IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 May 2024, accepted 6 May 2024, date of publication 23 May 2024, date of current version 29 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404474

RESEARCH ARTICLE

Uncertainty-Aware Rank-One MIMO Q Network Framework for Accelerated Offline Reinforcement Learning

THANH NGUYEN[1], (Graduate Student Member, IEEE), TUNG M. LUU[1], (Graduate Student Member, IEEE), TRI TON[1], (Graduate Student Member, IEEE), SUNGWOONG KIM[2], (Member, IEEE), AND CHANG D. YOO[1], (Senior Member, IEEE)

[1]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea
[2]Department of Artificial Intelligence, Korea University, Seoul 02841, Republic of Korea

Corresponding author: Chang D. Yoo (cd_yoo@kaist.ac.kr)

ABSTRACT Offline reinforcement learning (RL) has garnered significant interest due to its safe and easily scalable paradigm, which essentially requires training policies from pre-collected datasets without the need for additional environment interaction. However, training under this paradigm presents its own challenge: the extrapolation error stemming from out-of-distribution (OOD) data. Existing methodologies have endeavored to address this issue through means like penalizing OOD Q-values or imposing similarity constraints on the learned policy and the behavior policy. Nonetheless, these approaches are often beset by limitations such as being overly conservative in utilizing OOD data, imprecise OOD data characterization, and significant computational overhead. To address these challenges, this paper introduces an Uncertainty-Aware Rank-One Multi-Input Multi-Output (MIMO) Q Network framework. The framework aims to enhance Offline Reinforcement Learning by fully leveraging the potential of OOD data while still ensuring efficiency in the learning process. Specifically, the framework quantifies data uncertainty and harnesses it in the training losses, aiming to train a policy that maximizes the lower confidence bound of the corresponding Q-function. Furthermore, a Rank-One MIMO architecture is introduced to model the uncertainty-aware Q-function, offering the same ability for uncertainty quantification as an ensemble of networks but with a cost nearly equivalent to that of a single network. Consequently, this framework strikes a harmonious balance between precision, speed, and memory efficiency, culminating in improved overall performance. Extensive experimentation on the D4RL benchmark demonstrates that the framework attains state-of-the-art performance while remaining computationally efficient. By incorporating the concept of uncertainty quantification, our framework offers a promising avenue to alleviate extrapolation errors and enhance the efficiency of offline RL.

INDEX TERMS Self-supervise learning, computer vision, contrastive learning, deep learning, transfer learning.

I. INTRODUCTION

Offline reinforcement learning (RL), also known as batch RL, addresses the challenge of training a policy solely from a fixed dataset without additional interaction with the environment. This approach offers benefits in terms of data efficiency, scalability, and safety, particularly in real-world applications such as navigation and healthcare [10], [25], [26], leading to a drastically increasing attention in recent years [13], [14], [23]. However, learning from pre-collected datasets presents a well-known challenge: the extrapolation error when performing policy evaluation on out-of-distribution

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

(OOD) data. Specifically, estimating the Q-values of the learned policy using the offline dataset is prone to bias due to the distributional shift caused by differences in visitation distribution between the learned policy and the behavior policy used for collecting the dataset. This bias, once combined with deep network fitting in deep RL algorithms, leads to a significant extrapolation error characterized by the Q-function's significant overestimation of OOD data, as evidenced in [13]. Addressing this error in offline RL proves challenging due to the absence of interaction for collecting additional data, a process common in conventional RL methods, to rectify it.

Confronted with the inherent limitations of the problem, offline RL algorithms overcome the challenge by proposing different losses or training procedures capable of mitigating extrapolation errors. Early methods directly limit the distributional shift by constraining the learned policy to be similar to the behavior policy [12], [13], [14], [50]. This approach has a notable drawback: the learned policy is heavily influenced by the behavior policies, leading to suboptimal results in datasets collected by non-optimal behavior ones. Later methods adopt a different approach by making conservative estimates of future rewards, aiming to learn a value function that serves as a strict lower bound to the true value function [23], [25], [26]. Typically, this involves penalizing Q-functions for OOD actions. While these methods offer more freedom for learning better policies, the penalizing term often lacks precise characterization of OOD data (e.g. equally penalizes the OOD actions), resulting in overly conservative value functions [7]. Recent methods, considering uncertainty about the value function [7], [21], [52], have been proposed to address the excessively pessimistic nature of the aforementioned approaches. These methods commonly employ separate Q-functions as a Q-ensemble, enabling uncertainty-aware penalization. By penalizing conflicting actions and favoring decisions that are consistent across the models, they generate pessimistic Q-values to train the learned policy [2], [6], [14], [53], and have proven to achieve state-of-the-art performances. However, the use of separate Q-functions incurs high computational and memory complexity. Moreover, these methods introduce additional hyperparameters and require a large number of ensemble members to be effective, posing difficulties for large and complex datasets.

Taking into account the shortcomings of prior approaches as a whole, which include a conservative use of OOD data, imprecise characterization of such data, and substantial computational overhead, we propose the end-to-end Uncertainty-Aware Rank-One MIMO Q Network framework for improving offline RL. Our framework fully leverages uncertainty quantification to effectively utilize OOD data, thereby enhancing the learning process while ensuring fast training and memory efficiency.

Our contribution can be listed in detail as follows:

- Firstly, we introduce a novel architecture, *i.e.* Rank-One MIMO Q network, for approximating the naive Q ensemble. The Rank-One MIMO Q network combines a common shared network with mini rank-one network adapters, allowing these adapters to fuse the common shared network to assemble ensemble members. This network can handle multiple inputs and generate multiple outputs simultaneously. As a result, the proposed network offers the same capability for uncertainty quantification as an ensemble of networks but with a cost nearly equivalent to that of a single network. This result is achieved by recognizing that members of an ensemble can collectively acquire and retain certain common knowledge about the environment, thereby eliminating the necessity for individual learning and storage. In our design, within each layer, the MIMO Q network stores shared knowledge in the shared network weight matrix. Individual members can then augment this shared knowledge with their unique insights using their independent weights, modeled by two vectors. This design minimizes the parameter overhead of our MIMO Q compared to the naive ensemble, thanks to the utilization of a shared body. Furthermore, the MIMO Q is facilitated with matrix vectorization to enable uncertainty prediction in a single forward pass, optimizing prediction speed efficiency.
- Secondly, we propose pessimistic training losses as a means to leverage uncertainty quantification for effectively utilizing OOD data. These losses are based on maximizing the lower confidence bounds (LCB) of Q values, which have been proven to be a precise characterization of OOD data. Specifically, our proposed losses are designed to train the MIMO Q network and the policy network based on the conservatively estimated Q value suggested by the min-valued MIMO Q head. This design enables efficient estimates of the LCB of Q-values with only one hyper-parameter, the number of Q heads, for controlling pessimism. Furthermore, the backward pass only needs to propagate through the min-valued Q member instead of all the ensemble members, as conventionally done in the naive ensemble method. This further enhances the speed of the learning process.
- Thirdly, we enhance training stability and mitigate overestimation without the need for an OOD sampling scheme by incorporating two components: (1) maximizing entropy for OOD actions while maximizing likelihood for in-distribution actions. It plays a crucial role in preventing excessive exploitation of OOD actions, thereby reducing the risk of diverging Q-values, while simultaneously encouraging the exploitation of trustworthy in-distribution actions. These additions are particularly effective for low-coverage datasets, such as expert datasets; (2) employing a lazy policy improvement trick. This not only saves computational costs but also enhances the stability of policy evaluation.
- Finally, we conduct extensive experimental results and rigorous ablation studies. The experimental result

