Structural Parameter Space Exploration for Reinforcement Learning via a Matrix Variate Distribution

Shaochen Wang, Rui Yang, Bin Li, *Member, IEEE*, and Zhen Kan, *Member, IEEE*

***Abstract*—The trade-off between exploration and exploitation is essential for reinforcement learning, where an agent needs to be aware of when to explore for high reward policies and when to exploit the optimal policy known so far. Parameter space exploration provides an elegant solution. As one of the principal methods, injecting noise into the model parameters greatly improves exploration. However, directly stretching the parameters of the neural network into a vector and generating noise for this vector ignore the structural information of the model. In this paper, we aim to incorporate spatial information into weight matrices and propose matrix-variate noise exploration, which exploits the structural weight uncertainty brought by matrix variate noise to enhance the stochasticity of the agent. Indeed, we construct a bridge between the matrix noise exploration and probabilistic neural networks, which theoretically explains the improved performance of parameter space exploration. Extensive experiments have shown that matrix variate noise exploration outperforms fully factorized noisy exploration on most Atari tasks and Super Mario Bros tasks and is competitive to the state-of-the-art methods.**

***Index Terms*—Reinforcement learning, parameter space exploration, weight uncertainty.**

I. Introduction

Reinforcement learning (RL) has attracted significant attention and achieved impressive progress in sequential decision making problems [1]. An intelligent agent acts in an unknown world through trial and error to learn optimal behaviors. During the interaction with the environment, the agents should not only exploit their current knowledge to maximize their rewards, but also explore other unknown possibilities to find potential solutions. Efficient exploration is needed to guide the agent in search of promising policies.

Recently, deep reinforcement learning has achieved amazing success in numerous fields. Neural networks as a key component have become an efficient function approximator to approximate the policy function or value function. Many popular algorithms equipped with deep neural networks like deep Q network (DQN) [2], and Asynchronous Advantage Actor Critic (A3C) [3] accomplish remarkable advance in game playing [4], robotics locomotion [5], and many other fields. However, most of these methods rely on simple exploration strategies. For instance, DQN adopts a ϵ-greedy strategy where an agent takes random actions with the probability of ϵ and acts greedily with the probability of $1-\epsilon$. These heuristics exploration strategies are slightly stodgy and perform poorly on some hard exploration tasks. Furthermore, these inefficient mechanisms tend to consume lots of interaction steps with the environment, which significantly degrade the sample efficiency of the algorithm. Thus, it is desirable and urgent to seek a more efficient exploration strategy which performs state-dependent exploration.

In recent years, a variety of elaborate and sophisticated exploration algorithms have emerged and greatly addressed partial difficulties faced by reinforcement learning [6]. Among these methods, uncertainty estimation, as an indicator to quantify surprise or novelty, consistently encourages agents to seek for promising policies. Many of the exploration approaches based on uncertainty estimation have been developed such as upper confidence bounds (UCB) [7], curiosity exploration [6], and so on. One leading principle of these solutions is to increase the stochasticity of agents. Despite these successes, there still exist a few limitations for current uncertainty-driven methods. First, count-based methods like UCB perform well in tractable MDPs but remain a challenge to high dimensional state space. Nevertheless, pseudo-counts [8] can alleviate this issue but still need an additional density estimation model. Second, most curiosity-driven methods which make use of prediction errors as novelty rewards, inevitably require additional modules as auxiliary tasks. For example, in [6], authors use neural networks to train a forward dynamics model which estimates the difference between the true next state s_{t+1} and the predicted next state.

Taking account of uncertainty in the parameter space serves as an elegant solution to the trade-off between exploration and exploitation. There is a wealth of information hiding in the parameter space. Undoubtedly, it is desired to exploit weight uncertainty in a high-dimensional parameter space. How to utilize weight uncertainty to explore and extract this information is attracting growing research attention. In fact, parameter space exploration has been well studied in the literature. Recent works [9], [10] in parameter space exploration obtained significant

Manuscript received 31 July 2021; revised 8 November 2021; accepted 25 November 2021. Date of publication 13 January 2022; date of current version 24 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grants U2013601, 62173314, U19B2044 and 61836011. *(Corresponding author: Zhen Kan.)*

Shaochen Wang and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: samwang@mail.ustc.edu.cn; zkan@ustc.edu.cn).

Rui Yang and Bin Li are with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application Systems, University of Science and Technology of China, Hefei 230052, China (e-mail: yr0013@mail.ustc.edu.cn; binli@ustc.edu.cn).

Digital Object Identifier 10.1109/TETCI.2022.3140380

2471-285X © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

performance improvements with minor modifications to the original algorithms. Detailed ablation studies in [11] confirmed that eliminating the Noisy Net from the Rainbow DQN causes a substantial drop in several Atari games. However, directly injecting diagonal Gaussian noise into parameters is clearly insufficient due to ignoring the correlation between weights.

Another family of noise exploration is the evolution strategies [12]. Evolutionary strategies construct a population by adding noise to the model parameters, which can effectively explore the environment. However, most evolution-based methods do not consider the temporal structure of the agent and gradient information, resulting in poor sample efficiency.

Today, neural networks (NN) can learn an expressive representation from high dimensional input space. Yet, most of the advances in neural networks have been made under a frequentist perspective. For a well-trained NN, it usually gives a deterministic output and the weights concentrate on a single mode of the parameter space. This is mainly because the current models still seek a point estimation of parameters. In supervised learning, this may partially result in overfitting to the observed data. Moreover, in RL, this implies that the parameter space has not been fully explored, which directly affects the quality of the learned policy. Especially for the tasks with inherent stochasticity, this issue will be further exacerbated. It is therefore essential to incorporate uncertainty into the parameter space.

In this paper, we propose a novel structural parameter space noise exploration method, where structural weight uncertainty brought by noise dramatically enhances the exploration. Our method pays attention to the dependencies between weight variables in neural networks and learns rich information in parameter space. In this work, we fully consider the characteristics featured by neural networks, in which the noisy exploration is based on the entire weight matrix of a layer in the neural network rather than an isolated weight point. Overall, the main contribution of our approach can be summarized as follows: first, we propose a structural weight uncertainty exploration method where the noise is generated via a matrix variate distribution. The matrix variate noise is based on the dependencies between rows and columns of the weight matrices. Second, we build a bridge between the matrix noise exploration and probabilistic neural networks, theoretically explaining the stochasticity brought by noise exploration to explore more diverse areas in the parameter space. Moreover, our approach allows for a natural balance between exploration and exploitation. The matrix variate noise is jointly learned with model weights, leading to a low uncertainty in explored regions versus high uncertainty in less explored regions. Third, the proposed method is evaluated on environments with high dimensional state spaces such as Super Mario Bros and Atari games. The results demonstrate that our method significantly improves the effectiveness of exploration and performs better than other baselines. Weight uncertainty exploration guides the agent to explore and discover a number of intriguing behaviors.

II. Related Work

Efficient exploration is a fundamental and critical problem for reinforcement learning, and there exist many challenging issues waiting to be addressed. How to maintain the trade-off between exploration and exploitation is still an open problem. The agent needs to select the best behavior via interactions with the environment, and explores new actions to discover potentially better policies. We roughly divide recent literature on exploration into three categories: action space exploration, reward space exploration, and parameter space exploration, and briefly introduce recent advances in the domain of exploration.

