Backward Imitation and Forward Reinforcement Learning via Bi-directional Model Rollouts

Yuxin Pan\textsuperscript{1} and Fangzhen Lin\textsuperscript{2}

Abstract—Traditional model-based reinforcement learning (RL) methods generate forward rollout traces using the learnt dynamics model to reduce interactions with the real environment. The recent model-based RL method considers the way to learn a backward model that specifies the conditional probability of the previous state given the previous action and the current state to additionally generate backward rollout trajectories. However, in this type of model-based method, the samples derived from backward rollouts and those from forward rollouts are simply aggregated together to optimize the policy via the model-free RL algorithm, which may decrease both the sample efficiency and the convergence rate. This is because such an approach ignores the fact that backward rollout traces are often generated starting from some high-value states and are certainly more instructive for the agent to improve the behavior. In this paper, we propose the backward imitation and forward reinforcement learning (BIFRL) framework where the agent treats backward rollout traces as expert demonstrations for the imitation of excellent behaviors, and then collects forward rollout transitions for policy reinforcement. Consequently, BIFRL empowers the agent to both the sample efficiency and the convergence rate. This is because it is ignored that the traces derived from backward rollouts typically start from some high-value states, and then explore from those reached states for the future novel actions. Moreover, a value-regularized generative adversarial network is introduced to augment the valuable states which are infrequently received by the agent. Theoretically, we provide the condition where BIFRL is superior to the baseline methods. Experimentally, we demonstrate that BIFRL acquires the better sample efficiency and produces the competitive asymptotic performance on various MuJoCo locomotion tasks compared against state-of-the-art model-based methods.

I. INTRODUCTION

In recent years, with the tremendous advances of deep learning, the model-free reinforcement learning (RL) that utilizes the neural network to represent the policy or the value function has made remarkable progress \cite{1}, \cite{2}, \cite{3}, and can potentially automate various applications, such as autonomous driving and robotics manipulation. However, the model-free RL algorithm requires extensive interactions with the environment to obtain large amounts of data for policy optimization, which thus restricts it to mostly controllable simulated environments. In contrast, model-based methods have been found to be able to learn a near-optimal policy without enormous number of interactions with the environment \cite{4}. This is because once a reliable model that is capable of precisely constructing the running mechanism of the environment is learnt, it can be used as a virtual environment to further drive the learning process, consequently improving the sample efficiency. But the downside of the model-based RL method is that it is often inferior to the model-free counterpart in terms of asymptotic performance.

Traditionally, in model-based RL methods, the agent interacts with the learnt transition function with uncertainty from the current state by executing a policy to generate the imaginary traces for policy optimization (called rollout). More recently, Goyal et al. \cite{5} proposed an impressive backward model that estimates the conditional probability of the previous state given the previous action and the current state to recall backward rollout traces, and showed that in many domains, especially those with sparse high-value states, this approach can acquire the improvement in terms of sample efficiency. Lai et al. \cite{6} then proposed a so-called bi-directional model-based policy optimization (BMPO) algorithm that uses both the traditional forward model (transition function) and the backward model to generate bi-directional rollout samples that reach into both the past and the future. However, such a model-based method simply aggregates the bi-directional model-generated data together to reinforce the policy via the model-free RL algorithm, which may cause the low sample efficiency and the slow convergence rate. This is because it is ignored that the traces derived from backward rollouts typically start from some high-value states and consequently can supervise the learning of agent like the expert demonstrations. To put it another way, the agent can efficiently learn to reach the high-value states through directly imitating the traces from backward rollouts, and then explore from those reached states for the future novel experiences to reinforce the policy. We thus believe that the bi-directional model rollout samples should be treated differently for the efficient policy learning.

To this end, we propose the backward imitation and forward reinforcement learning (BIFRL) framework where backward rollout traces generated from the presumed valuable states are used as expert demonstrations for the agent imitation of excellent behaviors, and then the interactions with the forward model are conducted to produce forward rollout traces for policy reinforcement. Specifically, BIFRL requires to generate high-value states as the starting points for the bi-directional model rollouts. From these states, backward rollouts are performed to generate the traces back to some initial states. These traces are then used like those from expert demonstrations to train the agent using imitation learning. Afterward, BIFRL conducts forward rollouts from these states and the resulting samples are utilized to further
reinforce the policy using the RL algorithm. As a result, BIFRL empowers the agent to both reach to and explore from the valuable states to accelerate its policy learning and reduce the extensive interactions. Thus it has the potential to limit the damage to the robot and its environment if being applied in the real word, making it more applicable for real-robot learning. Additionally, the valuable states are indeed rare, and should be leveraged effectively and efficiently. Thus, a value-regularized generative adversarial network (GAN) is developed to discover more those particular useful states as state augmentation without the need of state labelling. Furthermore, we theoretically provide the condition where BIFRL shows the superiority against the baselines. Experimentally, we demonstrate that compared with previous state-of-the-art model-based methods, BIFRL achieves higher sample efficiency and produces competitive asymptotic performance on various MuJoCo locomotion benchmark tasks [7].

