BRPO: Batch Residual Policy Optimization

Sungryull Sohn[*,1,2] , **Yinlam Chow**[*,1] , **Jayden Ooi**[*,1] ,
Ofir Nachum[1] , **Honglak Lee**[1,2] , **Ed Chi**[1] and **Craig Boutilier**[1]

[1]Google Research
[2]University of Michigan

srsohn@umich.edu, {yinlamchow, jayden, ofirnachum, honglak, edchi, cboutilier}@google.com

Abstract

In batch reinforcement learning (RL), one often constrains a learned policy to be close to the behavior (data-generating) policy, e.g., by constraining the learned action distribution to differ from the behavior policy by some maximum degree that is the *same at each state*. This can cause batch RL to be overly conservative, unable to exploit large policy changes at frequently-visited, high-confidence states without risking poor performance at sparsely-visited states. To remedy this, we propose *residual policies*, where the allowable deviation of the learned policy is *state-action-dependent*. We derive a new for RL method, *BRPO*, which learns both the policy and allowable deviation that jointly maximize a lower bound on policy performance. We show that BRPO achieves the state-of-the-art performance in a number of tasks.

1 Introduction

Deep reinforcement learning (RL) methods are increasingly successful in domains such as games [Mnih *et al.*, 2013], and robotic manipulation [Nachum *et al.*, 2019]. Much of this success relies on the ability to collect new data through *online* interactions with the environment during training, often relying on simulation. Unfortunately, this approach is impractical in many real-world applications where faithful simulators are rare, and in which active data collection through interactions with the environment is costly, time consuming, and risky.

Batch (or offline) RL [Lange *et al.*, 2012] is an emerging research direction that aims to circumvent the need for online data collection, instead learning a new policy using only offline trajectories generated by some behavior policy (e.g., the currently deployed policy in some application domain). In principle, any off-policy RL algorithm (e.g., DDPG [Lillicrap *et al.*, 2015], DDQN [Hasselt *et al.*, 2016]) may be used in this batch (or more accurately, "offline") fashion; but in practice, such methods have been shown to fail to learn when presented with arbitrary, static, off-policy data. This can arise for several reasons: lack of exploration [Lange *et al.*, 2012], generalization error on out-of-distribution samples in value estimation [Kumar *et al.*, 2019], or high-variance policy gradients induced by covariate shift [Mahmood *et al.*, 2014].

Various techniques have been proposed to address these issues, many of which can be interpreted as constraining or regularizing the learned policy to be *close* to the behavior policy [Fujimoto *et al.*, 2018; Kumar *et al.*, 2019] (see further discussion below). While these batch RL methods show promise, none provide improvement guarantees relative to the behavior policy. In domains for which batch RL is well-suited (e.g., due to the risks of active data collection), such guarantees can be critical to deployment of the resulting RL policies.

In this work, we use the well-established methodology of *conservative policy improvement (CPI)* [Kakade and Langford, 2002] to develop a theoretically principled use of behavior-regularized RL in the batch setting. Specifically, we parameterize the learned policy as a *residual policy*, in which a base (behavior) policy is combined linearly with a learned *candidate policy* using a mixing factor called the *confidence*. Such residual policies are motivated by several practical considerations. First, one often has access to offline data or logs generated by a deployed base policy which is known to perform reasonably well. The offline data can be used by an RL method to learn a candidate policy with better *predicted* performance, but if confidence in parts of that prediction is weak, relying on the base policy may be desirable. The base policy may also incorporate soft business constraints or some form of interpretability. Our residual policies blend the two in a learned, non-uniform fashion. When deploying a new policy, we use the CPI framework to derive updates that learn both the candidate policy and the confidence that jointly maximize a lower bound on performance improvement relative to the behavior policy. Crucially, while traditional applications of CPI, such as TRPO [Schulman *et al.*, 2015], use a constant or state-independent confidence, our performance bounds and learning rules are based on *state-action-dependent* confidences—this gives rise to bounds that are less conservative than their CPI counterparts.

In Sec. 2, we formalize residual policies and in Sec. 3 analyze a novel *difference-value function*. Sec. 4 holds our main result, a tighter lower bound on policy improvement for our residual approach (vs. CPI and TRPO). We derive the BRPO algorithm in Sec. 5 to jointly learn the candidate policy and confidence; experiments in Sec. 6 show its effectiveness.

[*]Equal contribution

2 Preliminaries

We consider a *Markov decision process (MDP)* $\mathcal{M} = \langle S; A; R; T; P_0 \rangle$, with state space S, action space A, reward function R, transition kernel T, and initial state distribution P_0. A policy π interacts with the environment, starting at $s_0 \sim P_0$. At step t, the policy samples an action a_t from a distribution $\pi(\cdot|s_t)$ over A and applies. The environment emits a reward $r_t = R(s_t; a_t) \in [0; R_{\max}]$ and next state $s_{t+1} \sim T(\cdot|s_t; a_t)$. In this work, we consider discounted infinite-horizon problems with discount factor $\gamma \in [0; 1)$.

Let $\Pi = \{\pi : S \times A \to [0; 1]; \sum_a \pi(a|s) = 1\}$ be the set of Markovian stationary policies. The expected (discounted) cumulative return of policy $\pi \in \Pi$, is $J_\pi := \mathbb{E}_{T;\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t; a_t) \mid s_0 \sim P_0]$. Our aim is to find an optimal policy $\pi^* \in \arg\max_{\pi \in \Pi} J_\pi$. In reinforcement learning (RL), we must do so without knowledge of $R; T$, using only trajectory data generated from the environment (see below) or access to a simulator (see above).

We consider pure offline or *batch RL*, where the learner has access to a fixed data set (or *batch*) of state-actions-reward-next-state samples $B = \{(s; a; r; s')\}$, generated by a (known) *behavior policy* $\beta(\cdot|s)$. No additional data collection is permitted. We denote by d_π the γ-discounted occupation measure of the MDP w.r.t. π.

In this work, we study the problem of *residual policy optimization (RPO)* in the batch setting. Given the behavior policy $\beta(a|s)$, we would like to learn a *candidate policy* $\pi(a|s)$ and a state-action *confidence* $\alpha(s; a)$, such that the final *residual policy* $\pi_\beta(a|s) = (1 - \alpha(s; a)) \cdot \beta(a|s) + \alpha(s; a) \cdot \pi(a|s)$ maximizes total return. As discussed above, this type of mixture allows one to exploit an existing, "well-performing" behavior policy. Intuitively, $\alpha(s; a)$ should capture how much we can trust π at each $s; a$ pair, given the available data. To ensure that the residual policy is a probability distribution at every state $s \in S$, we constrain the confidence α to lie in the set $\mathcal{A}(\pi) = \{\alpha : S \times A \to [0; 1] : \sum_a \alpha(s; a)(\pi(a|s) - \beta(a|s)) = 0\}$.

Related work. Similar to the above policy formulation, CPI [Kakade and Langford, 2002] also develops a policy mixing methodology that guarantees performance improvement when the confidence α is a constant. However, CPI is an online algorithm, and it learns the candidate policy independently of (not jointly with) the mixing factor; thus, extension of CPI to offline, batch setting is unclear. Other existing work also deals with online residual policy learning without jointly learning mixing factors [Johannink *et al.*, 2019; Silver *et al.*, 2018]. Common applications of CPI may treat α as a hyper-parameter, which specifies the maximum total-variation distance between the learned and behavior policy distributions (see standard proxies in [Schulman *et al.*, 2015; Pirotta *et al.*, 2013] for details). *Batch-constrained Q-learning (BCQ)* [Fujimoto *et al.*, 2018; Fujimoto *et al.*, 2019] incorporates the behavior policy when defining the admissible action set in Q-learning for selecting the highest-valued actions that are similar to data samples in the batch. *BEAR* [Kumar *et al.*, 2019] is motivated as a means to control the accumulation of out-of-distribution value errors; but its main algorithmic contribution is realized by adding a regularizer to the loss that measures the kernel maximum mean discrepancy (MMD) [Gretton *et al.*, 2007] between the learned and behavior policies similar to KL-control [Jaques *et al.*, 2019]. Algorithms such as SPI [Ghavamzadeh *et al.*, 2016] and SPIBB [Laroche and Trichelair, 2017] bootstraps the learned policy with the behavior policy when the uncertainty in the update for current state-action pair is high, where the uncertainty is measured by the visitation frequency of state-action pairs in the batch data. While these methods work well in some applications it is unclear if they have any performance guarantees.

