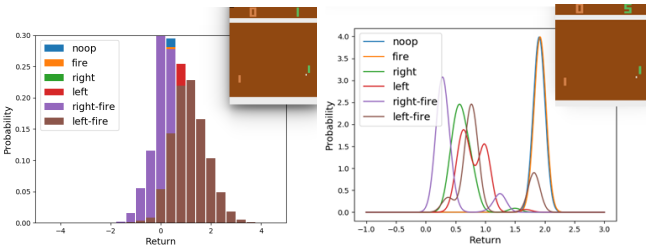
Distributional Deep Reinforcement Learning with a Mixture of Gaussians

Yunho Choi, Kyungjae Lee, and Songhwai Oh

***Abstract*— In this paper, we propose a novel distributional reinforcement learning (RL) method which models the distribution of the sum of rewards using a mixture density network. Recently, it has been shown that modeling the randomness of the return distribution leads to better performance in Atari games and control tasks. Despite the success of the prior work, it has limitations which come from the use of a discrete distribution. First, it needs a projection step and softmax parametrization for the distribution, since it minimizes the KL divergence loss. Secondly, its performance depends on discretization hyperparameters such as the number of atoms and bounds of the support which require domain knowledge. We mitigate these problems with the proposed parameterization, a mixture of Gaussians. Furthermore, we propose a new distance metric called the Jensen-Tsallis distance, which allows the computation of the distance between two mixtures of Gaussians in a closed form. We have conducted various experiments to validate the proposed method, including Atari games and autonomous vehicle driving.**



(a) *Baseline algorithm: C51* (b) *Proposed algorithm: MoG-DQN*

Fig. 1: Learned value distributions for each of the actions and game snapshots during an episode of PONG, which is one of Atari 2600 games. (a) Categorical value distributions trained with the C51 algorithm, which is our baseline. (b) Mixture of Gaussians value distributions trained with the proposed algorithm.

I. INTRODUCTION

The environment we interact with often has a probabilistic nature, inevitably. In such environments, a decision which is expected to generally produce the best result may also contain a risk of a catastrophic failure at the same time. Then, there may be a question about what decision to make, this risk-taking high-return action or a safer low-return action. Traditional reinforcement learning (RL) methods will choose the former without taking the risk into account.

In the setting of RL, an agent interacts with the stochastic environment, while receiving reward signals which feedbacks the agent's action. What the agent aims to learn in traditional RL is how to maximize the expected return, i.e., the expected sum of rewards. In value-based RL methods, such as SARSA [1] and Q-Learning [2], an agent estimates the state-action value which is the expected return when following the optimal policy given an initial state and action. Then, the agent chooses the action that maximizes the state-action value, since the decision theory implies that knowing the expected utility, or values for possible actions guarantees to act optimally.

On the other hand, with recent advances in the study of RL, it has been shown that learning a distribution rather than the expectation of the return is crucial and it leads to better performance in RL [3]–[5]. Rather than sticking to an unrealisable average of the return, an agent of the distributional perspective models the intrinsic randomness of the environment by learning the entire distribution of the return. Even in a deterministic environment, stochasticity can arise due to the state aliasing and the approximations in the state representation. Therefore, the distribution over returns, or the value distribution is a more stable learning target. The distribution also implies a multiple set of auxiliary predictions as investigated in [6], [7], which account for the high performance of the distributional approach. Knowing the distribution of the return also enables the risk-sensitive policy [5] that can encourage exploration.

Recently, algorithms that combine the distributional perspective on RL and the function approximaton method with neural networks have shown the state-of-the-art performance in Atari games [5], [8], [9] and robotic control problems [3]. These algorithms mainly build upon two distributional RL methods, categorical DQN (C51) [3] and quantile regrssion DQN (QR-DQN) [4]. C51 parametrizes the return distribution as a categorical distribution over finite atoms. It minimizes the cross-entropy loss between the projected Bellman update and the current estimate. On the other hand, QR-DQN uses a mixture of Diracs whose locations are estimated using quantile regression. However, in order for these algorithms to reduce the approximation error and get a decent performance, domain-specific knowledge is required when configuring the hyperparameters such as the number of atoms and the bounds of the support. Also in the case of C51, since the Bellman update of the categorical distribution distorts its original support, an additional projection step for

Y. Choi, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea. (e-mail: {yunho.choi, kyungjae.lee}@rllab.snu.ac.kr, songhwai@snu.ac.kr). This work was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017R1A2B2006136), and by SNU-Samsung Smart Campus Research Center (No. 0115-20160031).

the Bellman update is needed to avoid the disjoint support issue of the cross-entropy loss.