A. Action Space Exploration

Random exploration is easy to implement and widely used as a general exploration strategy in a large number of reinforcement learning approaches. For example, ϵ-greedy strategy is often applied to the valued based methods [2], [13], and in continuous control tasks, policy-based methods [14] explore the environment through random perturbations adding on the actions or entropy regularization to prevent premature to a local optimal policy. Boltzmann exploration uses a Boltzmann distribution to select the actions where the action is proportional to the Q value computed according to a softmax distribution. Apart from these, the previous work [15] combines model-free reinforcement learning and imitation learning which makes use of past good experiences for self-imitation. Self-imitation learning uses past good state-action pairs and indirectly promotes a positive exploration effect.

B. Reward Space Exploration

A prevalent solution in reward space exploration is reward shaping. The external reward of the task has certain delayed feedback and can be deceptive in many cases. It is intuitive to augment intrinsic rewards as a bonus to facilitate better exploration. A number of works [8], [16]–[18] have been developed from different perspectives as intrinsic rewards to improve exploration. Count-based exploration is well studied in the literature. Recent works [6], [19] inspired by psychology, explore the world through curiosity as an intrinsic reward. Authors address exploration by curiosity to guide agents finding novel states.

Another category is to utilize the intrinsic motivation of the agent to explore the environment [6]. Recent works use prediction error as curiosity to explore the environment, which requires building a model to estimate the difference between the value of the next state and the current state. However, most of these techniques require to design specific structure, and is challenging to scale to high dimensional state space.

C. Parameter Space Exploration

The idea of injecting noise into model's parameters is not new. It can go back at least to [20], which treats injection noise as a mutation operator to optimize the function. Recently, searching in parameter space to find a good policy is a pretty intuitive approach, and has been studied extensively in [9], [10], [21], [22]. NoisyNet [10] proposes fully factorized weight noise and achieved good performance on DQN and its variants. Moreover, concurrent work [9] uses a heuristics approach to adjust the variance of injected Gaussian distribution to improve exploration. The work of [12] considers evolutionary strategy

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

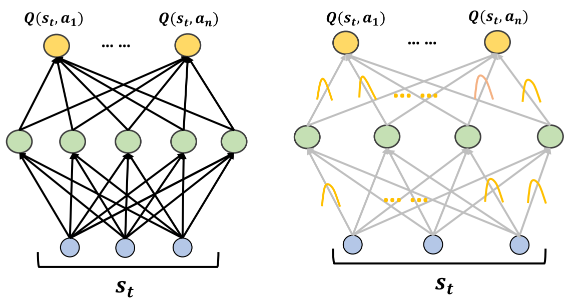


Fig. 1. Schematic representation of deterministic neural network(left) and fully factorized noise exploration (right). Isotropic Gaussian noise is added to each weight of the neural network.

as an alternative to reinforcement learning to solve sequential decision problems. Authors in [21] integrate parameter exploration on policy gradient methods by sampling policy parameters from parameter space. Indeed, the aforementioned methods all adopt the isotropic Gaussian distribution to generate noise and ignore the correlation between parameters. Due to the lack of consideration of the structure information, the exploration power of these methods is significantly diminished. Our matrix variate noise is based on the dependencies between rows and columns of the weight matrices and focuses on structural information in the parameter space. This is the major difference between our method and previous methods. In addition, our approach is closely related to the probabilistic neural network. A wide range of probabilistic neural networks have been studied in the field of Bayesian neural networks [23]–[26].

III. Background

A. Reinforcement Learning

The heart of reinforcement learning is to acquire an optimal policy through interactions with the environment via a reward function. The environment is generally modeled as Markov Decision Processes (MDPs) which is typically described by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. In this paper, we assume that $\mathcal{S}$ is super large but finite. $\mathcal{R}$ is the reward function and $\mathcal{P}$ is the dynamic transition probability. $p(s'|s, a_t)$ describes the transition probability from s to s' when action a_t is taken at time step t. $\gamma \in (0, 1)$ is the discount factor. The policy $\pi(a|s)$ is a probability distribution conditioned on the states, which maps a state to an action. At each discrete time step t, the agent selects the action a_t in s_t according to its current policy and acts the action to the environment. After the environment receives the action a_t in state s_t, the environment returns a scalar reward $r(s_t, a_t)$ and moves to the next state s_{t+1}. Then the agent continues this process until it encounters a terminal state. The objective function of RL is formulated as

$$\max_{\theta_P} \mathbb{E}_{\pi(s_t;\theta_P)} \left[\Sigma_t \gamma^t r(s_t, a_t) \right], \tag{1}$$

where the goal of the agent is to find a parameterized policy to maximize the expected cumulative discounted rewards defined in (1). Moreover, the cumulative return is usually used as an indicator to assess the quality of a policy.

In the framework of MDP, the problem is transformed to learn the value of each action at each state, namely state-action value $Q(s, a)$. The optimal state-action value can be learned by Q-learning [27] algorithm which is an off-policy algorithm based on dynamic programming. In valued-based method, Q-learning [27], as a very popular approach, is updated iteratively by a temporal difference form defined as follows:

$$Q^\pi(s, a) = Q(s, a) + \alpha(r + \gamma \max_a Q(s_{t+1}, a) - Q(s, a)), \tag{2}$$

where α is a step size. Q-learning works well and is guaranteed to converge to an optimal policy under tabular cases. The final optimal policy is the actions with the highest value.

However, tabular Q-learning is still not able to tackle high-dimensional problems. Mnih *et al.* [2] proposed deep Q network (DQN). DQN is the first deep reinforcement learning algorithm incorporating deep neural networks that achieves human-level performance on high-dimensional problems such as Atari games. The Q function is optimized by minimizing the Bellman residual expressed as follows:

$$\mathbb{E}_{s,a\sim\rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right], \tag{3}$$

where $y = r(s, a) + \gamma \max_{a'} Q(s', a'; \theta')$. θ is the parameters of main network and θ' is the frozen parameters of target network which periodically copies from the main network.

B. Noisy Exploration

The ability to handle uncertainty is critical for RL algorithms. The uncertainty originates from the unknown environment dynamics and is reflected on the observed data. [10] proposed NoisyNet to promote exploration, which added parametric noise to the weights of the neural network. NoisyNet explores the environment by perturbing in the parameter space, and therefore injecting stochasticity to the policy. In NoisyNet, the original linear layer is replaced by noisy layer and each weight in the model is imposed on a zero-mean Gaussian noise. The noisy layer is represented as follows:

$$y \overset{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w) x + \mu^b + \sigma^b \odot \varepsilon^b, \tag{4}$$

where w, b are weights and biases of the neural network respectively and x is the input of the current layer. σ is the standard deviation of Gaussian noise, which is learned jointly with the weight. ε are unit Gaussian random variables and $\odot$ means element-wise multiplication.