3 The Difference-value Function

We begin by defining and characterizing the *difference-value function*, a concept we exploit in the derivation of our batch RPO method in Secs. 4 and 5. For any $s \in S$, let $V_\pi(s)$ and $V_\beta(s)$ be the value functions induced by policies π and β, respectively. Using the structure of the residual policy, we establish two characterizations of the *difference-value function* $V_{\pi;\beta}(s) := V_\pi(s) - V_\beta(s)$.

Lemma 1. *Let $A_\beta(s; a) := Q_\beta(s; a) - V_\beta(s)$ be the advantage function w.r.t. residual policy π_β, where Q_β is the state-action value. The difference-value is $V_{\pi;\beta}(s) = \mathbb{E}_{T;\pi}[\sum_{t=0}^{\infty} \gamma^t \hat{A}_{\pi;\beta;\beta}(s_t) \mid s_0 = s]$; where $\hat{A}_{\pi;\beta;\beta}(s) = \sum_{a \in A} \beta(a|s) \cdot \alpha(s; a) \cdot \frac{\pi(a|s)}{\beta(a|s)} \cdot A_\beta(s; a)$ is the residual reward that depends on α and difference of candidate policy π and behavior policy β.*

This result establishes that the difference value is essentially a value function w.r.t. the residual reward. Moreover, it is proportional to the advantage of the target policy, the confidence, and the difference of policies. While the difference value can be estimated from behavior data batch B, this formulation requires knowledge of the advantage function A_β w.r.t. the target policy, which must be re-learned at every π-update in an off-policy fashion. Fortunately, we can show that the difference value can also be expressed as a function of the advantage w.r.t. the *behavior policy* β:

Theorem 2. *Let $A_\beta(s; a) := Q_\beta(s; a) - V_\beta(s)$ be the advantage function induced by β, in which Q_β is the state-action value. The difference-value is given by $V_{\pi;\beta}(s) = \mathbb{E}_{T;\beta}[\sum_{t=0}^{\infty} \gamma^t A_{\pi;\beta;\beta}(s_t) | s_0 = s]$, where*

$$A_{\pi;\beta;\beta}(s) = \sum_{a \in A} \beta(a|s) \; \alpha(s; a) \; \frac{\pi(a|s) \quad \beta(a|s)}{\beta(a|s)} \; A_\beta(s; a)$$

is the residual reward that depends on α and difference of candidate policy π and behavior policy β.

In our RPO approach, we exploit the nature of the difference-value function to solve the maximization w.r.t. the confidence and candidate policy: $(\alpha(s; \cdot); \pi(\cdot|s)) \in \arg\max_{\alpha \in \mathcal{A}(\pi); \pi} V_{\pi;\beta}(s), \forall s \in S$. Since $\alpha(s; \cdot) = 0$ implies $V_{\pi;\beta}(s) = 0$, the optimal difference-value function $V^*(s) := \max_{\alpha \in \mathcal{A}(\pi); \pi} V_{\pi;\beta}(s)$ is always lower-bounded by 0. We motivate computing $(\alpha; \pi)$ with the above difference-value formulation rather than as a standard RL problem as follows. In the tabular case, optimizing $(\alpha; \pi)$ with either formulation gives an identical result. However,

both the difference-value function in Theorem 2 and the standard RL objective require sampling data generated by the updated policy . In the batch setting, when fresh samples are unavailable, learning (;) with off-policy data may incur instability due to high generalization error [Kumar *et al.*, 2019]. While this can be alleviated by adopting the CPI methodology, applying CPI directly to RL can be overly conservative [Schulman *et al.*, 2015]. By contrast, we leverage the special structure of the difference-value function (e.g., non-negativity) below, using this new formulation together with CPI to derive a less conservative RPO algorithm.

4 Batch Residual Policy Optimization

We now develop an RPO algorithm that has stable learning performance in the batch setting and *performance improvement guarantees*. For the sake of brevity, in the following we only present the main results on performance guarantees of RPO. Proofs of these results can be found in the appendix of the extended paper. We begin with the following baseline result, directly applying Corollary 1 of the TRPO result to RPO to ensure the residual policy performs no worse than .

Lemma 3. *For any value function $U : S \to \mathsf{R}$, the difference-return satisfies* $J - J \geq \frac{1}{1-}\big[\hat{\mathsf{E}}_{U;\,;\,} - \frac{2}{(1-)^2}\cdot U;\,;\,\cdot \mathsf{E}_{s\ d}\ \frac{1}{2}D_{\mathsf{KL}}(\ (s)\|\ (s))\big]$ *; where the surrogate objective and the penalty weight are*

$$\hat{\mathsf{E}}_{U;\,;\,} := \mathop{\mathsf{E}}_{(s;a;s^0)\sim d}\ (s;a)\ \frac{(ajs)\quad (ajs)}{(ajs)}\ U(s;a;s')\ ;$$

$$U;\,;\,\ := \max_s j\mathsf{E}\ ;T[\ U(s;a;s')]j;$$

where $U(s;a;s^0) := R(s;a) + \ U(s^0) - U(s).$

When $U = V$, one has $\mathsf{E}_{a\ (s)}[\ U(s;a;s^0)] = 0, \forall s \in S$, which implies that the inequality is tight—this lemma then coincides Lemma 1. While this CPI result forms the basis of many RL algorithms (e.g., TRPO, PPO), in many cases it is very loose since $_{U;\,;\,}$ is a maximum over all states. Thus, using this bound for policy optimization may be *overly conservative*, i.e., algorithms which rely on this bound must take very small policy improvement steps, especially when the penalty weight $_{U;\,;\,}$ is large, i.e., $|\ _{U;\,;\,} = (1-\ )| >> |\hat{\mathsf{E}}_{U;\,;\,}|$. While this approach may be reasonable in online settings—when collection of new data (with an updated behavior policy $\leftarrow$) is allowed—in the batch setting it is challenging to overcome such conservatism.

To address this issue, we develop a CPI method that is specifically tied to the difference-value formulation, and uses a state-action-dependent confidence $(s;a)$. We first derive the following theorem, which bounds the difference returns that are generated by and .

Theorem 4. *The difference return of* (;) *satisfies*

$$J - J \geq \frac{1}{1-}\ L^{0}\,;\,;\, - \frac{}{1-}\cdot L^{00}\,;\,;\, \cdot \max_{s_0 2 S} L^{000}\,;\,;\,(s_0)\ ;$$

where the surrogate objective function, regularization, and penalty weight are given by

$$L'\,;\,;\, := \mathsf{E}_{(s;a)\sim d}\quad (s;a)\ \frac{(ajs)\quad (ajs)}{(ajs)}\ A\ (s;a)$$

$$L''\,;\,;\, := \mathsf{E}_{(s;a)\sim d}\quad (s;a)\ \frac{j\ (ajs)\quad (ajs)j}{(ajs)}$$

$$L'''\,;\,;\,(s_0) := \mathsf{E}_{(s;a)\sim d\ (s_0)}\quad (s;a)\ \frac{j\ (ajs)\quad (ajs)j}{(ajs)}\ jA\ (s;a)j$$

respectively, in which $d\ (s_0)$ is the discounted occupancy measure w.r.t. given initial state s_0.

Unlike the difference-value formulations in Lemma 1 and Theorem 2, which require the knowledge of advantage function A or the trajectory samples generated by , the lower bound in Theorem 4 is comprised only of terms that can be estimated directly using the data batch B (i.e., data generated by). This makes it a natural objective function for batch RL. Notice also that the surrogate objective, the regularization, and the penalty weight in the lower bound are each proportional to the confidence and to the relative difference of the candidate and behavior policies. However, the max operator requires state enumeration to compute this lower bound, which is intractable when S is large or uncountable.