II. RELATED WORK

Compared with model-free RL, model-based RL is often more sample efficient given that it can generate simulated samples using the learned dynamics model. However, this often comes in the expense of the asymptotic performance as the learnt model may not be accurate. Previous works have tried to mitigate the compounding errors coming from the multi-step model rollouts by tuning the ratio of real to model-generated samples [8], [9], [10]. Nguyen et al. [11] and Xiao et al. [12] considered different ways to estimate the cumulative model error by which the adaptive model-based planning horizon can be determined. MBPO [13] generates truncated model rollouts branched from real states to cripple the influence of model error and provides the condition for the return improvement in the true dynamics. At each time step \( t \), the agent in the state \( s_t \in \mathcal{S} \) executes the action \( a_t \in \mathcal{A} \), receives the reward \( r_t = r(s_t, a_t) \), and moves to the next state \( s_{t+1} \) according to the unknown transition function \( T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S} \). \( \rho_0 \) denotes the initial state distribution. The goal of reinforcement learning is to find an optimal policy \( \pi^* \) that maximizes the expected sum of discounted rewards (expected return) as:

\[
\pi^* = \arg \max_\pi \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],
\]

where \( \gamma \in [0, 1] \) is the discounted factor.

The transition function is assumed to be unknown in the model-free RL. In contrast, the Dyna-style algorithm [18] that is one branch of the model-based paradigms learns a dynamics model (forward model) \( p_\theta(s_{t+1}|s_t, a_t) \) parameterized by \( \theta \) as the true dynamics from agent interactions with the environment by conducting the policy \( \pi_\phi(a_t|s_t) \) parameterized by \( \phi \). In addition, in BIFRL, a backward model \( \tilde{p}_\psi(s_{t-1}|s_t, a_{t-1}) \) and a backward policy \( \tilde{\pi}_\phi'(a_{t-1}|s_t) \) are constructed for backward rollouts, which are parameterized by \( \theta' \) and \( \phi' \) respectively.

IV. METHOD

In this section, we detail BIFRL algorithm that leverages the simulated samples from backward model rollouts in a different way with those generated from interactions with the forward model by performing the policy. Consequently, the agent is able to imitate excellent behaviors from backward rollout traces to reach the high-value states and reinforce its policy through explorations starting from those high-value states. Besides, BIFRL is built on the Model-based Policy Optimization (MBPO) [13] algorithm that is one of state-of-the-art Dyna-style model-based methods and investigates forward truncated model rollouts.
A. BIFRL Framework

Figure 1 illustrates the overview of BIFRL algorithm that consists of five indispensible and interdependent stages. The algorithm actually iterates between those stages to progressively improve the policy. In the real transition collection stage, the agent collects the data for the replay buffer $D_{env}$ through interactions with the environment. In the model learning stage, the models including forward model, backward model, backward policy and value-regularized GAN are trained using the data in $D_{env}$. In the high-value state generation stage, the states in $D_{env}$ and those sampled from the value-regularized GAN are aggregated together, and then are prioritized and sampled for the backward imitation high-value state buffer $D^b_{hs}$ and the forward reinforcement high-value state buffer $D^f_{hs}$. In the bi-directional model rollouts stage, backward rollouts are performed starting from the states $s^b_{ts}$ randomly chosen from $D^b_{hs}$ to collect the samples for the backward rollout data buffer $D_b$. Meanwhile, forward rollouts start from the states $s^f_{ts}$ chosen from $D^f_{hs}$ to gather the rollout transitions which are stored in the forward rollout data buffer $D_f$. In the policy learning stage, the state-action pairs in $D_b$ are regarded as expert demonstrations for the agent to imitate excellent behaviors, and the transitions from $D_f$ are utilized to reinforce the policy via RL algorithm. Thus, the algorithm runs those five processes in cycles leading to the coupling between them. The BIFRL algorithm is presented in Algorithm 1.

Bi-directional Dynamics Models. The architecture choices of bi-directional dynamics models are crucial for the policy optimization on the model-generated data, since the compounding model error can severely degenerate the multi-step rollouts. Therefore, in BIFRL the ensemble of bootstrapped probabilistic dynamics model [19] is adopted for both the forward and backward models to capture two kinds of uncertainty. The one called aleatoric uncertainty arising from the inherent stochasticity of a system can be captured by each probabilistic model, and the other called epistemic uncertainty caused by the lack of sufficient training data can be remedied by the ensemble of models.