IV. Method

Fully factorized noise exploration, depicted in Fig. 1, maintains a perturbation factor for each parameter of the model, which directly stretches up the parameters of neural network into a vector, and then samples Gaussian noise for this vector to explore the parameter space. One major limitation is that fully factorized noise disregards the structural information in parameters. As we know, the parameters of each layer in the neural network are weight matrices. Stretching such a weight matrix into a vector would lose its spatial and structural information, and independent isotropic Gaussian noise ignores the correlation

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

between parameters. A deep motivation of this paper is to incorporate structural information from neural networks into parameter space exploration. Moreover, fully factorized noise doubles the number of parameters of the neural network. The dimension of the perturbation factor grows polynomially with respect to the number of neurons of the neural network. A very intuitive approach to ameliorate this issue is to consider the entire weight matrix instead of isolated parameters.

In this paper, we make use of matrix variate noise to generate parameter space noise, which considers the specific structures of neural networks. The overall noise exploration can be formulated as follows:

$$\tilde{\boldsymbol{W}} = \boldsymbol{W} + \boldsymbol{\epsilon}, \tag{5}$$

where $\boldsymbol{W}$ is the weight matrix of the neural network and $\boldsymbol{\epsilon}$ is the noise oriented for matrix exploration. The dependencies between parameters are brought in by taking into account the row and column covariance of weight matrices. $\tilde{\boldsymbol{W}}$ is the weight after adding noise, which brings in weight uncertainty. Structural uncertainty carried by matrix variate noise can better capture under-explored areas. The matrix variate distribution is a probability distribution over matrices variable. In particular, we take the matrix variate Gaussian as an instance to illustrate how to perform parameter space exploration. In addition, the matrix variate can be extended to other distributions and is not limited to the Gaussian distribution. In general, for a random matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, if it is distributed according to a matrix variate Gaussian, the p.d.f. of random matrix $\mathbf{X}$ is denoted in

$$\begin{aligned} p(\mathbf{X}) &= \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V}) \\ &= \frac{\exp\left(-\frac{1}{2}\operatorname{tr}\left[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})\right]\right)}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{n/2}}. \end{aligned} \tag{6}$$

$\mathcal{MN}$ refers to the matrix variate Gaussian distribution, where $\mathcal{MN}$ is the abbreviation of matrix normal. $\mathbf{M} \in \mathbb{R}^{m\times n}$ corresponds to its mean term and has the same size as the random matrix $\mathbf{X}$. $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are the matrices that govern the covariance of rows and columns of M respectively. In this way, we can explicitly model the correlations between rows and columns of the weight matrices of the neural network and even the relation with the input of each layer. Uncertainty from the structural noise brings stochasticity to the learned behavior and avoids premature convergence.

An immediate challenge is how to make the noise coadaptation with weights of the neural network. We use the reparameterization trick in [28], which makes the structural noise jointly learned with weights matrices. The randomness inherent in the weight noise is transformed into an auxiliary random matrix variable $\mathbf{E}$, and the parameters of matrix variate Gaussian are converted to be deterministic and can be learned jointly with the weights in the neural network. For example, for a scalar $x \sim \mathcal{N}(\mu, \sigma^2)$, a reparameterization is $x = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$. We specifically generalize it to the matrix noise:

$$\begin{aligned} \boldsymbol{\epsilon}^{(l)} &= \mathbf{0} + \mathbf{U}^{\frac{1}{2}}\mathbf{E}^{(l)}\mathbf{V}^{\frac{1}{2}} \\ \mathbf{E} &\sim \mathcal{MN}(\mathbf{0}, \mathbf{I}, \mathbf{I}) \quad (\text{ i.e. } E_{ij} \sim \mathcal{N}(0,1)), \end{aligned} \tag{7}$$

where $\boldsymbol{\epsilon}^{(l)}$ is the structural noise generated for l-th layer and $\mathbf{E}$ is the auxiliary matrix variable. Each element $\boldsymbol{E}_{ij}$ follows a standard Normal distribution.

A major prerequisite for row covariance and column covariance matrices is that the matrices need to be symmetric and positive definite. In order to solve this problem, instead of learning the parameters of U and V directly, we do a decomposition of the covariance matrices. Note that for any real matrix $\boldsymbol{L}$, $\boldsymbol{L}\cdot\boldsymbol{L}^T$ is guaranteed to be semi-positive definite. Therefore, we turn to learn the parameters of the decomposition matrices of the row and column covariance of the weight matrices.

The matrix variate noise is zero-mean and row and column covariances are learned from data, which reflects the dependencies among the weight matrices. Applying a matrix Gaussian distribution noise with zero-mean to a weight matrix with value $\mathbf{W}$, the original weight matrix is transferred to a matrix Gaussian distribution:

$$\boldsymbol{\epsilon} \sim \mathcal{MN}(\mathbf{0}, \mathbf{U}^{(l)}, \mathbf{V}^{(l)}) \Longrightarrow \tilde{\boldsymbol{W}} \sim \mathcal{MN}(\mathbf{W}, \mathbf{U}^{(l)}, \mathbf{V}^{(l)}). \tag{8}$$

$\tilde{\boldsymbol{W}}$ is the weight matrix after noise injection, which follows a matrix Gaussian distribution and $\mathbf{W}$ is the mean value of this distribution. When adding noise to each layer of the neural network, it is equivalent to sampling the weights from a $\mathcal{MN}(\mathbf{W}, \mathbf{U}^{(l)}, \mathbf{V}^{(l)})$ distribution. Equipped with matrix variate noise, the original weight matrix is transferred to a probability layer.

Deterministic neural network weights are injected with noise and the weights thus contain uncertainty. Our method builds on the weight matrix noise exploration, introducing structured uncertainty estimation in the parameter space. In value-based DRL, the action-value function, $Q(s, a)$, is usually represented by a neural network. In our work, the Q-value of each action $Q(s, a)$ is predicted by a neural network with weight uncertainty. s_t is fed into the current Q network and a_t is the action taken by the agent at time step t. The Q network is parameterized by θ with structural matrix noise injection. At each step, the final action is chosen based on the noisy prediction of the maximum Q value. The matrix noise exploration poses randomness on the weights and prevents excessively greedy action selection. The matrix variate noise is learnable and we cast the noise exploration as an optimization problem that can be trained through backpropagation. We refer to update the Q function by one-step noisy Bellman equation. Note that the loss function is defined in

$$\bar{L}(\zeta) = \mathbb{E}\left[r + \gamma \max\nolimits_{b\in A} Q\left(y, b, \varepsilon'; \theta^-\right) - Q(x, a, \varepsilon; \theta)\right]^2, \tag{9}$$

in which ε is the corresponding matrix variate noise.