In this paper, to mitigate the problems mentioned above, we propose to use a mixture of Gaussians to parametrize the value distribution. With the mixture of Gaussians parametrization, we can effectively approximate the return distribution without prior domain knowledge with a relatively small number of parameters. This parametrization has a weakness in that the cross-entropy between two mixtures of Gaussians is analytically intractable. We solve this issue by proposing a novel distance metric called the Jensen-Tsallis distance that can measure the distance between two mixtures of Gaussians in a closed form. We also show that it has a desirable property called the unbiased sample gradient for minimizing the Jensen-Tsallis distance loss to the target distribution.

With the proposed parametrization and distance metric applied to deep Q-learning, our algorithm is derived. We apply it to Atari games and autonomous vehicle driving, and find that it exhibits well-behaved optimization and has comparable performance.

II. Related Work

Studies on the distributional perspective before [3] had subordinated its purpose on modeling parametric uncertainty [11], designing risk-sensitive algorithms [12], [13], or for theoretical analysis [14]. However, [3] has opened up another possibilities on the distributional perspective. The authors provide a contraction property of the distributional Bellman operator in a maximal form of the Wasserstein distance. The contraction property implies that a distributional version of the Bellman equation is derived which has a unique fixed point. A practical algorithm they have proposed is called categorical deep Q-learning, or C51, which outperforms the previous deep RL algorithms [15]–[18] and needs less episodes to achieve the best performance. The algorithm models the distribution of the return as a categorical distribution whose support is the set of 51 atoms as seen in Figure 1. To fit the parametrized value distribution, the algorithm minimizes the cross-entropy term of the KL divergence between the target value distribution and the current estimation of the value distribution. Although the metric they justify through the theoretical study on the distributional Bellman operator is the Wasserstein distance, the cross-entropy loss is used instead. This is due to the biased gradients of the Wasserstein distance [10].

After this work, several studies have been done to narrow the gap between the theory and practical algorithm in C51. The convergence of the categorical algorithms considering the projection operator is proved in [19] in terms of the Cramér distance. In [4], the authors modeled the return distribution using the qunatile regression method instead of using the categorical distibution, which essentially forms a mixture of N Diracs with fixed uniform weights and variable locations. They exploited quantile regression loss to fit the mixture of Diracs distribution. To reduce the approximation error of the mixture of Diracs, large N is required which increases the number of the output parameters to learn. Subsequent work [5] extends [4] by learning the full quantile function, or the inverse cumulative distribution function. Latest work [8], [9] combines orthogonal advances such as dueling networks [17], multi-step learning [20], and noisy nets [21]. Also, D4PG algorithm [22] has adapted the categorical algorithm into actor-critic methods [23] to solve continuous control problems.

III. Background

A. Markov Decision Processes and Reinforcement Learning

We consider a problem of a Markov decision process (MDP) defined by a tuple $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$ with the state space $\mathcal{X}$, the finite action space $\mathcal{A}$, the reward function R, and $P(x'|x,a)$, the probability of transition from the state x to the state x' after taking the action a. A policy $\pi(\cdot|x)$ is a conditional distribution over actions at a given state $x \in \mathcal{X}$. We consider a γ-discounted infinite-horizon problem, with the discount factor $\gamma \in (0,1)$. The action value of a state-action pair (x,a) is defined as

$$Q^\pi(x,a) := \mathbb{E}\bigg[\sum_{t=0}^{\infty} \gamma^t R(x_t,a_t)|x_0 = x, a_0 = a, \pi\bigg],$$

where x_t is a trajectory generated by following π and P starting from the initial state, i.e., $a_t \sim \pi(\cdot|x_t), x_{t+1} \sim P(\cdot|x_t,a_t)$. The state-action value, or action value can be characterized by the following Bellman equation and Bellman operator $\mathcal{T}^\pi$ [24],

$$Q^\pi(x,a) = \mathcal{T}^\pi Q(x,a) = \mathbb{E}\big[R(x,a)\big] + \gamma \mathbb{E}_{P,\pi}\big[Q(x',a')\big].$$

The objective of RL is to find an optimal policy π^* which maximizes $Q^\pi(x,a)$. The optimal action value is given by $Q^*(x,a) = \max_\pi Q^\pi(x,a)$.