To be specific, each of the forward model ensembles $\{p_\theta^i\}_{i=1}^D$ is represented by a neural network, and we use $\theta$ to denote the parameters of forward model ensembles. Each individual model takes the state $s_t$ and the action $a_t$ as inputs, and outputs the parameters (i.e. means $\mu$ and diagonal covariances $\Sigma$) of the Gaussian distribution of the state variation $\Delta s_t = s_{t+1} - s_t$, denoted as:

$$p_\theta^i(\Delta s_t | s_t, a_t) = \mathcal{N}(\mu_\theta^i(s_t, a_t), \Sigma_\theta^i(s_t, a_t)).$$  \hspace{1cm} (2)

Likewise, the same reparameterization technique is put to use for the trustworthy backward model, where each individual neural network with $\theta$ of the model ensembles yields the parameters of a Gaussian distribution written as:

$$\tilde{p}_\theta^b(\Delta s_{t-1} | s_t, a_{t-1}) = \mathcal{N}(\mu_\theta^b(s_t, a_{t-1}), \Sigma_\theta^b(s_t, a_{t-1})).$$  \hspace{1cm} (3)

Note that these models can estimate the reward distribution as well, which is omitted for brevity.

Backward Policy. Relying on the learnt backward model, the previous state can be estimated given the current state and the previous action, which arouses a confusion about how to choose the previous actions for backward model rollouts. Thus, analogous to the setting of forward rollouts where the actions are naturally taken by the policy $\pi_\phi(a_t | s_t)$, we need the backward policy to predict the action $a_{t-1}$ given the state $s_t$. Since the multiple backward traces are required to be simultaneously generated from one starting state, the powerful probabilistic neural network with parameter $\phi'$ is employed to output the parameters of the Gaussian distribution of the action $a_{t-1}$:

$$\tilde{\pi}_\phi'(a_{t-1} | s_t) = \mathcal{N}(\mu_\phi'(s_t), \Sigma_\phi'(s_t))$$  \hspace{1cm} (4)

Value-regularized GAN. In order to increase the quantity of high-value states, we propose a generative model called value-regularized GAN, which aims to generate the states that can be criticized as the high-value ones under the current policy by the critic and cannot move out of the real-state distribution. Unlike prior works [17], [5] that rely on the state labelling to train the Goal GAN, the value-regularized GAN is strengthened by the value regularization by which the generator $G(\cdot)$ treats maximum state value computed by
the critic as another optimization objective besides the one defined by the discriminator $D(\cdot)$, resulting in removing the need for annotation. Additionally, due to the initially unstable learning of Q-value network $Q^\pi(\cdot)$, we consider the way to enable the generator to produce the high-reward states under the latest policy, which owes to the learnt dynamics model $p_0(\cdot)$ that can make reliable one-step prediction of the reward $r$. The value regularization can be described as:

$$
L_v = \alpha Q^\pi(G(z), \pi_\phi(G(z))) + (1 - \alpha)r
$$

(5)

$$
r \sim \mathcal{N}(\mu_\theta(G(z), \pi_\phi(G(z))), \Sigma_\phi(G(z), \pi_\phi(G(z))))
$$

(6)

where $\alpha \in (0, 1]$ is the weighting factor gradually increasing with the increasing number of training steps. The architecture of the value-regularized GAN is shown in Figure 2.

**State Sampling Strategies.** Since the state diversity benefits the exploration, we adopt to collect the states stored in $D_{env}$ and those generated from value-regularized GAN together, instead of directly using the GAN-generated data, so as to sample the high-value states for model rollouts. We design different applicable state sampling strategies for the diverse model rollouts, because of the different training schemes on those rollout data. In detail, for the backward rollouts, the state value is chosen as the criterion to measure the importance of each state. Generally, the value of each state can be accurately computed by summing up the rewards from the current state $s_t$ to the termination state $s_T$ along the trace $(\sum_{t'=t}^{T} r(s_{t'}, \pi_\theta(s_{t'})))$, which however is not computationally efficient. Thus, the learnt value function $V^\pi(\cdot)$ is used to estimate the value, which is accessible and effective. Thus, BIFRL greedily picks the top $K$ percent of the prioritized states and stores them in $D_{hs}^f$.