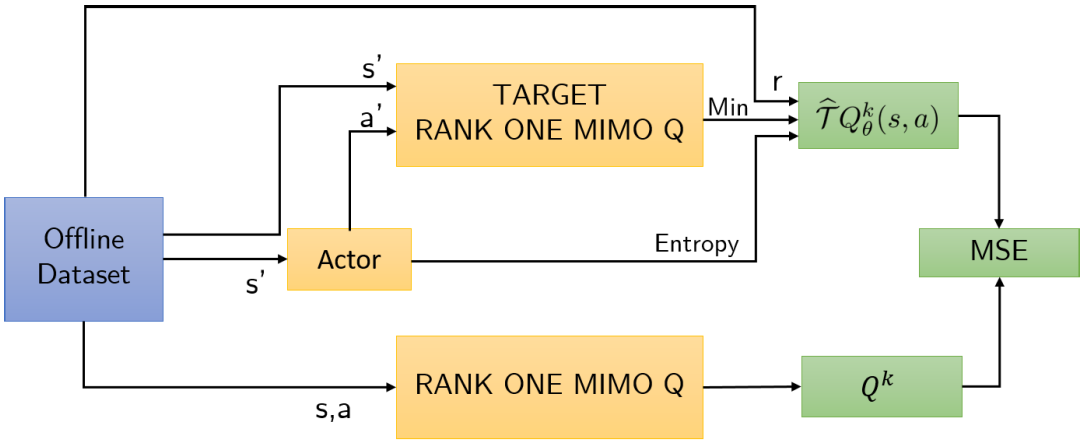

FIGURE 1. Illustration of the proposed workflow: Our framework adopts a conventional actor-critic setup, where the online Q network and target Q network are alternated by the proposed Rank-One MIMO Q, while the policy is modeled by a stochastic Gaussian actor. As the framework prohibits direct interaction with the environment, transitions are sampled from a pre-collected offline dataset. These sampled transitions are then utilized to calculate the target, employing the minimum-valued head of the target network and policy entropy. Finally, the calculated target is backed up to the online network using mean square error during training.

demonstrates that the framework achieves state-of-the-art performance on the D4RL benchmark [11] while being computationally friendly compared to strong baselines. The ablation studies are carefully chosen to provide a better understanding of the framework.

Broadly, this work highlights the importance of developing efficient, stable ensembling techniques specifically designed for offline RL. Additionally, it underscores the potential of offline RL as a testbed for validating uncertainty estimation techniques and raises intriguing research questions for future exploration.

The paper is organized as follows: Section I introduces the topic, followed by Section II, which delves into related work. Section III provides background information on offline RL, and Section IV thoroughly outlines our methodology, encompassing the Rank-One MIMO Q network and the Uncertainty-Aware Rank-One MIMO Q Network framework. Section V presents the experiment setup. Section VI shows the experimental results, and Section VII conducts an ablation study. Finally, Section VIII draws the paper to a conclusion and explores potential avenues for future research.

II. RELATED WORK

A. OFFLINE REINFORCEMENT LEARNING

This paper primarily delves into model-free offline Reinforcement Learning (RL). Early methods pinpoint the core issue in offline RL as extrapolation error [13] and suggest using policy constraints to ensure that the learned policy remains close to the behavior policy. These constraints include adding behavior cloning (BC) loss [46] in policy training [12], using the divergence between the behavior policy and the learned policy [13], [14], [25], applying advantage-weighted constraints to balance BC and advantages [39], penalizing the prediction-error of a variational auto-encoder [41], and learning latent actions from the offline data [55]. While policy-constraint methods excel in performance on datasets derived from expert behavior policies, they struggle to discover optimal policies when confronted with datasets featuring suboptimal policies. This limitation arises from the stringent constraints imposed on the learned policies [28], [35]. Furthermore, these methods necessitate precise estimation of the behavior policy, a task typically challenging in complex settings characterized by multiple sources of behavior or high-dimensional environments. Subsequent methods circumvent these limitations by instead learning a pessimistic Q-function, which serves as a lower bound estimate of the true value function. This is performed by penalizing their Q-value [8], [25], [26] or using V-learning [24], [32]. While offering increased flexibility for improved policy learning, the penalizing terms in these methods often lack a precise characterization of OOD data. For instance, CQL [26] uniformly penalizes the Q-values of all OOD samples, a practice proven to result in conservative value functions, as demonstrated in [7]. Our method overcomes the limitations of previous approaches by adopting uncertainty quantification to selectively penalize OOD data. While our approach shares similarities with CQL [26] in promoting conservatism in Q-learning, it distinguishes itself by explicitly measuring the uncertainty of OOD actions rather than applying a uniform penalty.

It's worth noting that uncertainty quantification has been used in online RL [4], [5], [33], [36], [43], [53]. However, offline RL poses their own challenge for uncertainty quantification compared to online RL, primarily due to the limited coverage of offline data and the distribution shift of learned policies. Several existing model-based and model-free approaches have been proposed to overcome this challenge. In model-based offline RL, representative

works include MOPO [54] and MOReL [22], which utilize an ensemble dynamics model for uncertainty quantification, while BOPAH [27] combines uncertainty penalization with behavior-policy constraints. However, model-based methods may suffer from additional computation costs and may perform sub-optimally in complex environments [9], [19]. In contrast, our work conducts model-free learning, which is less affected by these problems. In addition, model-free learning is also considered more favorable due to its simplicity and potential for high performance. To be noticed, while there are similar approaches proposed in model-free offline RL, our work stands apart from them. To be specific, UWAC [51] uses dropout-based uncertainty for model-free offline RL but relies on policy constraints for value function learning [15]. In comparison, our method with the proposed MIMO Q network demonstrates greater robustness to distributional shifts compared to the ensemble and dropout-based approaches. EDAC [3] also employs ensemble Q networks, but it diversifies gradients to penalize OOD actions, and PBRL [6] penalizes OOD actions through direct OOD sampling and associated uncertainty quantification, whereas our method doesn't require further OOD sampling.

