OPTIMIZING INFORMATION BOTTLENECK IN REINFORCEMENT LEARNING: A STEIN VARIATIONAL APPROACH

Anonymous authors
Paper under double-blind review

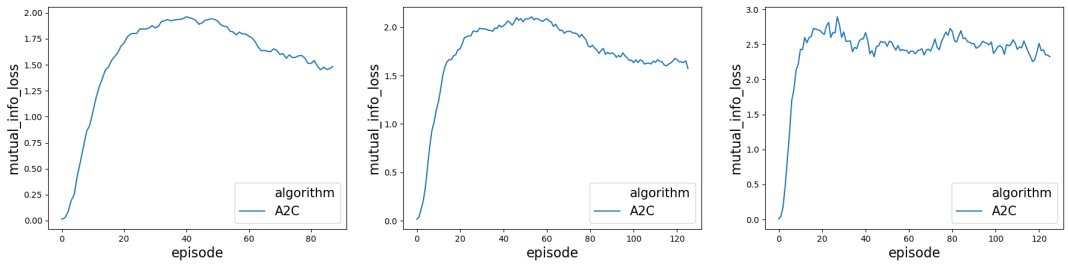
ABSTRACT

The information bottleneck (IB) principle is an elegant and useful learning framework for extracting relevant information that an input feature contains about the target. The principle has been widely used in supervised and unsupervised learning. In this paper, we investigate the effectiveness of the IB framework in reinforcement learning (RL). We first derive the objective based on IB in reinforcement learning, then we analytically derive the optimal conditional distribution of the optimization problem. Following the variational information bottleneck (VIB), we provide a variational lower bound using a prior distribution. Unlike VIB, we propose to utilize the amortized Stein variational gradient method to optimize the lower bound. We incorporate this framework in two popular RL algorithms: the advantageous actor critic algorithm (A2C) and the proximal policy optimization algorithm (PPO). Our experimental results show that our framework can improve the sample efficiency of vanilla A2C and PPO. We also show that our method achieves better performance than VIB and mutual information neural estimation (MINE), two other popular approaches to optimize the information bottleneck framework in supervised learning.

1 INTRODUCTION

In training a reinforcement learning model, an agent interacts with the environment, explores the (possibly unknown) state space, and learns a policy from the exploration sample data. Such samples may be quite expensive to obtain (e.g., requires interactions with the physical environment). Hence, improving the sample efficiency of the learning algorithm is a key problem in RL and has been studied extensively. In particular, an effective representation that has less redundant information can help improve the sample efficiency in RL. This can be seen from the following motivating example: According to the results of (Sygnowski & Michalewski, 2016), in the classical Atari game, Seaquest, it may require dozens of millions of samples to converge to an optimal policy when the input states are raw images (more than 28,000 dimensions), while it requires much less samples when the inputs are 128-dimension pre-defined RAM data. Clearly, the RAM data contains much less redundant information than the raw images does.

The information bottleneck (IB) framework (Tishby et al., 2000) is a popular framework for extracting relevant information that an input feature contains about the target and has been used extensively in supervised and unsupervised learning. In this paper, we adopt the information bottleneck framework in the context of RL, with the aim of producing more informative and less redundant representation and improving the sample efficiency. In the experiments in (Shwartz-Ziv & Tishby, 2017), one can see that during the training process in a standard supervised learning setting, the neural network first "remembers" the inputs by increasing the mutual information (MI) between the inputs and the representation variables, then compresses the inputs to efficient representation related to the learning task by discarding redundant information from inputs (decreasing the MI between inputs and representation variables). We call this phenomena "information extraction-compression process" (information E-C process). We verify empirically that such process also exists in some Atari games (in our experiments, we use A2C). We run simple experiments in three Atari games. Figure 1 shows the changes of MI between inputs and representations when training an RL agent (we need to use MINE (Belghazi et al., 2018) for MI estimation). We can see that the MI first increases, which



(a) MI in Pong (b) MI in MsPacman (c) MI in Assault

Figure 1: Visualizing the mutual information (MI) during the training in 3 Atari games. X-axis indicates the number of training steps and y-axis the MI.

implies that relevant information is extracted from the inputs, and then MI decreases, suggesting that the information is compressed (redundant information is discarded). This result is very similar in spirit to the results shown in (Shwartz-Ziv & Tishby, 2017). This interesting observation motivates us to adopt the IB framework in RL, in order to accelerate the extraction-compression process. The IB framework is intended to explicitly enforce RL agent to discard irrelevant information from raw input data, hence improving the sample efficiency. Our technical contributions can be summarized as follows:

1. We incorporate the information bottleneck (IB) principle to the training of reinforcement learning algorithms. We formulate the optimization problem of the IB framework, and prove a near-optimality property of our framework under certain conditions. (see Theorem 1).
2. We derive the optimal conditional distribution for the optimization problem (Theorem 2). Based on this, we construct a variational lower bound (similar to VIB (Alemi et al., 2016)). Then, we propose a new training algorithm, that leverages the amortized Stein Variational (SV) gradient method (Feng et al., 2017) to optimize the variational lower bound.
3. We conduct the experiments on 20 Atari games: Our experimental results show that actor-critic algorithms (A2C (Mnih et al., 2016) and PPO (Schulman et al., 2017)) with our framework are more sample efficient than their original versions.
4. Moreover, we show that our method outperforms VIB in optimizing IB in RL. In Appendix C.4, we find that using MINE (Belghazi et al., 2018) for optimizing IB is not suitable in RL. We discuss the possible reasons for this.

2 Related Work

Various information theoretic methods have been used in the context of RL. Here, we only mention some prior work which seek to maximize some kind of mutual information. Recall that the IB framework requires to minimize the MI between the inputs and representations under some constraints. (Oord et al., 2018) proposed Contrastive Predictive Coding (CPC) to maximize the MI between the current states and future states, which enables the model to predict the future states. We note that their method also gives a lower bound to estimate the MI. However, their bound is not tight and would yield a loose upper bound of IB framework, hence not suitable for optimizing IB framework. (Still & Precup, 2012; Kim et al., 2018; Kumar, 2018) also studied how to improve exploration of RL agents by maximizing the MI between states and action embedding representations.

As for applying IB principle in RL, (Peng et al., 2018) applied VIB (Alemi et al., 2016) in imitation learning and inverse reinforcement learning. (Wang et al., 2019) applied VIB in multi-agent communication. (Igl et al., 2019) applied VIB in Actor-Critic algorithms. However, VIB constrains the representations in the class of Gaussian distributions, while our optimization method can learn arbitrary distributions. (Abel et al., 2019) proposed to use IB in apprenticeship learning, and the method they used to optimize IB is Blahut-Arimoto algorithm (Tishby et al., 2000), which is difficult to apply in deep neural network. (Genewein et al., 2015; Sims, 2003; Leibfried & Braun, 2015) proposed to penalize regular RL objectives with an MI regularization term between states and actions, so that the policy can discard redundant information from the states. However, they only provided theoretical results. (Abdolmaleki et al., 2018; Grau-Moya et al., 2019; Leibfried & Grau-Moya, 2019)

applied the above theoretical results in deep RL and provided a variational lower bound like VIB (Alemi et al., 2016) to optimize the new objective. Yet because of the particularity of the policy distribution, they only needed to update Q function to implicitly optimize this lower bound. Their methods can not be used to solve general IB objectives, e.g., the IB objective in our paper.

3 PRELIMINARIES

A Markov Decision Process(MDP) is a tuple, $(\mathcal{X}, \mathcal{A}, R, \mathcal{P}, \mu)$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ is the set of actions, $R : X \times A \times X \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the transition probability function, where $\mathcal{P}(X^{'}|X, a)$ is the probability of transitioning to state $X^{'}$ given that the previous state is X and the agent took action a in X, and $\mu : \mathcal{X} \to [0, 1]$ is the starting state distribution. A policy $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is a probability function over actions and states, with $\pi(a|X)$ denoting the probability of choosing action a in state X.

In reinforcement learning, we aim to select a policy π which maximizes $\mathcal{R}(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(X_t, a_t, X_{t+1})]$, where we denote $R(X_t, a_t, X_{t+1}) = r_t$. Here $\gamma \in [0, 1)$ is a discount factor, τ denotes a trajectory $(X_0, a_0, X_1, a_1, ...)$. Define the state value function as $V^\pi(X) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r_t|X_0 = X]$, which is the expected return by policy π in state X. And the state-action value function $Q^\pi(X, a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t r_t|X_0 = X, a_0 = a]$ is the expected return by policy π after taking action a in state X.

In this paper, we consider actor-critic algorithms, which can take the advantage of both policy gradient methods and value-function-based methods, such as the well-known A2C algorithm (Mnih et al., 2016). Specifically, in the case that policy $\pi(a|X; \theta)$ is parameterized by θ, A2C uses the following as an approximation of the real policy gradient $\nabla_\theta \mathcal{R}(\pi)$:

$$\nabla_\theta J_{pg}(\theta) = \sum_{t=0}^{\infty} \nabla_\theta [\log \pi(a_t|X_t; \theta)(R_t - b(X_t)) + \alpha_2 H(\pi(\cdot|X_t))] = \sum_{t=0}^{\infty} \nabla_\theta J_{pg}(X_t; \theta), \quad (1)$$

where $\nabla_\theta J_{pg}(\theta)$ is the approximate policy gradient and $\nabla_\theta J_{pg}(X_t; \theta)$ is the approximate policy gradient on a specific state X_t, $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the accumulated return from time step t, $H(p)$ is the entropy of distribution p and $b(X_t)$ is a baseline function, which is commonly replaced by $V^\pi(X_t)$.

A2C also includes the minimization of the mean square error between R_t and value function $V^\pi(X_t)$. Thus in practice, the total objective function in A2C can be written as:

$$\begin{aligned} J(\theta) \approx & \sum_{t=0}^{\infty} \log \pi(a_t|X_t; \theta)(R_t - V^\pi(X_t)) + \alpha_2 H(\pi(\cdot|X_t)) \\ & - \alpha_1 \left\| R_t - V^\pi(X_t) \right\|^2 = \sum_{t=0}^{\infty} J(X_t; \theta), \end{aligned} \quad (2)$$

where α_1, α_2 are two coefficients.

In this paper, we use a neural network f to encode a state X to its representation Z. We use stochastic representation, i.e., we let $Z = f(X, \xi; \phi)$, where X is the state, ξ is a random noise and ϕ is the parameter of f. This naturally defines a probability distribution $Z \sim P_\phi(\cdot|X)$. A deterministic value function $V^\pi(.)$ takes either the state or its representation Z as the input. Hence, in the following, without causing any confusion, we write $V^\pi(X_t) = V^\pi(Z_t) \mid_{Z_t\sim P_\phi(\cdot|X_t)}$ (we can write $J(X_t; \theta)$ and $Q^\pi(X_t, a_t)$ similarly).