So as to find an optimal policy, we usually find a unique fixed point of the following Bellman optimality operator [24], which is a contraction mapping.

$$\mathcal{T}Q(x,a) = \mathbb{E}\big[R(x,a)\big] + \gamma \mathbb{E}_P \max_{a'} Q(x',a').$$

Q-learning [2] iteratively estimates a parametrized function Q_θ by minimizing the squared temporal difference (TD) error,

$$\delta_t^2 = \big[(r_t + \gamma \max_{a'} Q_\theta(x_{t+1},a')) - Q_\theta(x_t,a_t)\big]^2, \qquad (1)$$

where tuples (x_t, a_t, r_t, x_{t+1}) are sampled while following an ϵ-greedy policy. The ϵ-greedy policy on Q_θ chooses a random action with probability ϵ and otherwise follows the greedy policy, $\arg\max_a Q_\theta(x,a)$. Deep-Q Network (DQN) [15] parametrizes Q_θ by a convolutional neural network.

B. Distributional Reinforcement Learning

In distributional RL, taking away the expectations inside the Bellman equation, we consider the randomness of the reward function R. For a fixed policy π, the return $Z^\pi = \sum_{t=0}^{\infty} \gamma^t R_t$ is also a random variable describing the sum of discounted rewards along a trajectory while following π. From here, we will refer to the value distribution by its

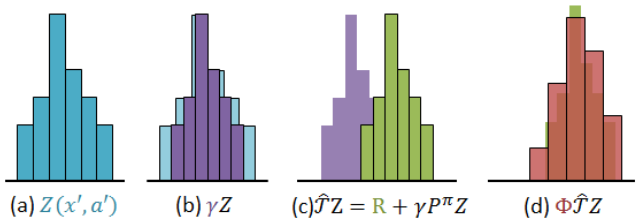


Fig. 2: The process of computing target distribution from sampled transition with a deterministic reward function: (a) An action-value distribution sampled with the next state and action, x' and a', (b) The discount factor γ shrinks the distribution towards 0, (c) The reward shifts the support, (d) The projection step projects it to the original support. It assigns atomic probabilities inversely proportional to distance from nearest support.

random variable. An action value can be represented using the random return,

$$Q^{\pi}(x,a) := \mathbb{E}[Z^{\pi}(x,a)] \\ = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) | x_0 = x, a_0 = a, \pi\Big],$$

where we say that the value function is the expectation of the value distribution.

As in RL, the value distribution can be computed through dynamic programming using a *distributional Bellman operator* defined as

$$\mathcal{T}^{\pi} Z(x,a) \stackrel{D}{=} R(x,a) + \gamma Z(X', A'),$$

where $X' \sim P(\cdot|x,a), A' \sim \pi(\cdot|X')$. The target value distribution is estimated from three sources of randomness, which are the reward R, transition P^{π} and the next-state value distribution $Z(X', A')$. In [3], the distributional Bellman operator $\mathcal{T}^{\pi}$ for a fixed policy π is shown to be a contraction mapping with respect to the maximal form of the Wasserstein metric.

C. Categorical Deep Q-learning

The contraction property of the distributional Bellman operator implies its convergence to the fixed point. Accordingly, it suggests a practical way to learn value distributions by minimizing the Wasserstein distance between a distribution Z and its Bellman update $\mathcal{T}^{\pi} Z$.

However, the result of [10] shows that the Wasserstein distance has a biased sample gradient so that we cannot use the Wasserstein metric as a loss in a stochastic gradient descent algorithm. Furthermore, in a practical context, since we must approximate the value distribution, an additional issue arises in the theoretical justification. That is, the theory of the contraction property does not take the projection operation into consideration. Nevertheless, the authors of [3] suggest a practical algorithm, categorical DQN or C51, which builds upon DQN [15] structure and approximates the value distribution as a categorical distribution. In [3], support of categorical distribution is noted by the set of N atoms $\{z_i = V_{min} + i\Delta z : 0 \leq i < N\}$, where $\Delta z := \frac{V_{max} - V_{min}}{N-1}$. To put it simply, the supporting set is a set of uniformly sliced bins between V_{min} and V_{max}. The atomic probabilities are given by a parametric model $\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^N$ which is a neural network in C51,

$$Z_{\theta}(x,a) = z_i \quad \text{with probability} \quad p_i(x,a) := \frac{e^{\theta_i(x,a)}}{\sum_j e^{\theta_j(x,a)}}.$$

In other words, the atomic probabilities are the softmax outputs of N parameters for each action at the last layer of the neural network. Figure 3-(a) describes the structure of a categorical DQN. It has $|\mathcal{A}| \times N$ network outputs while a vanilla DQN has $|\mathcal{A}|$ outputs. Their algorithm is called C51 since it has shown the best performance in experiments when using 51 bins.

