Generalization Guarantees for Imitation Learning

Allen Z. Ren, Sushant Veer, and Anirudha Majumdar
Department of Mechanical and Aerospace Engineering
Princeton University
{allen.ren, sveer, ani.majumdar}@princeton.edu

Abstract: Control policies from imitation learning can often fail to generalize to novel environments due to imperfect demonstrations or the inability of imitation learning algorithms to accurately infer the expert's policies. In this paper, we present rigorous generalization guarantees for imitation learning by leveraging the *Probably Approximately Correct (PAC)-Bayes* framework to provide upper bounds on the expected cost of policies in novel environments. We propose a two-stage training method where a latent policy distribution is first embedded with multi-modal expert behavior using a conditional variational autoencoder, and then "fine-tuned" in new training environments to explicitly optimize the generalization bound. We demonstrate strong generalization bounds and their tightness relative to empirical performance in simulation for (i) grasping diverse mugs, (ii) planar pushing with visual feedback, and (iii) vision-based indoor navigation, as well as through hardware experiments for the two manipulation tasks. [1] [2]

Keywords: Generalization, imitation learning, manipulation, indoor navigation

1 Introduction

Imagine a personal robot that is trained to navigate around homes and manipulate objects via *imitation learning* [1]. How can we guarantee that the resulting control policy will behave safely and perform well when deployed in a novel environment (e.g., in a previously unseen home or with new furniture)? Unfortunately, state-of-the-art imitation learning techniques do not provide any guarantees on *generalization* to novel environments and can fail dramatically when operating conditions are different from ones seen during training [2]. This may be due to the expert's demonstrations not being safe or generalizable, or due to the imitation learning algorithm not accurately inferring the expert's policy. The goal of this work is to address this challenge and propose a framework that allows us to provide *rigorous guarantees on generalization* for imitation learning.

The key idea behind our approach is to leverage powerful techniques from *generalization theory* [3] in theoretical machine learning to "fine-tune" a policy learned from demonstrations while also making guarantees on generalization for the resulting policy. More specifically, we employ *Probably Approximately Correct (PAC)-Bayes* theory [4, 5, 6]; PAC-Bayes theory has recently emerged as a promising candidate for providing strong generalization bounds for neural networks in supervised learning problems [7, 8, 9] (in contrast to other generalization frameworks, which often provide vacuous bounds [7]). However, the use of PAC-Bayes theory beyond supervised learning settings has been limited. In this work, we demonstrate that PAC-Bayes theory affords previously untapped potential for providing guarantees on imitation-learned policies deployed in novel environments.

Statement of Contributions: To our knowledge, the results in this paper constitute the first attempt to provide generalization guarantees on policies learned via imitation learning for robotic systems with rich sensory inputs (e.g., RGB-D images), complicated (e.g., nonlinear and hybrid) dynamics, and neural network-based policy architectures. We present a synergistic two-tier training pipeline that performs imitation learning in the first phase and then "fine-tunes" the resulting policy in a second phase (Fig. 1). In particular, the first phase performs multi-modal behavioral cloning [10] using a conditional variational autoencoder (cVAE) [11, 12] and diverse expert demonstrations. The

[1] Code is available at: https://github.com/irom-lab/PAC-Imitation

[2] A video showing the experiment results is available at: https://youtu.be/dfXyHvOTolc