We address this by introducing a slack variable ≥ 0 to replace the max-operator with suitable constraints. This allows the bound on the difference return to be rewritten as: $J - J \geq \frac{1}{1-}L^{0}\,;\,;\, - \min\ _{L^{000}\,;\,;\,(s_0);\ 8s_0\ \frac{}{(1-)^2}L^{00}\,;\,;\,}\ :$

Consider the Lagrangian of the lower bound:

$$\frac{L'\,;\,;\,}{1-}\quad \min_{\geq 0}\ \max_{(s)\geq 0;\forall s}\ \frac{L''\,;\,;\,}{(1-)^2}\quad \sum_s\ (s)(\quad L'''\,;\,;\,(s)):$$

To simplify this saddle-point problem, we restrict the Lagrange multiplier to be $(s) = \cdot P_0(s) \geq 0$, where ≥ 0 is a scalar multiplier. Using this approximation and the strong duality of linear programming [Boyd and Vandenberghe, 2004] over primal-dual variables (;), the saddle-point problem on (; ; ;) can be re-written as

$$L\ ;\,;\, := \max_{\geq 0}\min_{\geq 0}\ \frac{L'\,;\,;\,\quad L''\,;\,;\,\quad \frac{}{1-}\quad +\ L'''\,;\,;\,}{1-}$$

$$= \frac{1}{1-}\quad L'\,;\,;\,\quad \frac{}{1-}L''\,;\,;\,\quad L'''\,;\,;\,\quad ; \qquad (1)$$

where $L^{000}\,;\,;\, = \mathsf{E}_{s\ P_0}[L^{000}\,;\,;\,(s)]$. The equality is based on the KKT condition on (;). Notice that the only difference between the CPI lower bound in Theorem 4 and the objective function $\mathcal{L}\ ;\,;\,$ is that the max operator is replaced by expectation w.r.t the initial distribution.

With certain assumptions on the approximation error of the Lagrange multiplier parametrization $(s) \approx P_0(s)$, we can characterize the gap between the original CPI objective function in Theorem 4 and $\mathcal{L}\ ;\,;\,$. One approach is to look into the KKT condition of the original saddle-point problem and bound the sub-optimality gap introduced by this Lagrange parameterization. Similar derivations can be found in the analysis of approximate linear programming (ALP) algorithms [Abbasi-Yadkori *et al.*, 2019; Farias and Roy, 2003].

Compared with the vanilla CPI result from Lemma 3, there are two characteristics in problem (1) that make the optimization w.r.t. $\mathcal{L}\ ;\,;\,$ less conservative. First, the penalty weight

$L^{000}_{;;}$ here is smaller than $_{U;;;}$ in Lemma 3, which means that the corresponding objective has less incentive to force to be close to . Second, compared with entropy regularization in vanilla CPI, here the regularization and penalty weight are both linear in 2 $[0;1]^{|A|}$; thus, unlike vanilla CPI, whose objective is linear in , our objective is quadratic in —this modification ensures the optimal value is not degenerate extreme point of .[1]

5 The BRPO Algorithm

We now develop the BRPO algorithm, for which the general pseudo-code is given in Algorithm 1. Recall that if the candidate policy and confidence are jointly optimized

$$(\ ; \ ) \ 2 \ \arg\max_{2 \ ; \ 2} L_{\ ;\ ;\ } \ ; \qquad (2)$$

then the residual policy $(ajs) = (1 \quad (s;a)) \ (ajs) + \ (s;a) \ (ajs)$ performs no worse than behavior policy . Generally, solutions for problem (2) use a form of minorization-maximization (MM) [Hunter and Lange, 2004], a class of methods that also includes expectation maximization. In the terminology of MM algorithms, $L_{\ ;\ ;\ }$ is a surrogate function satisfying the following MM properties

$$J \quad J \quad L_{\ ;\ ;\ } \ ; \ J \quad J = L_{\ ;\ ;\ } = L_{\ ;\ ;\ 0} = 0 ; \qquad (3)$$

which guarantees that it minorizes the difference-return J with equality at $= 0$ (with arbitrary) or at $=$ (with arbitrary). This algorithm is also reminiscent of proximal gradient methods. We optimize and in RPO with a simple two-step coordinate-ascent. Specifically, at iteration $k \ 2 \ f0;1;:::;K \ g$, given confidence $_{k \ 1}$, we first compute an updated candidate policy , and with $_k$ fixed, we update $_k$, i.e., $L_{\ ;\ k;\ k} \ L_{\ ;\ k;\ k\ 1} \ L_{\ ;\ k\ 1;\ k\ 1}$. When and are represented tabularly or with linear function approximators, under certain regularity assumptions (the Kurdyka-Lojasiewicz property [Xu and Yin, 2013]) coordinate ascent guarantees global convergence (to the limit point) for BRPO.

However, when more complex representations (e.g., neural networks) are used to parameterize these decision variables, this property no longer holds. While one may still compute $(\ ; \ )$ with first-order methods (e.g., SGD), convergence to local optima is not guaranteed. To address this, we next further restrict the MM procedure to develop closed-form solutions for both the candidate policy and the confidence.

The Closed-form Candidate Policy . To effectively update the candidate policy when given the confidence 2 , we develop a closed-form solution for . Our approach is based on maximizing the following objective, itself a more conservative version of the CPI lower bound $L_{\ ;\ ;\ }$:

$$\max_2 \hat{L}_{\ ;\ ;\ } := E_{s \ d} \ E_a \quad (s;a) \frac{(ajs) \quad (ajs)}{(ajs)} A$$
$$\frac{\max f \ (s); \ _{jA \ j} \ (s) g}{2(1 \quad)} \ D_{KL}(\ k \ )(s) \quad \frac{1}{1} ; \qquad (4)$$

where $_g(s) = (1 + \log \ E \ [\exp(g(ajs)^2)]) > 0$ for any arbitrary non-negative function . To show that $\hat{L}_{\ ;\ ;\ }$ in (4) is an eligible lower bound (so that the corresponding is an MM), we need to show that it satisfies the properties in (3). When $=$, by the definition of $\hat{L}_{\ ;\ ;\ }$ the second property holds. To show the first property, we first consider the following problem:

$$\max_2 \frac{1}{1} \quad L^0_{\ ;\ ;\ } \quad \frac{}{1} E^{00}_{\ ;\ ;\ } \quad E^{000}_{\ ;\ ;\ } \ ; \qquad (5)$$

where $L^0(\ ; \ )$ is given in Theorem 4, and

$$E^{00}_{\ ;\ ;\ } = E_{s \ d} \left[\sqrt{ \ (s) \ D_{KL}(\ k \ )(s)=2} \right] ;$$
$$E^{000}_{\ ;\ ;\ } = E_{s \ d} \left[\sqrt{ \ _{jA \ j} \ (s) \ D_{KL}(\ k \ )(s)=2} \right] : \qquad (6)$$

The concavity of $\sqrt{\ }(\ )$ (i.e., $E_{s \ d}[\sqrt{\ }(\ )] \quad \sqrt{E_{s \ d}[(\ )]}$) and monotonicity of expectation imply that the objective in (4) is a lower bound of that in (7) below. Furthermore, by the weighted Pinsker's inequality [Boucheron and Villani, 2005] $\sqrt{}$ $_a \ jg(ajs)(\ (ajs) \quad (ajs))j \ \sqrt{_g(s)D_{KL}(\ k \ )(s)=2}$, we have: $0 \ E^{00}_{\ ;\ ;\ } \quad L^{00}_{\ ;\ ;\ } \ ; \ 0 \ E^{000}_{\ ;\ ;\ } \quad L^{000}_{\ ;\ ;\ }$; which implies the objective in (5) is a lower-bound of that in (2) and validates the first MM property.