4 FRAMEWORK

4.1 INFORMATION BOTTLENECK IN REINFORCEMENT LEARNING

The information bottleneck framework is an information theoretical framework for extracting relevant information that an input $X \in \mathcal{X}$ contains about an output $Y \in \mathcal{Y}$. An optimal representation of X would capture the relevant factors and compress X by diminishing the irrelevant parts which do not

contribute to the prediction of Y. In a Markovian structure $X \to Z \to Y$ where X is the input, Z is representation of X and Y is the label of X, IB seeks an embedding distribution $P^\star(Z|X)$ such that:

$$\begin{aligned} P^\star(Z|X) &= \arg\max_{P(Z|X)} I(Y,Z) - \beta I(X,Z) \\ &= \arg\max_{P(Z|X)} H(Y) - H(Y|Z) - \beta I(X,Z) \\ &= \arg\max_{P(Z|X)} -H(Y|Z) - \beta I(X,Z), \end{aligned} \tag{3}$$

which appears as the standard cross-entropy loss[1] in supervised learning with a mutual information regularizer. Here β is a coefficient that controls the magnitude of the regularizer.

Next we derive an information bottleneck framework in reinforcement learning. Just like the label Y in the context of supervised learning as shown in Equation 3, we assume the supervising signal Y in RL to be the accurate value R_t of a specific state X_t for a fixed policy π, which can be approximated by an n-step bootstrapping function $Y_t = R_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V^\pi(Z_{t+n})$ in practice. Let $P(Y|Z)$ be the following distribution:

$$P(Y_t|Z_t) \propto \exp(-\alpha(Y_t - V^\pi(Z_t))^2). \tag{4}$$

This assumption is heuristic but reasonable: Suppose we have an input X_t, its relative label $Y_t = R_t$, and we now have X_t's representation Z_t. Naturally we want to train our decision function $V^\pi(Z_t)$ to approximate the true label Y_t. If we set our target distribution to be $C \cdot \exp(-\alpha(R_t - V^\pi(Z_t))^2)$, the probability decreases as $V^\pi(Z_t)$ gets far from Y_t while increases as $V^\pi(Z_t)$ gets close to Y_t.

For simplicity, we just write $P(R|Z)$ instead of $P(Y_t|Z_t)$ in the following context.

With this assumption, Equation equation 3 can be written as:

$$\begin{aligned} P^\star(Z|X) &= \arg\max_{P(Z|X)} \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X), R\sim P(R|Z)}[\log P(R|Z)] - \beta I(X,Z) \\ &= \arg\max_{P(Z|X)} \mathbb{E}_{X,R,Z}[-\alpha(R - V^\pi(Z))^2] - \beta I(X,Z). \end{aligned} \tag{5}$$

The first term is simply the mean squared error. In a network with representation parameter ϕ and policy-value parameter θ, policy loss $J_{pg}(Z;\theta)$ in Equation 1 and IB loss in Equation 5 can be jointly written as:

$$L(\theta,\phi) = \mathbb{E}_{X,Z}[\underbrace{J_{pg}(Z;\theta) + \mathbb{E}_R[-\alpha(R - V^\pi(Z;\theta))^2]}_{J(Z;\theta)}] - \beta I(X,Z;\phi), \tag{6}$$

where $I(X,Z;\phi)$ denotes the MI between X and $Z \sim P_\phi(\cdot|X)$. Notice that $J(Z;\theta)$ itself is a standard loss function in RL as shown in Equation 2. Finally we get the ultimate formalization of the IB framework in reinforcement learning:

$$P_{\phi^*}(Z|X) = \arg\max_{P_\phi(Z|X)} \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X)}[J(Z;\theta)] - \beta I(X,Z;\phi). \tag{7}$$

The following theorem shows that if the mutual information $I(X,Z)$ of our framework and an optimal RL policy are close, then our framework is near optimal.

Theorem 1 (Near-optimality theorem). *Policy $\pi^r = \pi_{\theta^r}$, parameter ϕ^r, an optimal policy $\pi^\star = \pi_{\theta^\star}$ and its representation parameter $\phi^\star$ are defined as follows:*

$$\theta^r, \phi^r = \arg\min_{\theta,\phi} \mathbb{E}_{P_\phi(X,Z)} \left[\log \frac{P_\phi(Z|X)}{P_\phi(Z)} - \frac{1}{\beta} J(Z;\theta) \right]; \tag{8}$$

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} \mathbb{E}_{P_\phi(X,Z)} \left[-\frac{1}{\beta} J(Z;\theta) \right]. \tag{9}$$

Define J^{π^r} as $\mathbb{E}_{P_{\phi^r}(X,Z)}[J(Z;\theta^r)]$ and $J^{\pi^\star}$ as $\mathbb{E}_{P_{\phi^\star}(X,Z)}[J(Z;\theta^\star)]$. Assume that there exists an $\epsilon > 0$, $|I(X,Z;\phi^\star) - I(X,Z;\phi^r)| < \frac{\epsilon}{\beta}$, we have $|J^{\pi^r} - J^{\pi^\star}| < \epsilon$.

The proof can be found in Appendix B.1. In practice, β is very small, e.g., 0.001. Hence, if the gap between MI is ϵ, then the gap between J^{π^r} and $J^{\pi^\star}$ is 0.001ϵ.

[1] Mutual information $I(X,Z)$ is defined as $\int dXdZP(X,Z)\log\frac{P(X,Z)}{P(X)P(Z)}$, conditional entropy $H(Y|Z)$ is defined as $-\int dYdZP(Y,Z)\log P(Y|Z)$. In a binary-classification problem, $-\log P(Y|Z) = -(1-Y)\log(1-\hat{Y}(Z)) - Y\log(\hat{Y}(Z))$, where $\hat{Y}$ is the function of discriminator.

4.2 TARGET DISTRIBUTION DERIVATION AND VARIATIONAL LOWER BOUND CONSTRUCTION

In the section we first derive the optimal target distribution in Equation 7, prove the optimality of this distribution. Then, we optimize the distribution by constructing a variational lower bound.

We would like to solve the optimization problem in Equation 7:

$$\max_{P_\phi(Z|X)} \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X)}[\underbrace{J(Z;\theta) - \beta \log P_\phi(Z|X)}_{L_1(\theta,\phi)} + \underbrace{\beta \log P_\phi(Z)}_{L_2(\phi)}]. \tag{10}$$

Notice that L_1 and L_2 are both contained in the expectation. With the derivative of L_1 and L_2, setting their summation to 0, we can see the optimal conditional probability as the following form:

$$P_\phi(Z|X) \propto P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta)) \tag{11}$$

We provide a rigorous derivation of Equation 11 in Section B.2 of the appendix. We note that though our derivation is over the representation space instead of the whole network parameter space, the optimization problem Equation 10 and the resulting distribution Equation 11 are quite similar to those studied in (Liu et al., 2017) in the context of Bayesian inference. However, we stress that our formulations follow from the information bottleneck framework, and are mathematically different from those in (Liu et al., 2017). In particular, the difference lies in the term L_2, which depends on the the distribution $P_\phi(Z \mid X)$ we need to optimize (while in (Liu et al., 2017), the corresponding term is a fixed prior).

The following theorem shows that the distribution in Equation 11 is an optimal target distribution (with respect to the IB objective L). The proof can be found in Section B.3 of the appendix.

Theorem 2. *(Representation Improvement Theorem) Suppose we are given a fixed policy-value parameter θ, representation distribution $P_\phi(Z|X)$ and state distribution $P(X)$. Consider the objective function $L(\theta,\phi) = \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X)}[J(Z;\theta)] - \beta I(X,Z;\phi)$. Define a new representation distribution: $P_{\hat{\phi}}(Z|X) \propto P_\phi(Z)\exp(\frac{1}{\beta}J(Z;\theta))$. We have $L(\theta,\hat{\phi}) \geq L(\theta,\phi)$.*

Though we have derived the optimal target distribution, it is still difficult to compute $P_\phi(Z)$. In order to resolve this problem, following VIB (Alemi et al., 2016), we construct a variational lower bound with a prior distribution $U(Z)$ which is independent of ϕ. Notice that $\int dZ P_\phi(Z)\log P_\phi(Z) \geq \int dZ P_\phi(Z)\log U(Z)$, which means that $\mathbb{E}_Z[\log P_\phi(Z)] \geq \mathbb{E}_Z[\log U(Z)]$. Now, we can derive a lower bound of $L(\theta,\phi)$ in Equation 6 as follows:

$$\begin{aligned} L(\theta,\phi) &= \mathbb{E}_{X,Z}[J(Z;\theta) - \beta\log P_\phi(Z|X)] + \beta\mathbb{E}_Z[\log P_\phi(Z)] \\ &\geq \mathbb{E}_{X,Z}[J(Z;\theta) - \beta\log P_\phi(Z|X)] + \beta\mathbb{E}_Z[\log U(Z)] \\ &= \mathbb{E}_{X,Z}[J(Z;\theta) - \beta\log P_\phi(Z|X) + \beta\log U(Z)] = \hat{L}(\theta,\phi), \end{aligned} \tag{12}$$

where $\hat{L}(\theta,\phi)$ denotes the new lower bound loss. According to Theorem 2, the target distribution that maximizes the lower bound is:

$$P_\phi(Z|X) \propto U(Z)\exp(\frac{1}{\beta}J(Z;\theta)). \tag{13}$$

4.3 OPTIMIZATION BY STEIN VARIATIONAL GRADIENT DESCENT

Next we utilize the method in (Feng et al., 2017; Haarnoja et al., 2017) to optimize the lower bound.

Stein variational gradient descent (SVGD) is a non-parametric variational inference algorithm that leverages efficient deterministic dynamics to transport a set of particles $\{Z_i\}_{i=1}^n$ to approximate given target distribution $Q(Z)$. We choose SVGD to optimize the lower bound because of its ability to handle unnormalized target distributions such as Equation 13.

Briefly, SVGD iteratively updates the "particles" $\{Z_i\}_{i=1}^n$ via a direction function $\Phi^\star(\cdot)$ in the unit ball of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$:

$$Z_i \leftarrow Z_i + \epsilon\Phi^\star(Z_i). \tag{14}$$

In Equation 14, $\Phi^*(\cdot)$ is chosen as a direction to maximally decrease [2] the KL divergence between the particles' distribution $P(Z)$ and the target distribution $Q(Z) = \frac{\hat{Q}(Z)}{C}$ ($\hat{Q}$ is the unnormalized distribution, C is the normalization constant) in the sense that

$$\Phi^\star \leftarrow \arg\max_{\phi \in \mathcal{H}} \left\{ -\frac{d}{d\epsilon} \mathbb{D}_{KL}(P_{[\epsilon\phi]} || Q) \ s.t. \ \|\Phi\|_{\mathcal{H}} \leq 1 \right\}, \tag{15}$$

where $P_{[\epsilon\Phi]}$ is the distribution of $Z + \epsilon\Phi(Z)$. (Liu & Wang, 2016) provided a closed form of this direction:

$$\Phi(Z) = \mathbb{E}_{Z_j \sim P}\big[\mathcal{K}(Z_j, Z)\nabla_{\hat{Z}} \log \hat{Q}(\hat{Z}) \mid_{\hat{Z}=Z_j} + \nabla_{\hat{Z}} \mathcal{K}(\hat{Z}, Z) \mid_{\hat{Z}=Z_j} \big], \tag{16}$$

where $\mathcal{K}$ is a kernel function (typically an RBF kernel function). Notice that C does not appear in the equation.