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

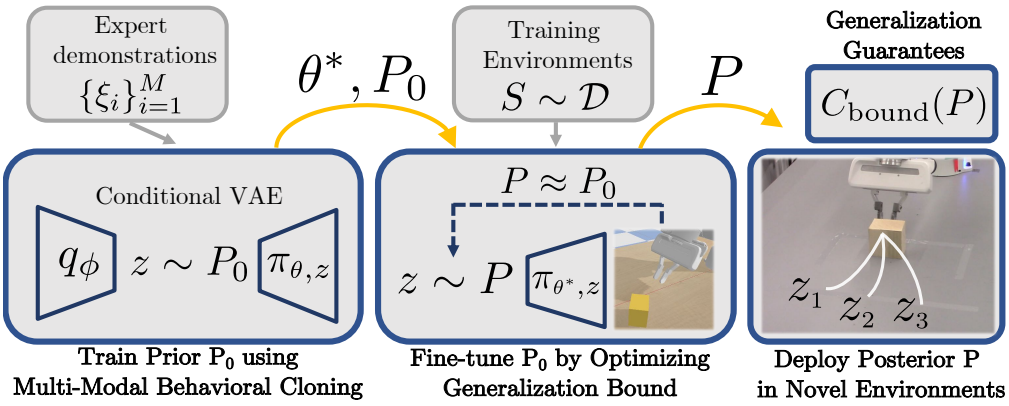


Figure 1: System overview. (Left) policies from expert demonstrations are embedded into a "prior" distribution P_0 (latent distribution in a cVAE). The cVAE decoder $\pi_{\theta,z}$ learns to generate actions close to an expert's; (middle) the prior P_0 is then "fine-tuned" by optimizing a generalization bound from PAC-Bayes theory. The decoder $\pi_{\theta,z}$ (with neural network parameters fixed as θ^*) is re-used to infer actions in training environments. The posterior distribution P is close to P_0, (right) and comes with a generalization bound $C_{\text{bound}}(P)$ when P is deployed in novel environments. Note that we work with a distribution over policies: different z sampled from P produce different pushing trajectories given the same observation.

resulting policy is then used as a *prior* for the second phase; this prior is specified by the distribution over the cVAE's latent variables. The second phase of training uses a fresh set of environments to optimize a *posterior* distribution over the latent variables by explicitly optimizing a bound on the generalization performance derived from PAC-Bayes theory. The resulting fine-tuned policy has an associated bound on the expected cost across novel environments under the assumption that novel environments and training environments are drawn from the same (but *unknown*) distribution (see Sec. 2 for a precise problem formulation). We demonstrate strong generalization bounds and their tightness relative to empirical generalization performance using simulation experiments in three settings: (i) grasping mugs with varying geometric and physical properties, (ii) vision-based feedback control for planar pushing, and (iii) navigating in cluttered home environments. We also present extensive hardware experiments for the manipulation examples using a Franka Panda arm equipped with an external RGB-D sensor. Taken together, our simulation and hardware experiments demonstrate the ability of our approach to provide strong generalization guarantees for policies learned via imitation learning in challenging robotics settings.

Related work

Multi-modal imitation learning. Imitation learning is commonly used in manipulation [13, 14, 2] and navigation tasks [15, 16] to accelerate training by injecting expert knowledge. Often, imitation data can be multi-modal, e.g., an expert may choose to grasp anywhere along the rim of a mug. Recently, a large body of work uses latent variable models to capture such multi-modality in robotic settings [17, 18, 19, 20, 10]. While these papers use multi-modal data to diversify imitated behavior, we embed the multi-modality into the prior policy distribution to accelerate the "fine-tuning" and produce better generalization bounds and empirical performance (see Sec. 4.3 and A4).

Learning from imperfect demonstrations. Another challenge with imitation learning is that expert demonstrations can often be imperfect [21]. Moreover, policies trained using only off-policy data can also cause cascading errors [22] when the robot encounters unseen states. One approach to addressing these challenges is to "fine-tune" an imitation-learned policy to improve empirical performance in novel environments [21, 23]. Other techniques that mitigate these issues include collecting demonstrations with noisy dynamics [24], augmenting observation-action data of demonstrations using noise [13], and training using a hybrid reward for both imitation and task success [2]. Some recent work also explores generalization in longer horizon tasks [23, 25]. However, none of these techniques provide rigorous generalization guarantees for imitated policies deployed in novel environments for robotic systems with discrete/continuous state and action spaces, nonlinear/hybrid dynamics, and rich sensing (e.g., vision). This is the primary focus of this paper.

Generalization guarantees for learning-based control. The PAC-Bayes Control framework [26, 27, 28] provides a way to make safety and generalization guarantees for learning-based control. In this work, we extend this framework to make guarantees on policies learned via imitation learning. To this end, we propose a two-phase training pipeline for learning a prior distribution over

policies via imitation learning and then fine-tuning this prior by optimizing a PAC-Bayes generalization bound. By leveraging multi-modal expert demonstrations, we are able to obtain significantly stronger generalization guarantees than [27, 28], which either employ simple heuristics to choose the prior or train the prior from scratch.

