Do What Nature Did To Us: Evolving Plastic Recurrent Neural Networks For Task Generalization

Fan Wang[†], **Hao Tian**[†], **Haoyi Xiong**[†], **Hua Wu**[†], **Yang Cao**[‡], **Yu Kang**[‡], **Haifeng Wang**[†]
† Baidu Inc.
‡ Department of Automation, University of Science and Technology of China
†{wang.fan, tianhao, xionghaoyi, wu_hua, wanghaifeng}@baidu.com
‡{forrest, kangduyu}@ustc.edu.cn

Abstract

While artificial neural networks (ANNs) have been widely adopted in machine learning, the gaps between ANNs and biological neural networks (BNNs) attracts attention increasingly. In this paper, we propose a framework named as *Evolutionary Plastic Recurrent Neural Networks* (**EPRNN**). Inspired by BNN, **EPRNN** composes Evolution Strategies, Plasticity Rules, and Recursion-based Learning all in one meta learning framework for generalization to different tasks. More specifically, **EPRNN** incorporates with nested loops for meta learning — an outer loop searches for optimal initial parameters of the neural network and learning rules; an inner loop adapts to specific tasks. In the inner loop of **EPRNN**, we effectively attain both long term memory and short term memory by forging plasticity with recursion-based learning mechanisms, both of which are believed to be responsible for memristance in BNNs. The inner-loop setting closely simulate that of BNNs, which neither rely on gradient based optimization methods nor require the exact forms of learning objectives. To evaluate the performance of **EPRNN**, we carry out extensive experiments in two groups of tasks: *Sequence Predicting*, and *Wheeled Robot Navigating*. The experiment results demonstrate the unique advantage of **EPRNN** compared to state-of-the-arts based on plasticity and recursion while yielding comparably good performance against deep learning based approaches in the tasks. The experiment results suggest the potential of **EPRNN** to generalize to variety of tasks and encourage more efforts in plasticity and recursion based learning mechanisms.

1 Introduction

Artificial Neural Networks (ANNs) achieve huge success in machine learning. Despite being initially inspired by Biological Neural Networks (BNNs), the optimization paradigm for ANNs has diverged from that of BNNs. BNNs are believed to learn through plasticity rules (Gerstner et al., 1993), among which Hebb's rule (Hebb, 1949) is most well known. In contrast, the optimization of ANNs is dominated by gradient descent methods, the evidences of which are less revealed in BNNs . Though up to now gradient descent methods are the most efficient optimizers for ANNs, their limitations are beginning to attract attentions, including the problems of catastrophic forgetting, over consumption of data, and the requirement for manual efforts in designing target functions. Those challenges are becoming the main impedance for further development of machine intelligence.

Recent studies show that under proper meta-parameter optimization, both recursion (Santoro et al., 2016; Mishra et al., 2018) and plasticity (Soltoggio et al., 2008; Najarro & Risi, 2020) can be valid alternatives to train ANNs for task generalization. Unlike gradient descending, both of the two learning mechanisms work in an unsupervised manner, without requiring exact forms of learning objectives, which are believed to more closely simulate BNNs.

Preprint. Under review.

Recursion based learning typically employs recurrent neural networks (*RNN*), *LSTM* (Hochreiter & Schmidhuber, 1997), and self-attention (Mishra et al., 2018; Chen et al., 2021) layers as learners. Learning takes place within the feed forward pass. The information are updated in the hidden states instead of the parameters. Recursion are found to be extremely sample efficient in generalized supervised tasks (Santoro et al., 2016), zero-shot generalization in language (Brown et al., 2020), and reinforcement learning (Mishra et al., 2018; Chen et al., 2021) when compared with various type of gradient descent methods. So far among those learners, though self-attention based learners such as *Transformers* have state-of-the-art performance, the computational costs for inference per step grow linearly with the sequence length, which makes them only available to relatively short sequences. On the other hand, recurrent learners such as *RNN* and *LSTM* have constant per-step inference costs, but suffer form poor asymptotic performances. That is, when sequences are getting longer, performances no longer improve or even deteriorate. This is partly due to the limitation of the memory spaces. For instance, an recurrent hidden layer of size n has a memory of scale $O(n)$. In contrast, its parameters scale with $O(n^2)$, which makes parameter-updating more powerful as learning mechanisms than recursion-only.

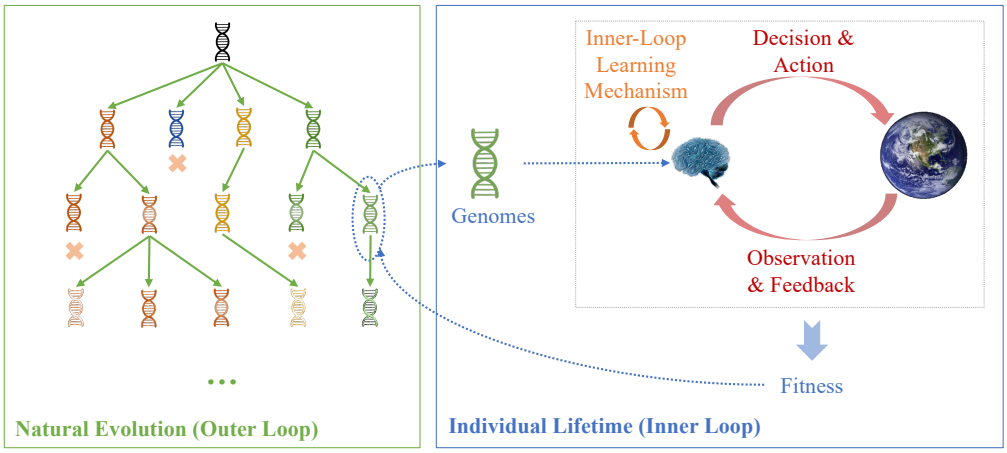


Figure 1: An illustration of the natural evolution: The evolution takes place in the outer loop, where the genomes are mutated and selected, and the population either thrive or become extinct based on the *Fitness*. Lifetime of each individual composes the inner loop. At the beginning, the genomes decide the learning mechanisms and initial neural configurations of the brain in the new born life. As the neural networks interact with the environment through actions and observations, its connections and hidden neuron states are further updated to better adapt to the environment. Plasticity are believed to be important part of the learning mechanisms. The fitness is dependent on the learning and adapting capability of each individual.