In the previous claim, we describe how to generate a structural parameter-based matrix noise. Armed with matrix noise exploration, we bridge our approach with the probability neural network. We maintain a fully probabilistic distribution over the model parameters rather than making a point estimate. We denote some symbols notation, where bold lowercase letters $\mathbf{x}$ are column vectors and capital bold letters $\mathbf{W}$ represent matrices. A transpose of a variable $\mathbf{x}$ is expressed by $\mathbf{x}^T$. Here we first consider the linear transformation of one layer in the neural

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

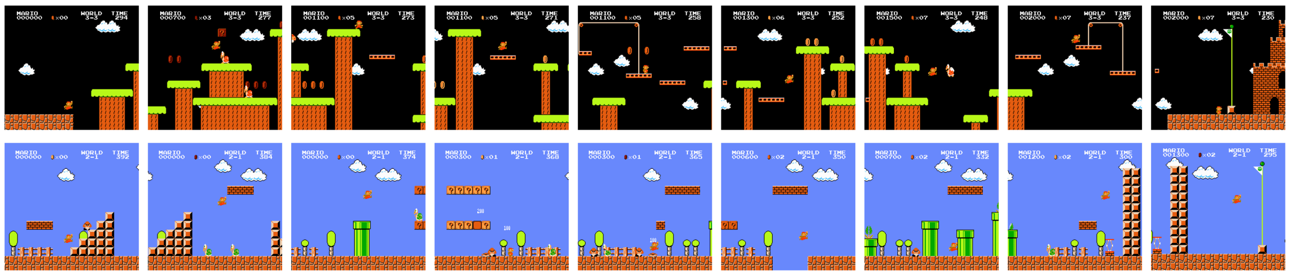


Fig. 2. Screenshots of the agent trained by matrix variate noise playing Mario. In two different levels, the agent has successfully passed all the obstacles and reached the flag.

network, which is usually defined as $y = wx + b$. For the sake of convenience, we extraly augment a dimension to the feature $\mathbf{x}$. Then we can absorb the bias into the weight, and the linear transformation is denoted as $y = w^T x$. Here w is (w, b) and x is $(x, 1)$.

Fortunately, the nice properties in multivariate Gaussian variables can all be generalized to matrix Gaussians [29][30]. The linear transformation of a matrix variable and a matrix Gaussian distribution remains a matrix variate Gaussian, and vice versa. In (10), $\mathbf{W} \sim \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$, where $\mathbf{M}$ is the mean value of this matrix distribution. The multiplication of a random matrix $\mathbf{A}$ with $\mathbf{W}$ still follows a matrix Gaussian distribution:

$$\mathbf{B} \triangleq \mathbf{AW} \sim \mathcal{MN}\left(\mathbf{AM}, \mathbf{AUA}^{\top}, \mathbf{V}\right). \tag{10}$$

In this section, we make a connection between our method and probabilistic neural network. The Q value can be viewed as a probabilistic model, $P(Q(s, a)|s, a)$. Structural uncertainty in weights allows more variation in decision making, which naturally promotes exploration. For clarity, we first consider a simple case. We assume that our deep Q network contains only one hidden layer and one output layer. Let $\mathbf{X}$ be the inputs to deep Q network, which are states sampled from the replay buffer with dimensions $N \times m$. N is the size of the minibatch and m is the dimension of the state s_t's feature. Then, we denote W_1 as the first weight matrix, W_2 as the second weight matrix. With our structural matrix noise injection, the weight matrix $\tilde{W}$ follows a Gaussian matrix distribution:

$$\mathbf{\tilde{W}_1} \sim \mathcal{MN}(\mathbf{W_1}, \mathbf{U}_1^{(l)}, \mathbf{V}_1^{(l)}); \mathbf{\tilde{W}_2} \sim \mathcal{MN}(\mathbf{W_2}, \mathbf{U}_2^{(l)}, \mathbf{V}_2^{(l)}).$$

All the weights in our model are expressed in terms of probability distributions, as opposed to having a fixed value. Moreover, the matrix distribution can carry more structural information. For the feedforward propagation of the deep Q network, it can be expressed as follows:

$$\mathbf{B} = \mathbf{X} \cdot \mathbf{\tilde{W}_1} \tag{11}$$

$$\mathbf{Q} = f(\mathbf{B}) \cdot \mathbf{\tilde{W}_2} \tag{12}$$

$\mathbf{B}$ denotes the result of the first weight matrix layer and f is a nonlinear activation function. $\mathbf{Q}$ is the output vector of deep Q net, and each entry in $\mathbf{Q}$ represents the $Q(s, a)$ value of each action at that state. Since $\mathbf{\tilde{W}_1}, \mathbf{\tilde{W}_2}$ all follow matrix Gaussian distribution, according to (10), the output of feedforward network also follows matrix Gaussian distribution as

$$\mathbf{B} \mid \mathbf{X} \sim \mathcal{MN}\left(\mathbf{XW_1}, \mathbf{XU}_1^{(l)}\mathbf{X}^T, \mathbf{V}_1^{(l)}\right) \tag{13}$$

$$\mathbf{Q} \mid \mathbf{B} \sim \mathcal{MN}\left(f(\mathbf{B})\mathbf{W_2}, f(\mathbf{B})\mathbf{U}_2^{(l)}f(\mathbf{B})^T, \mathbf{V}_2^{(l)}\right). \tag{14}$$

The output B of the middle layer is matrix variate Gaussian distribution conditioned on the input x. Likewise, the final output is a matrix variate Gaussian distribution dependent on the output of the middle layer. Note that all row covariances are input-dependent, indicating that matrix noise exploration is a state-dependent exploration. As we can see, the probabilistic form of the output can be represented in (14). Linear transformations between MVG distributions are still MVG distributions. Our method shows that the Q value in our deep Q network explores the environment through weight uncertainty, rather than a deterministic value at a given moment. The resulting state-action value $Q(s, a)$ is a matrix variate Gaussian distribution conditioned on current state s_t. In (14), the row covariance of output is the inner product of inputs and covariance of current weight matrix. This implicitly connects the correlation between the input and hidden units in the neural network and yields state-dependent exploration.

Now consider a complex case defined in

$$\boldsymbol{Q(\hat{s}, a)} = f_L \circ f_{L-1} \circ \cdots \circ f_1\left(\mathbf{s}_i\right), \tag{15}$$

where the neural network has l layers. $\circ$ means function composition where $f_L \circ f_{L-1}$ represents f_L is evaluated on the output of f_{L-1}. We impose the matrix variate noise on each layer, and eventually the Q value depends on the uncertainty brought by each layer of noise. Our approach is essentially an uncertainty driven method to exploration. Uncertainty drives the agent to explore areas with high uncertainty. As more observations are collected and the environment is better understood, the corresponding uncertainty decreases and then drives the agent to explore more uncertain regions. The mean of the weight matrix gains more information about the environment, and the noise injection increases the stochasticity of agents. Additionally, matrix based noise exploration makes it easier to dig out the correlation among the weights and discover information sharing between the parameters. In general, the probabilistic representation of weight matrix noise exploration could not only explain the uncertainty estimation of the value of actions, but also capture more information gain about the environment.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

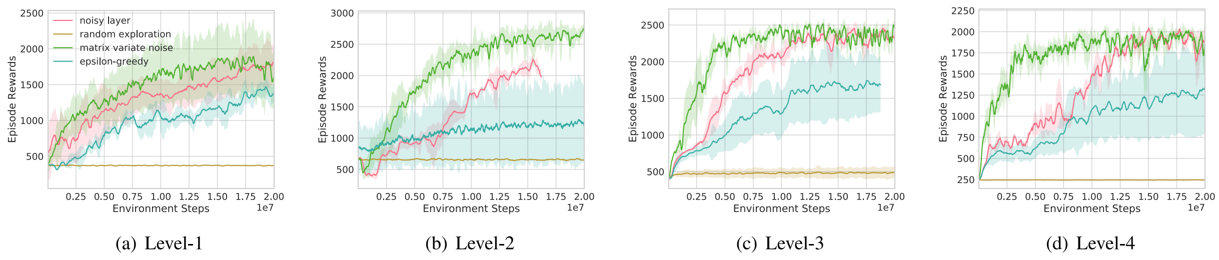


Fig. 3. Training curves of matrix variate exploration with other exploration baselines on Super Mario Bros tasks. The y-axis shows the episode rewards and x-axis shows steps. All curves are averaged with 4 random seeds.