2 Problem Formulation

We assume that the discrete-time dynamics of the robot are given by:

$$s_{t+1} = f_E(s_t, a_t), \tag{1}$$

where $s_t \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ is the state at time-step t, $a_t \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the action, and $E \in \mathcal{E}$ is the environment that the robot is operating in. We use the term "environment" here broadly to refer to external factors such as the object that a manipulator is trying to grasp, or a room that a personal robot is operating in. We assume that the robot has a sensor which provides observations $o_t \in \mathcal{O}$. For the first phase of our training pipeline (Fig. 1), we assume that we are provided a finite set of expert demonstrations $\{\zeta_i\}_{i=1}^M$, where each $\zeta_i := \{(o_t, a_t)\}_{t=1}^T$ is a sequence of observation-action pairs (e.g., human demonstrations of a manipulation task, where o_t and a_t correspond to depth images and desired relative-to-current poses respectively at step t of the sequence). For the second phase of training, we assume access to a dataset $S := \{E_1, \ldots, E_N\}$ of N training environments drawn independently from a distribution $\mathcal{D}$ (e.g., the distribution on the shapes, dimensions, and mass of mugs to be grasped). Importantly, we *do not* assume knowledge of the distribution $\mathcal{D}$ or the space $\mathcal{E}$ of environments (which may be extremely high-dimensional). We allow $\mathcal{D}$ to differ from the distribution from which environments for $\{\zeta_i\}_{i=1}^M$ are drawn.

Suppose that the robot's task is specified via a cost function and let $C(\pi; E)$ denote the cost incurred by a (deterministic) policy $\pi : \mathcal{O} \to \mathcal{A}$ when deployed in an environment E. Here, we assume that policy π belongs to a space Π of policies. We also allow policies that map *histories* of observations to actions by augmenting the observation space to keep track of observation sequences. The cost function is assumed to be bounded; without further loss of generality, we assume $C(\pi; E) \in [0, 1]$. We make no further assumptions on the cost function (e.g., continuity, smoothness, etc.).

Goal. Our goal is to utilize the expert demonstrations $\{\zeta_i\}_{i=1}^M$ along with the additional training environments in S to learn a policy that *provably generalizes* to novel environments S' drawn from the *unknown* distribution $\mathcal{D}$. In this work, we will employ a slightly more general formulation where we choose a *distribution* P over policies $\pi \in \Pi$ (instead of making a single deterministic choice). This will allow us to employ PAC-Bayes generalization theory. Our goal can then be formalized by the following optimization problem:

$$C^\star := \min_{P \in \mathcal{P}} C_{\mathcal{D}}(P), \;\; \text{where} \;\; C_{\mathcal{D}}(P) := \mathop{\mathbb{E}}_{E \sim \mathcal{D}} \mathop{\mathbb{E}}_{\pi \sim P} [C(\pi; E)], \tag{2}$$

and $\mathcal{P}$ refers to the space of probability distributions on the policy space Π. This optimization problem is challenging to tackle directly since the distribution $\mathcal{D}$ from which environments are drawn is not known to us. In the subsequent sections, we will demonstrate how to learn a distribution P over policies with a provable bound on the expected cost $C_{\mathcal{D}}(P)$, i.e., a provable guarantee on generalization to novel environments drawn from $\mathcal{D}$.

3 Approach

The training pipeline consists of two stages (Fig. 1). First, a "prior" distribution over policies P_0 is obtained by cloning multi-modal expert demonstrations. Second, the prior is "fine-tuned" by explicitly optimizing the PAC-Bayes generalization bound. The resulting "posterior" policy distribution P achieves strong empirical performance and generalization guarantees on novel environments.

3.1 Multi-Modal Behavioral Cloning using Latent Variables

The goal of the first training stage is to obtain a prior policy distribution P_0 by cloning expert demonstrations ζ_i. Behavioral cloning is a straightforward strategy to make robots mimic expert behavior. While simple discriminative models fail to capture diverse expert behavior, generative models such as variational autoencoders (VAEs) can embed such multi-modality in latent variables [11, 18]. We further use a conditional VAE (cVAE) [11] to condition the action output a on both the latent z and observation o (Fig. 2). The latent z encodes both o and a from expert demonstrations $\{\zeta_i\}_{i=1}^M$. In the example of grasping mugs, intuitively we can consider that z encodes the mug center location and the relative-to-center grasp pose from depth images and demonstrated grasps.

Both pieces of information are necessary for generating successful, diverse grasps along the rim using different sampled z (Fig. 4(a)).