In our case, we seek to minimize

$$\mathbb{D}_{KL}\left(P_\phi(\cdot|X) \ || \ \frac{1}{C} U(\cdot) \exp(\frac{1}{\beta} J(\cdot; \theta)) \right),$$

where $C = \int dZ U(Z) \exp(\frac{1}{\beta} J(Z; \theta))$. This is equivalent to maximizing $\hat{L}(\theta, \phi)$ (defined in Equation 12). The greedy direction yields:

$$\Phi(Z) = \mathbb{E}_{Z_j}\big[\mathcal{K}(Z_j, Z)\nabla_{\hat{Z}} (\frac{1}{\beta} J(\hat{Z}; \theta) + \log U(\hat{Z})) \mid_{\hat{Z}=Z_j} + \nabla_{\hat{Z}} \mathcal{K}(\hat{Z}, Z) \mid_{\hat{Z}=Z_j} \big]. \tag{17}$$

In practice we replace $\log U(\hat{Z})$ with $\zeta \log U(\hat{Z})$ where ζ is a coefficient that controls the magnitude of $\nabla_{\hat{Z}} \log U(\hat{Z})$. Notice that $\Phi(Z)$ is the greedy direction that Z moves towards $\hat{L}(\theta, \phi)$ (defined in Equation 12)'s target distribution as shown in Equation 13 (distribution that maximizes $\hat{L}(\theta, \phi)$). This means $\Phi(Z)$ is the gradient of $\hat{L}(Z, \theta, \phi)$: $\frac{\partial \hat{L}(Z,\theta,\phi)}{\partial Z} \propto \Phi(Z)$.

Since our ultimate purpose is to update ϕ, by the chain rule, $\frac{\partial \hat{L}(Z,\theta,\phi)}{\partial \phi} \propto \Phi(Z)\frac{\partial Z}{\partial \phi}$. Then for $\hat{L}(\theta, \phi) = \mathbb{E}_{P_\phi(X,Z)}[\hat{L}(Z, \theta, \phi)]$:

$$\frac{\partial \hat{L}(\theta, \phi)}{\partial \phi} \propto \mathbb{E}_{X \sim P(X), Z \sim P_\phi(\cdot|X)} \left[\Phi(Z) \frac{\partial Z}{\partial \phi} \right], \tag{18}$$

$\Phi(Z)$ is given in Equation 17. In practice we update the policy-value parameter θ by common policy gradient algorithm since:

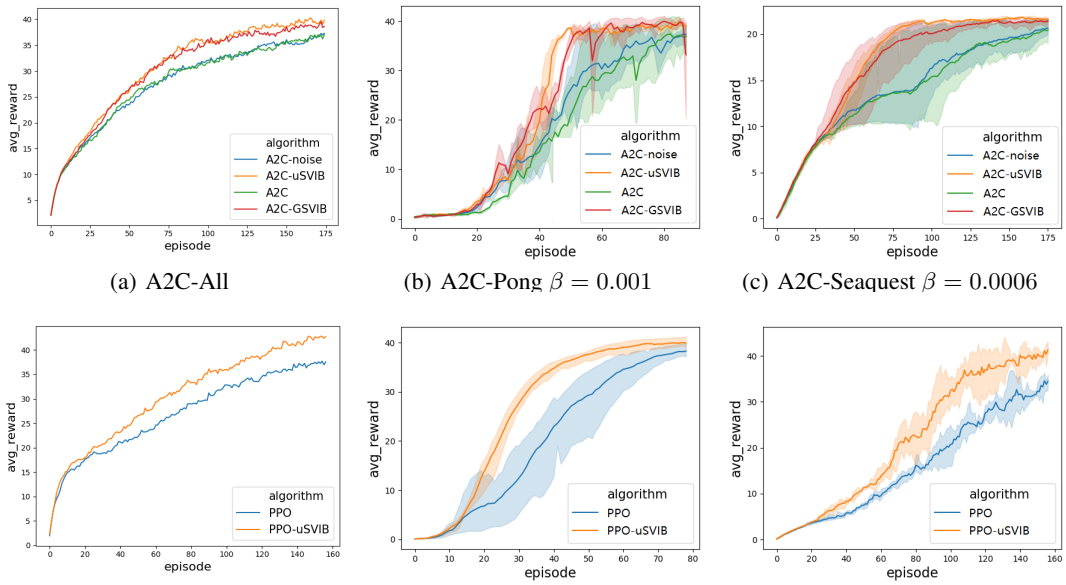
$$\frac{\partial \hat{L}(\theta, \phi)}{\partial \theta} = \mathbb{E}_{P_\phi(X,Z)} \left[\frac{\partial J(Z; \theta)}{\partial \theta} \right] \tag{19}$$

and update representation parameter ϕ by Equation 18.

5 EXPERIMENTS

In this section, we evaluate our proposed method SVIB (Stein Variational Information Bottleneck) in the Atari domain. Extensive experiments have shown the following results: 1) our framework can improve the sample efficiency of vanilla RL algorithms (A2C (Mnih et al., 2016) and PPO (Schulman et al., 2017)); 2) our method outperforms VIB (Alemi et al., 2016) in 16 out of 20 Atari games; 3) we run some simple experiments to show why our method works better: We show that our method is capable of accelerating information E-C process in 3 games as mentioned in Section 1 (Appendix C.4); 4) as MINE (Belghazi et al., 2018) is a popular method to estimate MI, we investigate the effectiveness of using MINE to optimize IB in RL (Appendix C.5); 5) we study the impact of different hyper-parameters β on the performance (Appendix C.6).

[2] In fact, Φ^* is chosen to maximize the directional derivative of $F(P) = -\mathbb{D}_{KL}(P||Q)$, which appears to be the "gradient" of F.



(a) A2C-All (b) A2C-Pong $\beta = 0.001$ (c) A2C-Seaquest $\beta = 0.0006$

(d) PPO-All (e) PPO-Pong $\beta = 0.0016$ (f) PPO-Qbert $\beta = 0.0016$

Figure 2: All reward curves are averaged over 3 random seeds, 10 episodes for each seed. Figure (a) and (d) show the averaged normalized reward across 20 Atari games for A2C and 18 Atari games for PPO. The rest four figures show the reward curves for two individual games, in which SVIB performs better than vanilla A2C and PPO. The details of other games can be found in Appendix C.2.

5.1 Improving Sample Efficiency for A2C and PPO

In A2C with our SVIB framework (the settings of PPO are almost the same as those of A2C), we sample Z from network $\phi(X, \xi)$ where $\xi \sim \mathcal{N}(\cdot; 0, 0.1)$ and the number of samples from each state X is 32. It turns out that the appropriate IB coefficient β varies among different games in our method, which we will discuss in more detail in Appendix C.6. The β we use for SVIB can be seen in the figures. We choose two prior distributions $U(Z)$ of our framework. The first one is the uniform distribution. The second one is a Gaussian distribution generated by the samples of representation Z. Further details can be found in Appendix C.1. We implement the following four algorithms for A2C:

A2C:Vanilla A2C in Openai-baselines (Dhariwal et al., 2017) with $\phi(X)$ as the embedding function.

A2C-uSVIB: Use $\phi(X, \xi)$ as the embedding function, optimized by our framework with $U(Z)$ being uniform distribution. Pseudocode of the algorithm can be found in Appendix A.1.

A2C-GSVIB: Use $\phi(X, \xi)$ as the embedding function, optimized by our framework with $U(Z)$ being Gaussian distribution.

A2C-noise: A2C with the same embedding function $\phi(X, \xi)$ as A2C-uSVIB and A2C-GSVIB. We design this algorithm for ablation study: Since our method needs additional noise as input while vanilla A2C does not need the noise.

Same as for A2C, we also implemented four algorithms for PPO: **PPO**, **PPO-uSVIB**, **PPO-GSVIB**, and **PPO-noise**. In our experiments, we find that in many games, PPO-GSVIB performs worse than PPO-uSVIB, and PPO-noise worse than PPO. So we just show and compare the performance of PPO-uSVIB and PPO. Figure 2 shows the reward curves of our method and the baselines. In Figure 2(a) and Figure 2(d), we normalize the episodic rewards to $[0, 50]$ and then average them across 20[3] Atari games (18 Atari games for PPO). We can see that our method improves sample efficiency of vanilla A2C and PPO in terms of normalized averaged rewards. The results of individual games can be seen in Appendix C.1. Specifically, A2C with SVIB is more sample efficient than vanilla A2C in 15 out of 20 games. PPO-uSVIB is more sample efficient than vanilla PPO in 15 out of 18 games.

[3]In fact, we have tested 35 games expect tough games like Pitfall and MontezumasRevenge. We find that A2C in Openai-baselines (Dhariwal et al., 2017) only manages to converge in 20 games. Thus we choose these 20 games for showing the performance of our method and vanilla A2C.

Recall that in Figure 1 in the introduction section, we show that the information extraction-compression phenomena exists in A2C training process for several games, by visualizing the evolution of mutual information $I(X, Z)$. We also study the changes of $I(X, Z)$ for our method. In particular, we visualize $I(X, Z)$ during the training using A2C-uSVIB in Pong. See Figure 3. Not only the information E-C process also exists for A2C-uSVIB, we can see that A2C-uSVIB in fact extracts and compresses information faster than vanilla A2C. It means that our method is able to accelerate the information E-C process in Pong, thus improving the sample efficiency. Further results and details can be found in Appendix C.4.

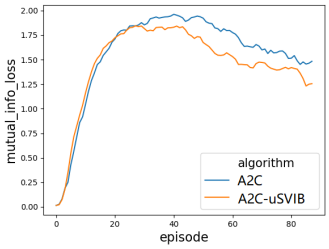


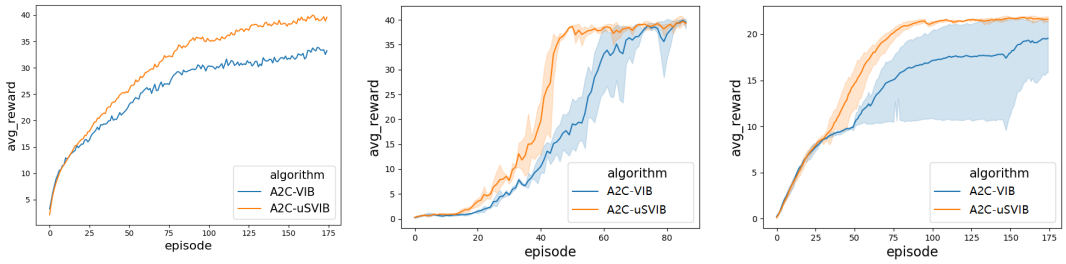
Figure 3: MI in Pong

5.2 Compare Our Method with VIB

We implement VIB based on A2C in RL. We name this algorithm "**A2C-VIB**". We have spent some time to tune the parameter β for VIB, but we found that the default $\beta = 0.001$ in (Alemi et al., 2016) works better than others. So we keep $\beta = 0.001$ for VIB in all games.