B. EFFICIENT UNCERTAINTY QUANTIFICATION IN SUPERVISED LEARNING

Our work draws inspiration from recent advancements in efficient uncertainty quantification in Supervised Learning to identify appropriate approaches. It is well-known that ensembles have demonstrated their effectiveness in estimating uncertainty in RL, just as they have in supervised deep learning [3], [6], [44]. Despite the availability of other technical methods for uncertainty estimation [30], [38], ensembles consistently outperform them, albeit at a higher computational cost. Therefore, in supervised learning, much of the current research on ensembles is aimed at improving computational efficiency, with proposals for reducing compute or memory footprint during training and inference on large ensembles [18], [31], [48]. Several architectures have been proposed to enable shared knowledge among ensemble members. The multi-head approach [29], [37], [47] designs ensembles that share a big ''trunk'' network and have separate ''head'' networks for each ensemble member. While multi-head approaches offer reduced computational costs compared to typical ensembles by sharing many layers, they often lack ensemble diversity [18]. On the other hand, MIMO [18] overcomes this problem by modifying both the input and output layers to be a multi-input multi-output network, allowing each ensemble member to take different paths throughout the full network. However, MIMO networks are reported to struggle with fitting more than 2 subnetworks [40], necessitating special care for the shared body layers [40], [45]. Our work shares similarities with the MIMO approach, an improved version of the multi-head approach. However, we explicitly model shared knowledge and each ensemble member individually, a strategy that has been proven to be effective for fitting multiple sub-networks [48]. Beyond the domain of supervised learning, within the realm of offline reinforcement learning, there is limited empirical evidence showcasing the effectiveness of these approaches. Inspired by efficient methods in supervised learning, our work aims to leverage these techniques to benefit offline reinforcement learning.

III. BACKGROUND

Offline reinforcement learning (RL) is a research field that distinguishes itself from online RL by enabling training without the need for active interaction with the environment. This involves utilizing a fixed dataset, collected by an unknown behavior policy, to learn a policy that maximizes the cumulative reward of a target environment. By leveraging previously collected data, offline RL offers a more practical and efficient solution for RL training in complex domains where online interaction is not feasible or too expensive.

This paper considers the fully-observed Markov Decision Process (MDP) as commonly utilized in offline RL research. Mathematically, the MDP is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, \rho_0, R, \gamma, H)$. Therein, $\mathcal{S}$ is a set of state s, $\mathcal{A}$ is a set of actions a, $\mathbb{P}$ is the transition probability of the dynamics in the form $\mathbb{P}(s_{t+1}|s_t, a_t)$, ρ_0 is the initial state distribution, R is reward function, $\gamma \in (0, 1]$ is a scalar discount factor, and H is the horizon.

Within a MDP, offline RL try to learn a policy $\pi(a_t|s_t)$ which is the probability of taking action a_t conditioned on the current state s_t. A trajectory distribution, which is a sequence of $H + 1$ states and H actions, can be further derived as $\tau = (s_0, a_0, \ldots, s_H)$ where H can be infinite. The probability density function for a given trajectory τ under policy π is as below:

$$p_\pi(\tau) = \rho_0(s_0) \prod_{t=0}^{H-1} \pi(a_t \mid s_t)\, \mathbb{P}(s_{t+1} \mid s_t, a_t). \quad (1)$$

In offline RL, a dataset comprising multiple pre-collected transitions is given, conveniently denoted as $D = \{(s, a, s', r)\}$. Here, s' represents the next state resulting from taking action a at the current state s and receiving a return r. This dataset is gathered by an unknown behavior policy π_β. The primary objective of offline RL is to learn an optimal π^* policy that maximizes the expected cumulative return of the learned policy π. The objective can be formulated as follows:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \quad \mathbb{E}_{\tau \sim p_\pi}\left[\Sigma_{t=0}^{H-1} \gamma^t R(s_t, a_t)\right]. \quad (2)$$

The optimal policy's corresponding Q-function satisfies the Bellman operator as shown below:

$$\mathcal{T}Q_\theta(s, a) := R(s, a) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)}\left[\max_{a'} Q_{\theta^-}(s', a')\right], \quad (3)$$

where θ represents the parameters of the Q network, and θ^- represents the parameters of the target-network, which is a copy of the Q network with momentum update for training stabilization [34].

Offline RL poses challenges due to the distribution shift when training policies from a pre-collected dataset. Value function evaluated on the greedy action a' in the Bellman operator $\mathcal{T}Q(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{a'} Q(s',a')\right]$ tends to have extrapolation errors, as the combination of state-action pairs (s',a') may have rarely occurred in the dataset D. To overcome this issue, early model-free offline RL methods incorporate conservatism by constraining learned policies to be similar to the behavior policy or penalizing values of OOD actions. However, these methods often limit the generalization of value functions beyond the offline data and lack precise characterization of OOD data. Uncertainty quantification is a promising way to enhance performance. Online RL typically uses upper-confidence bound (UCB) to encourage exploration, while offline RL focuses on fixed training data and uses lower-confidence bound (LCB) to estimate Q-values and avoid risky actions. Leveraging the LCB is supported by strong theoretical evidence in offline RL. Recent theoretical analyses, such as those discussed in [21], [52], prove the importance of uncertainty quantification in achieving provable efficiency in RL. Pessimistic Value Iteration [21] introduces an ϵ-uncertainty quantifier, which serves as a penalty and enables provable efficient pessimism in offline RL. In the context of linear MDPs, a LCB-penalty [1], [20] is proposed and is a known ϵ-uncertainty quantifier. In the context of function approximation, it is shown that the bootstrapped uncertainty provides an estimation of the LCB-penalty and enables efficient pessimism in many complex offline RL tasks [6]. This is achieved by utilizing the Q learning form below, where the K bootstrapped Q-functions in critic are used to quantify the epistemic uncertainty.

$$\begin{aligned}&\widehat{\mathcal{T}}Q^k_\theta(s,a)\\&\quad = R(s,a)\\&\qquad + \gamma \widehat{\mathbb{E}}_{s'\sim D, a'\sim\pi(\cdot|s)}\left[\bar{Q}_{\theta^-}\left(s',a'\right) - \beta \mathcal{U}_{\theta^-}\left(s',a'\right)\right] \quad (4)\\&\mathcal{U}_{\theta^-}(s,a)\\&\quad = \mathrm{Std}\left(Q^k_{\theta^-}(s,a)\right)\\&\quad = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(Q^k_{\theta^-}(s,a) - \bar{Q_{\theta^-}}(s,a)\right)^2}. \quad (5)\end{aligned}$$

In this formulation, Q^k_θ represents the k^{th} Q-function in the ensemble of bootstrapped Q-functions, while $\bar{Q}$ is the mean among the target-networks. The empirical Bellman operator $\hat{\mathcal{T}}$ aims to estimate the expected maximum Q-value $\mathbb{E}\left[R(s,a) + \gamma \max_{a'} Q_{\theta^-}\left(s',a'\right) \mid s,a\right]$ based on the offline dataset, as a' is sampled from the learned policy that is designed to maximize the Q-function. The epistemic uncertainty, $\mathcal{U}$, is quantified by the deviation among the bootstrapped Q-functions, and is used as a penalization in estimating the Q-functions. From a Bayesian perspective, the ensemble approach allows for estimation of the posterior distribution of the Q-functions, yielding similar values in areas with rich data and diverse values in areas with scarce data. A potential drawback of this approach is its high computational cost, which can be attributed to the use of a naive ensemble and its non-efficient objective forms [3], [6], [16].