V. Experiment

In order to verify the effectiveness of exploration, in this section, we empirically investigate how exploration affects the quality of learned policies and sample efficiency. We compare our approach with the classical action space exploration and parameter space exploration methods on a suite of Super Mario Bros tasks and further test it against state-of-the-art deep reinforcement learning methods on the famous Atari 2600 benchmark. The experiments are conducted on fifteen of all the Atari video games. We demonstrate that our weight uncertainty strategy enables the agent to make progress in all four levels of Mario. The results show that it outperforms the baselines. On all tasks, agents learn to play the games from the pixels.

Our matrix-variate parameter space exploration can be integrated with a value-based method or policy-based method. In this paper, we couple it with Deep Q network (DQN).

A. Environment

In order to verify the robustness of the algorithm, we evaluate the exploration ability of our method on different scenarios. We select Super Mario Bros tasks [31] and Atari 2600 games [32] as our testbed. Atari is a popular RL benchmark, which provides a platform for researchers to compare and evaluate the performance of RL algorithms. Super Mario Bros tasks (SMO) is a 2-D video game, which is a long horizon decision making and temporal extended task. For both tasks, deep RL agent needs to perceive from the pixel inputs. In the Mario Bros tasks, the agent goes through levels from left to right. In the process of advancing, the agent should avoid encountering monsters, while collecting gold coins as much as possible. Once a collision with a monster occurs, the game terminates. When picking up jewels, the agent will be rewarded. Also, killing monsters will be rewarded. The final rewards are also related to the time the agent takes to reach the flag. The shorter the time, the higher the reward. In the Mario Bros tasks, we considered 4 tasks including the day world and night world. The action space of SMO contains move forward, move backward and jump, etc. To allow for more diverse strategies to be explored, we allow the agent to take multiple actions at the same time. It is consistent with the players using joysticks in the game to press multiple buttons simultaneously. All experiments were conducted in OpenAI Gym [33] which is a general platform for evaluating RL algorithms. We then investigate how to exploit such structural weight uncertainty to drive exploration and test it on a nonlinear regression problem.

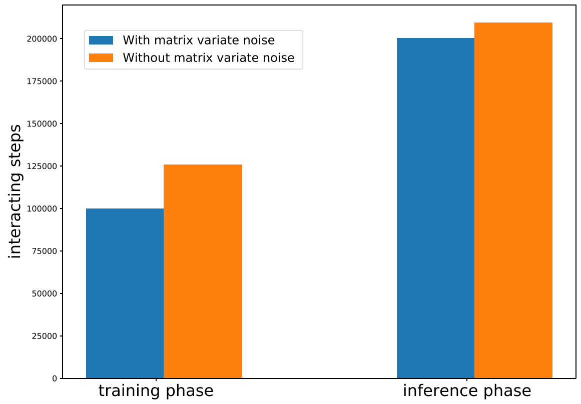


Fig. 4. Comparison of the number of interaction steps between the agent and environment with and without the matrix variate noise at 30 minutes during the training phase and the inference phase, respectively.

B. Ablation Studies

In this section, we conduct extensive experiments to validate the effectiveness of our matrix variate noise exploration. Firstly, the time performance of the algorithm is critical. In order to evaluate the impact of our matrix variate noise exploration on the time performance of the algorithm, we investigate the training and inference time of our model with and without the matrix variate noise. In Fig. 4, we present that the number of steps to interact with the environment in 30 minutes with and without matrix variate noise exploration in the training and inference phases, respectively. The experiments are performed on a machine with a 20-core Intel i9 CPU and NVIDIA 3070 GPU. We can observe that during the training phase the algorithm equipping matrix variate noise interacts with the environment for 1.01×10^5 steps in 30 min, and for 1.25×10^5 steps without applying matrix noise. In the inference stage, the algorithm interacts with the environment with and without matrix variate noise for 2.04×10^5 and 2.1×10^5 steps in 30 min, respectively. Due to additional learning of the matrix variate noise exploration during the training process, the speed of our approach is a bit slower. However, in the inference stage, the speed of the two methods is similar. We also compare the cumulative rewards of the algorithms over time in three Mario tasks. In Fig. 5, we discover that although more samples can be collected at the same timestep by using method without matrix variate noise, the agent can only find a suboptimal policy due to the lack of effective exploration. Moreover, under the same timestep, a better policy can be learned by using the matrix variate noise.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

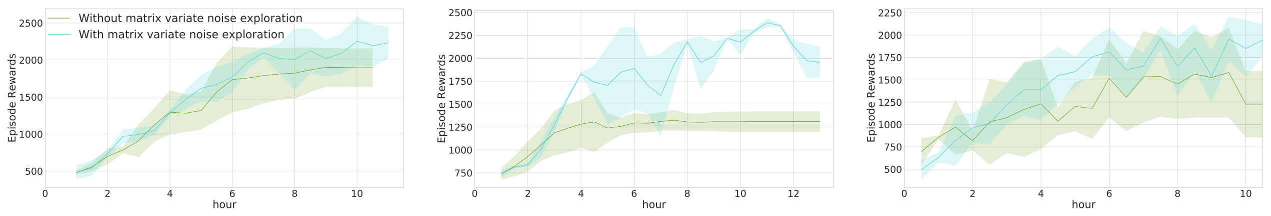


Fig. 5. Performance comparisons between our approach and the variant without using matrix variate noise. We can observe that given the same timesteps, our approach achieves higher rewards.

Then, in Fig. 3, we compare our approach with ϵ-greedy exploration, fully noise factorized noise exploration, and random exploration methods. Our matrix variate noise exploration outperforms ϵ-greedy by a large margin on Mario tasks and achieves better performance with fully factorized noise exploration method in the end. The driving force behind exploration in our approach mainly relies on uncertainty estimation. The uncertainty of events can be defined in two types: epistemic and aleatoric [34]. Aleatoric uncertainty primarily stems from randomness inherent in the environment. Uncertainty estimation is tightly interleaved with efficient exploration in reinforcement learning. To better explain the results of our exploration method, we visualized the effect of uncertainty estimation on the toy example for matrix variate noise exploration and deterministic neural network, respectively. Fig. 6 shows the exploration area driven by our matrix variate noise on the 1-dimensional dataset. The uncertainty is relatively low in the regions where data points are dense. In contrast, the model's uncertainty estimation is large where data is sparse. Accordingly, in order to reduce uncertainty, this requires collecting more training samples in this region. Consequently, uncertainty guides our model to discover novel states. In reinforcement learning, this implies exploring more previously unseen states. The uncertainty learned by our method can also be viewed as a kind of curiosity.