Prior to our work, evolving plasticity (Soltoggio et al., 2008, 2018; Lindsey & Litwin-Kumar, 2020; Yaman et al., 2021) has been proposed to reproduce the natural evolution and plasticity in simulation, as shown in Figure 1. Implementing plasticity is not straight-forward, unlike gradient-descent, plastic rules are not universal but have to be optimized beforehand, which is not possible without a further outer-loop optimizer over the inner-loop learning. Evolutionary algorithms (Zhang et al., 2011) are typically applied in the outer loop to search for meta-parameters shaping the learning rules, which can be regarded as information carried by genomes during evolution. Those optimized plasticity rules are then applied in the inner loop to further tune the parameter of NN for better adaptions to the environment. Due to the complexity in training, plasticity is insufficiently investigated and its potential is not well understood. Existing works on evolving plasticity are mainly focused on feed-forward layers. However, as recursion widely exists in BNNs, it is desirable to also investigate plasticity in recurrent connections.

Another motivation of combining recursion with plasticity is to simultaneously possess the merits of updating information in *hidden states* and in *connection weights*. Both recursion and plasticity mechanisms are believed to be relevant to memristance. While recursion that updates *hidden states* only is likely to be related to short-term memories, synaptic plasticity that updates *connection weights* is more related to long-term memories (Bliss & Lømo, 1973; Mel, 2002; Linares-Barranco & Serrano-Gotarredona, 2009). Based on this motivation, we propose a new learning framework named as

Methods	Inner Loop	Outer Loop
Memory Augmented NN, (Santoro et al., 2016)	Recursion	Gradient
MAML, (Finn et al., 2017)	Gradient	Gradient
Reptile, (Nichol et al., 2018)	Gradient	Average
Conditional Neural Processes, (Garnelo et al., 2018)	Average	Gradient
SNAIL, (Mishra et al., 2018)	Attention	Gradient
ES-MAML, (Song et al., 2019)	Gradient, Hill Climbing	Evolution
EPMLP, (Najarro & Risi, 2020)	Plasticity	Evolution
EPRNN (ours)	Plasticity & Recursion	Evolution

Table 1: A brief review of meta learning methods

Evolutionary Plastic Recurrent Neural Networks (**EPRNN**). Specifically, we apply plastic rules in recurrent connections by utilizing the information of time-dependent hidden states. Then, following the natural evolution of intelligence (Fig. 1), in the outer loop we employ evolution strategies to search for proper initial parameters and plastic rules. To validate our proposal, we carry out experiments on two groups of tasks: *Sequence Predicting* and *Wheeled Robot Navigating*. We compared the proposed framework with the other recursion and gradient based learners. The experiment results show that **EPRNN** can surpass decent models such as LSTM using the most naive recurrent structures. Although the validated model structure is relatively simple, we believe that validating such prototypes is an important step toward more general cases.

2 Related Works

2.1 Meta Learning

Meta learning aims at building learning machines that gain experience using task-specific data over a distribution of tasks. Inspired by human and animal brains that are born with both embedded skills and capability of acquiring new skills, meta learning implements two nested learning loops: The outer learning loops optimize the meta that typically involves initial parameters (Finn et al., 2017; Song et al., 2019), learning rules (Zoph & Le, 2017; Najarro & Risi, 2020; Pedersen & Risi, 2021), and model structures (Soltoggio et al., 2008; Li & Malik, 2016) over a distribution of tasks; The inner learning loops adapt the model to specific task by utilizing the meta. According to different inner-loop optimizers, we roughly classify the methods into *model-based* and *parameter-updating* methods. The *model-based* methods do not update the parameters in the inner-loop, where learning is accomplished by updating the hidden states (e.g., recursion); The *parameter-updating* methods update the parameter in the inner-loop (e.g., gradient descent (MAML), plasticity). From this point of view, our method can be classified into both groups. A brief review of the typical meta learning paradigms is presented in Table 1.

2.2 Evolving Plasticity

The proposal of the learning mechanism of BNNs is initially raised by Hebb's rule (Hebb, 1949), the most well-known part of which is "neurons fire together wire together". It is further polished by Spike Time-Dependent Plasticity (STDP) (Gerstner et al., 1993) indicating that the signal of learning is dependent on the temporal patterns of the pre-synaptic spike and post-synaptic spike. Learning could also appear in inhibitory connections, also known as anti-hebbian (Barlow, 1989). Also, relationships between STDP and memristance are also investigated (Linares-Barranco & Serrano-Gotarredona, 2009). Since many of those rules are related to spiking neural networks (Ghosh-Dastidar & Adeli, 2009), to apply them to ANNs, simplifed rules are proposed (Soltoggio et al., 2008) instead: given the pre-synaptic neuron state X and post-synaptic neuron state Y, the connections between X and Y are updated by

$$\delta W = m[A \cdot XY + B \cdot X + C \cdot Y + D], \tag{1}$$

m is the output from neuron modulators that adjust the learning rates of plasticity. Most of the existing rules are sub-classes of Equation 1. For instance, some works neglect the neural modulator m (Najarro & Risi, 2020; Miconi et al., 2018), others have set B, C, and D to 0 (Miconi et al., 2018, 2019). The learned rule A, B, C, D will inevitably be dependent on initial parameter of W, however,

learning plastic rules that is not dependent on the initial parameters was also investigated (Najarro & Risi, 2020; Yaman et al., 2021).