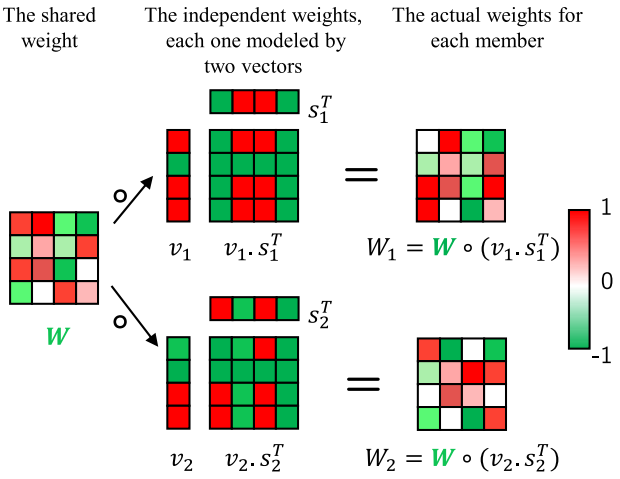

FIGURE 2. Demonstration of how to generate ensemble weights for two ensemble members from one rank-one layer. It's important to note that the training weight stored in each rank-one layer consists of only the shared weight and two vectors for each member. The actual weights for each member are calculated on demand following equation 6.

IV. METHODOLOGY

A. RANK-ONE MIMO Q NETWORK (MIMO Q)

Inspired by recent work in efficient ensemble [18], [48], we introduce the MIMO Q for fast and memory-efficient Q ensembling. The philosophy behind our method is that members of the ensemble can share certain common knowledge about the environment, which need not be learned and stored individually. Instead, this knowledge can be acquired collectively and stored in a shared weight matrix, which individual members can then use to learn their own unique knowledge with their own individual weights. As a result, we can model the ensemble as the product of a shared matrix and a rank-one matrix personalized for each member, enabling us to combine both collective and individual learning in a powerful and efficient way.

Following this philosophy, we design the architecture of MIMO Q network accordingly. The MIMO Q network will function as an ensemble of K members, as usual, meaning it will receive K inputs and produce corresponding K outputs. However, the MIMO Q network is constructed by stacking multiple special layers, referred to as rank-one layers, akin to building a multilayer perceptron from dense layers, rather than creating K independent networks for each member in the ensemble. The pivotal aspect unfolds within the rank-one layer, meticulously crafted to model both a shared weight and K individual weights. Specifically, the learnable weights for a rank-one layer consist of a shared weight that is common across all ensemble members, as well as K independent weights that correspond individually to each of the K ensemble members. Mathematically, the shared weight is essentially a conventional dense layer, denoted by $W \in \mathbb{R}^{m\times n}$, where m

represents the input dimension and n represents the output dimension. On the other hand, one independent weight consists of a pair of vectors, $v \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$, with dimensions m and n respectively. The layer comprises K ensemble independent weights, implying that there are K pairs of trainable vectors v_k and s_k, for k ranging from 1 to K. During the forward pass, the rank-one layer constructs the actual ensemble weights W_k by fusing the shared weight and the individual weights. This fusion is formulated using the following procedure:

$$W_k = W \circ (v_k s_k^\top), \tag{6}$$

where $\circ$ is element-wise multiplication and $(v_k s_k^\top)$ is the dot product to recover the rank-one matrix from two vectors. Figure 2 visualizes the process of creating actual weight for each member of the ensemble having two members.

To make the ensemble weight generation process parallelizable on a device, vectorization can be used. This allows the forward pass to be computed with respect to multiple ensemble members at once. The computation is accomplished through manipulation of matrix computations for a mini-batch [49]. Specifically, let x represent the feature map inputing to a neural network layer. The output feature map, y, are then given by:

$$\begin{aligned} y_{bk} &= \Phi\left(W_k^\top x_b\right) \\ &= \Phi\left(\left(W \circ v_k s_k^\top\right)^\top x_b\right) \\ &= \Phi\left(\left(W^\top (x_b \circ v_k)\right) \circ s_k\right), \end{aligned} \tag{7}$$

where Φ denotes the activation function and the subscript b represents the index in the mini-batch. The output, y_{bk}, represents next layer's feature map input from the k^{th} ensemble member. To enable vectorization of these computations, matrices V and S are defined such that their rows consist of the vectors v_k and s_k for all examples in the mini-batch. With this, the equation above can be expressed in a vectorized form as follows:

$$Y = \Phi(((X \circ V)W) \circ S), \tag{8}$$

where X is the mini-batch input. By computing equation 8, we can obtain the next layer's feature map input for each ensemble member in a mini-batch-friendly way. This allows us to take full advantage of parallel accelerators to implement the ensemble efficiently. To match the input and the ensemble weight, we can divide the input mini-batch into K sub-batches and each sub-batch receives an ensemble weight $W_k, k = \{1, \ldots, K\}$ or we can repeat the input mini-batch K times.

For evaluation, when the test batch size is B and there are K ensemble members, we optimize efficiency by repeating the input mini-batch K times, resulting in an effective batch size of $B \times K$. This allows all ensemble members to compute the output for the same B input data points in a single forward pass.

Diversity in the weights of ensemble members is a crucial factor for an ensemble to achieve high performance and reduce the number of members required for effective uncertainty quantification. Encouraging diversity during training can be achieved through the use of a loss function (e.g., diversity loss) [3], but this approach incurs additional computational costs. Instead, we adopt an alternative method by initializing individual weights as random sign vectors [48], which has been found to yield satisfactory results without incurring any extra computational overhead.

B. UNCERTAINTY-AWARE RANK-ONE MIMO Q NETWORK FRAMEWORK

Considering PBRL as the baseline, the proposed framework based on bootstrapped uncertainty quantification builds on the actor-critic scheme, consisting of policy evaluation and policy improvement phases. The Uncertainty-Aware Rank-One MIMO Q Network framework additionally modifies both the policy evaluation loss and the policy improvement loss to include the benefits of the framework.

In our policy evaluation, we integrate two key components: an effective term designed to approximate the lower confidence bound (LCB) of the Q-value predictions, and a value bonus derived from OOD action entropy. The proposed Bellman equation is as follows:

$$\begin{aligned} \widehat{\mathcal{T}} Q_\theta^k(s,a) = R(s,a) \\ + \gamma \widehat{\mathbb{E}}_{s' \sim D, a' \sim \pi(\cdot|s')} \\ \times \left[\min_{k=1,\ldots,K} Q_{\theta^-}^k\left(s', a'\right) - \alpha \log \pi_\phi\left(a' \mid s'\right) \right]. \end{aligned} \tag{9}$$