C. Results and Analysis

In this section, we have comprehensively analyzed the results of our experiments. In Fig. 7, the plots show the learning progress in Atari environments. We compare our method with DQN and Noisy-DQN. x-axis means training steps and y-axis means the cumulative rewards. Each curve is averaged across independent runs with different random seeds. The solid line shows the mean of game scores and shaded area represents the variance of each run. For clarity of presentation, each plot in the figure is smoothed with a window size of 10.

All methods have been trained over a long period of training steps with 20 M steps. We can see that our matrix variate noise exploration method performs better than baselines and learns faster on most tasks. We conjecture that matrix variate noise can incorporate the structural information in weight matrix and collect more informative training samples. In riverraid, videopinball tasks, fully factorized noise suffers from falling in local optimal. In contrast, our method does not present premature convergence. We can also observe that matrix variate exploration learns slowly initially in road runner environment, and achieves competitive performance on 10 M time steps. Although, we can still see that

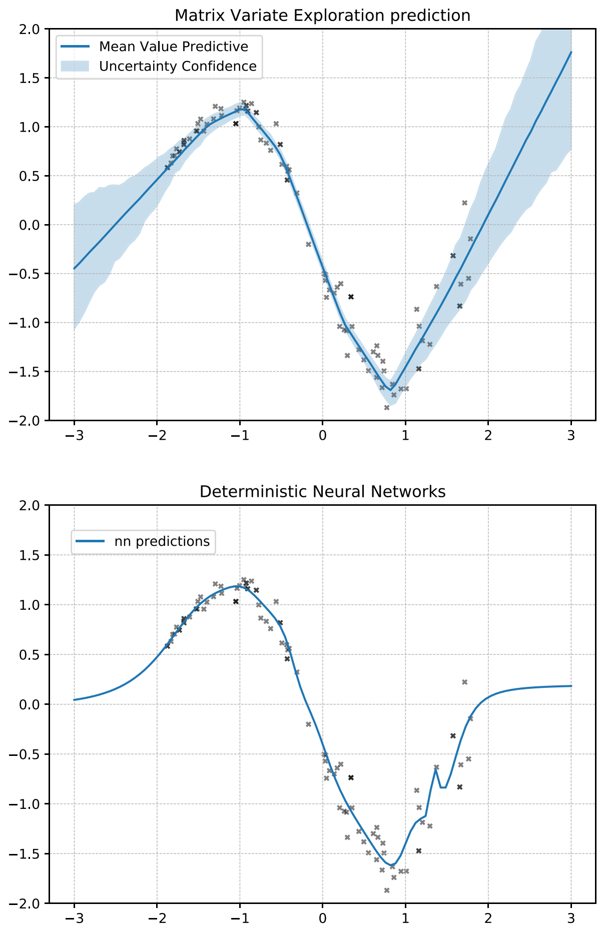
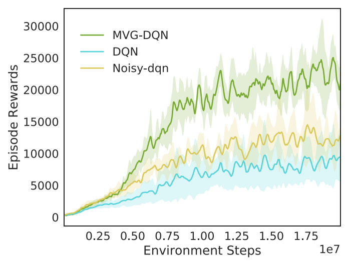


Fig. 6. A comparison between our matrix variate noise exploration and traditional deterministic neural network. Top: prediction from our matrix variate noise exploration. The shaded area is the uncertainty interval. Wide interval indicates high estimated uncertainty. Note that in areas with a lot of data, the uncertainty is low, and conversely in regions with little data, the uncertainty is high. Down: prediction from the traditional neural network. NN outputs a point estimation of the data, which does not reflect the uncertainty clearly.

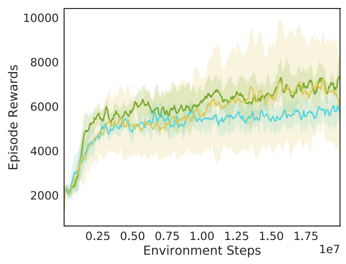
our method is slightly weaker than baselines in zaxxon task. In Fig. 2, we demonstrate that weight uncertainty drives our agent to experience more parts of the game and execute a variety of actions to the environment. Then the agent learns and identifies which specific behavior pattern is beneficial for achieving higher rewards.

In each task, our agents learn to play the game from scratch. To avoid trapping in suboptimal policy, this requires the algorithm to have sufficient exploration capabilities to explore more regions of the game space.

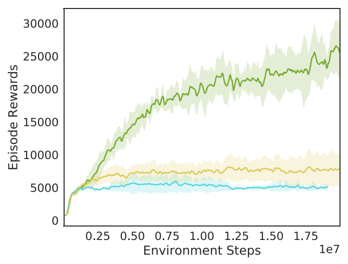
Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.