While grasping mugs can be achieved by executing an open-loop grasp from above, other tasks such as pushing boxes and navigating indoor environments require continuous, closed-loop actions. We embed either a short sequence of observation/action pairs ($T = 3$ steps) or the entire trajectory (as many as 50 steps) into a single latent z. Thus a sampled latent state can represent local, reactive actions (e.g. turn left to avoid a chair during navigation), or global "plans" (e.g. push at the corner of an object throughout the horizon while manipulating it).

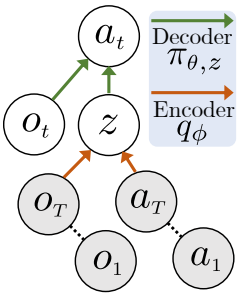


Figure 2: Graphical model of the cVAE

Fig. 2 shows the graphical model of the cVAE in our work. The encoder $q_\phi(z|o,a)$, parameterized by weights ϕ of a neural network, samples a latent variable z conditioned by each demonstration $\zeta_i := \{(o_t, a_t)\}_{t=1}^T$. The decoder, $\pi_{\theta,z} : o \mapsto a$, parameterized by the weights θ of another neural network and the sampled latent variable z, reconstructs the action from the observation at each step. Details of the neural network architecture are provided in A5. In the loss function below, the distribution $p(z)$ is chosen as a multivariate unit Gaussian. A KL regularization loss constrains the conditional distribution of z to be close to $p(z)$. Let $\mathcal{L}_{\text{rec}}(\pi_{\theta,z}(o_t), a_t)$ be the reconstruction loss between the predicted action and the expert's. The parameter $\lambda > 0$ balances the two losses:

$$\mathcal{L}(\phi, \theta; \zeta_i) = \mathrm{E}_{z\sim q_\phi(z|o_{1:T}, a_{1:T})}\left[\sum_{t=1}^{T} \mathcal{L}_{\text{rec}}\big(\pi_{\theta,z}(o_t), a_t\big)\right] + \lambda D_{\text{KL}}\big((q_\phi(z|o_{1:T}, a_{1:T})||p(z)\big). \quad (3)$$

The primary outcomes of behavioral cloning are: (i) a latent distribution $p(z) = \mathcal{N}(0, I)$ and (ii) weights θ^* of the decoder network $\pi_{\theta,z}$ that together encode multi-modal policies from experts. We now restrict the weights θ of the decoder network $\pi_{\theta,z}$ to θ^* giving rise to the space of policies $\Pi := \{\pi_{\theta^*,z} : \mathcal{O} \to \mathcal{A} \mid z \in \mathbb{R}^{n_z}\}$ parameterized by z. Hence, the latent distribution $p(z)$ can be equivalently viewed as a distribution on the space Π of policies. In the next section, we will consider $p(z)$ as a *prior* distribution P_0 on Π and "fine-tune" it by searching for a *posterior* distribution P in the space $\mathcal{P}$ of probability distributions on Π by solving (2). In particular, we choose $\mathcal{P}$ as the space of Gaussian probability distributions with diagonal covariance $\mathcal{N}(\mu, \Sigma)$. For the sake of notational convenience, let $\sigma \in \mathbb{R}^d$ be the element-wise square-root of the diagonal of Σ, and define $\psi := (\mu, \sigma)$, $P = \mathcal{N}_\psi := \mathcal{N}(\mu, \text{diag}(\sigma^2))$, and $P_0 = \mathcal{N}_{\psi_0} := \mathcal{N}(0, I)$.

3.2 Generalization Guarantees through PAC-Bayes Control

Although behavioral cloning provides a meaningful policy distribution P_0, the policies drawn from this distribution can fail when deployed in novel environments due to: unsafe or non-generalizable demonstrations by the expert, or the inability of the cVAE training to accurately infer the expert's policies. In this section, we leverage the PAC-Bayes Control framework introduced in [26, 27] to "fine-tune" the *prior* policy distribution P_0 and provide "certificates" of generalization for the resulting *posterior* policy distribution P. In particular, we will tune the distribution by approximately minimizing the true expected cost $C_{\mathcal{D}}(P)$ in equation (2), thus promoting generalization to environments drawn from $\mathcal{D}$ that are different from S. Although $C_{\mathcal{D}}(P)$ cannot be computed due to the lack of an explicit characterization of $\mathcal{D}$, the PAC-Bayes framework allows us to obtain an upper bound C_{PAC} for $C_{\mathcal{D}}(P)$, which can be computed despite our lack of knowledge of $\mathcal{D}$; see Theorem 1.