Let's delve into the details of each component. Addressing extrapolation error using the uncertainty quantification approach is done by maximizing the LCB of the Q-values. Instead of using equation 4 to compute the expected target, our framework chooses the worst-case Q-value, which can be interpreted as an approximation of the lower confidence bound (LCB) of the Q-value predictions. Mathematically, suppose $Q(s,a)$ follows a Gaussian distribution with mean μ and standard deviation σ. Let $\left\{Q^k(s,a)\right\}_{k=1}^K$ be realizations of $Q(s,a)$. The expected minimum of the realizations can be approximated using the work of Royston et al. [42], which also employed in [3], [12], [14], and [17] as:

$$\begin{aligned} &\mathbb{E}\left[\min_{k=1,\ldots,K} Q^k(s,a)\right] \\ &\approx \mu(s,a) - \mathcal{F}^{-1}\left(\frac{K - \frac{\pi}{8}}{K - \frac{\pi}{4} + 1}\right)\sigma(s,a). \end{aligned} \tag{10}$$

In the equation, $\mathcal{F}$ denotes the cumulative distribution function of the standard Gaussian distribution. This relation reveals that using the minimum value approximates the penalty on the ensemble mean of the Q-values minus the standard deviation scaled by a coefficient that depends on the number of ensembles, denoted by K. This approximation enables the estimation of the LCB in a computationally efficient manner. Furthermore, in the backward pass, employing this approximation results in the loss being backpropagated solely

TABLE 1. The average normalized scores and standard deviations of all algorithms across four seeds in the D4RL benchmark [11]. The highest-performing scores are highlighted for each dataset. The state-of-the-art scores referenced in this study are from [6] and [53].

Task Name		BCQ [13]	IQL [24]	BEAR [25]	UWAV [51]	CQL [26]	MOPO [54]	TD3-BC [12]	EDAC-10 [3]	PBRL [6]	OURS
Random	HalfCheetah	2.3 ±0.0	14.4 ±2.3	2.3 ±0.0	2.3 ±0.0	17.5 ±1.5	35.9 ±2.9	11.0 ±1.1	13.4 ± 1.1	11.0 ±5.8	**31.5±2.3**
	Hopper	10.5 ±0.2	11.1 ±0.1	3.9 ±2.3	2.7 ±0.3	7.9 ±0.4	16.7 ±12.2	8.5 ±0.6	16.9±10.1	26.8 ±9.3	**31.3±0.6**
	Walker2d	4.2 ±1.5	5.8 ±0.3	12.8 ±10.2	2.0 ±0.4	5.1 ±1.3	4.2 ±5.7	1.6 ±1.7	6.7±8.8	8.1 ±4.4	**21.3±0.2**
Medium	HalfCheetah	40.1 ±0.5	41.7 ±6.2	43.0 ±0.2	42.2 ±0.4	47.0 ±0.5	**73.1 ±2.4**	48.3 ±0.3	64.1±1.1	57.9 ±1.5	68.6±2.6
	Hopper	52.5 ±22.8	64.1±0.3	51.8 ±4.0	50.9 ±4.4	53.0 ±28.5	38.3 ±34.9	59.3 ±4.2	**103.6±0.2**	75.3 ±31.2	93.2±7.4
	Walker2d	47.7±7.8	78.3±0.6	-0.2 ±0.1	75.4 ±3.0	73.3 ±17.7	41.2 ±30.8	83.7 ±2.1	87.6±11.0	89.6 ±0.7	**92.2±0.3**
Medium Replay	HalfCheetah	39.0±1.9	42.5±0.3	36.3 ±3.1	35.9 ±3.7	45.5 ±0.7	69.2 ±1.1	44.6 ±0.5	60.1±0.3	45.1 ±8.0	**59.9±2.4**
	Hopper	11.1±7.7	81.9±17.9	52.2 ±19.3	25.3 ±1.7	88.7 ±12.9	32.7 ±9.4	60.9 ±18.8	102.8±0.3	100.6 ±1.0	**102.9±0.7**
	Walker2d	15.2±4.6	70.9±3.4	7.0 ±7.8	23.6 ±6.9	81.8 ±2.7	73.7 ±9.4	81.8 ±5.5	94.0±1.2	77.7 ±14.5	**100.6±3.4**
Medium Expert	HalfCheetah	60.1±11.1	75.4±3.5	46.0 ±4.7	42.7 ±0.3	75.6 ±25.7	70.3 ±21.9	90.7 ±4.3	**107.2±1.0**	92.3 ±1.1	100.9 ±3.3
	Hopper	110.5±2.8	91.5±0.2	50.6 ±25.3	44.9 ±8.1	105.6 ±12.9	60.6 ±32.5	98.0 ±9.4	58.1±22.3	110.8 ±0.8	**111.6±0.7**
	Walker2d	43.6±14.0	109.6±1.4	22.1 ±44.9	96.5 ±9.1	107.9 ±1.6	77.4 ±27.9	110.1 ±0.5	**115.4±0.5**	110.1 ±0.3	112.9±1.2
Expert	HalfCheetah	90.4±7.5	94.8±2.5	92.7 ±0.6	92.9 ±0.6	96.3 ±1.3	81.3 ±21.8	96.7 ±1.1	104.0±0.8	92.4 ±1.7	**105.4±2.3**
	Hopper	103.6±5.0	106.2±8.9	54.6 ±21.0	110.5 ±0.5	96.5 ±28.0	62.5 ±29.0	107.8 ±7	77.0±43.9	**110.5 ±0.4**	107.8±0.9
	Walker2d	110.4±0.2	109.0±0.3	106.6 ±6.8	108.4 ±0.4	108.5 ±0.5	62.4 ±3.2	110.2 ±0.3	57.8±55.7	108.3 ±0.3	**112.7±0.3**
Average		49.4	66.5	38.78	50.41	67.35	53.3	67.55	71.2	74.37	**83.6**

through the minimum-value Q network rather than through all ensemble members. This property is particularly advantageous in the context of our framework, as its computational costs remain insensitive to K.

Regarding the value bonus, the Q-function receives an additional boost from the entropy of actions generated by the learned policy in the next stage, which has a high probability of being OOD data. This approach encourages the Q-function to avoid over-reliance on any specific high-value OOD data, thereby preventing the exploitation of OOD data, which easily leads to over-estimation.

In the policy improvement, we use the combination of (1) minimum MIMO Q value, (2) the entropy of policy-generated action, and (3) the likelihood of action in the dataset to derive the corresponding policy by solving the following maximization problem:

$$\pi_\phi = \max_\phi \widehat{\mathbb{E}}_{s,a\sim\mathcal{D},a'\sim\pi(\cdot|s)}[\min_{k=1,\ldots,K} Q^k(s,a') - \alpha \log \pi_\phi\left(a' \mid s\right) + \beta \log \pi_\phi\left(a \mid s\right)], \quad (11)$$

where ϕ represents the policy parameters.