Due to the greedy prioritization that focuses on a small subset of the states, the temporal difference (TD) errors on those low-priority states shrink slowly when using function approximation, which makes the system prone to over-fitting. Therefore, for the forward rollouts, the stochastic sampling method is introduced to ensure the probability of being sampled is monotonic in a state’s priority and guarantee a non-zero probability for the low-value state. To further alleviate the lack of diversity, we treat the absolute TD error $\delta(\cdot)$ of the transition that is from the state to be prioritized to the next state under the current policy as another criterion. Concretely, we use the Boltzmann distribution to model the probability of sampling state $s^j$ defined as:

$$
p(s^j) = \frac{e^{\beta V^\pi(s^j) + (1 - \beta)\delta(s^j)}}{\sum_{k=1}^{n} e^{\beta V^\pi(s^k) + (1 - \beta)\delta(s^k)}},
$$

(7)

$$
\delta(s^j) = |V^\pi(s^j) - r^j + \gamma V^\pi(\Delta s^j + s^j)|,
$$

(8)

$$\Delta s^j, r^j \sim \mathcal{N}(\mu_\theta(s^j, \pi_\phi(s^j)), \Sigma_\phi(s^j, \pi_\phi(s^j)))
$$

(9)

where $j$ is the index of state in the state collection with size $n$, and $\beta \in (0, 1)$ is to balance those two criterions. The states sampled in this way are stored in $D_{hs}^f$.

![Fig. 2: The architecture of value-regularized GAN.](image)

B. Training

In this part, we dedicate to introducing how to train the models in the model learning stage, and how to perform the policy learning to increasingly improve the agent’s behavior.

**Bi-directional Dynamics Models.** In BIFRL, we choose the ensemble of bootstrapped probabilistic networks as the bi-directional dynamics models. In the training phase, each individual network of the ensembles is trained with different initializations and samples from the real transitions via maximizing likelihood. The corresponding loss function of the forward model can be written as:

$$
L_f(\theta) = - \sum_{t=1}^{N} \log p_\theta(\Delta s_t | s_t, a_t)
$$

(10)

$$
= \sum_{t=1}^{N} [\mu_\theta(s_t, a_t) - \Delta s_t]^\top \Sigma_\theta^{-1}(s_t, a_t) + \log \det \Sigma_\theta(s_t, a_t)
$$

where $N$ is the total number of samples in $D_{env}$. The loss function of the backward model can be further derived as:

$$
L_b(\theta') = - \sum_{t=1}^{N} \log \tilde{p}_{\theta'}(\Delta s_{t-1} | s_t, a_{t-1})
$$

(11)

$$
= \sum_{t=1}^{N} [\mu_{\theta'}(s_t, a_{t-1}) - \Delta s_{t-1}]^\top \Sigma_{\theta'}^{-1}(s_t, a_{t-1}) + \log \det \Sigma_{\theta'}(s_t, a_{t-1})
$$

**Backward Policy.** The backward policy aims to make the backward rollouts resemble the real traces sampled by the current policy, which entails that it should be trained on the real data $(s_t, a_{t-1})$ by maximum likelihood estimation. Thus, the corresponding loss function is defined as:

$$
L_{bp}(\phi') = - \sum_{t=1}^{N} \log \tilde{\pi}_{\phi'}(a_{t-1} | s_t)
$$

(12)

$$
= \sum_{t=1}^{N} [\mu_{\phi'}(s_t) - a_{t-1}]^\top \Sigma_{\phi'}^{-1}(s_t) + \log \det \Sigma_{\phi'}(s_t)
$$

**Value-regularized GAN.** In our proposed GAN, the generator $G(\cdot)$ parameterized by $\theta^G$ is trained to output
the states which not only follow the real-state distribution in $D_{env}$, but also are the critic-recognized valuable ones. The discriminator $D(\cdot)$ parameterized by $\theta^D$ is trained to distinguish the states in $D_{t{rain}}$ from those being out of the presumed real-state distribution. The value regularized GAN is implemented on the basis of LSGAN [20]. The loss functions of generator and discriminator can be written as:

$$L_G(\theta^G) = \mathbb{E}_{z \sim p_z(z)}[(D(G(z))) - 1]^2] - \lambda L_v$$

(13)

$$L_D(\theta^D) = \mathbb{E}_{s \sim p_s(s)}[(D(s) - 1)^2] + \mathbb{E}_{z \sim p_z(z)}[D(G(z))^2]$$

(14)

where $p_z(z)$ is the standard normal distribution, $s$ is the real state following the real-state distribution $p_s(s)$, and $\lambda$ is the hyperparameter.

**Policy Learning.** The backward imitation of excellent behaviors and the forward reinforcement of policy are severally carried out on the data from $D_b$ and $D_f$. In detail, to imitate excellent behaviors from the traces leading to the high-value states, the policy is learnt on the simulated data in $D_b$ via minimizing the loss function:

$$L_I(\phi) = -\sum_{t=1}^M \log \pi_\phi(a^b_t|s^b_t)$$

(15)

where the tuple $(s^b_t, a^b_t)$ is sampled from $D_b$ with size $M$.

