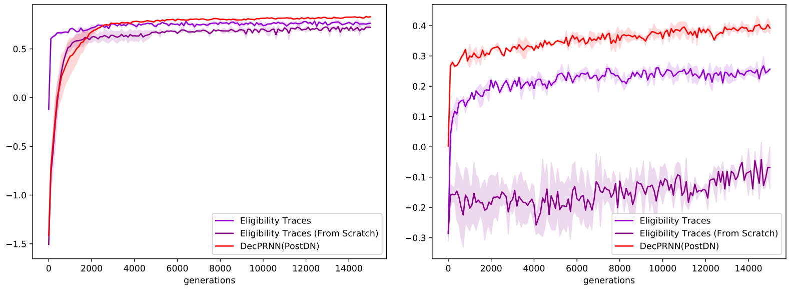
We present the average fitness score evaluated on validating tasks ($\mathcal{T}_{valid}$) against the evolved generations of all the compared methods in Figure 7. For clarity, we group model-based and plasticity-based learners into 6 groups and show each group independently to avoid confusion. For comparison, we also add DecPRNN(PostDN) in each group. Notice that we follow a curriculum learning process for the meta-training.



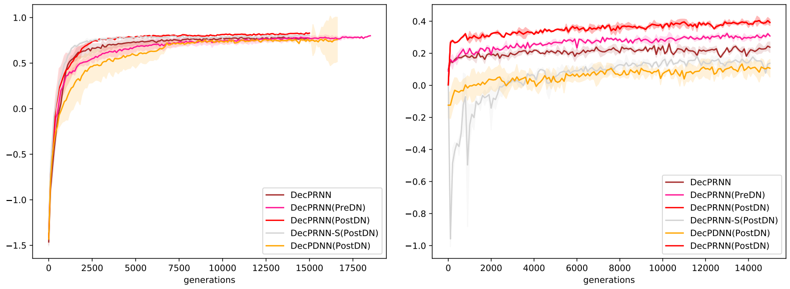
(g) 9 × 9 (h) 15 × 15 (i) 21 × 21



(j) 9 × 9 (k) 15 × 15



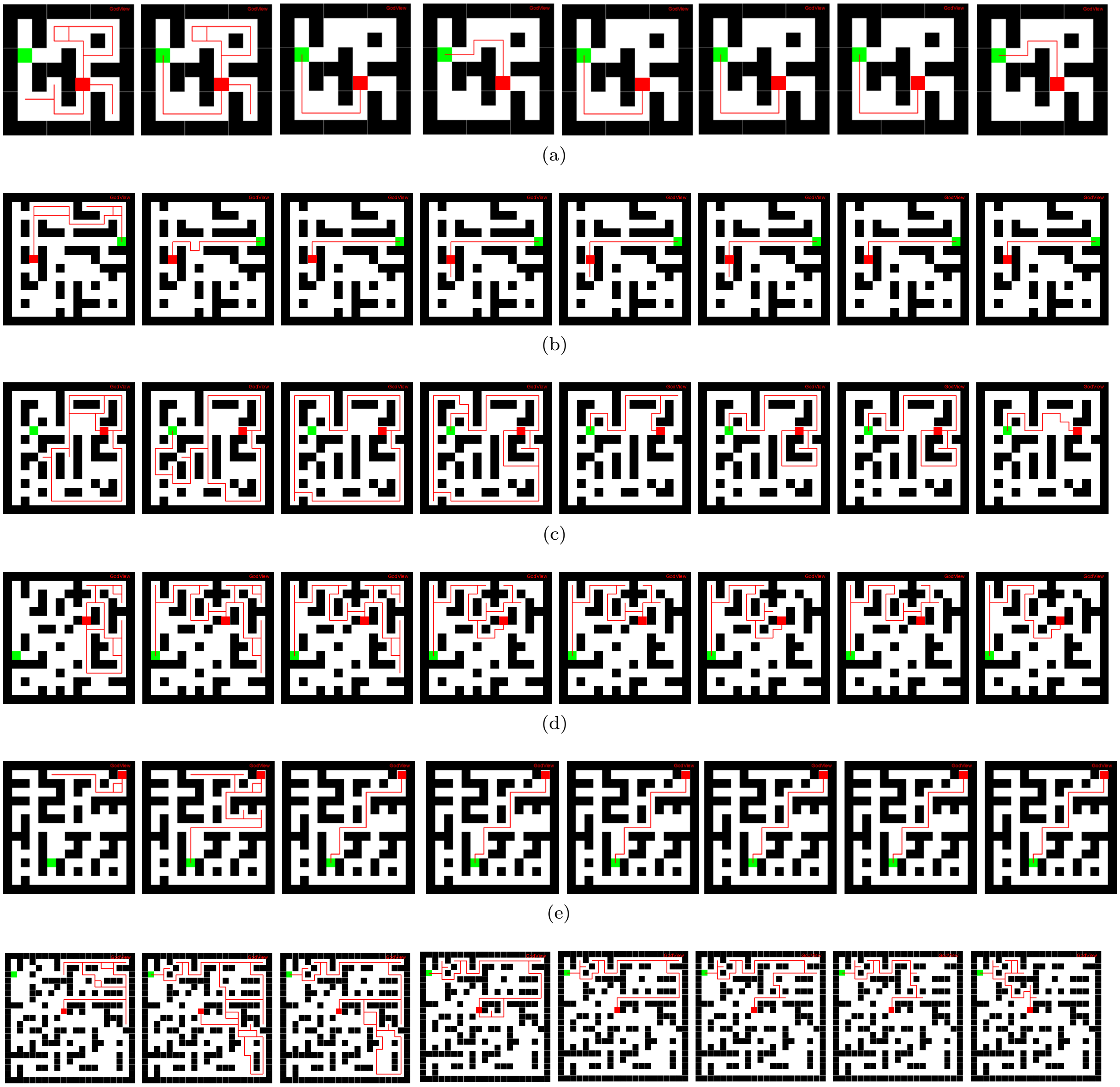
(l) 9 × 9 (m) 15 × 15



(n) 9 × 9 (o) 15 × 15

Figure 7: Plotting the mean and variance of the fitness score in validating tasks.

We train the learners on 9 × 9 mazes first and apply a warm start in 15 × 15, and 21 × 21 sequentially to reduce the meta-training cost. Also, it is not fair to compare DecPRNN(PostDN) with Evolving&Merging directly since we must start from a well-trained PRNN(PostDN) model in Evolving&Merging. Also, we occasionally reset the covariance to avoid some pitfalls. Generally, we found that DecPRNN(PostDN) is not only one of the bests in performances but also converges fastly thanks to fewer meta-parameters. Model-based learners such as Meta-LSTM are generally harder to reach convergence. We sometimes run models longer than the other models in case we judge the convergence is yet far.