The technique of maximizing entropy is utilized to encourage a diverse set of actions in a given state. This is achieved by treating the policy as a probability distribution over actions given a state and maximizing its entropy, as done in SAC [17]. By doing so, the policy becomes less deterministic, allowing the agent to explore more effectively and reducing the risk of overfitting the training data. Furthermore, maximizing entropy has a regularizing effect on the policy evaluation process. Encouraging the policy to take a diverse set of actions also reduces the risk of exploiting maximum actions (highly likely to be adversarial examples), which can lead to diverging Q-values in the long run. From another perspective, using a high-entropy Gaussian policy has the effect of smoothing out the Q target, which improves the robustness of the Q-function to outliers. Furthermore, in the context of offline RL, we must balance the use of in-distribution and out-of-distribution data, which respectively be sampled from the dataset and sampled from the learned policy. While OOD data can be useful for seeking a better optimistic result, in-distribution data is more trustworthy and should be given some priority. Therefore, the log-likelihood term in the loss function is designed to incentivize the policy to favor actions that are in distribution, ensuring a balanced and effective learning strategy. The maximization of the log-likelihood term is particularly useful in low-coverage environments, such as expert datasets. In high-coverage environments, this term may be less important and can be eliminated without any impact on performance. As a whole, our objective for policy improvement allows us to avoid penalizing OOD data using some additional high computational cost losses, as is done in some other approaches such as PBRL, CQL [6], [26].

It is worth noting that updating the policy network too frequently not only incurs higher computational costs but can also introduce instability during policy evaluation. Therefore, in our approach, we update the policy network after a certain number of policy evaluation updates. This ensures a balance between computational efficiency and stability in the learning process.

The overall architecture of our framework is illustrated in figure 1.

V. EXPERIMENTAL SETUP

Our method is evaluated on the D4RL benchmark [11], which consists of various continuous-control tasks and datasets. Specifically, we evaluate our method on three environments (HalfCheetah, Hopper, and Walker2d) in the Gym domain, using five types of datasets for each environment: random-v2, medium-v2, medium-replay-v2,

medium-expert-v2, and expert-v2. The medium-replay dataset contains experiences collected during the training up to a medium-level policy, while the random/medium/expert dataset is generated by a single random/medium/expert policy. The medium-expert dataset is a combination of the medium and expert datasets.

In all experiments, we train our algorithms for 3000 epochs, which corresponds to 1000 training steps per epoch and a total of 3 million steps. This training setup and common hyper-parameters follow the guidelines of PBRL and EDAC [3], [6]. The reported results are normalized using d4rl scores, which provide a measure of performance relative to expert and random scores. The normalization formula is as follows:

$$score_{normalized} = 100 * \frac{score - score_{random}}{score_{expert} - score_{random}}. \quad (12)$$

For evaluation, each algorithm is assessed based on 10 trajectories, with 1000 steps per trajectory, and the returns are averaged over 4 random seeds.

For specific hyper-parameters, we employ an adaptive learning rate α based on (SAC) algorithm [17]. The value of α is determined dynamically during training. The number of members in the ensemble, denoted as K, is selected through a random search within the range of 2 to 20. As for β, in the case of expert datasets, we perform a random search within the range of [0, $1e3$], while for other datasets, the random search is conducted within the range of [0, 1]. We update the policy network every 2 policy evaluation updates to strike a balance between computational efficiency and stability.

VI. EXPERIMENTAL RESULT

A. RESULT ON BENCHMARK DATASETS

We compare our method with several state-of-the-art algorithms: (1) BCQ, [13] aiming to mitigate extrapolation error by constraining the action space of the trained policy to be similar to the behavior policy, learned from the dataset, (2) IQL [24], adopting an implicit Q-learning approach where the Q-function is learned based on a learned V-function without directly querying a Q-function with OOD actions. (3) BEAR [25], which enforces policy constraints using the Maximum Mean Discrepancy (MMD) distance, (4) UWAC [51], an improvement on BEAR that incorporates dropout uncertainty-weighted updates, (5) CQL [26], which learns conservative value functions by minimizing Q-values of OOD actions, (6) MOPO [54], which quantifies uncertainty through ensemble dynamics in a model-based setting, (7) TD3-BC [12], which incorporates adaptive behavior cloning constraints to regularize the policy during training, (8) EDAC [3], which utilizes a diversified Q-ensemble to enforce conservatism, (9) PBRL [6], which applies uncertainty penalization and OOD sampling. It is worth noting that EDAC and PBRL are related to our method since all these methods employ a Q-ensemble for conservatism.

Table 1 reports the performance of the average normalized score with standard deviation. Based on the results, our framework achieves state-of-the-art performance, significantly surpasses other techniques. Especial, on average, our approach nearly doubles the scores achieved by methods such as BCQ and BEAR. Moreover, it outpaces the closest competitor (PBRL) by a substantial margin of +9.23.

A detailed examination of individual tasks reveals that our framework demonstrates superior performance compared to, or is at least on par with, the previous strongest approach. Notably, our framework exhibits significant improvements over the second-best result with a large margin when handling messy data, such as random or medium replay. This remarkable enhancement can be attributed to the effective uncertainty quantification capabilities of our framework, which is particularly advantageous in handling this type of data.

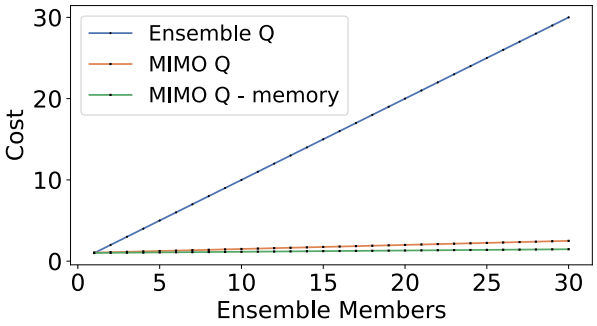

FIGURE 3. The forward time cost and memory of MIMO Q w.r.t the ensemble size. The result is relative to the single model cost.

TABLE 2. Computational costs on hopper-medium using Tesla V100.

	Runtime (s/epoch)	GPU memory (GB)
CQL	32.4	1.4
PBRL	102.96	1.8
OUR	17.8	0.97

B. TIME AND SPACE COMPLEXITY

The naive ensemble relies on employing separate models for each member, resulting in linear increases in forward cost and memory cost as the ensemble size grows. In contrast, our approach utilizes the shared weight across the ensemble. The only extra weights introduced are individual weights, which incur an extremely small overhead thanks to the utilization of rank-one vectors. As a result, the cost remains nearly unchanged, equivalent to that of a single network, even as the ensemble size increases as shown in Figure 3.

Mathematically, in our approach, the only additional memory required is for storing the sets of vectors $v_1, \ldots, v_k$ and $s_1, \ldots, s_k$, which have low memory overhead compared to weight matrices. Assuming the Q network consists of L fully connected layers with dimensions $m \times n$, the ensemble weight size is reduced from $LmnK$ in the naive ensemble to $Lmn + K(m + n)$. This reduction in memory usage is a significant advantage of our approach.

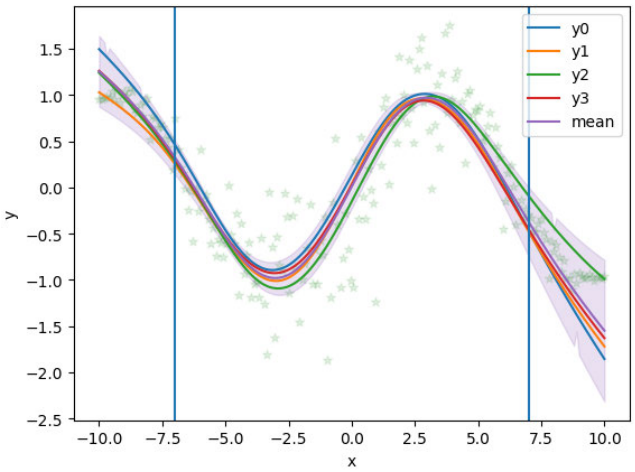

FIGURE 4. An illustration of the behavior of the Rank-One Q Network on the synthetic regression dataset. y_i denotes the prediction of each head.