What differentiates categorical DQN from DQN is the loss function. It replaces the squared TD error of DQN by the cross-entropy term of the KL divergence to compute the distance between two categorical distributions, a currently estimated distribution and a target distribution. The sample loss $\mathcal{L}_{x,a}(\theta)$ is as follows.

$$\mathcal{L}_{x,a}(\theta) = D_{KL}(\Phi(R(x,a) + \gamma Z_{\theta}(x', a')) \parallel Z_{\theta}(x,a)), \quad (2)$$

where $a' = \arg\max_a Q_{\theta}(x', a) = \arg\max_a \mathbb{E}[Z_{\theta}(x', a)]$.

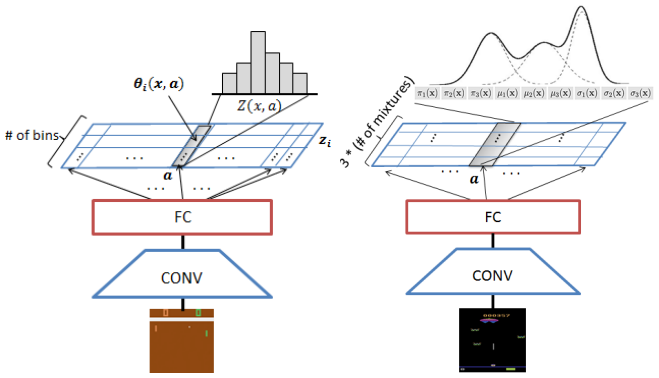
The process of computing the target distribution, or the sample Bellman update with the projection step is well explained in Figure 2. Since $Z_{\theta}(x,a)$ is a categorical distribution, the sample Bellman update $R(x,a) + \gamma Z_{\theta}(x', a')$ and $Z_{\theta}(x,a)$ almost always have disjoint supports as shown in Figure 2 (c) which disables us to directly minimize the KL divergence. Therefore the projection step Φ, which projects the sample Bellman update $R + \gamma Z_{\theta}(x', a')$ onto the prespecified support, is adopted in order to compute the sample loss readily. The projection step is done through a linear interpolation, mapping the target atomic probabilities to the adjacent bins.

IV. Distributional Reinforcement Learning with a Mixture of Gaussians

We now propose a new parameterization for distributional RL and a new distance metric for it, which are a mixture of Gaussians and the Jensen-Tsallis distance (JTD), respectively. With this new parameterization and distance metric, an additional projection step is no longer needed. In other words, the proposed method well approximates the value distribution without the cumbersome hyperparameter tuning including adjusting the support of value distribution. We refer to the resulting algorihtm with the newly proposed parametrization and distance metric as the mixture of Gaussians DQN (MoG-DQN).

A. Mixture Density Network for the Value Distribution

A parametric model with a mixture of Gaussians, also called a Gaussian mixture model (GMM), is well-known for its expressiveness. It is shown to be able to approximate any given density function to arbitrary accuracy by adjusting the number of mixtures [25]. In our case, the state-action value



(a) *C51* (b) *Proposed algorithm: MoG-DQN*

Fig. 3: Network structures. CONV and FC represent convolutional and fully connected layers, respectively. (a) The structure of the categorical DQN. Atomic probabilities are paremetrized by the final output, $\theta_i(x, a)$. (b) The structure of the MoG-DQN. It is a mixture density network (MDN) which outputs $|\mathcal{A}|$ sets of mixture parameters.

distribution can be characterized using a GMM parametric model $\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^{3K}$, where K is the number of mixtures, as follows.

$$Z_\theta(x, a) = z, \quad \text{with probability}$$
$$p(z|x, a) := \sum_{j=1}^{K} \pi_j(x, a)\mathcal{N}(z; \mu_j(x, a), \sigma_j(x, a)^2),$$

where $\pi_j(x, a)$, $\mu_j(x, a)$, and $\sigma_j(x, a)^2$ are the j-th mixture weight, mean, and variance function, respectively.

For natural modification of the Q network, we construct a mixture density network (MDN) [26], a neural network whose output consists of parameters of a GMM as seen in Figure 3-(b). To deal with the raw outputs, we use tricks used in [27]. Our network can effectively approximate the value distribution with the relatively small K. It has $3K \times |\mathcal{A}|$ output parameters which is a lot less compared to $51 \times |\mathcal{A}|$ parameters in C51 and $200 \times |\mathcal{A}|$ in QR-DQN [4].