Figure 4 shows the performance of A2C-VIB and A2C-uSVIB. In Figure 4(a), we plot the normalized averaged reward curves (average across 20 Atari games) for A2C-uSVIB and A2C-VIB, and Figure 4(b) and (c) plot the reward curves for two individual games Pong and Seaquest. More experiment results for individual games can be found in Appendix C.3. From the figures, we can see that A2C-uSVIB generally outperforms A2C-VIB (in average, and in 16 out of 20 individual games).

There is another way to solve general IB objectives: Using MINE (originally for estimating MI) to optimize IB. Yet in RL, our empirical results show that it works even worse than vanilla RL algorithms. Further discussion and results can be found in Appendix C.5. So among these methods that can solve general IB objectives in RL, our method achieves the best performance.



(a) VIB-All (b) VIB-Pong $\beta = 0.001$ (c) VIB-Seaquest $\beta = 0.0006$

Figure 4: All curves are averaged over 3 random seeds, 10 episodes for each seed. Figure (a) shows the averaged normalized reward across 20 Atari games. The rest two figures show the reward for 2 example games reporting SVIB's superior performance over VIB. Other individual games can be found in Appendix C.3. The β in SVIB is as written in the figures, while it is fixed at 0.001 in VIB.

6 Conclusion

We propose an information-bottleneck-based framework for reinforcement learning and derive the closed form of the optimal target distribution. We construct a lower bound and utilize amortized Stein Variational gradient method to optimize it. We experimentally show that our framework can improve the performance of vanilla RL algorithms. To the best of our knowledge, our method achieves the best performance among information-bottleneck-based algorithms in reinforcement learning. Moreover, in order to gain a better understanding that why our method works better, we verify that the information extraction and compression process also exists in the training process of some simple games, and our framework can accelerate this process.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3134–3142, 2019.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.

Tessler Chen, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. *arXiv preprint arXiv:1707.06626*, 2017.

Tim Genewein, Felix Leibfried, Jordi Grau-Moya, and Daniel Alexander Braun. Bounded rationality, abstraction, and hierarchical decision-making: An information-theoretic optimality principle. *Frontiers in Robotics and AI*, 2:27, 2015.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyEtjoCqFX`.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *Proceedings of the 34th International Conference on Machine Learning*, 70:1352–1361, 2017.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bklr3j0cKX`.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, pp. 13978–13990, 2019.

Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. *arXiv preprint arXiv:1810.01176*, 2018.

Navneet Madhu Kumar. Empowerment-driven exploration using mutual information estimation. *arXiv preprint arXiv:1810.05533*, 2018.

Felix Leibfried and Daniel A Braun. A reward-maximizing spiking neuron as a bounded rational decision maker. *Neural computation*, 27(8):1686–1720, 2015.

Felix Leibfried and Jordi Grau-Moya. Mutual-information regularization in markov decision processes and actor-critic learning. *arXiv preprint arXiv:1909.05950*, 2019.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pp. 2378–2386, 2016.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Christopher A Sims. Implications of rational inattention. *Journal of monetary Economics*, 50(3): 665–690, 2003.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. *arXiv preprint arXiv:1910.06222*, 2019.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Jakub Sygnowski and Henryk Michalewski. Learning from the memory of atari 2600. *arXiv preprint arXiv:1605.01335*, 2016.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *University of Illinois*, 411(29-30):368–377, 2000.

Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning efficient multi-agent communication: An information bottleneck approach. *arXiv preprint arXiv:1911.06992*, 2019.

A Appendix of algorithm

A.1 Algorithm

Algorithm 1 Information-bottleneck-based state abstraction in RL

$\theta, \phi \leftarrow$ initialize network parameters
$\beta, \zeta \leftarrow$ initialize hyper-parameters
$\epsilon \leftarrow$ learning rate
$M \leftarrow$ number of samples from $P_\phi(\cdot|X)$
repeat
  Draw a batch of data $\{X_t, a_t, R_t, X_{t+1}\}_{t=1}^n$ from environment
  for each $X_t \in \{X_t\}_{t=1}^n$ **do**
    Draw M samples $\{Z_i^t\}_{i=1}^M$ from $P_\phi(\cdot|X_t)$
  end for
  Get the batch of data $\mathcal{D} = \{X_t, \{Z_i^t\}_{i=1}^M, a_t, R_t, X_{t+1}\}_{t=1}^n$
  Compute the representation gradients $\nabla_\phi L(\theta, \phi)$ in $\mathcal{D}$
  Compute the RL gradients $\nabla_\theta L(\theta, \phi)$ in $\mathcal{D}$
  Update ϕ: $\phi \leftarrow \phi + \epsilon \nabla_\phi L(\theta, \phi)$
  Update θ: $\theta \leftarrow \theta + \epsilon \nabla_\theta L(\theta, \phi)$
until Convergence

B APPENDIX OF THEOREMS AND DERIVATIONS

B.1 PROOF OF THEOREM 1

Theorem. *(Theorem 1 restated)Policy $\pi^r = \pi_{\theta^r}$, parameter ϕ^r, optimal policy $\pi^\star = \pi_{\theta^\star}$ and its relevant representation parameter $\phi^\star$ are defined as follows:*

$$\theta^r, \phi^r = \arg\min_{\theta,\phi} \mathbb{E}_{P_\phi(X,Z)} \left[\log \frac{P_\phi(Z|X)}{P_\phi(Z)} - \frac{1}{\beta} J(Z;\theta) \right]; \tag{20}$$

$$\theta^\star, \phi^\star = \arg\min_{\theta,\phi} \mathbb{E}_{P_\phi(X,Z)} \left[-\frac{1}{\beta} J(Z;\theta) \right]. \tag{21}$$

Define J^{π^r} as $\mathbb{E}_{P_{\phi^r}(X,Z)}[J(Z;\theta^r)]$ and $J^{\pi^\star}$ as $\mathbb{E}_{P_{\phi^\star}(X,Z)}[J(Z;\theta^\star)]$. Assume that there exists an $\epsilon > 0$, $|I(X,Z;\phi^\star) - I(X,Z;\phi^r)| < \frac{\epsilon}{\beta}$, we have $|J^{\pi^r} - J^{\pi^\star}| < \epsilon$. Specifically, in value-based algorithm, this theorem also holds between expectation of two value functions.

Proof. *From Equation 20 we can get:*

$$I(X,Z;\phi^\star) - \frac{1}{\beta} J^{\pi^\star} \geq I(X,Z;\phi^r) - \frac{1}{\beta} J^{\pi^r} \tag{22}$$

From Equation 21 we can get:

$$-\frac{1}{\beta} J^{\pi^r} \geq -\frac{1}{\beta} J^{\pi^\star} \tag{23}$$

These two equations give us the following inequality:

$$\beta(I(X,Z;\phi^\star) - I(X,Z;\phi^r)) \geq J^{\pi^\star} - J^{\pi^r} \geq 0 \tag{24}$$

According to the assumption, naturally we have:

$$|J^{\pi^r} - J^{\pi^\star}| < \epsilon \tag{25}$$

Notice that if we use our IB framework in value-based algorithm, then the objective function J^π can be defined as:

$$\begin{aligned} J^\pi = V^\pi &= (1-\gamma)^{-1} \int_X dX d^\pi(X) R^\pi(X) \\ &= (1-\gamma)^{-1} \int_X dX d^\pi(X) [\int_Z dZ P_\phi(Z|X) R^\pi(Z)] \end{aligned} \tag{26}$$

where $R^\pi(Z) = \int_{X \in \{X' : \phi(X') = Z\}} dX R^\pi(X)$ and d^π is the discounted future state distribution, readers can find detailed definition of d^π in the appendix of (Chen et al., 2018). We can get:

$$|V^{\pi^r} - V^{\pi^\star}| < \epsilon \tag{27}$$

B.2 TARGET DISTRIBUTION DERIVATION

We show the rigorous derivation of the target distribution.

Denote P as the distribution of X, $P_\phi^Z(Z) = P_\phi(Z)$ as the distribution of Z. We use P_ϕ as the short hand notation for the conditional distribution $P_\phi(Z|X)$. Moreover, we write $L(\theta,\phi) = L(\theta, P_\phi)$ and $\langle p, q \rangle_X = \int dX p(X) q(X)$. Notice that $P_\phi^Z(Z) = \langle P(\cdot), P_\phi(Z|\cdot) \rangle_X$. Take the functional derivative with respect to P_ϕ of the first term L_1:

$$\begin{aligned} &\left\langle \frac{\delta L_1(\theta, P_\phi)}{\delta P_\phi}, \Phi \right\rangle_{XZ} \\ &= \int dZ dX \frac{\delta L_1(\theta, P_\phi(Z|X))}{\delta P_\phi(Z|X)} \Phi(Z,X) = \left[\frac{d}{d\epsilon} L_1(\theta, P_\phi + \epsilon\Phi) \right]_{\epsilon=0} \\ &= \left[\frac{d}{d\epsilon} \int dX P(X) \Big\langle P_\phi(\cdot|X) + \epsilon\Phi(\cdot,X), J(\cdot;\theta) - \beta \log(P_\phi(\cdot|X) + \epsilon\Phi(\cdot,X)) \Big\rangle_Z \right]_{\epsilon=0} \\ &= \int dX P(X) \left[\Big\langle \Phi(\cdot,X), J(\cdot;\theta) - \beta \log P_\phi(\cdot|X) \Big\rangle + \Big\langle P_\phi(\cdot|X), -\beta \frac{\Phi(\cdot,X)}{P_\phi(\cdot|X)} \Big\rangle_Z \right] \\ &= \Big\langle P(\cdot)[J(\cdot;\theta) - \beta \log P_\phi(\cdot|\cdot) - \beta], \Phi(\cdot,\cdot) \Big\rangle_{XZ} \end{aligned}$$

Hence, we can see that

$$\frac{\delta L_1(\theta, P_\phi)}{\delta P_\phi(Z|X)} = P(X)[J(Z;\theta) - \beta \log P_\phi(Z|X) - \beta]. \tag{28}$$