To introduce the PAC-Bayes generalization bounds, we will first define the *empirical cost* of P as the average expected cost across training environments in S:

$$C_S(P) := \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{z\sim P}[C(\pi_{\theta^*,z}; E)]. \quad (4)$$

The following theorem can then be used to bound the true expected cost $C_{\mathcal{D}}(P)$ from Sec. 2.

Theorem 1 (PAC-Bayes Bound for Control Policies; adapted from [26, 29]) *Let $P_0 \in \mathcal{P}$ be a prior distribution. Then, for any $P \in \mathcal{P}$, and any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over sampled environments $S \sim \mathcal{D}^N$, the following inequality holds:*

$$C_{\mathcal{D}}(P) \le C_{\text{PAC}}(P, P_0) := C_S(P) + \sqrt{R(P, P_0)}, \text{ where } R(P, P_0) := \frac{\text{KL}(P\|P_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}.$$

Intuitively, minimizing the upper bound C_{PAC} can be viewed as minimizing the empirical cost $C_S(P)$ along with a regularizer R that prevents overfitting by penalizing the deviation of the posterior from

the prior. Due to the presence of a "blackbox" physics simulator for rollouts, the gradient of $C_S(P)$ cannot be computed analytically. Thus we employ blackbox optimizers for minimizing C_{PAC}.

Optimizing PAC-Bayes bound using Natural Evolutionary Strategies. To minimize C_{PAC}, we use the class of blackbox optimizers known as Evolutionary Strategies (ES) [30] that estimate the gradient of a loss function through Monte-Carlo methods, without requiring an analytical gradient of the loss function. To minimize C_{PAC} using ES, we express the gradient of the empirical cost $C_S(P)$ (4) as an expectation w.r.t. the posterior distribution $P = \mathcal{N}_\psi$:

$$\nabla_\psi C_S(\mathcal{N}_\psi) = \frac{1}{N}\sum_{E\in S} \nabla_\psi \mathop{\mathbb{E}}_{z\sim\mathcal{N}_\psi}[C(\pi_{\theta^*,z}; E)] = \frac{1}{N}\sum_{E\in S} \mathop{\mathbb{E}}_{z\sim\mathcal{N}_\psi}[C(\pi_{\theta^*,z}; E)\nabla_\psi \ln \mathcal{N}_\psi(z)]. \quad (5)$$

Although the gradient of the regularizer R can be computed analytically, we found that it can heavily dominate the noisy gradient estimate of the empirical cost during training. Therefore, we compute its gradient using ES as well, expressing the regularizer in terms of an expectation on the posterior:

$$\nabla_\psi R = \frac{1}{2N}\nabla_\psi \text{KL}(\mathcal{N}_\psi \| \mathcal{N}_{\psi_0}) = \frac{1}{2N}\nabla_\psi \mathop{\mathbb{E}}_{z\sim\mathcal{N}_\psi}\left[\log \frac{\mathcal{N}_\psi(z)}{\mathcal{N}_{\psi_0}(z)}\right]. \quad (6)$$

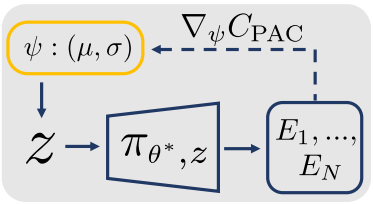


Figure 3: PAC-Bayes Control training loop. Each training environment E_i is simulated with a random policy z sampled from $\mathcal{N}_\psi$, which is then updated using gradient estimate of the bound.

In practice we use Natural Evolutionary Strategies (NES) [31] that transforms the ES gradient to the natural gradient [32] to accelerate training. During each epoch, for each of the N environments we sample a certain number of z's from the posterior $\mathcal{N}_\psi$ and then compute the corresponding empirical costs by performing rollouts in simulation. Sampled z's and their empirical costs $C(\pi_{\theta^*,z}; E)$ are then used in (5) and (6) to compute the gradient estimate, which is passed to the Adam optimizer [33] to update ψ. The training loop is visualized in Fig. 3.