Now recall the optimization problem $\max_2 \hat{L}_{\ ;\ ;\ }$. Since this optimization is over the state-action mapping , the Interchangeability Lemma [Shapiro et al., 2009] allows swapping the order of $E_{s \ d}$ and $\max_2$. This implies that at each $s \ 2 \ S$ the candidate policy can be solved using:

$$2 \ \arg\max_{(\ js) 2} E_a \quad \quad A \quad (s) \log \frac{(ajs)}{(ajs)}$$
$$= \arg\max_{(\ js) 2} E_a \quad (s;a) A \quad (s) \log \frac{(ajs)}{(ajs)} \ ; \qquad (7)$$

where $(s) = \max f \ (s); \ _{jA \ j} \ (s) g=(2 \quad 2 \ )$ is the state-dependent penalty weight of the relative entropy regularization. By the KKT condition of (7), the optimal candidate policy has the form

$$(ajs) = \frac{(ajs) \ \exp \ \frac{(s;a)A}{(s)}}{E_{a^0} \ [\exp(\ (s;a^0)A \ (s;a^0)= \ (s))]} : \qquad (8)$$

Notice that the optimal candidate policy is a relative softmax policy, which is a common solution policy for many entropy-regularized RL algorithms [Haarnoja et al., 2018]. Intuitively, when the mixing factor vanishes (i.e., $(s;a) = 0$), the candidate policy equals to the behavior policy, and with confidence we obtain the candidate policy by modifying the behavior policy via exponential twisting.

The Closed-form Confidence . Given candidate policy , we derive efficient scheme for computing the confidence that solves the MM problem $\max_2 L_{\ ;\ ;\ }$. Recall that this optimization can be reformulated as a concave quadratic program (QP) with linear equality constraints, which has a unique optimal solution [Faybusovich and Moore, 1997]. However, since the decision variable (i.e., the confidence mapping) is infinite-dimensional, solving this QP is intractable without some assumptions about this mapping, To

[1] For example, when is state-dependent (which automatically satisfies the equality constraints in), the linear objective in vanilla CPI makes the optimal value $(\ js)$ a 0-1 vector. Especially when $\frac{2}{1} E_{s \ d} \left[\sqrt{\frac{1}{2} D_{KL}(\ (s) k \ (s))} \right]$ is large, then most entries of become zero, i.e., will be very close to .

resolve this issue, instead of using the surrogate objective $L_{\pi,\mu}$ in MM, we turn to its sample-based estimate. Specifically, given a batch of data $\mathcal{B} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^{|\mathcal{B}|}$ generated by the behavior policy, denote by

$$\bar{L}^{0}_{\pi,\mu} := \frac{1}{1-\gamma}\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|} \rightarrow_i \left(\left(\quad\right) A\right)_i$$

$$\bar{L}^{00}_{\pi,\mu} := \frac{1}{1-\gamma}\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|} \rightarrow_i \quad j \quad j_i$$

$$\bar{L}^{000}_{\pi,\mu} := \frac{1}{1-\gamma}\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|} \rightarrow_i \left(j \quad j \; jA \; j\right)_i$$

the sample-average approximation (SAA) of functions $L^{0}_{\pi,\mu}$, $L^{00}_{\pi,\mu}$, and $L^{000}_{\pi,\mu}$ respectively, where

$$\left(\quad\right)A = \{\left(\quad js_i\right) \quad \left(js_i\right)A \; (s_i, \cdot)\}_{s_i \in \mathcal{B}}; \tag{9}$$

$$j \quad jjA \; j = \{j \; \left(js_i\right) \quad \left(js_i\right)jjA \; (s_i, \cdot)j\}_{s_i \in \mathcal{B}}; \tag{10}$$

and $j \quad j = \{j \; \left(js_i\right) \quad \left(js_i\right)j\}_{s_i \in \mathcal{B}}$ are $|A|\,|\mathcal{B}|$-dimensional vectors, where each element is generated by a state sample from $\mathcal{B}$, and $\bar{\quad} = \{\left(js_i\right)\}_{s_i \in \mathcal{B}}$ is a $|A|\,|\mathcal{B}|$-dimensional decision vector, where each $|A|$-dimensional element vector corresponds to the confidence w.r.t. state samples in $\mathcal{B}$. Since the expectation in $L^{0}_{\pi,\mu}$, $L^{00}_{\pi,\mu}$, and $L^{000}_{\pi,\mu}$ is over the stationary distribution induced by the behavior policy, all the SAA functions are unbiased Monte-Carlo estimates of their population-based counterparts. We now define $\bar{L}_{\pi,\mu} := \bar{L}^{0}_{\pi,\mu} \quad \bar{L}^{00}_{\pi,\mu} \quad \bar{L}^{000}_{\pi,\mu}$ as the SAA-MM objective and use this to solve for the confidence vector $\bar{\quad}$ over the batch samples.

Now consider the following maximization problem:

$$\max_{\bar{\quad} \in \bar{\quad}} h\left(\quad\right) A \; ; \bar{\quad}i \quad \frac{}{|\mathcal{B}|(1-\quad)} h j \quad j; \bar{\quad}i \; hj \quad jjA \; j; \bar{\quad}i; \tag{11}$$

where the feasible set $\bar{\quad} = \{\quad \in [0,1]^{|\mathcal{B}|} : \sum_{a \in A} (s_i, a)\left(\left(a|s_i\right) \quad \left(a|s_i\right)\right) = 0, \forall i \in \{1, \ldots, |\mathcal{B}|\}\}$ only imposes constraints on the states that appear in the batch $\mathcal{B}$.

This finite-dimensional QP problem can be expressed in the following quadratic form:

$$\max_{\bar{\quad} \in \bar{\quad}} \rightarrow^{\top} \left(\quad\right) A \quad \frac{1}{2} \rightarrow^{\top} \bar{\quad};$$

where the symmetric matrix is given by

$$; := \frac{\left(D_{jA \; j} \quad ; + \quad ; \quad D^{\top}_{jA \; j}\right)}{|\mathcal{B}|(1-\quad)};$$

$; = j \quad j \; j \quad j^{\top}$, and $D_{jA \; j} = \mathrm{diag}(\{j \; A \; jg_{a \in A; s \in \mathcal{B}})$ is a $|\mathcal{B}|\,|A| \times |\mathcal{B}|\,|A|$-diagonal matrix whose elements are the absolute advantage function. By definition, $$ is positive-semi-definite, hence the QP above is concave. Using its KKT condition, the unique optimal confidence vector over batch $\mathcal{B}$ is given as

$$\bar{\quad} = \min\{1, \max\{0, \quad ;^{-1}\left(\left(\quad\right) A \; + M^{\top}_{\quad} \quad ; \quad ; \quad g\right)\}, \tag{12}$$

where $M_{\quad} = \mathrm{blkdiag}(\{[\quad (a|x) \quad (a|x)]_{a \in A} g_{x \in \mathcal{B}})$ is a $|\mathcal{B}|\,|A| \times |\mathcal{B}|\,|A| \; |\mathcal{B}|$-matrix, and the Lagrange multiplier $\quad ; \in \mathbb{R}^{|\mathcal{B}|}$ w.r.t. constraint $M_{\quad} \; \bar{\quad} = 0$ is given by

$$\quad ; = \left(M^{\top}_{\quad} \quad ;^{-1} M_{\quad}\right)^{-1}\left(M^{\top}_{\quad} \quad ;^{-1}\left(\quad\right) A\right): \tag{13}$$

We first construct the confidence function (s,a) from the confidence vector $\bar{\quad}$ over $\mathcal{B}$, in the following tabular fashion:

$$(s,a) = \begin{cases} \bar{\quad}_{s,a} & \text{if } (s,a) \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}.$$

While this construction preserves optimality w.r.t. the CPI objective (2), it may be overly conservative, because the policy equates to the behavior policy by setting $= 0$ at state-action pairs that are not in $\mathcal{B}$ (i.e., no policy improvement). To alleviate this conservatism, we propose to learn a confidence function that generalizes to out-of-distribution samples.