3 Algorithms

Problem Settings. We consider an agent (learner) that is dependent on meta-parameter θ. It has the capability of adapting itself to a distribution of tasks $T_j \in \mathcal{T}$ by interacting with the environment T_j through observation i_t and action a_t. In K-shot learning, the agent is allowed to first observe samples of length K (this stage can be referred as *meta-training-training*, see Beaulieu et al. (2020)), then its fitness is calculated in *meta-training-testing* rollouts. In Generalized Supervised Learning tasks (**GSL**), the observations typically include features (x_t) and labels (y_t) in *meta-training-training* stage ($i_t = \{x_t, y_t\}$), and the labels are left out for predicting in the *meta-training-testing* stage (Santoro et al., 2016; Garnelo et al., 2018). In Generalized Reinforcement Learning tasks (**GRL**), the observations typically include states (s_t), actions (a_{t-1}), and feedbacks (r_{t-1}) ($i_t = \{s_t, a_{t-1}, r_{t-1}\}$, sometimes r_{t-1} can not be observed) (Mishra et al., 2018). The goal of *meta-training* is to optimize θ such that the agent achieves higher fitness in *meta-training-testing*. In *meta-testing*, similarly, the learned parameters are given *meta-testing-training* and *meta-testing-testing* in order, the performances in *meta-testing-testing* are evaluated.

Plastic Recurrent Neural Networks (PRNN). Given a sequence of observations $i_1, ..., i_t, ...$, we first consider an recurrent neural network (*RNN*) that propagates forward and yields sequence of outputs a_t following:

$$h_{t+1} = \sigma(W_t \cdot h_t + W_i \cdot i_t + b), \tag{2}$$

$$a_t = f(h_{t+1}) \tag{3}$$

where h_t is the hidden states at step t. In PRNN, we kept W_i stationary, but we set W_t to be plastic, so that we add a subscript t to mark different W_t at different steps. Regarding h_t as pre-synaptic neuron states, and h_{t+1} as post-synaptic neuron states, by applying Equation 1, we update W_t with:

$$W_{t+1} = W_t + \delta W_t \tag{4}$$

$$\delta W_t = W_A \odot (\hat{h}_{t+1} \cdot h_t^{\mathrm{T}}) + W_B \odot (m_t \cdot h_t^{\mathrm{T}}) + W_C \odot (\hat{h}_{t+1} \cdot \mathbf{1}^{\mathrm{T}}) + W_D \cdot m_t \tag{5}$$

$$\hat{h}_{t+1} = m_t \odot h_{t+1}, \tag{6}$$

where we use $\odot$ and $\cdot$ to represent "element-wise multiplication" and "matrix multiplication" respectively. h and $\mathbf{1}$ are column vectors. W_A, W_B, W_C, W_D are collection of plastic rules of A, B, C, D from Equation 1, which has the same shape as W_t. m_t is the neural modulators that adjusts the learning rates of plasticity. We calculate m_t by applying a neuron modulating layer denoted with:

$$m_t = \sigma(W_h^{(m)} \cdot h_t + W_i^{(m)} \cdot i_t + b^{(m)}). \tag{7}$$

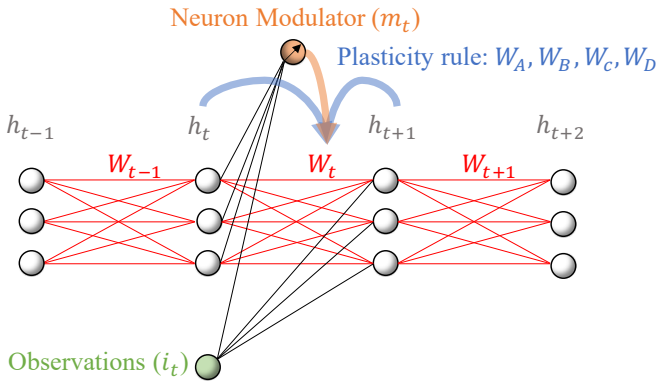


Figure 2: A sketch of the information flow in plastic recurrent neural networks (PRNN)

A sketch of PRNN is presented in Figure 2. The main difference between PRNN and naive *RNN* is that PRNN updates both the hidden states and the connection weights during the forward pass.

Evolving PRNN. Given task $T_j \in \mathcal{T}$, by continuously applying Equation 2 to 7 over *meta-training-training* and *meta-training-testing*, the fitness is eventually dependent on the initial parameters, learning rules, and the sampled task T, which is denoted as:

$$\textit{Fit}(\theta, T) = \textit{Fitness}(i_{K+1}, a_{K+1}, i_{K+2}, a_{K+2}, ...,) \tag{8}$$

$$W_i, W_0, W_A, W_B, W_C, W_D, W_h^{(m)}, W_i^{(m)}, b, b^{(m)} \in \theta \tag{9}$$

Following Evolution Strategies (ES) (Salimans et al., 2017), in kth outer-loop iteration, we sample different tasks from $\mathcal{T}$, and meta-parameters $\theta_{k,i}$ ($i \in [1, n]$) from the neighbourhoods of θ_k. We evaluate the fitness of sampled meta-parameters, and update the meta-parameters by applying:

$$\theta_{k+1} = \theta_k + \alpha \frac{1}{n} \sum_{i=1}^{n} \textit{Fit}(\theta_{k,i}, T_k)(\theta_{k,i} - \theta_k) \tag{10}$$

Why Recurrent Neural Networks? As stated in Equation 1, plasticity in feed-forward-only NNs allows NNs to gain experiences from single-frame observation only. In cases of non-sequential GSL, the plasticity has chances to tune the connection weights to the specific task by relying on observing one single frame of data ($i_t = \{x_t, y_t\}$), since its information of the feature and the supervision is complete. However, in general cases, learning can not happen without summarizing sequences of observations. For instances, a human driver getting used to a new car through continuously interacting and observing. Moreover, in GRL, there are time lag between the observation of states and feed-backs. Recursion helps to summarize historical observations to give the correct update for the connection weights.

Although, compared with naive *RNN*, there are obviously bunches of more sophisticated neural structure such as *GRU* and *LSTM*, we believe it is more desirable to start from simplest recurrent structure to study the potential of combining recursion and plasticity.