After that, the agent should explore from the valuable states to further reinforce the policy. Thus, we optimize the policy on the samples in $D_f$ through Soft Actor-Critic (SAC) [3] by minimizing the expected KL-divergence:

$$L_R(\phi) = \mathbb{E}_{s^f \sim D_f}[D_{KL}(\pi_\phi||\exp(Q^\pi - V^\pi))]$$

(16)

where Q-value network $Q^\pi(\cdot)$ and state value network $V^\pi(\cdot)$ are trained via minimizing the variants of TD error. Please refer to [3] for the details of SAC algorithm.

V. THEORETICAL ANALYSIS

In this section, we theoretically provide the condition where BIFRL algorithm can produce the better asymptotic performance than previous methods, which is also instructive for hyperparameter settings in experiments. On the one hand, prior works [13], [6] have studied the discrepancy between the expected return in the actual environment and that in the branched rollouts, which is applicable for the analysis on the forward rollout return in BIFRL. On the other hand, unlike those works, BIFRL views backward rollout traces $s^b = [s^b_T, a^b_{T-1}, ..., a^b_1, s^b_0]$ as expert demonstrations $s^e = [s^e_T, a^e_{T-1}, ..., a^e_1, s^e_0]$ to train the policy, and assumes those expert samples are generated from the interactions with a presumed model $p^e(s^e_{t+1}|s^e_t, a^e_t)$ by executing a presumed expert policy $\pi^e(a_t|s_t)$. We thus investigate the discrepancy between the return from the current policy $\pi$ interactions with the real dynamics $p$, denoted as $\eta[\pi]$, and the return from the expert policy $\pi^e$ interactions with the presumed model $p^e$, denoted as $\eta[\pi^e]$. Let $k_b$ be the length of backward rollouts, $k_f$ be length of forward rollouts.

**Theorem 1.** We assume that the expected total variation distance between $\pi$ and $\pi^e$ at each timestep $t$ is bounded as $\epsilon_\pi = \max_t E_{(s,a) \sim \pi,e} [D_{TV}(\pi^e_t(a|s)||\pi_t(a|s))]$, and the expected total variation distance between $p$ and $p^e$ at each time step $t$ is bounded as $\epsilon_\pi = \max_t E_{(s,a,s',a') \sim \pi,e} [D_{TV}(p^e_t(s'|s,a)||p(s'|s,a))]$. Let $r_{max}$ denote the maximum reward. Then the upper bound of the discrepancy between $\eta[\pi]$ and $\eta[\pi^e]$ is:

$$|\eta[\pi] - \eta[\pi^e]| \leq 2r_{max}(1 - \gamma^{k_b+1}) (\epsilon_\pi + k_b(\epsilon_\pi + \epsilon_m))$$

(17)

Proof. Please refer to Appendix for the proof. □

We observe that BMPO [6] ignores the variation distance between the forward policy and the backward policy (corresponding to $\epsilon_\pi$ in BIFRL), and consequently set $k_b = k_f$ to obtain the tighter return discrepancy bound, compared to MBPO [13]. However, based on Theorem 1, $\epsilon_\pi$ cannot be eliminated in the return discrepancy. Thus, compared to forward rollouts that only have the model error, backward rollouts have both the model error and the policy divergence. In consequence, we propose the condition $k_b < k_f$ where BIFRL can perform better asymptotically. The experiments also support our viewpoints.

VI. EXPERIMENTS

The aims of our experimental evaluation are to answer two primary questions: 1) How does BIFRL algorithm perform on various benchmarks compared with state-of-the-art model-free and model-based methods? 2) what are the contributions of the critical components in BIFRL?

A. Performance Comparisons

In this section, we evaluate BIFRL and other methods on five MuJoCo continuous control tasks [7] where the agent requires to learn the moving forward gaits. The agents we used are Hopper, Walker2D and Ant. And we add two variants of Hopper and Walker2D without early termination states, denoted as Hopper-NT and Walker2D-NT [21]. Additionally, we choose Pendulum task [22] to evaluate the performance of BIFRL on traditional control task. Please refer to [7], [21], [22] for the environment settings.