Learning the Confidence. Given a confidence vector $\bar{\quad}$ corresponding to samples in batch $\mathcal{B}$, we learn the confidence function (s,a) in supervised fashion. To ensure that the confidence function satisfies the constraint: $\sum_a (s,a)\left(\left(a|s\right) \quad \left(a|s\right)\right) = 0$, i.e., $(s,a) \in [0,1], \forall s,a$[2], we parameterize it as

$$(s,a) := \frac{\left(a|s\right) \quad \left(a|s\right)}{\left(a|s\right) \quad \left(a|s\right)}, \; \forall (s,a) \in S \quad A, \tag{14}$$

where $\quad 2 \quad$ is a learnable policy mapping, such that $\min\{\left(a|s\right), \left(a|s\right)\} \quad \left(a|s\right) \quad \max\{\left(a|s\right), \left(a|s\right)\}$, $\forall s,a$. We then learn $$ via the following KL distribution-fitting objective [Rusuet al., 2015]:

$$\min_{} \frac{1}{|\mathcal{B}|} \sum_{(s,a) \in \mathcal{B}} \left(a|s\right) \log \frac{\left(a|s\right)}{(1 - \bar{\quad}_{s,a})\left(a|s\right) + \bar{\quad}_{s,a} \quad \left(a|s\right)}.$$

While this approach learns $$ by generalizing the confidence vector to out-of-distribution samples, when $$ is a NN, one challenge is to enforce the constraint $\min\{\left(a|s\right), \left(a|s\right)\} \quad \left(a|s\right) \quad \max\{\left(a|s\right), \left(a|s\right)\}, \forall s,a$. Instead, using an in-graph convex optimization NN [Amos and Kolter, 2017], we parameterize $$ with a NN with the following constraint-projection layer $\quad : S \rightarrow A$ before the output:

$$\left(\quad s\right) \in \arg\min_{\quad \in \mathbb{R}^{|A|}} \frac{1}{2} \sum_{a \in A} k \quad a \quad e_{s,a} k^2;$$

$$\text{s.t.} \sum_a \quad a\left(\left(a|s\right) \quad \left(a|s\right)\right) = 0; \quad 0 \quad \quad 1; \tag{15}$$

where, at any $s \in S$, the $|A|$-dimensional confidence vector label $\{e_{s,a}\}_{a \in A}$ is equal to $\{\bar{\quad}_{s,a}\}_{a \in A}$ chosen from the batch confidence vector $\bar{\quad}$ such that s in $\mathcal{B}$ is closest to s. Indeed, analogous to the closed-form solution in (12), this projection layer has a closed-form QP formulation with linear constraints: $\left(\quad s\right) = \min\{1, \max\{0, e_{s,} + \left(\quad js\right) \quad \left(js\right)\) \quad ; \quad g\}$, where Lagrange multiplier $\quad ;$ is given by $\quad ; = \left(\quad js\right) \quad \left(js\right)\right)^{\top} e_{s,} = k \quad js\right) \quad \left(js\right)k^2$.

Although the $$-update is theoretically justified, in practice, when the magnitude of (s) becomes large (due to the conservatism of the weighted Pinsker inequality), the relative-softmax candidate policy (8) may be too close to the behavior policy $$, impeding learning of the residual policy (i.e.,). To avoid this in practice, we can upper bound the temperature, i.e., $(s) \quad \min\{ _{\max}; \; (s)\}$, or introduce a weak temperature-decay schedule, i.e., $(s) \quad (s) \quad ^{k}$, with a tunable $\in [0,1)$.

[2] If one restricts $$ to be only state-dependent, this constraint immediately holds.

Environment, ε	DQN		BRPO-C		BRPO (ours)		BCQ		KL-Q		SPIBB		BC		Behavior Policy
Acrobot-0.05	-91.2	9.1	-94.6	3.8	-91.9	9.0	-96.9	3.7	-93.0	2.6	-103.5	24.1	-102.3	5.0	-103.9
Acrobot-0.15	-83.1	5.2	-91.7	4.0	-86.1	10.1	-97.1	3.3	-92.1	3.2	-91.1	44.8	-113.1	5.6	-114.3
Acrobot-0.25	-83.4	3.9	-91.2	4.1	-85.3	4.8	-96.7	3.1	-90.0	2.9	-86.0	5.8	-124.1	7.0	-127.2
Acrobot-0.50	-84.3	22.6	-90.9	3.4	-83.7	16.6	-77.8	13.5	-84.5	3.8	-106.8	102.7	-173.7	8.1	-172.4
Acrobot-1.00	-208.9	174.8	-156.8	22.0	-121.7	10.2	-236.0	85.6	-227.5	148.1	-184.8	150.2	-498.3	1.7	-497.3
CartPole-0.05	82.7	0.5	220.8	117.0	336.3	122.6	255.4	11.1	323.0	13.5	28.8	1.2	205.6	19.6	219.1
CartPole-0.15	299.3	133.5	305.6	95.2	409.9	64.4	255.3	11.4	357.7	84.1	137.7	11.7	151.6	27.5	149.5
CartPole-0.25	368.5	129.3	405.1	74.4	316.8	64.1	247.4	128.7	441.4	79.8	305.2	119.7	103.0	20.4	101.9
CartPole-0.50	271.5	52.0	358.3	114.1	433.8	93.5	282.5	111.8	314.1	107.0	310.4	128.0	39.7	5.1	37.9
CartPole-1.00	118.3	0.3	458.6	51.5	369.0	42.3	194.0	25.1	209.7	48.4	147.1	0.1	22.6	1.5	21.9
LunarLander-0.05	-236.4	177.6	35.6	61.7	88.2	32.0	81.5	14.9	84.4	26.3	-200.4	81.7	75.8	17.7	73.7
LunarLander-0.15	-215.6	140.4	79.6	29.7	103.9	49.8	80.3	16.8	61.4	39.0	86.1	73.3	76.4	16.6	84.9
LunarLander-0.25	2.5	101.3	109.5	40.7	141.6	11.0	83.5	14.6	78.7	48.8	166.0	90.6	57.9	13.1	57.3
LunarLander-0.50	-104.6	68.3	42.5	71.4	101.0	39.6	-13.2	44.9	66.2	78.0	-134.6	17.1	-32.6	6.5	-36.0
LunarLander-1.00	-65.6	45.9	53.5	44.1	81.8	42.1	-69.1	44.0	-139.2	29.1	-107.1	94.4	-177.4	13.1	-182.6

Table 1: The mean and st. dev. of average return with the best hyperparameter con guration (with the top-2 results boldfaced). Full training curves are given in the appendix. For BRPO-C, the optimal con dence parameter is found by grid search.

Algorithm 1 BRPO algorithm

Require: B: batch data; Tunable parameter $\in [0, 1]$

1: for $t = 1, \ldots, N$ do
2: Sample mini-batch of transitions (s, a, r, d, s') from B
3: Compute $\bar{\alpha}$ from Eq. (12)
4: Update con dence β by Eq. (15)
5: Update candidate policy π_ω by Eq. (8)
6: Construct target critic network $V_{\omega'}(s') := (1 - \beta) \mathbb{E}_{a \sim \pi_\omega}[Q_{\omega'}(s', a')] + \beta \max_{a \sim \pi_\omega} Q_{\omega'}(s', a')$
7: Update $\omega \leftarrow \arg\max_\omega \frac{1}{2}(Q_\omega(s, a) - r - \gamma V_{\omega'}(s'))^2$
8: Update target network: $\omega' \leftarrow \tau\omega + (1 - \tau)\omega'$

6 Experimental Results

To illustrate the effectiveness of BRPO, we compare against six baselines: DQN [Mnih et al., 2013], discrete BCQ [Fujimoto et al., 2019], KL-regularized Q-learning (KL-Q) [Jaques et al., 2019], SPIBB [Laroche and Trichelair, 2017], Behavior Cloning (BC), [Kober and Peters, 2010], and BRPO-C, which is a simpli ed version of BRPO that uses a constant (tunable) parameter as con dence weight.[3] We do not consider ensemble models, thus do not include methods like BEAR [Kumar et al., 2019] among our baselines. CPI is also excluded since it is subsumed by BRPO-C with a grid search on the con dence. It is also generally inferior to BRPO-C because candidate policy learning does not optimize the performance of the nal mixture policy. We evaluated on three discrete-action OpenAI Gym tasks [Brockman et al., 2016]: Cartpole-v1, Lunarlander-v2, and Acrobot-v1.

The behavior policy in each environment is trained using standard DQN until it reaches 75% of optimal performance, similar to the process adopted in related work (e.g., [Fujimoto et al., 2019]). To assess how exploration and the quality of behavior policy affect learning, we generate ve sets of data for each task by injecting different random exploration into the same behavior policy. Speci cally, we add ε-greedy exploration for $\varepsilon = 1$ (fully random), 0.5, 0.25, 0.15, and 0.05, generating 100K transitions each for batch RL training.