Then we consider the second term. By the chain rule of functional derivative, we have that

$$\begin{aligned}\frac{\delta L_2(\theta, P_\phi)}{\delta P_\phi(Z|X)} &= \left\langle \frac{\delta L_2(\theta, P_\phi)}{\delta P_\phi^Z(\cdot)}, \frac{\delta P_\phi^Z(\cdot)}{\delta P_\phi(Z|X)} \right\rangle_{\hat{Z}} = \beta \left\langle 1 + \log P_\phi^Z(\cdot), \frac{\delta P_\phi^Z(\cdot)}{\delta P_\phi(Z|X)} \right\rangle_{\hat{Z}} \\ &= \beta \int d\hat{Z}(1 + \log P_\phi^Z(\hat{Z}))\delta(\hat{Z} - Z)P(X) = \beta P(X)(1 + \log P_\phi^Z(Z))\end{aligned} \tag{29}$$

Combining the derivative of L_1 and L_2 and setting their summation to 0, we can get that

$$P_\phi(Z|X) \propto P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta)) \tag{30}$$

B.3 Proof of Theorem 2

Theorem. *(Theorem 2 restated)For $L(\theta, \phi) = \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X)}[J(Z;\theta)] - \beta I(X, Z; \phi)$, given a fixed policy-value parameter θ, representation distribution $P_\phi(Z|X)$ and state distribution $P(X)$, define a new representation distribution:$P_{\hat{\phi}}(Z|X) \propto P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta))$, we have $L(\theta, \hat{\phi}) \geq L(\theta, \phi)$.*

Proof. *Define $I(X)$ as:*

$$I(X) = \int_Z dZ P_{\hat{\phi}}(Z|X) = \int_Z dZ P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta)) \tag{31}$$

$$\begin{aligned}L(\theta, \hat{\phi}) &= \mathbb{E}_X\{\mathbb{E}_{Z\sim P_{\hat{\phi}}(Z|X)}[J(Z;\theta)] - \beta \mathbb{E}_{Z\sim P_{\hat{\phi}}(Z|X)}[\log \frac{P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta))}{I(X) P_{\hat{\phi}}(Z)}]\} \\ &= \mathbb{E}_X\{\beta \mathbb{E}_{Z\sim P_{\hat{\phi}}(Z|X)}[\log I(X)] - \beta \mathbb{E}_{Z\sim P_{\hat{\phi}}(Z|X)}[\log \frac{P_\phi(Z)}{P_{\hat{\phi}}(Z)}]\} \\ &= \beta \mathbb{E}_X[\log I(X)] - \beta \mathbb{E}_{X,Z\sim P_{\hat{\phi}}(X,Z)}[\log \frac{P_\phi(Z)}{P_{\hat{\phi}}(Z)}] \\ &= \beta \mathbb{E}_X[\log I(X)] - \beta \mathbb{E}_{Z\sim P_{\hat{\phi}}(Z)}[\log \frac{P_\phi(Z)}{P_{\hat{\phi}}(Z)}] \\ &= \beta \mathbb{E}_{X\sim P(X)}[\log I(X)] + \beta \mathbb{D}_{KL}(P_{\hat{\phi}}(Z)||P_\phi(Z))\end{aligned} \tag{32}$$

$$\begin{aligned}L(\theta, \phi) &= \mathbb{E}_X\{\beta \mathbb{E}_{Z\sim P_\phi(Z|X)}[\log \exp(\frac{1}{\beta} J(Z;\theta))] + \beta \mathbb{E}_{Z\sim P_\phi(Z|X)} \log \frac{P_\phi(Z)}{P_\phi(Z|X)}\} \\ &= \mathbb{E}_X\{\beta \mathbb{E}_{Z\sim P_\phi(Z|X)}[\log \frac{P_\phi(Z) \exp(\frac{1}{\beta} J(Z;\theta))}{P_\phi(Z|X) I(X)}] + \beta \log I(X)\} \\ &= \beta \mathbb{E}_X[\log I(X)] + \beta \mathbb{E}_{X\sim P(X), Z\sim P_\phi(Z|X)}[\log \frac{P_{\hat{\phi}}(Z|X)}{P_\phi(Z|X)}] \\ &= \beta \mathbb{E}_{X\sim P(X)}[\log I(X)] - \beta \mathbb{E}_{X\sim P(X)}[\mathbb{D}_{KL}(P_\phi(Z|X)||P_{\hat{\phi}}(Z|X))]\end{aligned} \tag{33}$$

$$L(\theta, \hat{\phi}) - L(\theta, \phi) = \beta \mathbb{D}_{KL}(P_{\hat{\phi}}(Z)||P_\phi(Z)) + \beta \mathbb{E}_{X\sim P(X)}[\mathbb{D}_{KL}(P_\phi(Z|X)||P_{\hat{\phi}}(Z|X))] \tag{34}$$

According to the positivity of the KL-divergence, we have $L(\theta, \hat{\phi}) \geq L(\theta, \phi)$.

C Experimental Results

C.1 Experiment Settings

In A2C with our SVIB framework, we sample Z from network $\phi(X, \xi)$ where $\xi \sim \mathcal{N}(\cdot; 0, 0.1)$ and the number of samples from each state X is 32. It turns out that the appropriate IB coefficient β

varies among different games in our method. The β we use for SVIB can be seen in the figures. We choose two prior distributions $U(Z)$ of our framework. The first one is the uniform distribution. Apparently, when $U(Z)$ is the uniform distribution, $\nabla_{\hat{Z}} \log U(\hat{Z}) \mid_{\hat{Z}=Z}$ can be omitted. The second one is the Gaussian distribution, which is defined as follows: for a given state X_i, sample a batch of $\{Z_j^i\}_{j=1}^n$, then define $U(Z) = \mathcal{N}(Z; \mu = \frac{1}{n}\sum_{j=1}^n Z_j^i, \sigma^2 = \frac{1}{n}\sum_{j=1}^n (Z_j^i - \mu)^2)$, in which $n = 32$. The setting of this Gaussian distribution is a little bit heuristic: We roughly assume that a Gaussian distribution generated by the samples of $\{Z_j^i\}_{j=1}^n$ can approximate the true representation distribution $P_\phi(Z)$. Yet later we find that Gaussian distribution performs nearly the same as uniform distribution in A2C with our framework, and in PPO with our framework, uniform distribution is even better than Gaussian distribution.

To calculate $\Phi(Z_i)$ (defined in Equation 17 in Section 4.3), we replace $\log U(\hat{Z})$ with $\zeta \log U(\hat{Z})$ to control the magnitude of $\nabla_{\hat{Z}} \log U(\hat{Z})$. We set ζ as $0.005\|\nabla_{\hat{Z}} \frac{1}{\beta} J(\hat{Z}; \theta) / \nabla_{\hat{Z}} \log U(\hat{Z})\| \mid_{\hat{Z}=Z}$. Following (Liu et al., 2017), the kernel function we use for $\Phi(Z_i)$ is Gaussian RBF kernel $\mathcal{K}(Z_i, Z_j) = \exp(-\|Z_i - Z_j\|^2/h)$ where $h = med^2/2\log(n+1)$, med denotes the median of pairwise distances between the particles $\{Z_j^i\}_{j=1}^n$. As for the hyper-parameters in RL, we simply choose the default parameters in A2C of Openai-baselines (Dhariwal et al., 2017). The number of total time steps is 14 million (56 million frames specifically) except that the number of time steps for Pong is 7 million, since the agent in Pong converges to the optimal score much faster than other games.

The settings of PPO are almost the same as those of A2C. Here are some differences: 1)The number of samples from each state is 26. 2)Since PPO converges faster than A2C in most games, we set the number of total time steps to be 10 million (40 million frames) except 5 million in Pong. And we find that in some games, PPO-GSVIB is worse than PPO-uSVIB, and PPO-noise is worse than regular PPO. So we just compare the performance of PPO-uSVIB and PPO.

C.2 Individual Atari games for A2C and PPO

Figure 5, 6, 7 and 8 show the cumulative reward curves. We use exponential moving average to smooth each curve (In Pong, we move the y-axis up 21 units so that the lowest score is 0, which is convenient to make exponential moving average). All curves are averaged over 3 random seeds, 10 episodes for each seed.

Figure 5 and 6 show the cumulative rewards of four A2C-based algorithms. The yellow curve and red curve show the performance of A2C with our method. We can see that in MsPacman, Carnival, SpaceInvaders, DemonAttack and YarsRevenge (Figure 5(d), (h), (i) and Figure 6(e), (f)), A2C with our method performs nearly same as vanilla A2C. While in other 15 games, our method improves sample efficiency of vanilla A2C. Figure 7 and 8 show the cumulative rewards of two PPO-based algorithms. The yellow curve shows the performance of PPO with our method. We can see that in MsPacman, Seaquest, Asterix (Figure 7(c) and Figure 8(d), (f)), PPO with our method performs nearly same as vanilla PPO. While in other 15 games, our method improves sample efficiency of vanilla PPO.

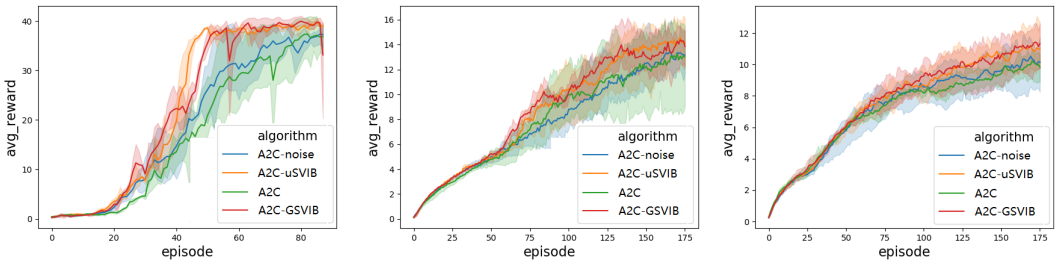
C.3 Individual Atari games for VIB

Figure 9 and 10 show the cumulative rewards of A2C-VIB and A2C-uSVIB. The yellow curve shows the performance of A2C-uSVIB. We can see that in Qbert, Assault, CrazyClimber and YarsRevenge (Figure 9(b) and Figure 10(a), (l), (n)), A2C-uSVIB performs nearly same as A2C-VIB. While in other 16 games, A2C-uSVIB is more sample efficient than A2C-VIB.

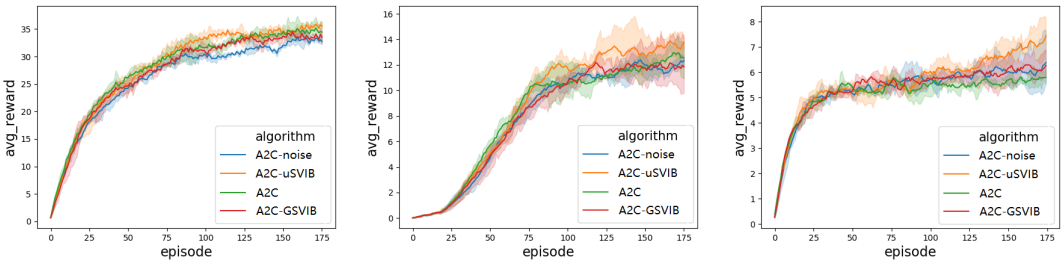
C.4 SVIB accelerates the information E-C process

This section we verify that the information E-C process exists in several Atari games when we train agents using A2C (using MINE to estimate the mutual information) and our framework accelerates this process.