We compare BIFRL algorithm with previous state-of-the-art algorithms in those six domains. To be specific, for model-based methods, we compare against MBPO [13] that is the backbone model of our method, and BMPO [6] that treats the samples from backward rollouts in the same way as those from forward rollouts. We also compare to PETS [19] that incorporates the learnt model ensembles into model predictive control (MPC) [23] for planning. For model-free methods, we compare to SAC [3], which is used for policy reinforcement in BIFRL. We do not compare with the backtracking model method [5] as it is not a purely model-based method and its improvements on both the performance and efficiency are limited compared with SAC. In addition, some previous model-based methods only involving the forward model rollouts (e.g. MAAC [24], MoPAC [25]) can be integrated with BIFRL. It is unfair to directly compare BIFRL against those model-based methods, and the compatibility of BIFRL with those baselines can
be researched in future works. In experiments, we use the default hyperparameter settings in both the baselines and the backbone model of BIFRL for fair comparison. And the unique hyperparameters of BIFRL are included in Appendix.

The learning curves of all the methods are shown in Figure 3. We observe that in the four domains, including Ant, Hopper, Walker2D and Walker2D-NT, BIFRL acquires better sample efficiency resulting in the faster convergence rate, and meanwhile to some extent improves the asymptotic performance. In Hopper-NT, BIFRL does not make considerable improvements. This is probably because the high-value states cannot be frequently encountered in this domain. And in Pendulum, PETS is superior to all the other model-based methods. One possible reason is that MPC-based methods are more applicable for the traditional control tasks.

More qualitative results are shown in our supplementary materials, which provides the qualitative comparisons among BIFRL, BMPO, MBPO and SAC.

B. Design Evaluation

In this section, we investigate the contribution of each component and study the important hyperparameters of BIFRL on Hopper task.

Ablation Study. As illustrated in Figure 4(a), we conduct the ablation study to evaluate the contributions of two main components in BIFRL: 1) backward imitation that views backward rollout samples as expert demonstrations, denoted as BI; 2) value-regularized GAN that augments the high-value states, denoted as VGAN. To evaluate the importance of BI, we compare the performance among three models: 1) MBPO that optimizes the policy on forward rollout samples via RL algorithm (Baseline); 2) BMPO that adds the backward rollout samples to the baseline model (Baseline+BR); 3) the model that employs backward imitation (Baseline+BI). We observe that the baseline model with backward imitation convergences faster, which demonstrates that using backward rollout samples as expert demonstrations is more efficient than simply feeding those samples to RL algorithm. To characterize the importance of VGAN, we additionally choose two models: 1) using vanilla LSGAN [17] to generate states (Baseline+BI+GAN); 2) using value-regularized GAN for high-value states (Baseline+BI+VGAN). We find that VGAN can contribute more to the performance in terms of convergence rate, which verifies the effectiveness of VGAN.

Hyperparameter Study. We further investigate the important hyperparameters in our algorithm. Firstly, we conduct experiments with different $K$ which indicates the quantity of sampled high-value states for backward rollouts. We set $K = 1, 10, 50$ respectively to study its influence. The results are shown in Figure 4(b). We find that the large number of sampled states can impair the performance. This probably because the larger $K$ makes the high-value states sparse in $D_{hs}^k$. We observe that too few high-value states also degrades the performance. Thus, it is crucial to choose an appropriate quantity of high-value states for backward rollouts.

The strategy of increasing rollout length linearly has shown the effectiveness in MBPO and BMPO. However, we require to study how to choose the backward rollout length $k_b$ in our algorithm when keeping the same forward rollout length $k_f$ as BMPO. In section V, we derive the condition...
where $\text{BIFRL}$ can perform better, we thus vary $k_b$ from 0 to $k_f$. As shown in Figure 4(c), setting $k_b = \frac{2}{3}k_f$ provides the better results, while neither too short or too long backward rollout length improves the performance. In practice, we use $k_b = \frac{2}{3}k_f$ in most domains except Walker2D and Walker2D-NT where $k_b$ is set to the same as $k_f$ due to $k_f = 1$ in the original MBPO paper. Please refer to Appendix for the unique hyperparameters of BIFRL.

To prove that the policy divergence $\epsilon_{\pi}$ cannot be eliminated in Theorem 1, we choose the mean squared error (MSE) to measure both the bi-directional model errors and the policy divergence on validation set. Figure 4(d) shows the changes of these errors in the training phase. We notice that the policy divergence is two orders of magnitude larger than the bi-directional model errors at the end of training, which supports our view, i.e. $\epsilon_{\pi}$ cannot be eliminated, and further experimentally verifies the validity of condition $k_b < k_f$.

VII. CONCLUSION

In this paper, we propose a new model-based RL framework called backward imitation and forward reinforcement learning (BIFRL) framework where the agent views backward rollout traces as expert demonstrations to imitate the excellent behaviors from those traces, and uses forward rollout transitions to optimize the policy via RL algorithm. Theoretically, we provide the condition where BIFRL can produce the better asymptotic performance, and the experiment results actually support our view and derived condition in the analysis. Furthermore, BIFRL can empirically improves the sample efficiency and produces the competitive asymptotic performance on various locomotion tasks. For future work, we will investigate the compatibility of BIFRL with other model-based methods, and study its effectiveness in more real scenarios.