To better understand the empirical efficiency of our framework, we present a runtime analysis in Table 2. The results demonstrate that our method not only achieves the best runtime performance but also maintains exceptional memory efficiency. Remarkably, it operates 1.82 times faster than CQL and 5.87 times faster than PBRL. In terms of memory usage, our framework is the most efficient, requiring only 0.97GB, compared to CQL's 1.4GB and PBRL's 1.8GB. Given that our method outperforms others while also being significantly more efficient in terms of runtime and memory usage compared to other strong baselines, we consider the improvements to be substantial.

VII. ABLATION STUDY

A. UNCERTAINTY QUANTIFICATION ABILITY

To gain a better understanding of the effectiveness of uncertainty quantification, we present a simple prediction task as an illustration. In this task, we train the Rank-One MIMO Q network using training data on the $\mathbb{R}^1$ plane, specifically within the range of $[-7.0, 7.0]$. The input variable x is generated from a Gaussian-distributed sinusoid function. For test data, we extend the data range to $[-10, 10]$, following the same underlying function. We visualize the testing data points along with the corresponding uncertainty quantification in figure 4. As depicted in the figure, the uncertainty quantification gradually increases as we move from the in-distribution data points to the OOD data points. This visual representation demonstrates the Rank-one MIMO Q's ability to perform regression with reliable uncertainty quantification, even on OOD data.

B. THE EFFECT OF K

We conducted experiments by varying the parameter K and recorded the corresponding outcomes in table 3. The experimental results confirm that adjusting the parameter K effectively controls pessimistic behavior, aligning with the theoretical expectations. Lower values of K tend to yield more optimistic results, while higher values of K lead to greater pessimism. The optimal outcome lies in finding an approximate value of K that strikes a balance between optimism and pessimism, resulting in the best overall performance.

TABLE 3. The effect of K on final performance and Q estimation on walker2d-medium-expert.

K	2	5	10	15	20
Avg Return	0.19	92.9	112.8	23	0.4
Avg $Q(s, \pi(s))$	3e11	410	373.46	-5e6	-2e12

TABLE 4. Component-wise analysis on the walker-2d medium-expert dataset.

Entropy	Likelihood	Avg Return	Avg $Q(s, \pi(s))$
✓	✓	**112.9**	373.4
✓		107.3	267.1
	✓	111.0	258.9
		109.6	453.2

C. COMPONENT-WISE ANALYSIS

We provide experimental results of the framework when different components are omitted. The baseline model we used is the basic framework, which does not incorporate the entropy bonus or the in-distribution likelihood maximization. Table 4 presents the ablation results, indicating the impact of these additional components. It is evident that our framework achieves the best performance when leveraging both the entropy and in-distribution likelihood maximization.

Interestingly, both the entropy and likelihood terms introduce a certain level of pessimism, as evidenced by the corresponding average Q values of 267.1 and 258.9, respectively. On the other hand, not using either term yields a higher Q value. Nevertheless, the performance remains satisfactory even without the inclusion of these components. It is worth noting that in the case of expert datasets such as walker2d-expert, the inclusion of in-distribution likelihood maximization plays a crucial role. Without it, achieving good results becomes highly unstable and challenging. The in-distribution likelihood maximization component provides stability and enhances the performance of the model when working with expert datasets.

VIII. CONCLUSION

In conclusion, this paper introduces a novel framework for offline reinforcement learning that utilizes uncertainty quantification to effectively leverage reliable OOD data. The proposed framework for precisely learning a policy through maximizing the lower confidence bound of the Q-function, along with the Rank-One Multi-Input Multi-Output architecture, strikes a balance between computational cost and precision, providing good overall performance. Extensive experiments on the D4RL benchmark show that our proposed framework achieves state-of-the-art performance

while being computationally friendly. Our findings contribute to the advancement of offline RL research and pave the way for future research in leveraging uncertainty quantification in an efficient way for addressing challenges in offline RL.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2021-0-01381, Development of Causal AI through Video Understanding and Reinforcement Learning, and its applications to real environments), partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2022-0-00951, Development of Uncertainty-Aware Agents Learning by Asking Questions), and partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. RS-2019-II190079, Artificial Intelligence Graduate School Program, Korea University).

REFERENCES

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2312–2320.
[2] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 104–114.
[3] G. An, S. Moon, J.-H. Kim, and H. O. Song, "Uncertainty-based offline reinforcement learning with diversified Q-ensemble," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 7436–7447.
[4] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar, "Efficient exploration through Bayesian deep Q-networks," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–9.
[5] C. Bai, L. Wang, L. Han, J. Hao, A. Garg, P. Liu, and Z. Wang, "Principled exploration via optimistic bootstrapping and backward induction," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 577–587.
[6] C. Bai, L. Wang, Z. Yang, Z. Deng, A. Garg, P. Liu, and Z. Wang, "Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning," 2022, *arXiv:2202.11566*.
[7] J. Buckman, C. Gelada, and M. G. Bellemare, "The importance of pessimism in fixed-dataset policy optimization," 2020, *arXiv:2009.06799*.
[8] C.-A. Cheng, T. Xie, N. Jiang, and A. Agarwal, "Adversarially trained actor critic for offline reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 3852–3878.
[9] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 4754–4765.
[10] A.-M. Farahmand, "Action-gap phenomenon in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2011, pp. 172–180.
[11] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," 2020, *arXiv:2004.07219*.
[12] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 20132–20145.
[13] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 2052–2062.
[14] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Mach. Learn., (ICML)*, vol. 80, 2018, pp. 1582–1591.
[15] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
[16] K. Ghasemipour, S. S. Gu, and O. Nachum, "Why so pessimistic? Estimating uncertainties for offline RL through ensembles, and why their independence matters," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 18267–18281.
[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[18] M. Havasi, R. Jenatton, S. Fort, J. Zhe Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran, "Training independent subnetworks for robust prediction," 2020, *arXiv:2010.06610*.
[19] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12519–12530.
[20] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan, "Provably efficient reinforcement learning with linear function approximation," in *Proc. Conf. Learn. Theory*, 2020, pp. 2137–2143.
[21] Y. Jin, Z. Yang, and Z. Wang, "Is pessimism provably efficient for offline RL?" in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5084–5096.
[22] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "MOReL: Model-based offline reinforcement learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 21810–21823.
[23] I. Kostrikov, R. Fergus, J. Tompson, and O. Nachum, "Offline reinforcement learning with Fisher divergence critic regularization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5774–5783.
[24] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit Q-learning," 2021, *arXiv:2110.06169*.
[25] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy Q-learning via bootstrapping error reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11784–11794.
[26] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1179–1191.
[27] B. Lee, J. Lee, P. Vrancx, D. Kim, and K.-E. Kim, "Batch reinforcement learning with hyperparameter gradients," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5725–5735.
[28] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "SUNRISE: A simple unified framework for ensemble learning in deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 6131–6141.
[29] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why M heads are better than one: Training a diverse ensemble of deep networks," 2015, *arXiv:1511.06314*.
[30] P. Li, J. Cao, and Z. Wang, "Robust impulsive synchronization of coupled delayed neural networks with uncertainties," *Phys. A, Stat. Mech. Appl.*, vol. 373, pp. 261–272, Jan. 2007.
[31] S. Liu, T. Chen, Z. Atashgahi, X. Chen, G. Sokar, E. Mocanu, M. Pechenizkiy, Z. Wang, and D. C. Mocanu, "Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity," 2021, *arXiv:2106.14568*.
[32] X. Ma, Y. Yang, H. Hu, Q. Liu, J. Yang, C. Zhang, Q. Zhao, and B. Liang, "Offline reinforcement learning with value-based episodic memory," 2021, *arXiv:2110.09796*.
[33] B. Mavrin, H. Yao, L. Kong, K. Wu, and Y. Yu, "Distributional reinforcement learning for efficient exploration," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4424–4434.
[34] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[35] A. Nair, A. Gupta, M. Dalal, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," 2020, *arXiv:2006.09359*.
[36] N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause, "Information-directed exploration for deep reinforcement learning," 2018, *arXiv:1812.07544*.
[37] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4026–4034.
[38] N. Pawlowski, A. Brock, M. C. H. Lee, M. Rajchl, and B. Glocker, "Implicit weight uncertainty in neural networks," 2017, *arXiv:1711.01297*.
[39] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," 2019, *arXiv:1910.00177*.
[40] A. Ramé, R. Sun, and M. Cord, "MixMo: Mixing multiple inputs for multiple outputs via deep subnetworks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 803–813.
[41] S. Rezaeifar, R. Dadashi, N. Vieillard, L. Hussenot, O. Bachem, O. Pietquin, and M. Geist, "Offline reinforcement learning as anti-exploration," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 8106–8114.
[42] J. P. Royston, "Expected normal order statistics (exact and approximate)," *J. Roy. Stat. Soc. C, Appl. Statist.*, vol. 31, no. 2, pp. 161–165, 1982.
[43] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 8583–8592.