All models use the same architecture for a given environment—details (architectures, hyper-parameters, etc.) are described in the appendix of the extended paper. While training is entirely of ine, policy performance is evaluated online using the simulator, at every 1000 training iterations. Each measurement is the average return with 100 evaluation episodes and 5 random seeds, and results are averaged over a sliding window of size 10.

Table 1 shows the average return of BRPO and the other baselines under the best hyper-parameter con gurations in each task setting. Behavior policy performance decreases as ε increases, as expected, and BC matches that very closely. DQN performs poorly in the batch setting. Its performance improves as ε increases from 0.05 to 0.25, due to increased state-action coverage, but as ε goes higher (0.5, 1.0), the state space coverage decreases again since the (near-) random policy is less likely to reach a state far away from the initial state.

BCQ, KL-Q and SPIBB follow the behavior policy in some ways, and showing different performance characteristics over the data sets. The underperformance relative to BRPO is more prominent for very low or very high ε, suggesting deficiency due to overly conservative updates or following the behavior policy too closely, when BRPO is able to learn.

Since BRPO exploits the statistics of each (s, a) pair in the batch data, it achieves good performance in almost all scenarios, outperforming the baselines. The stable performance and robustness across various scenarios make BRPO an appealing algorithm for batch/of ine RL in real-world, where it is usually dif cult to estimate the amount of exploration required prior to training, given access only to batch data.

7 Concluding Remarks

We have presented Batch Residual Policy Optimization (BRPO) for learning residual policies in batch RL settings. Inspired by CPI, we derived learning rules for jointly optimizing both the candidate policy and state-action dependent con dence mixture of a residual policy to maximize a conservative lower bound on policy performance. BRPO is thus more exploitative in areas of state space that are well-covered by the batch data and more conservative in others. While we have shown successful application of BRPO to various benchmarks, future work includes deriving nite-sample analysis of BRPO, and applying BRPO to more practical batch domains (e.g., robotic manipulation, recommendation systems).

[3] For algorithms designed for online settings, we modify data collection to sample only from of ine / batch data.

References

[Abbasi-Yadkori et al., 2019] Y. Abbasi-Yadkori, P. Bartlett, X. Chen, and A. Malek. Large-scale markov decision problems via the linear programming dual. arXiv:1901.01992, 2019.

[Amos and Kolter, 2017] B. Amos and Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In ICML, pages 136–145, 2017.

[Bolley and Villani, 2005] F. Bolley and C. Villani. Weighted csiszár-kullback-pinsker inequalities and applications to transportation inequalities. Annales de la Faculté des sciences de Toulouse: Mathématiques, volume 14, pages 331–352, 2005.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. Convex optimization. Cambridge university press, 2004.

[Brockman et al., 2016] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[Farias and Roy, 2003] P. De Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. Operations research, 51(6):850–865, 2003.

[Faybusovich and Moore, 1997] L. Faybusovich and J. Moore. In nite-dimensional quadratic optimization: interior-point methods and control applications. Applied Mathematics and Optimization, 36(1):43–66, 1997.

[Fujimoto et al., 2018] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. arXiv:1812.02900, 2018.

[Fujimoto et al., 2019] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. arXiv:1910.01708, 2019.

[Ghavamzadeh et al., 2016] M. Ghavamzadeh, M. Petrik, and Y. Chow. Safe policy improvement by minimizing robust baseline regret. In NIPS, 2016.

[Gretton et al., 2007] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel approach to comparing distributions. In AAAI, 2007.

[Haarnoja et al., 2018] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. ICML, 2018.

[Hasselt et al., 2016] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-learning. In AAAI, 2016.

[Hunter and Lange, 2004] D. Hunter and K. Lange. A tutorial on MM algorithms. The American Statistician, 58(1):30–37, 2004.

[Jaques et al., 2019] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. arXiv:1907.00456, 2019.

[Johannink et al., 2019] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In ICRA, pages 6023–6029. IEEE, 2019.

[Kakade and Langford, 2002] S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In ICML, pages 267–274, 2002.

[Kober and Peters, 2010] J. Kober and J. Peters. Imitation and reinforcement learning. IEEE Robotics & Automation Magazine, 17(2):55–62, 2010.

[Kumar et al., 2019] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. arXiv:1906.00949, 2019.

[Lange et al., 2012] S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In Reinforcement learning, pages 45–73. Springer, 2012.

[Laroche and Trichelair, 2017] R. Laroche and P. Trichelair. Safe policy improvement with baseline bootstrapping. arXiv:1712.06924, 2017.

[Lillicrap et al., 2015] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arXiv:1509.02971, 2015.

[Mahmood et al., 2014] A. Mahmood, H. van Hasselt, and R. Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In NIPS, pages 3014–3022, 2014.

[Mnih et al., 2013] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv:1312.5602, 2013.

[Nachum et al., 2019] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. CoRL, 2019.

[Pirotta et al., 2013] Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe policy iteration. In ICML, pages 307–315, 2013.

[Rusu et al., 2015] A. Rusu, S. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. arXiv:1511.06295, 2015.

[Schulman et al., 2015] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In ICML, pages 1889–1897, 2015.

[Shapiro et al., 2009] A. Shapiro, D. Dentcheva, and A. Ruszczyński. Lectures on stochastic programming: modeling and theory. SIAM, 2009.

[Silver et al., 2018] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. arXiv:1812.06298, 2018.

[Xu and Yin, 2013] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM Journal on imaging sciences, 6(3):1758–1789, 2013.

A Proofs for Results in Section 3

A.1 Proof of Lemma 1

Before going into the derivation of this theorem, we first have the following technical result that studies the distance of the occupation measures that are induced by and .

Lemma 5. The following expression holds for any state-next-state pair (s, s'):

$$(I - \gamma T_{\pi})^{-1} - (I - \gamma T_{\tilde{\pi}})^{-1}(s'|s) = \gamma (I - \gamma T_{\pi})^{-1} T_{\pi;\tilde{\pi}} (I - \gamma T_{\tilde{\pi}})^{-1}(s'|s),$$

where $T_{\pi}(s'|s)$ and $T_{\tilde{\pi}}(s'|s)$ represent the transition probabilities from state to next-state following policy and respectively, and for any state-next-state pair (s, s'), $T_{\pi;\tilde{\pi}}(s'|s) = \sum_{a \in A} T(s'|s,a)\, \tilde{\pi}(a|s) \left(\pi(s,a) \frac{\pi(a|s) - \tilde{\pi}(a|s)}{\tilde{\pi}(a|s)}\right)$.

Proof. Consider the following chain of equalities from matrix manipulations:

$$\begin{aligned}
(I - \gamma T_{\pi})^{-1} - (I - \gamma T_{\tilde{\pi}})^{-1} &= (I - \gamma T_{\pi})^{-1} \left(I - \sum_{a \in A} T(s'|s,a)\left(\tilde{\pi}(a|s) + \pi(s,a)\left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right)\right)_{s;s'} \\
&= (I - \gamma T_{\pi})^{-1} \left(\sum_{a \in A} T(s'|s,a)\, \pi(s,a)\left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right)_{s;s' \in S}.
\end{aligned}$$

By multiplying the matrix $(I - \gamma T_{\tilde{\pi}})^{-1}$ on both sides of the above expression, it implies that

$$(I - \gamma T_{\pi})^{-1} - (I - \gamma T_{\tilde{\pi}})^{-1} = \left((I - \gamma T_{\pi})^{-1} \sum_{a \in A} T(s'|s,a)\, \pi(s,a)\left(\pi(a|s) - \tilde{\pi}(a|s)\right)\right)_{s;s' \in S} (I - \gamma T_{\tilde{\pi}})^{-1}.$$

Using the definition of $T_{\pi;\tilde{\pi}}$ completes the proof of this lemma. □