Computing the final bound. After ES training, we can calculate the generalization bound using the optimal ψ^*. First, note that the empirical cost $C_S(P) = C_S(\mathcal{N}_{\psi^*})$ involves an expectation over the posterior and thus cannot be computed in closed form. Instead, it can be estimated by sampling a large number of policies $z_1, \cdots, z_L$ from $\mathcal{N}_{\psi^*}$: $\hat{C}_S(\mathcal{N}^*_\psi) := \frac{1}{NL}\sum_{E\in S}\sum_{i=1}^{L} C(\pi_{\theta^*,z_i}; E)$, and the error due to finite sampling can be bounded using a sample convergence bound $\bar{C}_S$ [34]. The final bound $C_{\text{bound}}(\mathcal{N}^*_\psi) \geq C_{\mathcal{D}}(\mathcal{N}^*_\psi)$ is obtained from $\bar{C}_S$ and $R(\mathcal{N}^*_\psi, P_0)$ by a slight tightening of C_{PAC} from Theorem 1 using the KL-inverse function [27]. Please refer to A1, A2, A3 for detailed derivations and implementations.

Overall, our approach provides generalization guarantees in novel environments for policies learned from imitation learning: as policies are randomly sampled from the posterior $\mathcal{N}^*_\psi$ and applied in test environments, the expected success rate over all test environments is guaranteed to be at least $1 - C_{\text{bound}}(\mathcal{N}^*_\psi)$ (with probability $1-\delta$ over the sampling of training environments; $\delta = 0.01$ for all examples in Sec. 4).

4 Experiments

We demonstrate the efficacy of our approach on three different robotic tasks: grasping a diverse set of mugs, planar box pushing with external vision feedback, and navigating in home environments with onboard vision feedback. Our experimental results demonstrate: (i) strong theoretical generalization bounds, (ii) tightness between theoretical bounds and empirical performance in test environments, and (iii) zero-shot generalization to hardware in challenging manipulation settings.

Expert demonstrations are collected in the PyBullet simulator [35] using a 3Dconnexion 6-dof mouse for manipulation and a keyboard for navigation; no data from the real robot or camera is used for the training. Following behavioral cloning using collected data, we fine-tune the policies using rollout costs in the PyBullet simulator as well. Results of manipulation tasks are then transferred to the real hardware with no additional training (zero-shot). More details of synthetic training (including code) and hardware experiments (including a video) are provided in A5 and A6.

4.1 Grasping a diverse set of mugs

The goal is to grasp and lift a mug from the table using a Franka Panda Arm (Fig. 5). An open-loop action $a_{\text{grasp}} = (x, y, z, \theta)$ is applied for each rollout and corresponds to desired 3D positions and

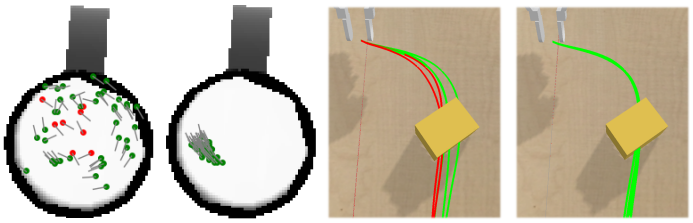


Figure 4: Sampled grasps [(a),(b)] and gripper trajectories when pushing boxes [(c),(d)] before and after polices being "fine-tuned". Grey tails in (a), (b) indicate grasp orientation. Red indicates failure and green indicates success. PAC-Bayes Control training "shrinks" the space of actions applied by policies.

yaw orientation of the grasp. The action is computed based on an observation of a 128×128 depth image from an overhead camera.

We gathered 50 mugs of diverse shapes and dimensions from the ShapeNet dataset [36]. These mugs are split into 3 sets for expert demonstrations, PAC-Bayes Control training environments S, and test environments S'. They are then randomly scaled in all dimensions into multiple different mugs. Their masses are sampled from a uniform distribution. Each training or test environment consists of a unique mug from the set and a unique initial SE(2) pose on the table. A rollout is considered successful (zero cost) if the center of mass (COM) of the mug is lifted by 10 cm and the gripper palm makes no contact with the mug; otherwise, a cost of 1 is assigned to the rollout.

Expert data. In each of 60 environments, we specify 5 grasp poses along the rim. The initial depth image of the scene and corresponding grasp poses are recorded. It takes about an hour to collect the 300 trials.