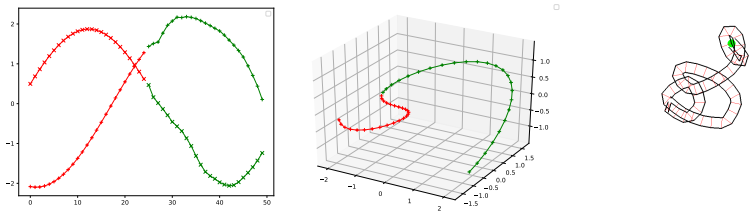
4 Experiments

4.1 Tasks for Evaluation

In generalized tasks, we have each task T_j dependent on some configuration parameters that are hidden from the agent. Below we introduce two groups of generalized tasks that we experiment on.

Sequence Predicting (Generalized Supervised Learning tasks). We randomly generate sequences of vectors $\boldsymbol{y}(t) = (y_1(t), ..., y_l(t))$, where $y_i(t) = A_i sin(\frac{2\pi t}{n_i} + \phi_i)$, $t = 1, 2, 3, ...$ and $A_i \sim \mathcal{U}(1, 3)$, $n_i \sim \mathcal{U}(10, 100)$, $\phi_i \sim \mathcal{U}(0, 2\pi)$, and $\mathcal{U}(a, b)$ represents the uniform distribution between a and b. A, n and ϕ are hidden from the agent. The front part of the sequence $\boldsymbol{y}(1), \boldsymbol{y}(2), ..., \boldsymbol{y}(K)$ is exposed to the agent, and the left part $\boldsymbol{y}(K+1), \boldsymbol{y}(K+2), ..., \boldsymbol{y}(N)$ is to be predicted. The fitness is the opposite of mean square error (MSE) between the predicted sequence and the ground truth. We test the methods for comparison in four groups of tasks including ($l = 1, K = 10, N = 20$), ($l = 1, K = 25, N = 50$), ($l = 3, K = 10, N = 20$), and ($l = 3, K = 25, N = 50$) (see Figure 3(a)(b)).

Wheeled Robot Navigating (Generalized Reinforcement Learning tasks). The agent is to navigate a two-wheeled robot to a randomly generated goal in 2-D space $\boldsymbol{g} = (g_x, g_y)$. We assume that there is a signal transmitter on the goal and a receiver on the robot. The robot observes the signal intensity decided by $A_t = A_0 - k \cdot log(d_t/d_0) + \epsilon$ (inspired by the attenuation for electromagnetic wave, see Friis (1946)), where d_t is the current distance between the robot and the goal, $\epsilon \sim \mathcal{N}(0, \sigma)$ is the white noise in the observation. g_x, g_y, A_0, k are environment related configurations that is hidden from the agent. For each task, we sample configurations by $g_x, g_y \sim \mathcal{U}(-0.5, 0.5)$, $A_0 \sim \mathcal{U}(0.5, 2.0)$, and $k \sim \mathcal{U}(0.1, 0.5)$. The action is the rotation speed of its two wheels that controls the orientation and velocity of the robot. The reward at each step is $r_t = -d_t$, an episode terminates when the robot approaches the goal or steps reaches the maximum of 100. We also hide its own position and orientation from the agent, such that the agent relies on recording its own action the signal strength A_t to locate itself. We investigate three types of navigating circumstances with different level of noises in the observed signal: Low Noise ($\sigma = 0.01$), Median Noise ($\sigma = 0.05$), and High Noise ($\sigma = 0.2$) (see Figure 3(c)).



(a) (b) (c)

Figure 3: Demonstration of the tasks.(a) Two tasks sampled from *Sequence Predicting* (l=1,K=25,N=25), the red lines are training sets and the green lines are testing sets. (b) One task sampled from *Sequence Predicting* (l=3,K=25,N=25). (c) A trajectory generated by an agent in a *Wheeled Robot Navigating* task.

4.2 Experiment Settings

We add the following methods into comparison, the methods share exactly the same outer loop and differ in the inner loop.

- **ES-MAML** (Song et al., 2019) : We use four gradient descent steps in the inner loop, the learning rate of each step is treated as meta-parameters which is to be optimized by the outer loop. As MAML can not utilize instant observation in zero-shot case, we show results of both **ES-MAML (zero-shot)** and **ES-MAML (one-rollout)** in *Wheeled Robot Navigating*. Except for **ES-MAML**, the other methods are measured with zero-shot *meta-testing* score only in *Wheeled Robot Navigating*.
- **ES-RNN**: Vanilla *RNN* as the inner loop learner.
- **ES-LSTM**: Long Short Term Memory (Hochreiter & Schmidhuber, 1997) as the inner loop learner.
- **EPMLP** (Soltoggio et al., 2018): Multi-Layer Perceptrons (MLP) with plasticity rules implemented.
- **EPMLP (Random)** (Najarro & Risi, 2020): The main difference with **EPMLP** is randomly setting the parameters of the plastic layers at the beginning of each inner loop instead of using fixed trainable initial parameters.
- **EPRNN (w/o m)**: Removing neuron modulator (m_t) from the plasticity rule in PRNN.

We add additional non-plastic fully connected layers before and after the plastic layers to increase the representation capability of the models. For fairness, we kept those layers identical for all compared methods. For every 100 outer loops in *meta training*, we add an extra *meta testing* epoch, evaluating the average fitness of current meta-parameters on testing tasks. Each run includes 15000 outer loops and 150 meta-testing epochs. Each result is concluded from independent 3 runs. Our code[1] relies on PARL[2]for parallelization. We leave the detailed illustration of the model structures and hyper-parameters in the Appendices.