In order to train the aforementioned network, a distance metric between the currently estimated mixture of Gaussians of the return distribution and the target distribution should be minimized from the sampled transitions, which can also be viewed as minimizing the temporal difference. Meanwhile, when the reward R is assumed to be a scalar r, the one-step sample backup distribution from a sampled transition (x, a, r, x') can be characterized as $r+\gamma Z_\theta(x', a')$, where $a' = \arg\max_a Q_\theta(x', a) = \arg\max_a \mathbb{E}[Z_\theta(x', a)]$. Let μ_i, σ_i, π_i be the mean, variance, and mixture weight for each Gaussian of mixture $Z_\theta(x', a')$. Then, the sample backup distribution is also a mixture of the same number of Gaussians where each mean and variance are shifted as, $\mu_i \to \gamma\mu_i + r$, $\sigma_i^2 \to (\gamma\sigma_i)^2$, with the mixture weights unchanged. This follows from shifting the support via an affine shift map $f : \mathbb{R} \to \mathbb{R}$, defined by $f(x) = r + \gamma x$. Since the support of the distribution is kept as the real line, we do not need an projection step.

Then, minimizing the temporal difference is equivalent with minimizing a distance metric between two mixtures of Gaussians, which are $Z_\theta(x, a)$ and $r+\gamma Z_\theta(x', a')$. However, neither the KL-divergence nor the Wasserstein distance between the two mixtures of Gaussians can be computed in a closed form and the estimation of it by a Monte Carlo method is computationally expensive [22]. Therefore, we propose a new distance metric for the mixtures of Gaussian in the following subsection IV-B.

B. Jensen-Tsallis Distance for the Distance Metric

The *Jensen-Tsallis divergence* is a generalization of the *Jensen-Shannon divergence (JSD)* which uses the *Tsallis entropy* instead of the *Shannon entropy* [28], [29]. The *Jensen-Tsallis divergence* between two probability distribution functions (PDFs) p_1, p_2 with the entropic index q and positive real constant k is defined as

$$JT_{q,k}(p_1, p_2) = S_{q,k}\Big(\frac{p_1 + p_2}{2}\Big) - \frac{S_{q,k}(p_1) + S_{q,k}(p_2)}{2},$$

where $S_{q,k}(p)$ is the *Tsallis entropy* defined as [29] $S_{q,k}(p) = \frac{k}{q-1}(1 - \int p(x)^q dx)$. Note that if $q \to 1$ and $k = 1$, $S_{1,1}$ is same as the Shannon entropy and $JT_{1,1}$ is same with JSD.

We find that with $q = 2$ and $k = 4$, $JT_{2,4}$ is resolved into a simple form as follows.

$$JT_{2,4}(p_1, p_2) = \int (p_1(x) - p_2(x))^2 dx$$

We refer to the square root of the $JT_{2,4}$ as the *Jensen-Tsallis distance* (JTD) since it satisfies the conditions of a metric, which is proved starting from the definition as below.

Definition 1 (*Jensen-Tsallis Distance*)**.** *Let X, Y be random variables defined on $\mathbb{R}$ and $f_X(r)$, $f_Y(r)$ be PDFs of X and Y respectively, which are assumed to be square integrable. We define the Jensen-Tsallis distance (JTD) as below.*

$$JTD(X, Y) = \Big(\int_R (f_X(r) - f_Y(r))^2 \, dr\Big)^{1/2},$$

Theorem 1. *JTD is a metric over distributions.*

Proof. Among the conditions of a metric (non-negativity, identity, symmetry, and triangle inequality), the only nontrivial property is the triangle inequality.

$$JTD(X, Y) + JTD(Y, Z) =$$
$$\Big(\int_R (f_X(r) - f_Y(r))^2 \, dr\Big)^{1/2} + \Big(\int_R (f_Y(r) - f_Z(r))^2 \, dr\Big)^{1/2}$$
$$\geq \Big(\int_R (f_X(r) - f_Z(r))^2 \, dr\Big)^{1/2} = JTD(X, Z),$$

where the inequality holds from the Minkowski's inequality. □

The simple formula of the JTD in Definition 1 makes it possible to calculate the distance between two mixtures of Gaussians in a closed form, while other distance metrics including the KL-divegence, Wasserstein distance, and *JSD* cannot. Making use of the fact that

$\int_R \mathcal{N}(x;\mu,\sigma^2)\mathcal{N}(x;\mu',\sigma'^2)dx = \mathcal{N}(\mu;\mu',\sigma^2+\sigma'^2)$, *JTD* between two mixtures of Gaussians Z_1, Z_2 can be easily computed as follows.