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 8: Example trajectories of the DecPRNN (PostDN) agents in each of the 8 rollouts in different mazes (9 × 9, 15 × 15, and 21 × 21). The red squares mark the start position, the green square marks the goal, and the red lines denote the agents' trajectories.

A.3.3 Lengths of the Life Cycles

The life cycle of the agents lasts for eight episodes. Each episode has at most 200 steps. The agent takes 1600 steps at most, but it is shortened as the agent performs better. The actual steps of a life cycle vary for different learners and different meta-training stages. In Figure 9 we show the summarized steps of the life cycle of the DecPRNN(PostDN) learner against the evolved generations.

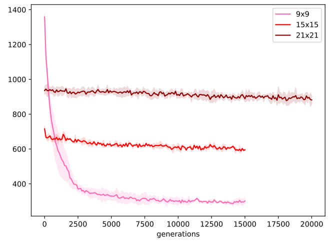


Figure 9: Summarized steps of a life cycle of DecPRNN(PostDN) agent in $\mathcal{T}_{valid}$

A.3.4 Case Demonstration

We randomly sample several mazes and show the actual trajectory of the agents of each rollout in Figure 8. We deliver cases where the agents do or do not find the global optimum. This could give an intuitive interpretation of inner-loop learning. We can also observe behaviors of *exploitation* to maintain high performance in the current rollout and *exploration* to reveal better routes for the following rollouts. For instance, the agent tends to explore new directions in case its previous rollout is not successful enough and take the shortcut discovered in the previous rollout.