4.3 Results

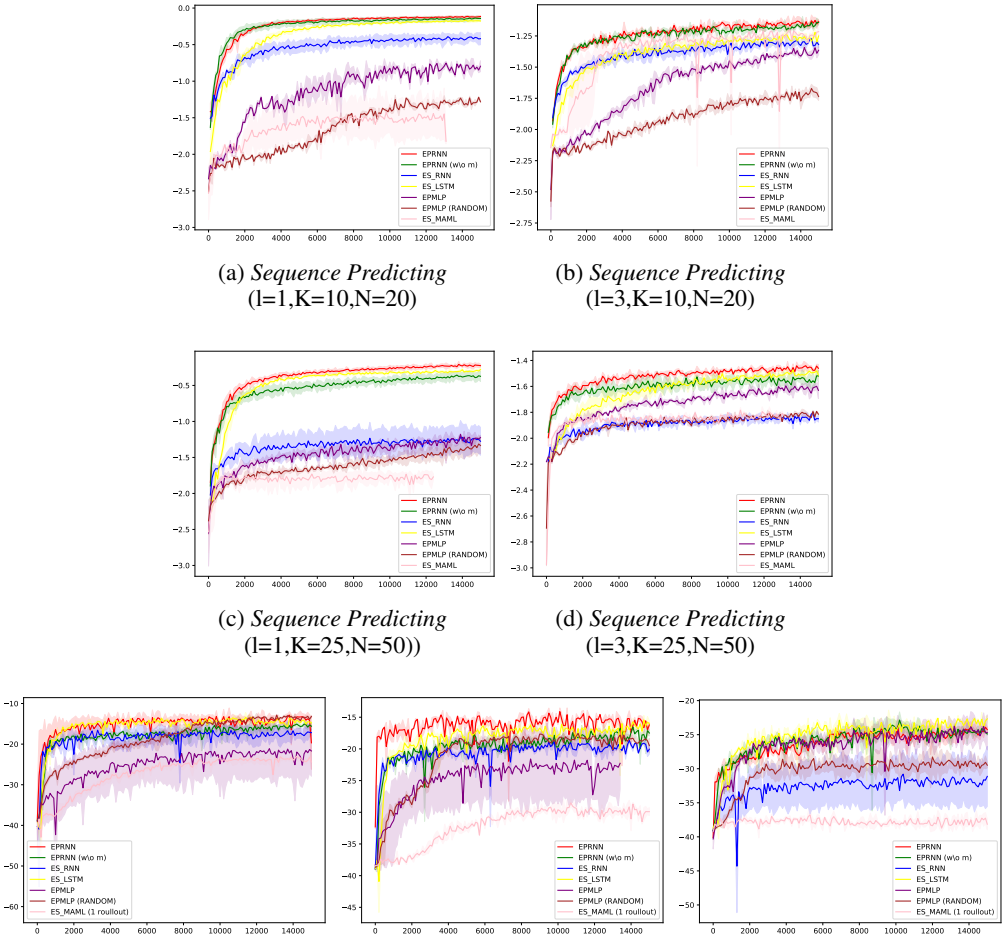
We present the experiment results in Figure 4 (In Table 2, and Table 3, we also list the summarized performance by averaging the Top-3 *meta-testing* scores in the latest 10 *meta-testing* epochs of each run over 3 independent runs). Generally, we can conclude that PRNN performs substantially better when compared with naive *RNN*. In some cases, it even produces better results compared with *LSTM*, despite the simpler model architecture. It is also interesting to notice that the gap between *RNN* and PRNN are smaller in shorter sequences or low-noise environments, but larger in more challenging tasks with longer sequence or higher noise (In *Wheeled Robot Navigating* tasks, higher noise pushes the agent to maintain a longer memory in order to filter the noise and figure out the way to goal). This

[1]https://github.com/WorldEditors/EvolvingPRNN

[2]https://github.com/PaddlePaddle/PARL

phenomenon reaffirms the lack of long-term memories in *RNN*, and shows that PRNN significantly improves this drawback. Comparing **EPRNN (w/o m)** with *RNN* and **EPRNN** clearly demonstrates that simple ABCD rule (without the neural modulator) may also work to some extent, but introducing the neuron modulator m_t can further benefit the learner.

Among plasticity based methods, we show that recursion is more advantageous than MLP in evaluated tasks. It is also worth noticing that **EPMLP** and **EPMLP (Random)** perform steadily beyond the gradient-based learner (**ES-MAML**). In **ES-MAML**, the gradient can only be calculated after an episode is completed, while **EPMLP** is able to perform sequential learning even though no feedback is available during its life time. This demonstrates the possibility of surpassing human-designed gradient-based learning rules with automatically learned unsupervised rules. Moreover, comparison between **EPMLP** and **EPMLP (Random)** validate the proposal of Najarro & Risi (2020), implying the possibility of discovering global plastic learning rules instead of rules coupled with the initial parameters. Yet we see optimization of the initial parameters is still advantageous, which can also be validated by evidences in nature that the newborn lives already have certain embedded skills (e.g., Newborn human babies have reflexes of suckling and grasping; Foals can stand shortly after being born).



(a) *Sequence Predicting* (l=1,K=10,N=20)

(b) *Sequence Predicting* (l=3,K=10,N=20)

(c) *Sequence Predicting* (l=1,K=25,N=50))

(d) *Sequence Predicting* (l=3,K=25,N=50)

(e) *Wheeled Robot Navigating* (Low Noise)

(f) *Wheeled Robot Navigating* (Median Noise)

(g) *Wheeled Robot Navigating* (High Noise)

Figure 4: Plotting *meta-testing* scores against *meta-training* iterations.

Table 2: Summarized performances comparison in *Sequence Predicting* tasks

Methods	l=1,K=10,N=20	l=1,K=25,N=50	l=3,K=10,N=20	l=3,K=25,N=50
ES-RNN	-0.385 ± 0.060	-1.228 ± 0.191	-1.273 ± 0.009	-1.811 ± 0.015
ES-LSTM	-0.165 ± 0.014	-0.283 ± 0.013	-1.229 ± 0.010	-1.475 ± 0.016
ES-MAML	-1.452 ± 0.292	-1.747 ± 0.064	-1.218 ± 0.013	-1.796 ± 0.013
EPMLP	-0.732 ± 0.031	-1.185 ± 0.045	-1.339 ± 0.004	-1.735 ± 0.013
EPMLP-(Random)	-1.233 ± 0.115	-1.319 ± 0.035	-1.489 ± 0.023	-1.788 ± 0.003
EPRNN	$\mathbf{-0.114 \pm 0.012}$	$\mathbf{-0.208 \pm 0.025}$	$\mathbf{-1.107 \pm 0.015}$	$\mathbf{-1.430 \pm 0.019}$
EPRNN (w/o m)	-0.135 ± 0.026	-0.351 ± 0.058	-1.128 ± 0.014	-1.520 ± 0.041