$$\begin{aligned} &JTD^2(Z_1,Z_2) \\ &= \int_R \big(\sum_i \pi_i \mathcal{N}(x;\mu_i,\sigma_i^2) - \sum_i \omega_i \mathcal{N}(x;m_i,s_i^2)\big)^2 dx \\ &= \sum_{i,j} \pi_i\pi_j \mathcal{N}(\mu_i;\mu_j,\sigma_i^2+\sigma_j^2) + \sum_{i,j} \omega_i\omega_j \mathcal{N}(m_i;m_j,s_i^2+s_j^2) \\ &\quad -2\sum_{i,j} \pi_i\omega_j \mathcal{N}(\mu_i;m_j,\sigma_i^2+s_j^2) \end{aligned} \tag{3}$$

In addition to this computational advantage when using the mixture of Gaussians parametrization, *JTD* also has nice properties that it is free from the disjoint support issue which arises when using the KL-divergence loss, and exhibits unbiased sample gradients unlike the Wasserstein distance. The latter is proven in the following theorem in a similar way with [10].

Theorem 2. *The Jensen-Tsallis distance has unbiased sample gradients. That is, given* $\mathbf{X}_m := X_1, ..., X_m$ *drawn from a distribution* P*, the empirical distribution* $\hat{P}_m := \frac{1}{m}\sum_{i=1}^m \delta_{X_i}$*, and a distribution* Q_θ *under the sufficient smooth condition,*

$$\mathop{\mathbb{E}}_{\mathbf{X}_m \sim P} \nabla_\theta JTD^2(\hat{P}_m, Q_\theta) = \nabla_\theta JTD^2(P, Q_\theta).$$

Proof. By definition,

$$\begin{aligned} \nabla_\theta JTD^2(P,Q_\theta) &= \nabla_\theta \int_{-\infty}^{\infty} \left(f_{Q_\theta}(x) - f_P(x)\right)^2 dx \\ &= \int_{-\infty}^{\infty} 2(f_{Q_\theta}(x) - f_P(x))\nabla_\theta f_{Q_\theta}(x)\, dx \\ &= \int_{-\infty}^{\infty} 2(f_{Q_\theta}(x) - \mathop{\mathbb{E}}_{\mathbf{X}_m\sim P} f_{\hat{P}_m}(x))\nabla_\theta f_{Q_\theta}(x)\, dx \\ &= \int_{-\infty}^{\infty} 2 \mathop{\mathbb{E}}_{\mathbf{X}_m\sim P}(f_{Q_\theta}(x) - f_{\hat{P}_m}(x))\nabla_\theta f_{Q_\theta}(x)\, dx \\ &\overset{(a)}{=} \mathop{\mathbb{E}}_{\mathbf{X}_m\sim P} \int_{-\infty}^{\infty} 2(f_{Q_\theta}(x) - f_{\hat{P}_m}(x))\nabla_\theta f_{Q_\theta}(x)\, dx \\ &= \mathop{\mathbb{E}}_{\mathbf{X}_m\sim P} \nabla_\theta JTD^2(\hat{P}_m, Q_\theta), \end{aligned}$$

where (a) follows from Fubini's theorem. □

If the above property is satisfied, the sampled estimation of the gradient is asymptotically same with the true gradient so that we can use the sampled gradient to make the objective function converge when using gradient-based method. Thus, we can minimize the JTD with the stochastic gradient descent (SGD) method to update the value distribution.

C. Mixture of Gaussians DQN

We now propose a concrete algorithm, MoG-DQN, that builds upon the DQN structure [15]. To adopt the mixture of Gaussians parametrization and the JTD metric, there comes two major differences from the original DQN. First, the network structure is modified to output the mixture parameters as shown in Figure 3 (b). K was set to five throughout the experiments. Second, the squared TD error loss given in Equation 1 is modified to the JTD loss to measure the distance between two mixtures of Gaussians. A procedure to compute a sample loss is shown in Algorithm 1. As in DQN, we construct the sample Bellman update with the action that maximizes the state-action value, which is the expected value of the value distribution or the mixture-weighted sum of Gaussian means in our case.

Algorithm 1 Sample Loss of MoG Q-Learning

Require: Number of Gaussian components K
input $x, a, r, x', \gamma \in (0,1)$
  # Compute distributional Bellman update
  $Q(x',a') := \Sigma_j \pi_j(x',a')\mu_j(x',a')$
  $a^* \leftarrow \arg\max_{a'} Q(x',a')$
  $\mathcal{T}Z(x,a) \leftarrow$
    $\Sigma_j \pi_j(x',a^*)\,\mathcal{N}\big(\gamma\mu_j(x',a^*) + r, \gamma^2\sigma_j(x',a^*)^2\big)$
  # Compute JTD loss (Equation 3)
output $JTD(Z(x,a), \mathcal{T}Z(x,a))$

While C51 requires an additional projection step onto the predefined bounded supports, MoG-DQN rather moves the Gaussian components and adjusts its variances to approximate any return distributions, which is possible through the use of *JTD* loss. Consequently, MoG-DQN does not need domain-specific hyperparameters such as bounds of the support and the number of bins. What MoG-DQN only requires as a hyperparameter is the number of Gaussian components K. It suffices for small number of components in experiments, consequently requring less output parameters to learn compared to C51 or QR-DQN.