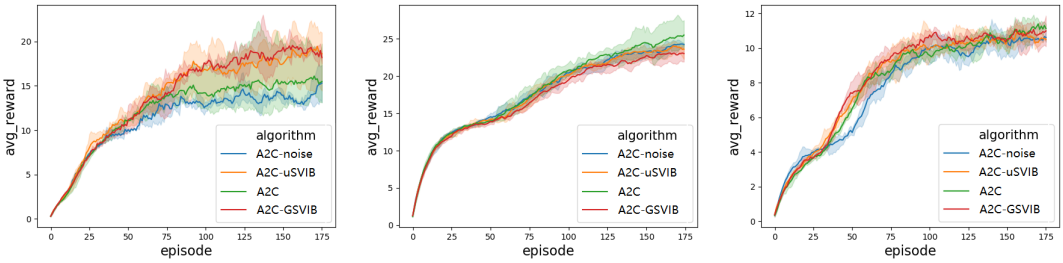
Mutual information neural estimation (MINE) (Belghazi et al., 2018) is an algorithm that can estimate mutual information (MI) between two high dimensional random variables more accurately and



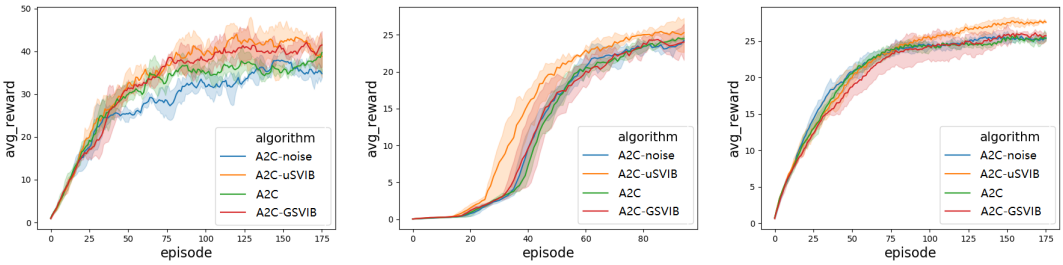
(a) A2C-Pong $\beta = 0.001$ (b) A2C-Qbert $\beta = 0.001$ (c) A2C-BeamRider $\beta = 0.001$



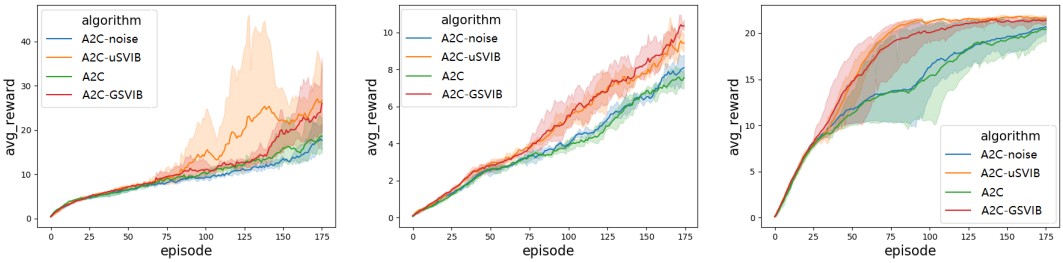
(d) A2C-MsPacman $\beta = 0.001$ (e) A2C-Breakout $\beta = 0.001$ (f) A2C-Asteroids $\beta = 0.001$



(g) A2C-Assault $\beta = 0.0006$ (h) A2C-Carnival $\beta = 0.0006$ (i) A2C-SpaceInvaders $\beta = 0.0006$

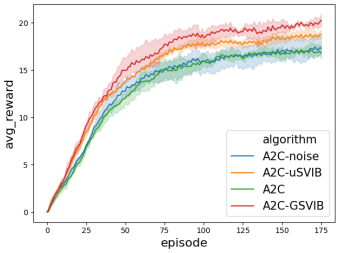


(j) A2C-AirRaid $\beta = 0.0006$ (k) A2C-BankHeist $\beta = 0.0006$ (l) A2C-Alien $\beta = 0.0006$

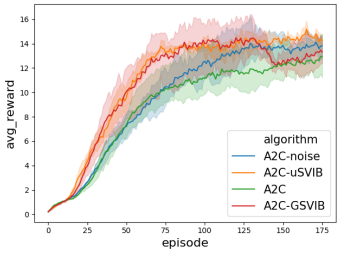


(m) A2C-Atlantis $\beta = 0.0006$ (n) A2C-StarGunner $\beta = 0.0006$ (o) A2C-Seaquest $\beta = 0.0006$

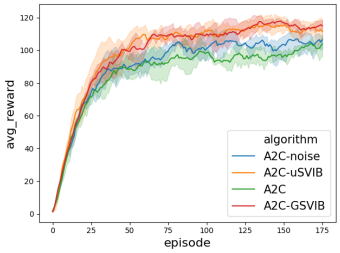
Figure 5: Cumulative reward curves on A2C. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.



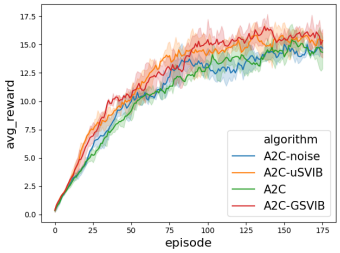
(a) A2C-Amidar $\beta = 0.0006$



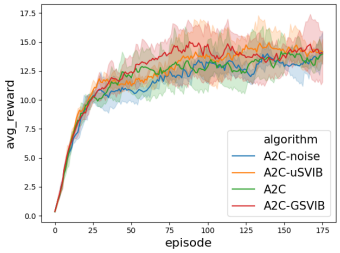
(b) A2C-Asterix $\beta = 0.0006$



(c) A2C-CrazyClimber $\beta = 0.0006$

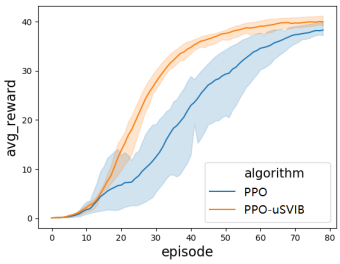


(d) A2C-DemonAttack $\beta = 0.0006$

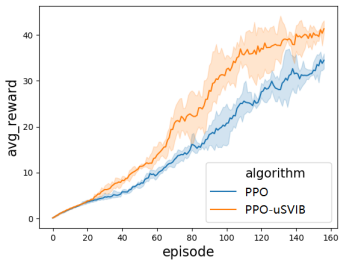


(e) A2C-YarsRevenge $\beta = 0.0006$

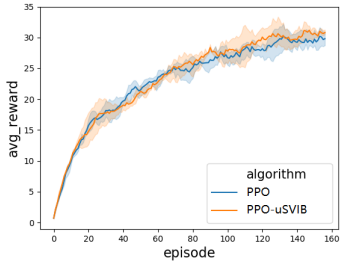
Figure 6: Cumulative reward curves on A2C. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.



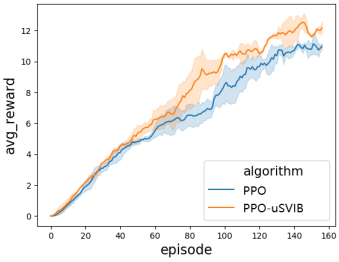
(a) PPO-Pong $\beta = 0.0016$



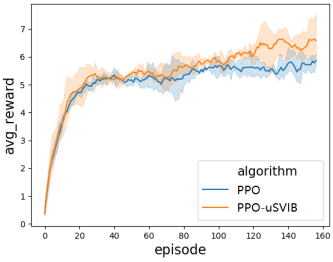
(b) PPO-Qbert $\beta = 0.0016$



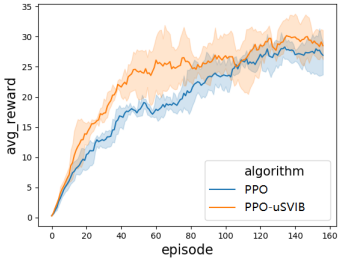
(c) PPO-MsPacman $\beta = 0.0006$



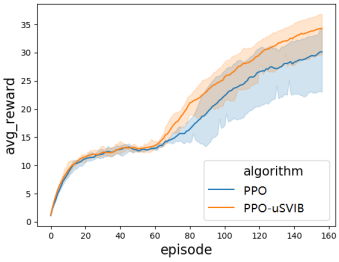
(d) PPO-Breakout $\beta = 0.001$



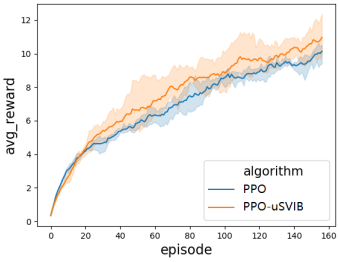
(e) PPO-Asteroids $\beta = 0.0016$



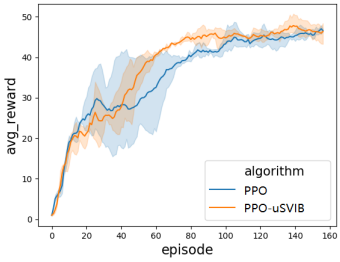
(f) PPO-Assault $\beta = 0.0016$



(g) PPO-Carnival $\beta = 0.0016$

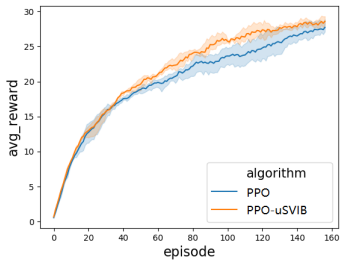


(h) PPO-SpaceInvaders $\beta = 0.0016$

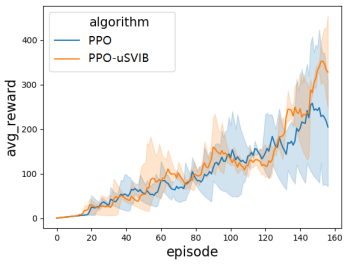


(i) PPO-AirRaid $\beta = 0.8$

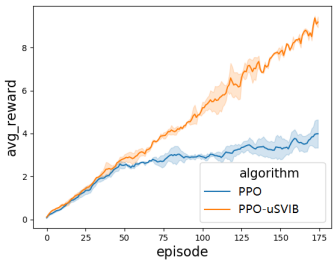
Figure 7: Cumulative reward curves on PPO. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.