Table 3: Summarized performances comparison in *Wheeled Robot Navigating* tasks

Methods	Low Noise ($\sigma = 0.01$)	Median Noise ($\sigma = 0.05$)	High Noise ($\sigma = 0.2$)
ES-RNN	-16.90 ± 1.30	-18.99 ± 0.14	-31.23 ± 3.95
ES-LSTM	-14.04 ± 0.08	-15.50 ± 0.58	$\mathbf{-22.53 \pm 0.45}$
ES-MAML (zero-shot)	-37.18 ± 0.45	-37.91 ± 0.80	-37.90 ± 0.05
ES-MAML (1 rollout)	-23.10 ± 0.46	-29.21 ± 0.46	-37.12 ± 0.12
EPMLP	-21.24 ± 6.08	-16.65 ± 0.39	-23.75 ± 2.02
EPMLP (Random)	-13.02 ± 0.19	-18.25 ± 0.23	-28.85 ± 1.1
EPRNN	$\mathbf{-12.71 \pm 0.75}$	$\mathbf{-15.07 \pm 0.03}$	-23.79 ± 0.55
EPRNN (w/o m)	-14.93 ± 0.49	-16.72 ± 0.33	-24.11 ± 0.34

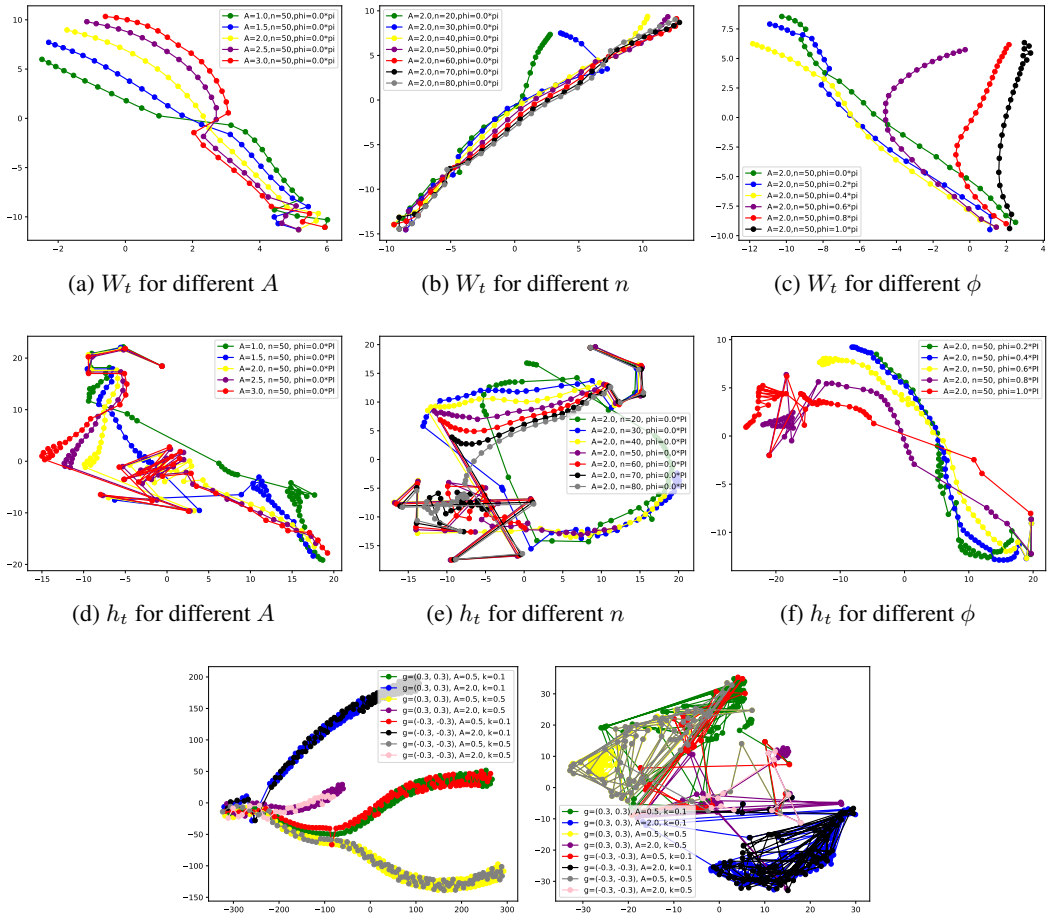
4.4 Analysis

To investigate whether plasticity rules update the connection weights as expected, we test the trained model with different tasks and record the updating trajectories of plastic connection weights W_t and hidden states h_t. We run t-SNE visualization to map those tensors (W_ts and h_ts) to 2-D space and show their trajectories in Figure 5. The connection weights W_t typically start from the same position and gradually move in different directions depending on the task configurations. The final weights effectively capture environmental configuration that was hidden from the agent. Particularly, in Figure 5(d) for *Wheeled Robot Navigating* tasks, we can see that W_t captures only the signal transmission patterns (A_0, k), but neglects the position of the goal (g_x, g_y). We guess that A_0, k are important stationary patterns that helps the agent to interpret the observed signal strength (A_t), while the absolute position of goal is less important as its relative position to the robot is changing continuously. This is a clear demonstration showing that plasticity has perform meaningful update on the connections weights depending on the tasks. On the other hand, the hidden states (Figure **??**(c)(d)(e)(h)) are noisier and less distinguishable among different tasks. The small repeated circles in the trajectory of hidden states might correspond to short-term patterns such as the periodicity of *Sequence Predicting* task and the circular movements in *Wheeled Robot Navigation* tasks. (see Appendices **??**)

5 Conclusions

In this paper we present a nature-inspired learning framework composed of Evolution Strategies, Plastic rules, and Recursion. Experiment results show that plasticity can be effectively forged with recursion to enhance the learning capability. The proposed framework can achieve equivalent or even better performances compared with more sophisticated neural structures, by applying the simplest recurrent neural structures. Moreover, we also show that under proper meta parameters, plasticity has a chance to surpass gradient descent methods.