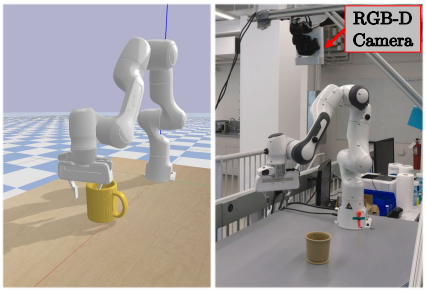


Figure 5: Grasping setup (left) in PyBullet simulation and (right) on hardware.

Prior performance. Each pair of initial observation and action from expert data is embedded into a latent $z \in \mathbb{R}^{10}$. Thus the length of time sequence in each demonstration ζ_i is $T_{\text{grasp}} = 1$. The reconstruction loss $\mathcal{L}_{\text{rec,grasp}}$ is a combination of l_2 and l_1 loss between predicted and expert's actions. The prior policy distribution achieves 83.3% success in novel environments in simulation. Shown in Fig. 4(a), the prior P_0 captures the multi-modality of expert data: different latent z sampled from P_0 generate a diverse set of grasps along the rim.

Posterior performance. $N = 500$ environments are used for fine-tuning via PAC-Bayes. The resulting PAC-Bayes bound $C^*_{\text{bound}} := C_{\text{bound}}(P)$ of the posterior P is 0.070. Thus, with probability 0.99 the optimized posterior policy is guaranteed to have an expected success rate of 93.0% in novel environments (assuming that they are drawn from the same underlying distribution $\mathcal{D}$ as the training examples). The policy is then evaluated on 500 test environments in simulation and the success rate is 98.4%. Fig. 4(b) shows sampled grasps of a mug using the posterior distribution, which are concentrated at a relatively fixed position compared to grasps along the rim sampled from the prior.

$1 - C^*_{\text{bound}}$	True expected success (estimate)		
	Prior in simulation	Posterior in simulation	Posterior on hardware
0.930	0.833	0.984	1.000, 1.000, 0.960

Table 1: Prior performance, posterior performance, and generalization guarantees in grasping mug example using $N = 500$ training environments.

Hardware implementation. The posterior policy distribution trained in simulation is deployed on the hardware setup (Fig. 5) in a zero-shot manner. We pick 25 mugs with a wide variety of shapes and materials (Fig. A5). Among three sets of experiments with different seeds (for sampling initial poses and latent z), the success rates are 100% (25/25), 100% (25/25), and 96% (24/25). The hardware results thus validate the PAC-Bayes bound trained in simulation (Table 1).

4.2 Planar box pushing with real-time visual feedback

In this example, we tackle the challenging task of pushing boxes of a wide range of dimensions to a target region across the table. Real-time external visual feedback is applied using a 150×150 overhead depth image of the whole environment at 5Hz. The observation comprises of the depth map augmented with the proprioceptive x/y positions of the end-effector. The action is the desired relative-to-current x/y displacement of the end-effector $a_{\text{push}} = (\Delta x, \Delta y)$. A low-level Jacobian-

based controller tracks desired pose setpoints. The gripper fingers maintains a fixed height from the table and a fixed orientation, and the gripper width is set to 3 cm to maintain a two-point contact.

Rectangular boxes are generated by sampling the three dimensions (4-8 cm in length, 6-10 cm in width, and 5-8 cm in height) and the mass (0.1-0.2 kg). Each environment again consists of a unique box and a unique initial SE(2) pose. Based on the dimensions of starting and target regions, we define an "Easy" task and a "Hard" one (Fig. 6). A continuous cost is assigned based on how far the COM of the box is from the target region at the end of a rollout.

Expert data. In each of 150 environments, we specify 2 different, successful pushing trajectories. The overhead depth image, the end-effector pose, and the desired end-effector displacement are recorded at 5Hz. It takes about 90 minutes to collect the 300 trials.

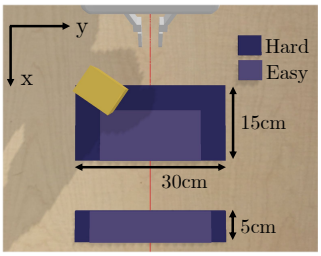


Figure 6: Starting and target regions for "Easy" and "Hard" tasks ("Hard" overlaps "Easy").