APPENDIX

A. Proof

Proof. Firstly, the state-action space is assumed to be finite due to the restrictions on states or actions of the robot. According to the definition of the return in terms of the occupancy measure and the total variation distance, the return discrepancy bound can be derived as:

$$|\eta[\pi] - \eta[\pi^e]| = \left| \sum_{s,a} \sum_{t=0}^{k_b} \gamma^t r(s,a)(p_t(s,a) - p_t^e(s,a)) \right|$$

$$\leq 2r_{max} \sum_{s,a} \sum_{t=0}^{k_b} \gamma^t \frac{1}{2} |p_t(s,a) - p_t^e(s,a)|$$

$$\leq 2r_{max} \sum_{t=0}^{k_b} \gamma^t D_{TV}(p_t(s,a)||p_t^e(s,a)),$$

(18)

Next, according to Lemma B.1 in MBPO [13], we convert joint distribution to marginal distribution, thus we have:

$$D_{TV}(p_t(s,a)||p_t^e(s,a)) \leq D_{TV}(p_t(s)||p_t^e(s)) + \max_{a|s} E_{s \sim p_t(s)}[D_{TV}(\pi_t(a|s)||\pi_t^e(a|s))].$$

(19)

Then, let $\xi_t = D_{TV}(p_t(s)||p_t^e(s))$, and inspired by Lemma B.1 in MBPO [6], we have:

$$\xi_t \leq E_{(s',a) \sim p_{t+1}(s',a)}[D_{TV}(p(s||s',a)||p^e(s||s',a)) + D_{TV}(p_{t+1}(s',a)||p_{t+1}^e(s',a))]$$

(20)

Here, we make the assumption without loss of generality:

$$\epsilon_m \geq \max_t E_{(s',a) \sim p_t(s',a)}[D_{TV}(p(s||s',a)||p^e(s||s',a))]$$

(21)

After that, we can iteratively do the above decomposition to obtain:

$$\xi_t \leq (\epsilon_m + \epsilon_{\pi}) + \xi_{t+1} \leq (\epsilon_m + \epsilon_{\pi})(k_b - t) + \xi_{k_b}$$

(22)

where $\xi_{k_b} = 0$, because the real states sampled from $D_{ha}^b$ are used as the starting states for the backward rollouts.

Finally, the upper bound of discrepancy between $\eta[\pi]$ and $\eta[\pi^e]$ can be described as:

$$|\eta[\pi] - \eta[\pi^e]| \leq \frac{2r_{max}(1 - \gamma^{k_b+1})}{1 - \gamma}(\epsilon_{\pi} + k_b(\epsilon_{\pi} + \epsilon_m))$$

(23)
TABLE I: Hyperparameter settings in BIFRL.

| Environment | $k_b$ | $k_f$ | $\epsilon_v$ |
|-------------|-------|-------|-------------|
| Pendulum    | $1 \rightarrow 3 \mid 1 \rightarrow 5$ | $1 \rightarrow 5 \mid 1 \rightarrow 5$ | 10 |
| Hopper      | $1 \rightarrow 10 \mid 20 \rightarrow 150$ | $1 \rightarrow 15 \mid 20 \rightarrow 150$ | 20 |
| Hopper-NT   | $1 \rightarrow 10 \mid 20 \rightarrow 150$ | $1 \rightarrow 15 \mid 20 \rightarrow 150$ | 20 |
| Walker2D    | 1     | 1     | 40          |
| Walker2D-NT | 1     | 1     | 40          |

Algorithm 1 BIFRL Algorithm

Initialization: policy $\pi_{\phi}$, backward policy $\pi_{\phi'}$, forward model $\tilde{p}_{\theta}$, backward model $\tilde{p}_{\theta'}$, and VGAN (generator, discriminator)

1. **for** $N$ epochs **do**
2. Collect data from environment; add to $\mathcal{D}_{env}$
3. Train $p_{\theta}, p_{\theta'}, \pi_{\phi'},$ and VGAN using $\mathcal{D}_{env}$ via applying gradient descent on Equation 10, 11, 12, 13 and 14
4. Aggregate states from $\mathcal{D}_{env}$ and VGAN
5. Sample top $K$ percent of high-value states from aggregated states; add to $\mathcal{D}_{hs}^b$
6. Sample high-value states from aggregated states according to Equation 7; add to $\mathcal{D}_{hs}^f$
7. **for** $M_1$ backward rollouts **do**
8. Perform $k_b$ steps backward rollouts; add to $\mathcal{D}_b$
9. **end for**
10. **for** $G_1$ gradient updates **do**
11. Update policy on $\mathcal{D}_b$ by optimizing Equation 15
12. **end for**
13. **for** $M_2$ forward rollouts **do**
14. Perform $k_f$ steps forward rollouts; add to $\mathcal{D}_f$
15. **end for**
16. **for** $G_2$ gradient updates **do**
17. Update policy on $\mathcal{D}_f$ by optimizing Equation 16
18. **end for**
19. **end for**