We believe the learning framework of Figure 1 can be extended to more sophisticated plastic rules and model structures, uncovering better learners in the future. Also, it would be more interesting if such framework can be validated in more complex environments such as natural language processing



(a) W_t for different A (b) W_t for different n (c) W_t for different ϕ

(d) h_t for different A (e) h_t for different n (f) h_t for different ϕ

(g) W_t for different A and goals (h) h_t for different A and goals

Figure 5: t-SNE visualization of the transformation of the connection weights (W_t) and hidden states (h_t) in the inner loop. (a)(b)(c)(d)(e)(f) are from *Sequence Predicting* (l=1,K=25,N=50) task, (g)(h) are from *Wheeled Robot Navigating* (Low Noise) task.

tasks and vision-related tasks. Finally, we are looking forward that this work can shed light to new paradigm of building intelligent machines and inspire more efforts in this line of research,

References

Horace B Barlow. Adaptation and decorrelation in the cortex. *The computing neuron*, 1989.

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.

Tim VP Bliss and Terje Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356, 1973.

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.

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.

Harald T Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 34(5):254–256, 1946.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018.

Wulfram Gerstner, Raphael Ritz, and J Leo Van Hemmen. Why spikes? hebbian learning and retrieval of time-resolved excitation patterns. *Biological cybernetics*, 69(5):503–515, 1993.

Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

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.

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

Bernabé Linares-Barranco and Teresa Serrano-Gotarredona. Memristance can explain spike-time-dependent-plasticity in neural synapses. *Nature precedings*, pp. 1–1, 2009.

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. *arXiv preprint arXiv:2006.09549*, 2020.

Bartlett W Mel. Have we been hebbing down the wrong path? *Neuron*, 34(2):175–177, 2002.

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, 2019. URL `https://openreview.net/forum?id=r1lrAiA5Ym`.

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

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

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Joachim Winther Pedersen and Sebastian Risi. Evolving and merging hebbian learning rules: Increasing generalization by decreasing the number of rules. *arXiv preprint arXiv:2104.07959*, 2021.

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.

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)*, number CONF, 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.

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.

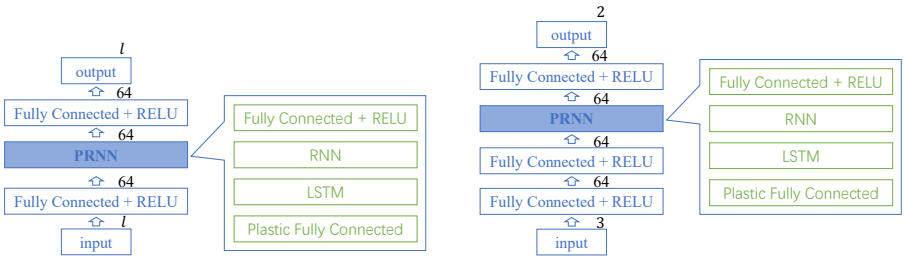
Jun Zhang, Zhi-hui Zhan, Ying Lin, Ni Chen, Yue-jiao Gong, Jing-hui Zhong, Henry SH Chung, Yun Li, and Yu-hui Shi. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine*, 6(4):68–75, 2011.

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 Appendices

A.1 Model Architecture and Training Details

To maintain fairness for comparison, the model architectures of the compared methods are similar, where the differences lie in only one of their layers (Figure 6). We use 3 hidden layers for *Sequence Predicting* tasks and 4 hidden layers for *Wheeled Robot Navigating* tasks. For **ES-MAML**, we replace the PRNN layer with fully connected layer; For **ES-RNN** and **ES-LSTM**, we replace it with *RNN* and *LSTM* respectively; For **EPMLP**, we replace it with plastic fully connected layer, where the plasticity rule is stated by Equation 1, and the neural modulator m is calculated by an additional dense layer with sigmoid activation. The hidden sizes of all the hidden layers are 64 (for LSTM it is 64 hidden states and 64 cell states).



(a) *Sequence Predicting* (b) *Wheeled Robot Navigating*

Figure 6: A sketch of the model architectures for evaluated tasks

For *Sequence Predicting* tasks, the input observation and the output action has the dimension of l. In the *meta-training-training* and *meta-testing-train* stages, we use the ground truth $\boldsymbol{y}_{t-1}$ as input observation; In the the *meta-training-test* and *meta-testing-test* stages, its previous action (a_{t-1}) are taken as inputs. For *Wheeled Robot Navigating* tasks, the output action is the control command of its two wheels (a length 2 vector), the input observation is the concatenation of its previous action and the current observed signal intensity (A_t).