Using the above result, for any initial state $s \in S$, the value functions that are induced by and have the following relationship:

$$\begin{aligned}
V_{\pi}(s) - V_{\tilde{\pi}}(s) &= \sum_{s_0 = s} (I - \gamma T_{\pi})^{-1} R - \sum_{s_0 = s} (I - \gamma T_{\tilde{\pi}})^{-1} R \\
&= \sum_{s_0 = s} \left[(I - \gamma T_{\pi})^{-1} - (I - \gamma T_{\tilde{\pi}})^{-1} \right] R + \sum_{s_0 = s} (I - \gamma T_{\tilde{\pi}})^{-1} (R - R) \\
&= E_{T;\tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \, \delta_{\pi}(s_t, a_t) \, \frac{\pi(a_t|s_t) - \tilde{\pi}(a_t|s_t)}{\tilde{\pi}(a_t|s_t)} \, V_{\pi}(s_{t+1}) \,\middle|\, s_0 = s \right] \\
&\quad + E_{T;\tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \, \delta_{\pi}(s_t, a_t) \, \frac{\pi(a_t|s_t) - \tilde{\pi}(a_t|s_t)}{\tilde{\pi}(a_t|s_t)} \, R(s_t, a_t) \,\middle|\, s_0 = s \right] \\
&= E_{T;\tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \, \delta_{\pi}(s_t, a_t) \, \frac{\pi(a_t|s_t) - \tilde{\pi}(a_t|s_t)}{\tilde{\pi}(a_t|s_t)} \, Q_{\pi}(s_t, a_t) \,\middle|\, s_0 = s \right] \\
&= E_{T;\tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \, \delta_{\pi}(s_t, a_t) \, \frac{\pi(a_t|s_t) - \tilde{\pi}(a_t|s_t)}{\tilde{\pi}(a_t|s_t)} \, A_{\pi}(s_t, a_t) \,\middle|\, s_0 = s \right].
\end{aligned} \tag{16}$$

The second equality follows from the fact that $Q_{\pi}(s,a) = R(s,a) + \sum_{s' \in S} T(s'|s,a) V_{\pi}(s')$. The third equality follows from the result in Lemma 5 and the fact that for any state $s \in S$, $(I - \gamma T_{\tilde{\pi}})^{-1} R(s) = V_{\tilde{\pi}}(s)$. The last equality is based on the fact of the confidence constraint that

$$E_{T;\tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t V_{\pi}(s_t) \sum_{a} \delta_{\pi}(s_t, a) \left(\tilde{\pi}(a|s_t) - \tilde{\pi}(a|s_t)\right) \,\middle|\, s_0 = s \right] = 0, \; \forall s \in S.$$

A.2 Proof of Theorem 2

Denote by V_{π} and $V_{\tilde{\pi}}$ the vectors of value functions $V_{\pi}(s)$ and $V_{\tilde{\pi}}(s)$ at every state $s \in S$ respectively. Re-writing the result in (16) in matrix form, it can be expressed as

$$V_{\pi} - V_{\tilde{\pi}} = (I - \gamma T_{\pi})^{-1} \left(R_{\pi;\tilde{\pi}} + \gamma T_{\pi;\tilde{\pi}} V_{\pi} + \gamma T_{\pi;\tilde{\pi}} (V_{\pi} - V_{\tilde{\pi}}) \right);$$

where for any state $s \in S$, $R_{\pi,\pi}(s) = \sum_a \pi(a|s) \frac{\pi(a|s) - \pi(a|s)}{\pi(a|s)} R(s,a)$, and $T_{\pi,\pi}(s'|s) = \sum_{a \in A} T(s'|s,a) \pi(a|s) \frac{\pi(a|s) - \pi(a|s)}{\pi(a|s)}$. This expression implies that

$$(I - (I - T)^{-1} T_{\pi,\pi})(V - V) = (I - T)^{-1}(R_{\pi,\pi} + T_{\pi,\pi} V);$$

which further implies that

$$V - V = (I - (I - T)^{-1} T_{\pi,\pi})^{-1}(I - T)^{-1}(R_{\pi,\pi} + T_{\pi,\pi} V):$$

Here based on the definition of $T_{\pi,\pi}$ and the confidence constraint, one can show that $(T + T_{\pi,\pi})$ is a stochastic matrix (all the elements are non-negative, and $\sum_{s' \in S}(T + T_{\pi,\pi})(s'|s) = 1, \forall s \in S$). Therefore the matrix $(I - (I - T)^{-1} T_{\pi,\pi})$ is invertible.

Using the matrix inversion lemma, one has the following equality:

$$(I - (I - T)^{-1} T_{\pi,\pi})^{-1} = (I - (T + T_{\pi,\pi}))^{-1}(I - T):$$

Therefore the difference of value function V and V can further be expressed as

$$V - V = (I - T - T_{\pi,\pi})^{-1}(R_{\pi,\pi} + T_{\pi,\pi} V):$$

In other words, at any state $s \in S$, the corresponding value function $V(s)$ is given by the following expression:

$$\begin{aligned} V(s) - V(s) &= E\left[\sum_{t=0}^{\infty} \gamma^t (R_{\pi,\pi} + T_{\pi,\pi} V)(s_t) \mid T'_{\pi,\pi}; s_0 = s\right] \\ &= E\left[\sum_{t=0}^{\infty} \gamma^t Q_{\pi,\pi}(s_t, a_t) \mid T'_{\pi,\pi}; s_0 = s\right] \\ &= E\left[\sum_{t=0}^{\infty} \gamma^t A_{\pi,\pi}(s_t, a_t) \mid T'_{\pi,\pi}; s_0 = s\right]; \end{aligned} \tag{17}$$

where the transition probability $T'_{\pi,\pi}(s'|s) = (T + T_{\pi,\pi})(s'|s)$ is given by $\sum_{a \in A} \pi(a|s) T(s'|s,a) \left(1 + \frac{\pi(a|s) - \pi(a|s)}{\pi(a|s)}\right)$ at state-next-state pair (s, s'). By noticing that $T'_{\pi,\pi}(s'|s)$ is indeed $T(s'|s)$ (the transition probability that is induced by residual policy π), the proof of Theorem 2 is completed.

B Proofs for Results in Section 4

B.1 Proof of Theorem 4

De ne the state-action discounted stationary distribution w.r.t. an arbitrary policy as $d(s,a) = (1-\gamma)\sum_{t=0}^{1}\gamma^t P(s_t = s, a_t = a|s_0, P_0, \pi)$ and its state-only counterpart as $d(s) = \sum_{a\in A} d(s,a)\pi(a|s)$. Immediately one can write the difference of return (objective function of this problem) with the following chain of equalities/inequalities:

$$\begin{aligned}
&E\left[\sum_{t=0}^{1}\gamma^t A_{\pi,\pi}(s_t) \mid T, \pi, s_0 \sim P_0\right] = \frac{1}{1-\gamma}\sum_{s\in S} d(s) A_{\pi,\pi}(s) \\
&= \frac{1}{1-\gamma}\sum_{s\in S} d(s) A_{\pi,\pi}(s) + \left(\frac{1}{1-\gamma}\sum_{s\in S} d(s) A_{\pi,\pi}(s) - \frac{1}{1-\gamma}\sum_{s\in S} d(s) A_{\pi,\pi}(s)\right) \\
&= \frac{1}{1-\gamma}\sum_{s\in S} d(s) A_{\pi,\pi}(s) + \left(\sum_{s\in S} d(s) A_{\pi,\pi}(s) - \sum_{s\in S} d(s) A_{\pi,\pi}(s)\right).
\end{aligned}$$

Recall that $T_{\pi,\pi}(s'|s) = \sum_{a\in A} T(s'|s,a)\pi(a|s)\frac{\pi(a|s)}{\pi(a|s)}$. At any state $s \in S$, the difference of stationary distribution $d(s) - d(s)$ can be further expressed as

$$\begin{aligned}
(d - d)(s) &= P_0^{\top}(I - T)^{-1}(I - T)^{-1}(s) \\
&= P_0^{\top}(I - T)^{-1} + (I - T)^{-1}T_{\pi,\pi}(I - (I - T)^{-1}T_{\pi,\pi})^{-1} \\
&\quad (I - T)^{-1}(I - T)^{-1}(s) \\
&= P_0^{\top}(I - T)^{-1}T_{\pi,\pi}(I - (I - T)^{-1}T_{\pi,\pi})^{-1}(I - T)^{-1}(I - T)^{-1}(s).
\end{aligned}$$