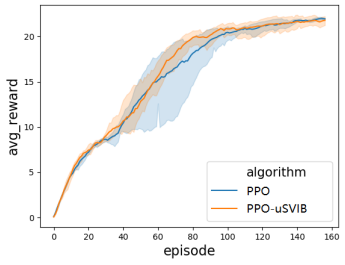
(a) PPO-Alien $\beta = 0.002$



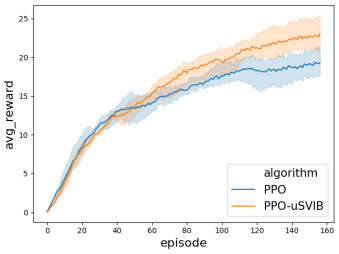
(b) PPO-Atlantis $\beta = 0.003$



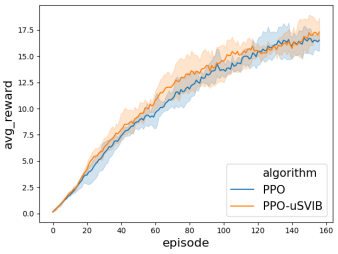
(c) PPO-StarGunner $\beta = 0.0016$



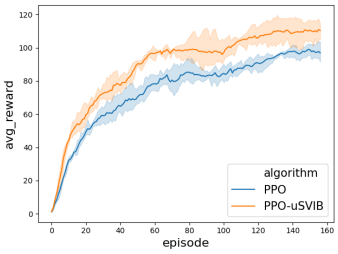
(d) PPO-Seaquest $\beta = 0.0016$



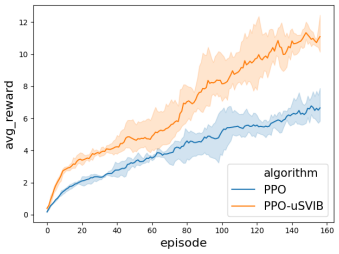
(e) PPO-Amidar $\beta = 0.0016$



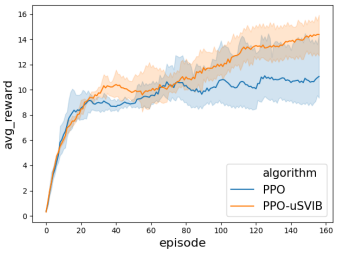
(f) PPO-Asterix $\beta = 0.0016$



(g) PPO-CrazyClimber $\beta = 0.0016$

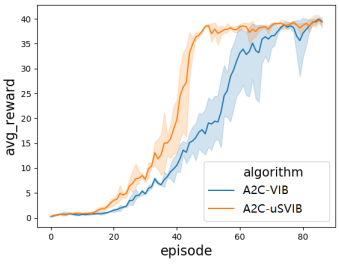


(h) PPO-DemonAttack $\beta = 0.0016$

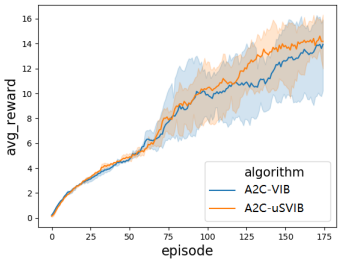


(i) PPO-YarsRevenge $\beta = 0.0016$

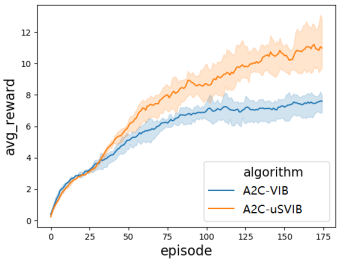
Figure 8: Cumulative reward curves on PPO. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.



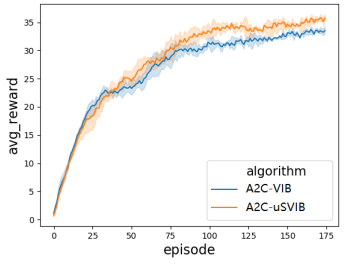
(a) A2C-Pong $\beta = 0.001$



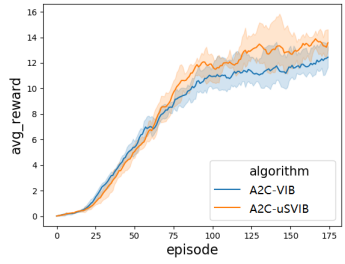
(b) A2C-Qbert $\beta = 0.001$



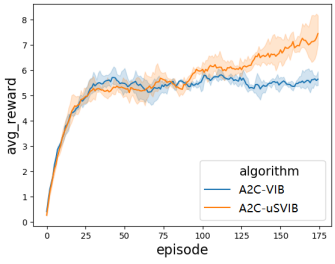
(c) A2C-BeamRider $\beta = 0.001$



(d) A2C-MsPacman $\beta = 0.001$

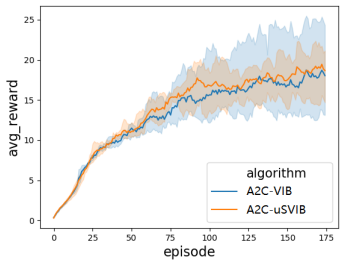


(e) A2C-Breakout $\beta = 0.001$

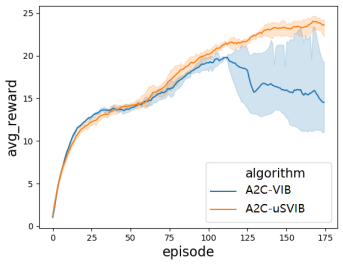


(f) A2C-Asteroids $\beta = 0.001$

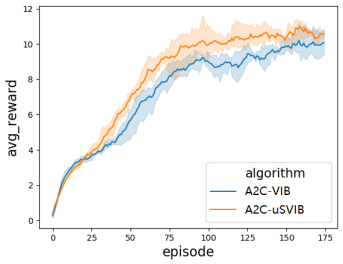
Figure 9: Cumulative reward curves of A2C-VIB and A2C-uSVIB. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.



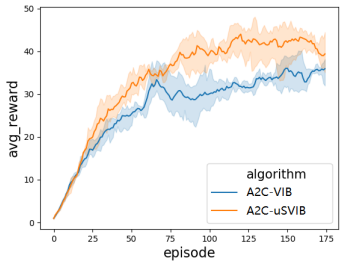
(a) A2C-Assault $\beta = 0.0006$



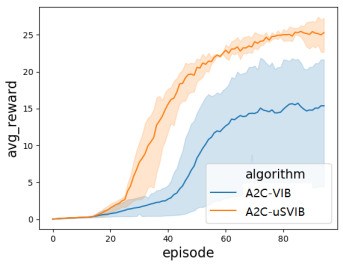
(b) A2C-Carnival $\beta = 0.0006$



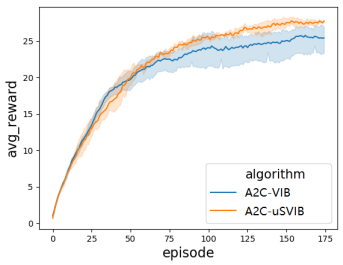
(c) A2C-SpaceInvaders $\beta = 0.0006$



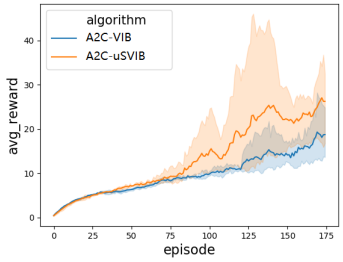
(d) A2C-AirRaid $\beta = 0.0006$



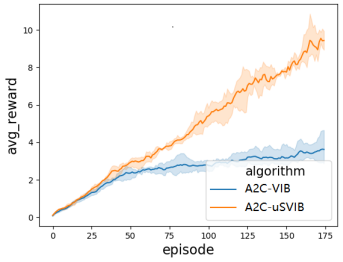
(e) A2C-BankHeist $\beta = 0.0006$



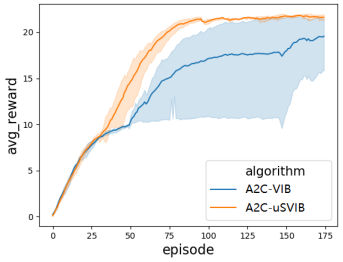
(f) A2C-Alien $\beta = 0.0006$



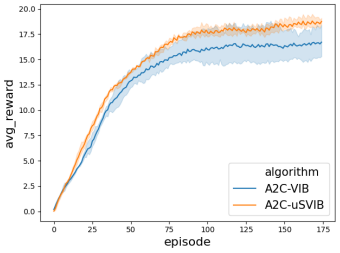
(g) A2C-Atlantis $\beta = 0.0006$



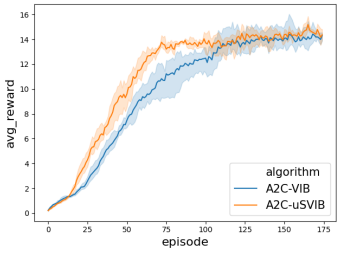
(h) A2C-StarGunner $\beta = 0.0006$



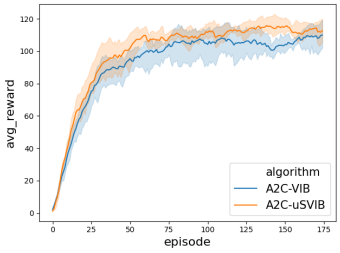
(i) A2C-Seaquest $\beta = 0.0006$



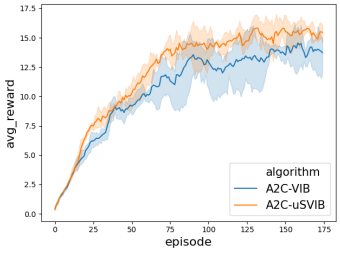
(j) A2C-Amidar $\beta = 0.0006$



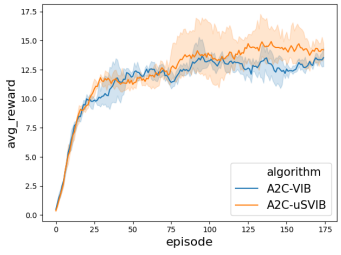
(k) A2C-Asterix $\beta = 0.001$



(l) A2C-CrazyClimber $\beta = 0.001$



(m) A2C-DemonAttack $\beta = 0.0006$



(n) A2C-YarsRevenge $\beta = 0.0006$

Figure 10: Cumulative reward curves of A2C-VIB and A2C-uSVIB. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.

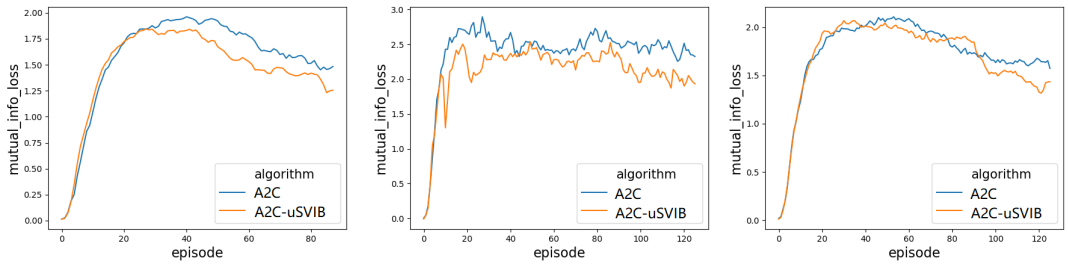
efficiently. Specifically, for random variables X and Z, assume T to be a function of X and Z, the calculation of $I(X, Z)$ can be transformed to the following optimization problem according to (Belghazi et al., 2018):

$$I(X, Z) = \max_{T} \mathbb{E}_{P(X,Z)}[T] - \log(\mathbb{E}_{P(X)\otimes P(Z)}[\exp^{T}]). \tag{35}$$

The optimal function $T^{\star}(X, Z)$ can be approximated by updating a neural network $T(X, Z; \eta)$.