[44] J. Smit, C. T. Ponnambalam, M. T. J. Spaan, and F. A. Oliehoek, "PEBL: Pessimistic ensembles for offline deep reinforcement learning," in *Proc. Robust Reliable Autonomy Wild Workshop 30th Int. Joint Conf. Artif. Intell.*, 2021.
[45] R. Sun, A. Ramé, C. Masson, N. Thome, and M. Cord, "Towards efficient feature sharing in MIMO architectures," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 2696–2700.
[46] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018, *arXiv:1805.01954*.
[47] L. Tran, B. S. Veeling, K. Roth, J. Swiatkowski, J. V. Dillon, J. Snoek, S. Mandt, T. Salimans, S. Nowozin, and R. Jenatton, "Hydra: Preserving ensemble diversity for model distillation," 2020, *arXiv:2001.04694*.
[48] Y. Wen, D. Tran, and J. Ba, "BatchEnsemble: An alternative approach to efficient ensemble and lifelong learning," 2020, *arXiv:2002.06715*.
[49] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," 2018, *arXiv:1803.04386*.
[50] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," 2019, *arXiv:1911.11361*.
[51] Y. Wu, S. Zhai, N. Srivastava, J. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh, "Uncertainty weighted actor-critic for offline reinforcement learning," 2021, *arXiv:2105.08140*.
[52] T. Xie, C.-A. Cheng, N. Jiang, P. Mineiro, and A. Agarwal, "Bellman-consistent pessimism for offline reinforcement learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 6683–6694.
[53] R. Yang, C. Bai, X. Ma, Z. Wang, C. Zhang, and L. Han, "RORL: Robust offline reinforcement learning via conservative smoothing," 2022, *arXiv:2206.02829*.
[54] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "MOPO: Model-based offline policy optimization," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 14129–14142.
[55] W. Zhou, S. Bajracharya, and D. Held, "PLAS: Latent action space for offline reinforcement learning," in *Proc. Conf. Robot Learn.*, 2021, pp. 1719–1735.

THANH NGUYEN (Graduate Student Member, IEEE) received the B.Sc. degree in electronic and automation engineering from Ho Chi Minh City University of Science and Technology, in 2015. He is currently pursuing the M.Sc. and Ph.D. degrees with Korea Advanced Institute of Science and Technology. His research interests include machine learning, deep learning, and reinforcement learning.

TUNG M. LUU (Graduate Student Member, IEEE) received the B.Sc. degree in electronic and telecommunication engineering from Hanoi University of Science and Technology, in 2017, and the M.Sc. degree in electrical engineering from Korea Advanced Institute of Science and Technology, in 2020, where he is currently pursuing the Ph.D. degree. His research interests include machine learning, deep learning, and reinforcement learning.

TRI TON (Graduate Student Member, IEEE) received the B.Sc. degree in automation and control engineering from Ho Chi Minh University of Technology, in 2021. He is currently pursuing the M.Sc. and Ph.D. degrees with Korea Advanced Institute of Science and Technology. His research interests include computer vision, with a focus on 3D understanding.

SUNGWOONG KIM (Member, IEEE) received the B.S. and Ph.D. degrees from KAIST, in 2004 and 2011, respectively. He is currently an Associate Professor with the Department of Artificial Intelligence, Korea University. He directs the AGI Laboratory, where the research focuses on realizing artificial general intelligence to make an AI agent perform task generalization and self-learning. When he was a graduate student, his research area was machine learning, especially applied to speech and image processing, under the supervision of Prof. Chang D. Yoo. In the middle of the Ph.D. degree, he performed research internships with the National ICT Australia under the supervision of Dr. Alex Smola and Microsoft Research Cambridge under the supervision of Dr. Pushmeet Kohli and Dr. Sebastian Nowozin. After receiving the Ph.D. degree, he was with KAIST, as a Postdoctoral Researcher, and then worked as a Staff Engineer with Qualcomm Research Korea. In 2017, he joined Kakao Brain and worked as a Research Scientist conducting several research projects mostly related to artificial general intelligence. Then, in March 2023, he joined Korea University.

CHANG D. YOO (Senior Member, IEEE) received the B.S. degree in engineering and applied science from California Institute of Technology, the M.S. degree in electrical engineering from Cornell University, and the Ph.D. degree in electrical engineering from Massachusetts Institute of Technology. He was the Dean of the Office of Special Projects and the Office of International Relations. From January 1997 to March 1999, he was a Senior Researcher with Korea Telecom (KT). Since 1999, he has been a Faculty Member with Korea Advanced Institute of Science and Technology (KAIST), where he is currently a Full Professor with tenure in the School of Electrical Engineering and an Adjunct Professor with the Department of Computer Science.