Let $D = (1-\gamma)E[\sum_{t=0}^{1}\gamma^t P(s_t = s'|s_0 = s, \pi)]_{s,s'\in S}$ be the occupation measure matrix induced by π. Combining the above arguments one has

$$\begin{aligned}
&|\langle d - d, A_{\pi,\pi}\rangle| \\
&= |\langle P_0^{\top}(I - T)^{-1}T_{\pi,\pi}(I - (I - T)^{-1}T_{\pi,\pi})^{-1}(I - T)^{-1}, A_{\pi,\pi}\rangle| \\
&= \frac{1}{1-\gamma}|\langle P_0^{\top}D T_{\pi,\pi}((1-\gamma)I - D T_{\pi,\pi})^{-1}, D A_{\pi,\pi}\rangle| \\
&= \frac{1}{1-\gamma}|\langle P_0^{\top}D T_{\pi,\pi}, ((1-\gamma)I - D T_{\pi,\pi})^{-1}D A_{\pi,\pi}\rangle|.
\end{aligned} \tag{18}$$

Now $(1-\gamma)I + D T_{\pi,\pi}$ is a stochastic matrix, which is because for any state $s \in S$,

$$\begin{aligned}
((1-\gamma)I + D T_{\pi,\pi})e(s) &= (1-\gamma)E\left[\sum_{t=0}^{1}\gamma^t \sum_{s'} T_{\pi,\pi}(s'|s) \mid T, \pi, s_0 = s\right] \\
&= (1-\gamma)E\left[\sum_{t=0}^{1}\gamma^t \sum_{a\in A} (s,a)(\pi(a|s) - \pi(a|s)) \mid T, \pi, s_0 = s\right] \\
&= 0.
\end{aligned}$$

Using this property one can upper bound the magnitude of each element of the matrix $((1-\gamma)I + D T_{\pi,\pi})^{-1}$ by $\frac{1}{1-\gamma}$. Therefore, using Holder inequality one can further upper bound the expression in (18) as follows:

$$\begin{aligned}
&|\langle d - d, A_{\pi,\pi}\rangle| \\
&\le \frac{1}{(1-\gamma)^2} P_0^{\top}D T_{\pi,\pi}\|_1 \|D A_{\pi,\pi}\|_1 \\
&\le \frac{1}{(1-\gamma)^2}\sum_{s\in S,a\in A} d(s)\pi(a|s)(s,a)\frac{|\pi(a|s) - \pi(a|s)|}{\pi(a|s)} \max_{s_0\in S}\sum_{s\in S} d(s|s_0)|A_{\pi,\pi}|.
\end{aligned}$$

Plugging in the de nition of the discounted occupation measure w.r.t. into the above expression, the proof is this theorem is completed.

C Practical Implementation Details of BRPO

In this section, we discuss several practical techniques to further boost training stability and effectiveness of BRPO.

Improving CPI with Optimal Advantage The derivation in Section 4 and Section 5 shows that optimizing $L_{\pi,\beta}$ finds a residual policy that performs no-worse than the behavior policy (modulo any Lagrangian approximation error). While we argue that this optimization is less conservative than existing methods (like TRPO) due to the state-action-dependent learned confidence, it might not be aggressive enough in leveraging the function approximation to generalize to unseen state and action. One major reason is that by design $L_{\pi,\beta}$ only uses the long-term value of π (in the form of A^π), in order to circumvent the issue of bad generalization to unseen state-action pair. This also makes policy improvement local. This is a fundamental challenge of batch RL, but can be relaxed depending on the domain. As a remedy to this issue, by a convex ensemble of the results from Lemma 1 and Theorem 2 (with any combination weight $\alpha \in [0, 1]$), notice that the difference-return also satisfies

$$J(\pi') - J(\pi) \geq \frac{1}{1-\gamma} E^{\alpha}_{\pi',\pi,\beta} - \frac{(1-\alpha)\gamma}{1-\gamma} L^{\alpha}_{\pi',\pi}(\pi') - \max_{s_0 \in S} L^{\alpha}_{\pi',\pi}(\pi')(s_0),$$

where

$$E^{\alpha}_{\pi',\pi,\beta} := \mathbb{E}_{(s,a)\sim d_{\pi'}(s,a)} \frac{\pi'(a|s) - \pi(a|s)}{\beta(a|s)} W(s,a)$$

with a weighted advantage function $W := (1-\alpha)A^\pi + \alpha A^{\pi'}$. Therefore, without loss of generality one can replace $A^{\pi'}$ in $L_{\pi',\pi}$ with the weighted advantage function. Furthermore, to avoid estimating $A^{\pi'}$ at each policy update and assuming that CPI eventually finds $\pi' \approx \pi^*$, one may directly estimate the optimal weighted advantage function

$$W^*(s,a) = Q_{\pi,\pi}(s,a) - V_{\pi,\pi}(s),$$

in which the value function $Q_{\pi,\pi}$ is a Bellman fixed-point of $Q(s,a) = R(s,a) + \sum_{s'} T(s'|s,a) V(s')$, with

$$V(s) = (1-\alpha)\mathbb{E}_a[Q(s,a)] + \alpha \max_a Q(s,a).$$

This approach of combining the optimal Bellman operator with the on-policy counterpart belongs to the general class of hybrid on/off-policy RL algorithms of 2016combining.

Therefore, we learn an advantage function W that is a weighted combination of A^π and A^*. Using the batch data $\mathcal{B}$, the expected advantage A^* can be learned with any critic-learning technique, such as SARSA sutton2018reinforcement. We can learn A^* by DQN mnih2013playing or other Q-learning algorithm. We provide pseudo-code of our BRPO algorithm in Algorithm 1.

D Experimental Details

This section describes more details about our experimental setup to evaluate the algorithms.

D.1 Behavior policy

We train the behavior policy using DQN, using architecture and hyper-parameters speci ed in Section D.2. The behavior policy was trained for each task until the performance reaches around 75% of the optimal performance similar to [Fujimoto et al., 2018] and [Kumar et al., 2019]. Speci cally, we trained the behavior policy for 100,000 steps for Lunarlander-v2, and 50,000 steps for Cartpole-v1 and Acrobot-v1. We used two-layers MLP with FC(32)-FC(16). The replay buffer size is 500,000 and batch size is 64. The performance of the behavior policies are given in Table 1.

D.2 Hyperparameters

For fair comparison, we generally used the same set of hyper-parameters and architecture across all methods and experiments, which are de ned in Table 2 and Table 3. Similar to the behavior policy, we used two-layers MLP with FC(32)-FC(16) for all the critic agents and Behavioral cloning agent's policy. The nal hyperparameters are found using grid search, with candidate set speci ed in Table 2 and Table 3.

Hyperparameters for BC, BCQ, SARSA, DQN, and KL-Q	Sweep range	Final value
Soft target update rate ( )	-	0.5
Soft target update period	150, 500, 1500	500
Discount factor	-	0.99
Mini-batch size	-	64
Q-function learning rates	0.0003, 0.001, 0.002	0.001
Neural network optimizer	-	Adam
[BCQ] Behavior policy threshold (in [Fujimoto et al., 2019])	0.1, 0.2, 0.3, 0.4	0.3
[SPIBB] Bootstrapping set threshold	0.1, 0.2, 0.3, 0.4	0.2
[KL-Q] KL-regularization weight	0.01, 0.03, 0.1, 0.3	0.1

Table 2: The range of hyperparameters sweeped over and the nal hyperparameters used for the baselines (BC, BCQ, SARSA, DQN, and KL-Q).

Hyperparameters for BRPO and BRPO-C	Sweep range	Final value
Soft target update rate ( )	-	0.5
Soft target update period	150, 500, 1500	500
Discount factor	-	0.99
Mini-batch size	-	64
Q-function learning rates	0.0003, 0.001, 0.002	0.001
Neural network optimizer	-	Adam
[BRPO, BRPO-C] Mixing (for critic training)	0.1, 0.5, 0.9	0.9
[BRPO] Con dence ( ) learning rates	0.0002, 0.0001	0.0001
[BRPO-C] Constant (for constant residual policy)	0.25, 0.33, 0.5, 0.66, 0.75	0.5

Table 3: The range of hyperparameters sweeped over and the nal hyperparameters used for the proposed methods (BRPO and BRPO-C).

E Additional Results

Here are the learning curves for each environment and behavior policy over the course of batch training.