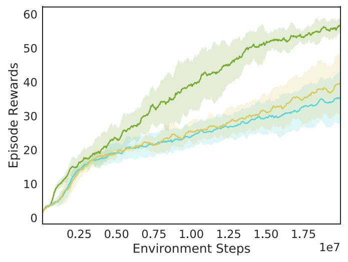
(a) gopher



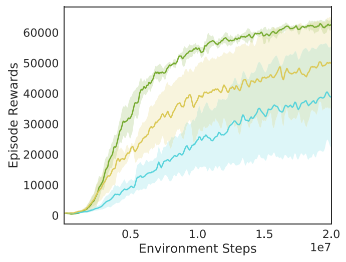
(b) name this game



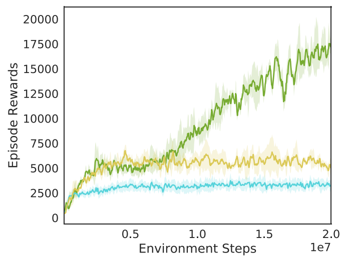
(c) phoenix



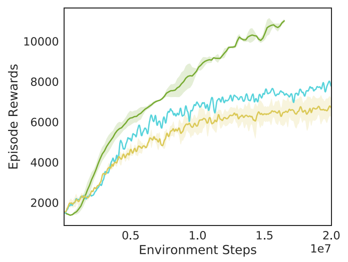
(d) robotank



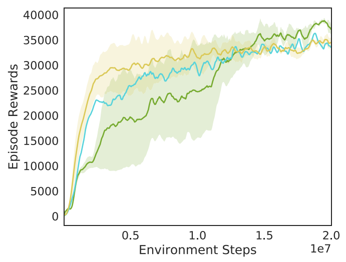
(e) star gunner



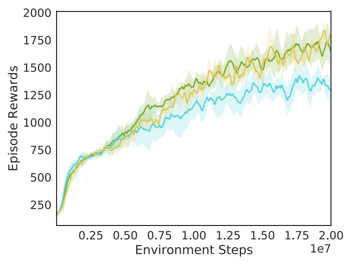
(f) up n down



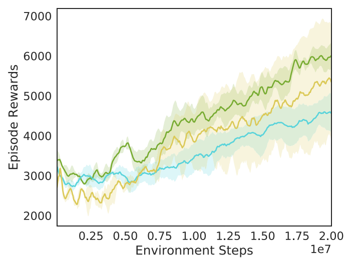
(g) riverraid



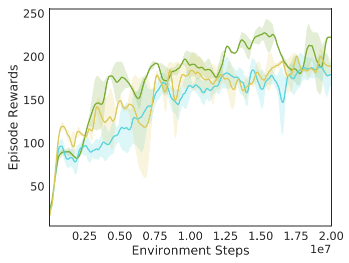
(h) road runner



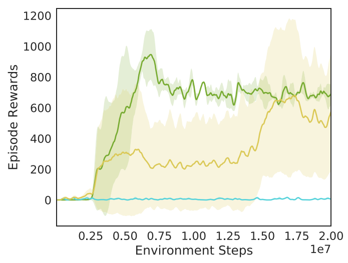
(i) space invaders



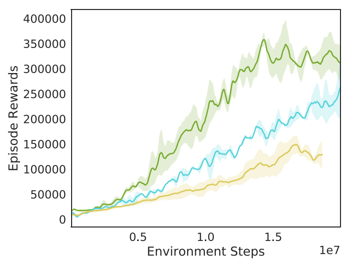
(j) time pilot



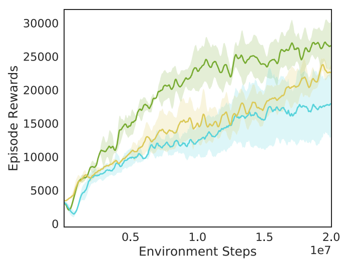
(k) tutankham



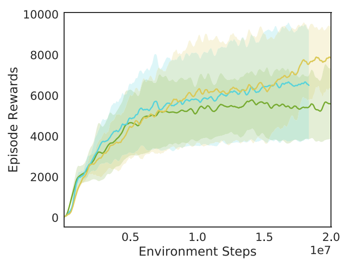
(l) venture



(m) video pinball



(n) yars revenge



(o) zaxxon

Fig. 7. Training curves of MVE-DQN with DQN and Noisy DQN on Atari tasks.

In Super Mario Bros tasks, the agent automatically discovers running from the left side to the right side. Interestingly, the agent itself finds several novel behaviors to accelerate passing the games levels or to avoid fatal events. For instance, the agent learns to escape or kill enemies, as well as cross obstacles and collect gold coins. We note that our method performs better in Super Mario Bros than other action or parameter space exploration baselines. In level-2, there exist more different obstacle structures and various enemies. The agent is likely to walk on mushrooms or get stuck in obstacles. Hence, the algorithm needs to explore more options to avoid these traps to reach the flag. Other baselines are comparatively inefficient and do not take into account weight uncertainty. The parameters of classical NN are deterministic and fail to reflect the uncertainty of the

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SCORES FOR MVE-DQN WITH DQN, NOISY-DQN ON ATARI GAMES

Task	Random	Human	DQN (Nature)	Noisy DQN [10]	DQN (Ours)	Noisy DQN (Ours)	Our approach
gopher	258	2412	6149	10516	5214	10346	**23323**
name this game	2292	**8049**	6183	7153	6524	7519	7715
phoenix	761	7243	6798	12540	5963	8352	**23241**
robotank	2	12	47	53	31	38	**54**
star gunner	664	10250	24582	28861	34544	4365	**58425**
up n down	553	11693	5209	9364	2433	5306	**14041**
riverraid	1338	**17118**	5717	8766	6487	6045	11893
road runner	12	7845	25650	34459	34531	35297	**35841**
space invaders	148	**1669**	1081	1524	1119	1251	1602
time Pilot	3568	5229	4392	**6116**	4312	4868	5812
tutankham	11	168	141	135	137	141	**228**
venture	0	**1188**	586	8	3	362	758
video pinball	16257	17668	88092	152022	210142	14893	**340107**
yars revenge	3093	**54577**	14027	16437	16098	18761	22021
zaxxon	32	9173	4218	**9703**	5745	6353	6853

parameters. The regions are not well explored where data is sparse. Empirically, they require more training samples to learn to dodge or kill enemies.

The agent trained by our matrix variate noise consistently receives rewards during the interaction with the environment. However, it is still challenging to effectively exploit extrinsic rewards to discover novel patterns. The agent should not only learn some basic skills such as jumping and walking, but also more complex skills such as collecting coins and dodging enemies. In Fig. 2, there exist many successive barriers in this level. For an agent that adopts ϵ-greedy strategy, it is likely to hit boulders and get stuck here. When ϵ was very small, it takes many attempts to jump over. In contrast to our approach, when the agent approaches the boulders, it itself explores a specific sequence of key presses to fly up and jump over the successive obstacles.

D. Implementation Details

For all Super Mario and Atari tasks, we apply the same neural network architecture and hyperparameter configuration. Since both environments are pixel-based tasks, we use a convolutional neural network. Our CNN contains three convolutional layers and two fully connected layers and uses ReLU activation functions. In order to satisfy the Markov property, the state consists of the current frame with three previous frames. The parameters of neural network are optimized with Adam optimizer [35] with a learning rate of 1e-4. In our experiments, we found Adam performs better than RMSProp. The selection of learning rate in Adam is not as sensitive as RMSProp. In the original setup, our agent is running on a single CPU machine, and a 10 M steps training with Atari tasks over more than 10 days. To speed up training, we utilize multi-core CPUs and use vectorized environment mechanisms. The agent interacts asynchronously with multiple environments in parallel and an entire training of 20 M steps takes approximately 13 hours of wall-clock time. Our implementation is based on OpenAI Spinningup [1] and the code is available at https://github.com/WangShaoSUN/Structural-Parameter-Space-Exploration-for-Reinforcement-Learning-via-a-Matrix-Variate-Distribution. For DQN agents, we follow the procedure in [2] which utilizes an anneal ϵ decreasing from 1 to 0.1 during the first 1 M steps. In our matrix variate noise method, in order to reflect the exploration ability of our approach, we adopt a faster annealing ϵ which reduces to 0.01 in the first 20 K steps. In practice, it is computationally expensive and memory intensive to learn the entire parameters of rows and columns covariance matrix. For simplicity, we utilize a diagonal approximation to the covariance matrix. In an $m \times n$ weight matrix, we only need $m + n$ perturbation factor to govern the two diagonal matrices. In contrast, a fully factorized Gaussian exploration requires $m \times n$ parameters to generate the exploration noise per layer. Our matrix noise exploration greatly reduces the number of parameters. In fact, we found that the approximate matrix noise exploration performs well.

[1]https://github.com/openai/spinningup

VI. DISCUSSION

In this paper, we focus on integrating the matrix variate approach with a value-based method, which can also benefit policy-based RL algorithms. We compare our method with DQN, Noisy DQN, and human players. Detailed comparisons are shown in Table I which summarizes the results of several DQN methods. To ensure a fair comparison, we use the data of DQN and Noisy DQN from the original paper [10]. The highest score of each game is indicated by bold font. The results show that our method performs significantly better than Noisy DQN. Statistically, our method performs better than Noisy DQN on an average improvement of 48% on Atari games (13 out of 15) and outperforms DQN by a large margin of 98%. Concretely, our approach surpasses human players in nine environments and outperforms the baselines in more than two-thirds of these tasks. In the tasks like gopher, up n down, and riverraid, our method exceeds Noisy DQN by 124.8%, 41.9%, and 19.8%, respectively. These results demonstrate that our approach scales better than fully factorized noise exploration on large-scale state space problems and make more efficient exploration. We find that good exploration relies heavily on structured information in the parameter space. Structural exploration makes the agent more flexible in the environment and more likely to achieve high scores. Our matrix variate noise exploration incentivizes the agent to explore novel states, which brings positive feedback to the environment and results in more rewards. In contrast, we find that our compared strategies are prone to fall into

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

sub-optimal policies. For example, deterministic policies tend to trap in some specific states. In the Mario game, the agent trained by ϵ-greedy is easy to get stuck in the water pipe. In the implementation of our DQN agent, we suggest that the annealing epsilon should be gradually decreased to a threshold value to ensure adequate exploration. It is also observed that too much intensive exploitation during the early stage of training process can affect the final performance. In Venture, DQN fails to handle this task well. Note that fully factorized noise exploration does not make progress in the early stage while our matrix-variate noise exploration learns quickly.