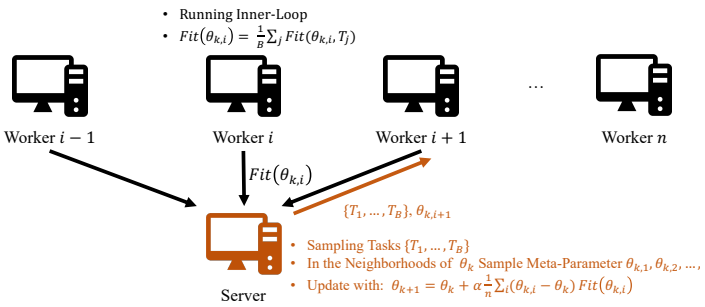
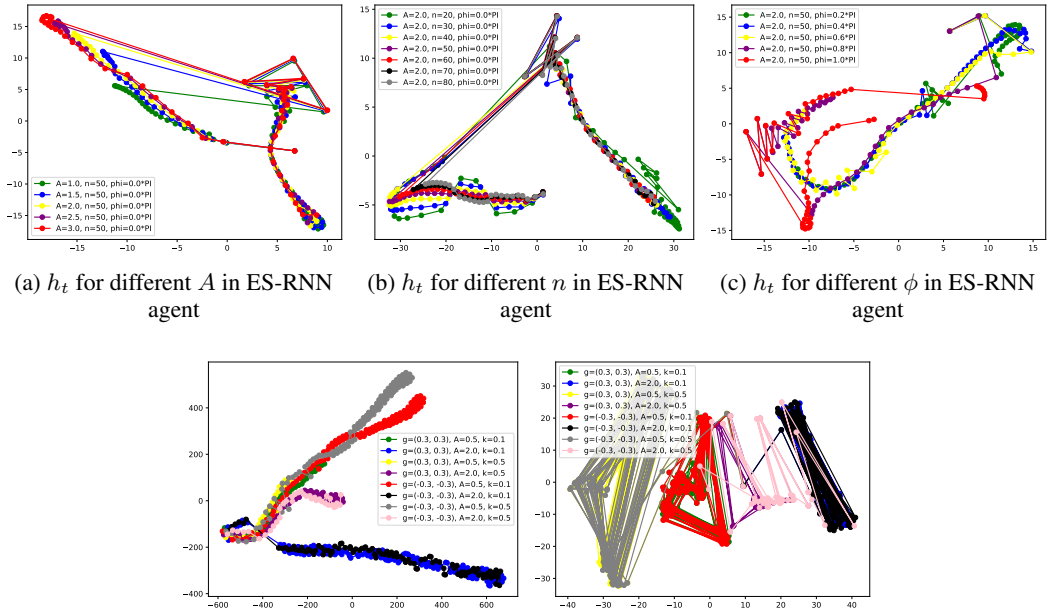


Figure 7: A sketch of the parallel training framework.

The training process is accelerated by utilizing the paralleling mechanism of PARL. We employ 400 workers (400 Intel(R) Xeon(R) CPU E5-2650) running inner-loops for 400 off-springs in each generation of the evolution, and additional 1 CPU to perform evolutionary update (shown in Figure 7(b)). It takes 3 to 12 hours for each run depending on the length of the inner loop and model architectures. Following the previous work (Salimans et al., 2017), we rank normalize the fitness among 400 workers. The learning rate α is set to 0.2. The mutation is performed by adding independent Gaussian noises to each parameter. During the *meta-training*, we sample B different tasks and 400 meta-parameters. Each worker run B *meta-training-training* and B *meta-training-testing* given the assigned meta-parameter, then the fitness is averaged. We set $B = 16$ for *Sequence Predicting* tasks, and $B = 4$ for *Wheeled Robot Navigating* tasks. For each *meta-testing* epoch, we evaluated the current meta parameters in newly sampled 1600 tasks.

A.2 Supplementary Experimental Results and Analyses

We show more t-SNE visualization of the connection weights and hidden states in Figure 8 which can be compared with Figure 5. The transformation of hidden states in **ES-RNN** (Figure 8(a)(b)(c) and (e)) are even less distinguishable among different tasks compared with that of **EPRNN** (Figure 5(d)(e)(f) and (h)), showing naive RNNs fail to capture task configurations. The weights of **EPMLP** can also capture some hidden configurations (Figure 8(d)) but are less accurate compared with that of **EPRNN** (Figure 5(g)). For instance, the gold and grey line share the same hidden configuration of $A = 0.5$ and $k = 0.5$. The connection weights effectively captures this information and the two lines overlap with each other in **EPRNN**, but it does not happen in **EPMLP**. Similar phenomenon can be observed for the red and the green lines ($A = 0.5$ and $k = 0.1$).



(a) h_t for different A in ES-RNN agent

(b) h_t for different n in ES-RNN agent

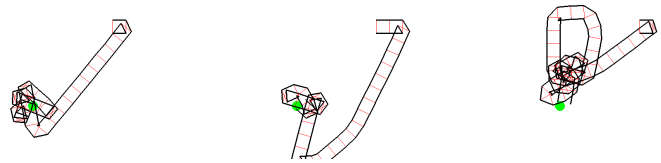
(c) h_t for different ϕ in ES-RNN agent

(d) W_t for different A,k and goals in EPMLP agent

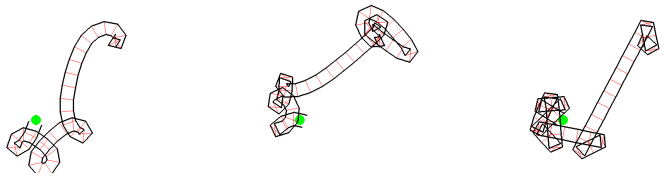
(e) h_t for different A,k and goals in ES-RNN agent

Figure 8: Supplementary t-SNE visualization of the transformation of the connection weights (W_t) and hidden states (h_t) in the inner loop. (a)(b)(c) are from *Sequence Predicting* (l=1,K=25,N=50) task, (d)(e) are from *Wheeled Robot Navigating* (Low Noise) task.

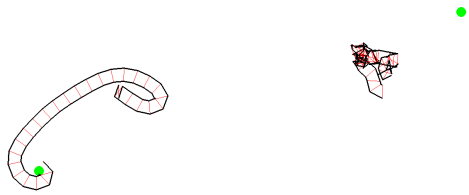
We also plot the trajectories of the robot in *Wheeled Robot Navigation* tasks. All the agents struggle to find the goal by continuously shifting their directions, which is more frequent in cases of higher observation noise. A reasonable guess is that in higher noise cases, the agents are less confident in their estimations. They have to more frequently adjust their directions to acquire diverse observations for more effective denoising.



(a) EPRNN (Low Noise) (b) EPRNN (Median Noise) (c) EPRNN (High Noise)



(d) EPRNN (w/o m) (Median Noise) (e) EPMLP (Median Noise) (f) ES-RNN (Median Noise)



(g) ES-LSTM (Median Noise) (h) Random Controller

Figure 9: Trajectories of the agents from *Wheeled Robot Navigating* task with the task configuration of ($A = 0.5$, $k = 0.2$, $(g_x, g_y) = (-0.3, -0.3)$). We show the trajectories of the wheeled robot in black and red lines and the goal in green circles.