With the aid of this powerful tool, we would like to visualize the mutual information (MI) between input state X and its relative representation Z: Every a few update steps, we sample a batch of inputs and their relevant representations $\{X_i, Z_i\}_{i=1}^{n=64}$ and compute their MI with MINE. The learning rate of updating η is 0.0007 and the number of training steps is 256. Every time we train MINE (update η), we just shuffle $\{Z_i\}_{i=1}^{n}$ and roughly assume the shuffled representations $\{Z_i^{\text{shuffled}}\}_{i=1}^{n}$ to be independent with $\{X_i\}_{i=1}^{n}$:

$$I(X, Z) \approx \max_{\eta} \frac{1}{n}\sum_{i=1}^{n}[T(X_i, Z_i; \eta)] - \log(\frac{1}{n}\sum_{i=1}^{n}[\exp^{T(X_i, Z_i^{\text{shuffled}}; \eta)}]). \tag{36}$$



(a) MI in Pong (b) MI in MsPacman (c) MI in Assault

Figure 11: MI estimation in 3 Atari games.

Figure 11 shows the mutual information estimation between X and Z in 3 Atari games. The x-axis is the number of update steps and the y-axis is the MI estimation. As we can see, in both A2C-uSVIB and vanilla A2C, the MI first increases to encode more information from inputs ("remember" the inputs), then decreases to drop irrelevant information from inputs ("forget" the useless information). And clearly, A2C-uSVIB extracts faster and compresses faster than regular A2C in all 3 games.

C.5 Optimize IB with MINE in RL

This section we show the performance of using MINE as an IB optimizer in RL. And discuss the possible reasons that may cause the bad performance.

According to Equation 35, the whole optimization problem can be written as follows:

$$\begin{aligned} L(\theta, \phi, \eta) = &\underbrace{\mathbb{E}_{P_\phi(X,Z)}[J(Z; \theta)]}_{\text{RL loss term}} \\ &- \beta\mathbb{E}_{P_\phi(X,Z)}[T(X, Z; \eta)] + \beta \log(\mathbb{E}_{P(X)\otimes P_\phi(Z)}[\exp^{T(X,Z;\eta)}]), \end{aligned} \tag{37}$$

where η is the parameter of the function T. Then the optimal parameters of this optimization problem are:

$$\theta^{\star}, \phi^{\star}, \eta^{\star} = \arg\max_{\theta,\phi} \arg\min_{\eta} L(\theta, \phi, \eta). \tag{38}$$

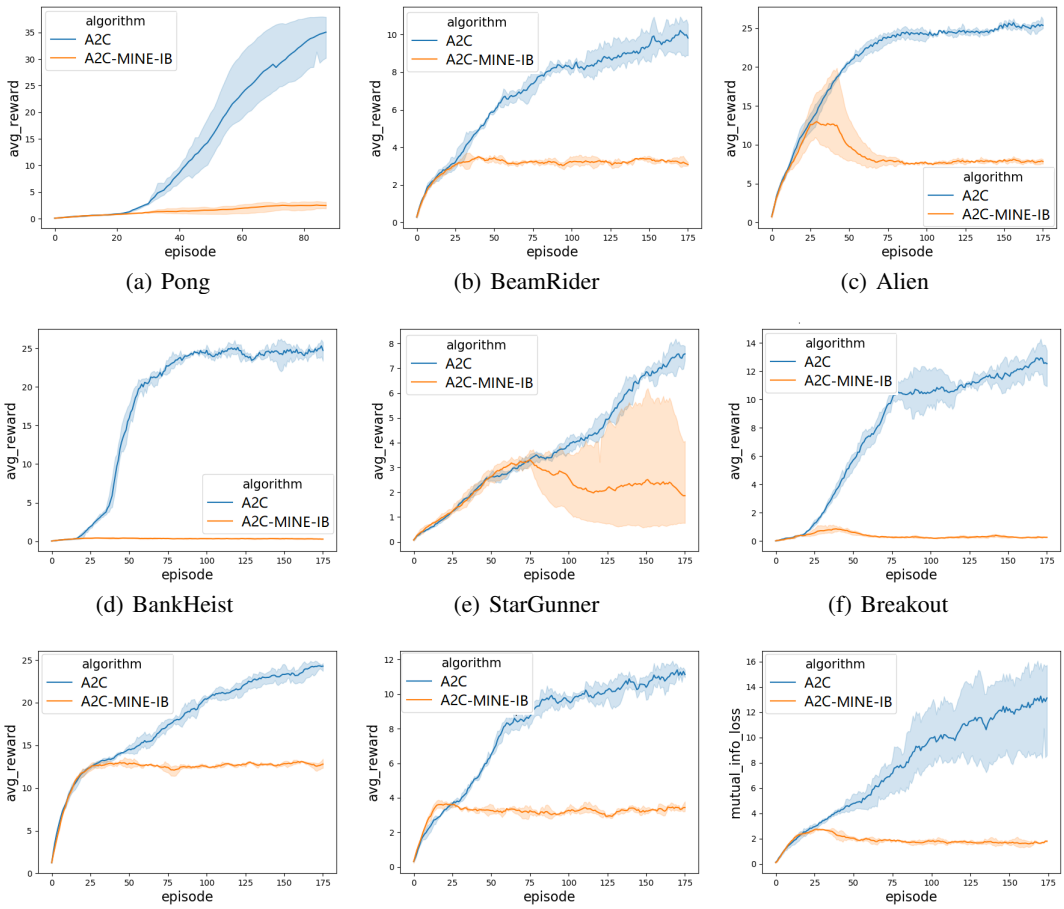
Thus the key steps to optimize $L(\theta, \phi, \eta)$ is to update θ, ϕ, η iteratively as follows:

$$\eta_{t+1} \leftarrow \arg\min_{\eta_t} -\beta\mathbb{E}_{P_{\phi_t}(X,Z)}[T(X, Z; \eta_t)] + \beta \log(\mathbb{E}_{P(X)\otimes P_{\phi_t}(Z)}[\exp^{T(X,Z;\eta_t)}]), \tag{39}$$

$$\theta_{t+1}, \phi_{t+1} \leftarrow \arg\max_{\theta_t,\phi_t} L(\theta_t, \phi_t, \eta_{t+1}). \tag{40}$$

We call this algorithm "**MINE-IB**". Unfortunately, though MINE is a powerful tool as a mutual information estimator, MINE-IB is quite unstable and may be even hard to converge in reinforcement

learning. We implement MINE-IB based on A2C (named "**A2C-MINE-IB**"). Figure 12 shows the cumulative reward of A2C-MINE-IB and vanilla A2C in 9 Atari games. We can see that A2C-MINE-IB performs much worse than vanilla A2C. In some games, e.g., Pong and BankHeist (Figure 12(a) and (d)), and the performance of A2C-MINE-IB is nearly same as a random policy.



(a) Pong (b) BeamRider (c) Alien

(d) BankHeist (e) StarGunner (f) Breakout

(g) Carnival (h) SpaceInvaders (i) Qbert

Figure 12: Cumulative reward curves of A2C-MINE-IB and A2C. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.

We suspect the following two reasons that may cause such performance:

1. MINE suffers from exponential variance according to (Song & Ermon, 2019). Considering training the regular RL objective is already quite difficult and somewhat unstable, if we combine RL objective with MINE as a mutual information regularizer, the training becomes even more unstable.
2. According to Equation 38, optimizing IB with MINE is a min-max optimization problem. In deep learning, this kind of problems are extremely hard to converge.

We have mentioned that MINE is a useful tool for mutual information maximization according to (Hjelm et al., 2019). Since mutual information maximization with MINE is a maximization optimization problem. Here is an example about the difference of MI maximization and IB optimization with MINE. Suppose X is the input, $Z = f(X, \epsilon; \xi)$ is the representation, where f is the embedding function, ξ is the noise and ϕ is the parameter of f. If we want to maximize the MI between X and Z using MINE, then the optimization problem can be written as follows:

$$\max_{\phi,\eta}\{\mathbb{E}_{P_{\phi_t}(X,Z)}[T(X,Z;\eta_t)] - \log(\mathbb{E}_{P(X)\otimes P_{\phi_t}(Z)}[\exp^{T(X,Z;\eta_t)}])\}, \tag{41}$$

which is easy to optimize in deep learning as shown in (Hjelm et al., 2019). While if we want to minimize the MI under some constraints $g(Z;\phi)$ (assume that we want to maximize $g(\phi)$, like Equation 38), then the optimization problem can be written as follows:

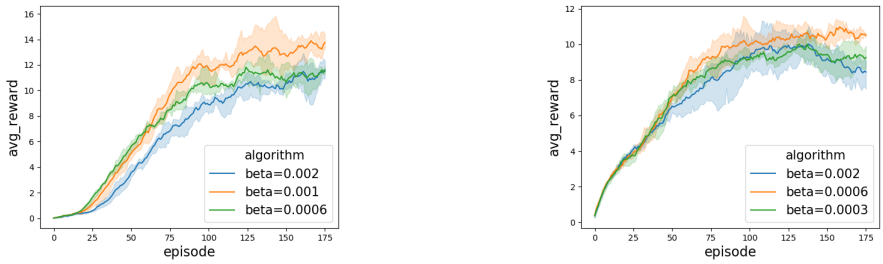
$$\max_{\phi} \min_{\eta} \{-\mathbb{E}_{P_{\phi_t}(X,Z)}[T(X,Z;\eta_t)] + \log(\mathbb{E}_{P(X)\otimes P_{\phi_t}(Z)}[\exp^{T(X,Z;\eta_t)}]) + g(\phi)\}, \tag{42}$$

which is extremely hard to optimize in deep learning.

C.6 THE IMPACT OF β

This section we show the impact of hyper-parameter β.

It turns out that that the proper IB coefficient β varies among different games in our method. Either large or small β will yield bad performance. Figure 13 shows the cumulative reward curves of A2C-uSVIB with different β. In Figure 13(a), the proper β in Breakout is 0.001, either the performance of 0.002 or 0.0006 is worse than 0.001. While in Figure 13(b), the proper β in Breakout is 0.0006, whose performance is better than 0.002 and 0.0003.



(a) Breakout (b) SpaceInvaders

Figure 13: Cumulative reward curves of A2C-uSVIB across different hyper-parameter β. All reward curves are averaged over 3 random seeds, 10 episodes for each seed.