VII. Conclusion

In this work, we concentrate on incorporating structural information into parameter space and make better use of weight uncertainty to explore the environment. We propose a novel parameter space exploration approach called matrix variate noise exploration (MVE). Rather than an isolated view on each weight, our method treats the entire weight matrix as a whole and utilizes the matrix variate noise to explore. Experimental results show that our approach achieves state-of-the-art performance against other variants of DQN and dramatically improves performance compared to other action and parameter space exploration baselines.

In the era of deep reinforcement learning, the value function or policy function is generally parameterized by a neural network. Models of neural networks often contain millions of parameters [36], and their parameter spaces are often very complex. How to explore in such a huge parameter space is still a challenging problem. Our future work is to introduce curiosity exploration into the parameter space and matrix noise exploration can be complementary to other reinforcement learning methods.

References

[1] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33nd Int. Conf. Mach. Learn.*, 2016, vol. 48, pp. 1928–1937. [Online]. Available: http://proceedings.mlr.press/v48/mniha16.html

[4] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *Comput. Res. Repository*, 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[5] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020. [Online]. Available: https://doi.org/10.1177/0278364919887447

[6] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, pp. 2778–2787, 2017.

[7] A. Garivier and O. Cappé, "The KL-UCB algorithm for bounded stochastic bandits and beyond," in *Proc. 24th Annu. Conf. Learn. Theory JMLR Workshop Conf. Proc.*, 2011, pp. 359–376.

[8] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst. 29*, 2016, pp. 1471–1479.

[9] M. Plappert *et al.*, "Parameter space noise for exploration," in *Proc. 6th Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ByBAl2eAZ

[10] M. Fortunato *et al.*, "Noisy networks for exploration," in *Proc. 6th Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rywHCPkAW

[11] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3215–3222.

[12] T. Salimans, J. Ho, X. Chen, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *Comput. Res. Repository*, 2017, *arXiv:1703.03864*. [Online]. Available: http://arxiv.org/abs/1703.03864

[13] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[14] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[15] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, pp. 3875–3884, 2018.

[16] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *Comput. Res. Repository*, 2017, *arXiv:1703.01732*. [Online]. Available: http://arxiv.org/abs/1703.01732

[17] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "UCB and infogain exploration via q-ensembles," *Comput. Res. Repository*, 2017, *arXiv:1706.01502*. [Online]. Available: http://arxiv.org/abs/1706.01502

[18] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. Adv. Neural Inf. Process. Syst. 29*, 2016, pp. 1109–1117.

[19] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by random network distillation," *Comput. Res. Repository*, 2018, *arXiv:1810.12894*. [Online]. Available: http://arxiv.org/abs/1810.12894

[20] I. Rechenberg, "Evolutionsstrategien," in *Proc. Simulationsmethoden Der Medizin Und Biologie*, 1978, pp. 83–114.

[21] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Netw.*, vol. 23, no. 4, pp. 551–559, 2010. [Online]. Available: https://doi.org/10.1016/j.neunet.2009.12.004

[22] T. Rückstieß, M. Felder, and J. Schmidhuber, "State-dependent exploration for policy gradient methods," in *Proc. Mach. Learn. Knowl. Discov. Databases, Euro. Conf.*, vol. 5212, pp. 234–249, 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-87481-2_16

[23] C. Blundell *et al.*, "Weight uncertainty in neural networks," *Proc. Int. Conf. Mach. Learn.*, vol. 37, pp. 1613–1622, 2015.

[24] S. Sun, C. Chen, and L. Carin, "Learning structured weight uncertainty in bayesian neural networks," *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, pp. 1283–1292, 2017.

[25] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," *Proc. 33nd Int. Conf. Mach. Learn.*, vol. 48, pp. 1708–1716, 2016.

[26] R. Zhang, C. Li, C. Chen, and L. Carin, "Learning structural weight uncertainty for sequential decision-making," *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 48, pp. 1137–1146, 2018.

[27] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.

[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. 2nd Int. Conf. Learn. Representations*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6114

[29] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*, vol. 104. Boca Raton, FL, USA: CRC Press, 2018.

[30] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. Baltimore: Johns Hopkins Univ. Press, 1996.

[31] C. Kauten, "Super mario bros for OpenAI gym," *GitHub*, 2018. [Online]. Available: https://github.com/Kautenja/gym-super-mario-bros

[32] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents (extended abstract)," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 4148–4152.

[33] G. Brockman *et al.*, "Openai gym," 2016, *arXiv:1606.01540*.

[34] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Proc. Adv. Neural Inf. Process. Syst. 30*, 2017, pp. 5574–5584.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.

Shaochen Wang received the B.E. degree from the Hefei Unversitiy of Technology, Hefei, China, in 2016. He is currently working toward the Ph.D. degree in control science and engineering with the University of Science and Technology of China, Hefei, China.

His current research interests include reinforcement learning and robotics.

Rui Yang received the B.E. degree in electronic science and technology from the Hefei Unversity of Technology, Hefei, China, in 2019. He is currently working toward the Eng.D. degree in information and communication engineering from the University of Science and Technology of China, Hefei, China.

His research interests include deep reinforcement learning, vision-based reinforcement learning, and machine learning.

Bin Li (Member, IEEE) received the B.S. degree from the Hefei University of Technology, Hefei, China, in 1992, the M.Sc. degree from the Institute of Plasma Physics, China Academy of Science, Hefei, China, in 1995, and the Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2001. He is currently a Professor with the School of Information Science and Technology, USTC. He has authored or coauthored more than 60 refereed publications. His current research interests include computation intelligence, artificial intelligence safety, optimal design, and human-computer interaction. Dr. Li is the Founding Chair of the IEEE CIS Hefei Chapter and the Founding Counselor of the IEEE USTC Student Branch.

Zhen Kan (Member, IEEE) received the Ph.D. degree from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2011. He was a Postdoctoral Research Fellow with Air Force Research Laboratory (AFRL), Eglin Air Force Base and with the University of Florida Research and Engineering Education Facility, Shalimar, FL, USA, from 2012 to 2016, and was an Assistant Professor with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA, USA, from 2016 to 2019. He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests include networked control systems, nonlinear control, formal methods, and robotics. He currently serves on program committees of several internationally recognized scientific and engineering conferences and is an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on July 22,2023 at 10:18:11 UTC from IEEE Xplore. Restrictions apply.