V. Experiments

To test the performance of the proposed algorithm, we conducted our experiments on Atari 2600 games [30] and the autonomous vehicle driving simulator we built with a Gazebo simulator in Robot Operating System (ROS) [31]. Throughout the experiment, for C51 and MoG-DQN network, we built from the standard DQN architecure and techniques used in [15] for fair comparisons.

A. Evaluation on Atari Games

Playing Atari games with RL agents can be viewd as a simple vision-based control problem since it uses the game snapshots as the agent's state and decides discrete control value. We provide evaluation results on the three games in Atari 2600 in Figure 4. As shown in the figure, MoG-DQN exhibits desirable properties we pursue. First, MoG-DQN enhances the sample-complexity over C51, i.e., requires less episodes to accomplish the same performance. Also, its final performance reaches comparable score with the reported best score [5]. Furthermore, the empirical results support that MoG-DQN requires the less domain-specific information than C51. As in the case of BOXING, the performance of C51 tends to degenerate when the hyperparameters such as the bounds of the support, number of bins, and target network

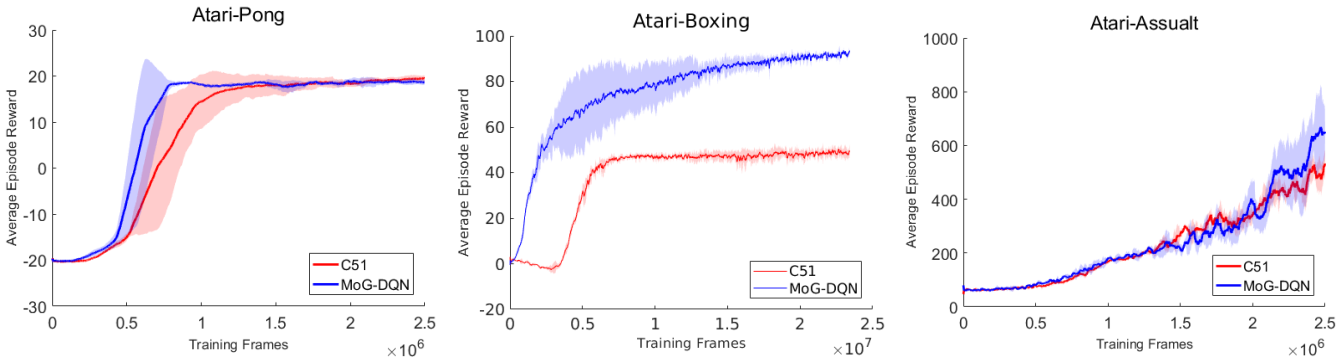
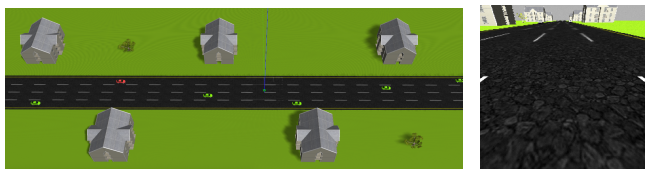


Fig. 4: Training curves for C51 and MoG-DQN on the three Atari games, PONG, BOXING, and ASSAULT. Moving average of the return over 100 episodes is plotted along the number of training frames. Curves are averages over three random seeds, and the shaded regions represents standard deviations.



(a) (b)

Fig. 5: The autonomous vehicle driving simulator used in the experiment. (a) Highway environment. (b) Front camera image of the car.

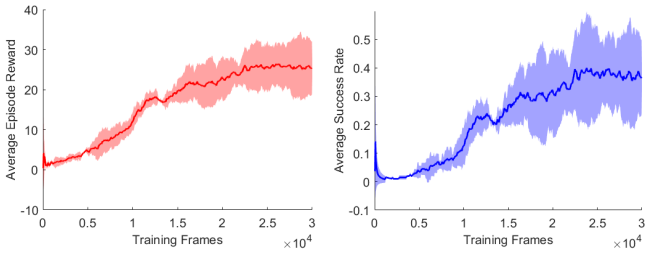


Fig. 6: Moving average of the return (left), and success rate (right) over 100 episodes in the autonomous driving task are plotted along the number of training frames.

update frequency are not tuned carefully. Meanwhile, MoG-DQN shows more robust behavior without changing hyperparameters such as the number of Gaussian components. From these observations, we can speculate that optimization under the JTD loss leads to the stable convergence in accord with Theorem 2 Thus, we can approximate the return distribution efficiently even with a small number of components.