B. Hyperparameters

Table I provides the unique hyperparameters in BIFRL. $x \rightarrow y \mid a \rightarrow b$ indicates clipped linear function, i.e. for epoch $c$, $f(c) = \text{clip}(x + \frac{c-a}{b-a}(x-y), x, y)$. $\alpha$ increases from 0.2 to 0.95 with the epoch changing from 1 to $e_\alpha$. $\beta$ is set as 0.7 in all the domains. $\lambda$ is 0.0001 in most domains except Pendulum where it is 0.001. The ratio between states from $\mathcal{D}_{env}$ and those from VGAN is 1. Other hyperparameters not listed here are the same as those in MBPO [6]. Additionally, the network architectures and the experimental settings in value-regularized GAN are the same as those in Goal GAN [17] that is also built on LSGAN [20].

REFERENCES

[1] 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, pp. 529–533, 2015.
[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, 2015, pp. 1889–1897.
[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International Conference on Machine Learning*, 2018.
[4] M. P. Deisenroth, G. Neumann, and J. Peters, *A survey on policy search for robotics*, now publishers, 2013.
[5] A. Goyal, P. Brakel, W. Fedus, S. Singhal, T. Lillicrap, S. Levine, H. Larochelle, and Y. Bengio, “Recall traces: Backtracking models for efficient reinforcement learning,” in *International Conference on Learning Representations*, 2019.
[6] H. Lai, J. Shen, W. Zhang, and Y. Yu, “Bidirectional model-based policy optimization,” in *International Conference on Machine Learning*, PMLR, 2020, pp. 5618–5627.
[7] E. Todorov, T. Erez, and Y. Tassa, “Mujoec: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
[8] N. Mishra, P. Abbeel, and I. Mordatch, “Prediction and control with temporal segment models,” in *International Conference on Machine Learning*, 2017.
[9] W. Whitney and R. Fergus, “Understanding the asymptotic performance of model-based rl methods,” 2018.
[10] Y.-H. Wu, T.-H. Fan, P. J. Ramadge, and H. Su, “Model imitation for model-based reinforcement learning,” in *International Conference on Learning Representations*, 2020.
[11] N. M. Nguyen, “Improving model-based rl with adaptive rollout using uncertainty estimation,” 2018.
[12] C. Xiao, Y. Wu, C. Ma, D. Schuurmans, and M. Müller, “Learning to combat compounding-error in model-based reinforcement learning,” in *NeurIPS 2019 Deep Reinforcement Learning Workshop*, 2019.
[13] M. Janner, J. Fu, M. Zhang, and S. Levine, “When to trust your model: Model-based policy optimization,” in *Advances in Neural Information Processing Systems*, 2019, pp. 12 519–12 530.
[14] A. D. Edwards, L. Downs, and J. C. Davidson, “Forward-backward reinforcement learning,” *arXiv preprint arXiv:1803.10227*, 2018.
[15] X. Liang, T. Wang, L. Yang, and E. Xing, “Crl: Controllable iterative reinforcement learning for vision-based self-driving,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 584–599.
[16] Y. Pan, J. Xue, P. Zhang, W. Ouyang, J. Fang, and X. Chen, “Navigation command matching for vision-based autonomous driving,” in *2020 IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 4343–4349.
[17] C. Florensa, D. Held, X. Geng, and P. Abbeel, “Automatic goal generation for reinforcement learning agents,” in *International conference on machine learning*. PMLR, 2018, pp. 1515–1528.
[18] S. Sutton, “Dyna, an integrated architecture for learning, planning, and reacting,” ACM Sigart Bulletin, vol. 2, no. 4, pp. 160–163, 1991.
[19] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.
[20] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.
[21] E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, “Benchmarking model-based reinforcement learning,” *arXiv preprint arXiv:1907.02057*, 2019.
[22] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
[23] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.
[24] I. Clavera, Y. Fu, and P. Abbeel, “Model-augmented actor-critic: Back-propagating through paths,” in *International Conference on Learning Representations*, 2019.
[25] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D’Eramo, A. M. Dollar, and J. Peters, “Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6672–6678.