B. Evaluation on the Autonomous Vehicle Driving Task

To test the proposed algorithm on a more realistic environment, we ran our algorithm on the autonomous vehicle driving simulator which we built. Figure 5 shows the environment, where we control the red car in the highway from its front camera images while eight green cars are moving along the lane with a constant speed. The red car is controlled with two speeds and three steering angles, which make six actions in total. The agent's state is defined as four consecutive front camera images of the red car which are downsampled to 84x84. We used the same network structure with the one for Atari games. Our goal is to drive the red car to the other end of the road. The stepwise reward is defined as $0.1 * d - r$, where d is the travel distance and r is the deviation from the center of the lane. Also the additional reward of 10 is given when the car reaches the other end and -10 is given when the car collides with the other cars or derails out of the road. Since the speeds of the red car is faster than the green cars and the steering is not fine enough to drive between the eight cars, it is quite a hard task.

The evaluation result is given in Figure 6. Although the agent was trained with only 30000 training frames, it managed to succesfully learn to drive in the highway, reaching the success rate of 0.4 on average. Meanwhile, the DQN agent [15] failed to learn the task with the same setting.

VI. CONCLUSION

In this paper, we have proposed a novel distributional RL method that models the value distribution as a mixture of Gaussians. The expressiveness of the mixture of Gaussians make the proposed method well approximate the continous distributions with less parameters compared to the other algorithms. In addition, we have proposed a new distance metric called the JTD that computes the distance between two mixtures of Gaussians in a closed form. We have proven a useful fact that the JTD has unbiased sample gradients. This makes the JTD loss applicable with sample gradient based methods, which results in a new practical algorithm, MoG-DQN. From the experiments on Atari games, we have shown that MoG-DQN has significant performance over the compared algorithm, C51. We have also verified that our algorithm works in a more realistic domain, the autonomous driving task. MoG-DQN only requires the number of Gaussian components as a hyperparameter, which makes the algorithm widely applicable with less domain-knowledge.

REFERENCES

[1] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994, vol. 37.

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

[3] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. of the 34th International Conference on Machine Learning (ICML)*, 2017.

[4] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2018.

[5] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proc. of the International Conference on Machine Learning (ICML)*, 2018.

[6] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[7] M. Jaderberg, V. Mnih, W. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

[8] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2018.

[9] A. Gruslys, M. G. Azar, M. G. Bellemare, and R. Munos, "The reactor: a fast and sample-efficient actor-critic agent for reinforcement learning," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[10] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[11] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *Proc. of the National Conference on Artificial Intelligence*, 1998.

[12] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka, "Parametric return density estimation for reinforcement learning," in *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[13] ——, "Nonparametric return distribution approximation for reinforcement learning," in *Proc. of the 27th International Conference on Machine Learning (ICML)*, 2010.

[14] M. G. Azar, R. Munos, and H. J. Kappen, "On the sample complexity of reinforcement learning with a generative model," in *Proc. of the 29th International Coference on Machine Learning (ICML)*, 2012.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[16] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *Proc. of the AAAI Conference on Artificial Intelligence*, 2016.

[17] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. of the International Conference on Machine Learning (ICML)*, 2016.

[18] "Prioritized experience replay," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2016.

[19] M. Rowland, M. Bellemare, W. Dabney, R. Munos, and Y. W. Teh, "An analysis of categorical distributional reinforcement learning," in *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[20] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[21] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[22] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. TB, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[23] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2016.

[24] R. Bellman, "Dynamic programming," *Princeton University Press*, 1957.

[25] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. Marcel Dekker, 1988, vol. 84.

[26] C. M. Bishop, "Mixture density networks," Citeseer, Tech. Rep., 1994.

[27] S. Choi, K. Lee, S. Lim, and S. Oh, "Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[28] C. Tsallis, "Possible generalization of boltzmann-gibbs statistics," *Journal of statistical physics*, 1988.

[29] A. Martins, P. Aguiar, and M. Figueiredo, "Nonextensive generalizations of the jensen-shannon divergence," *arXiv preprint arXiv:0804.1653*, 2008.

[30] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.