Prior performance. Entire trajectories of observations and actions from expert data are embedded into latent $z \in \mathbb{R}^5$. Thus T_{push} is the total number of steps in each trial. The reconstruction loss $\mathcal{L}_{\text{rec,push}}$ is again a combination of l_2 and l_1 loss between predicted and expert's actions. The prior policy distribution is able to achieve 84.2% success in novel environments in the "Easy" task, and 74.9% in the "Hard" task. Fig. 4(c) shows a challenging environment of "Hard" task where the box starts far away from the red centerline and with a large yaw angle relative to the gripper. While experts always push at the corner of the box in demonstrations, the prior P_0 learned via behavioral cloning fails to imitate this behavior perfectly.

Posterior performance. We train both tasks using 500, 1000, and 2000 training environments. The resulting PAC-Bayes bound and empirical success rates across 2000 test environments are shown in Table 2. Posterior performances are improved by about 10% from the priors. Fig. 4(c,d) shows that compared to prior policies, posterior policies perform better at the challenging task as the robot learns to push at the corner consistently.

Task difficulty	N (# of training environments)	$1 - C^*_{\text{bound}}$	True expected success (estimate)		
			Prior in simulation	Posterior in simulation	Posterior on hardware
Easy	500	0.861	0.842	0.929	-
Easy	1000	0.888		0.937	0.800, 0.867, 0.933
Easy	2000	0.904		0.945	-
Hard	500	0.754	0.749	0.863	-
Hard	1000	0.791		0.864	0.800, 0.800, 0.800
Hard	2000	0.810		0.864	-

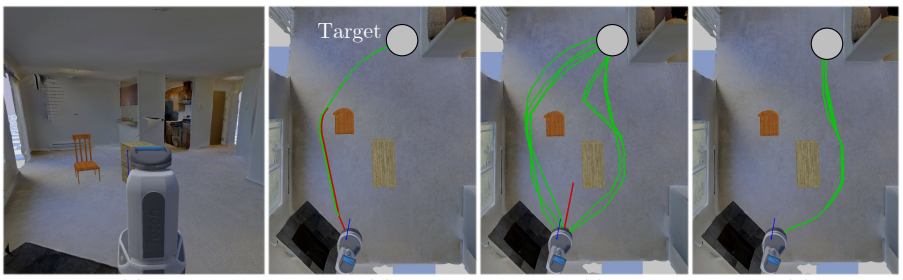
Table 2: Prior performance, posterior performance, and generalization guarantees in pushing box example.

Hardware implementation. The posterior policy distributions trained using 1000 environments are deployed on the real arm. For both "Easy" and "Hard" tasks, three set of experiments with different seeds are performed with 15 rectangular blocks (Fig. A7). The success rates are shown in Table 2. Note that the result for the "Easy" task falls short of the bound in some seeds. We suspect that the sim2real performance is affected by imperfect depth images from the real camera and minor differences in dynamics between simulation and the actual arm.

4.3 Vision-based indoor navigation

In this example, a Fetch mobile robot needs to navigate around furniture of different shapes, sizes, and initial poses, before reaching a target region in a home environment (Fig. 7). We use iGibson [37] to render photorealistic visual feedback in PyBullet simulations. Again we consider a challenging setting where the robot executes actions given front-view camera images and without any extra knowledge of the map. At each step, the policy takes a 200×200 RGB-D image and chooses the motion primitive with the highest probability from the four choices (move forward, move backward, turn left, and turn right), $a_{\text{nav}} = \arg\max(p_{\text{forward}}, p_{\text{backward}}, p_{\text{left}}, p_{\text{right}})$.

We collect 293 tables and 266 chairs from the ShapeNet dataset [36]. A fixed scene (Sodaville) from iGibson is used for all environments. One table and one chair are randomly spawned between the fixed starting location and the target region in each environment. The SE(2) poses of the furniture are drawn from a uniform distribution. A rollout is successful if the robot reaches the target within 100 steps without colliding with the furniture and the wall.



(a) (b) (c) (d)

Figure 7: (a) robot view in navigation, (b) 10 trajectories sampled from a single-modal prior, (c) from a multi-modal prior, and (d) from a posterior. Red indicates failure and green indicates success.

Expert data. In each of 100 environments, two different, successful robot trajectories are collected. The front-view RGB-D image from the robot and the motion primitive applied at each step are recorded. It takes about 90 minutes to collect the 200 demonstrations.

Prior performance. Short sequences ($T_{\text{nav}} = 3$) of observations and actions from expert data are embedded into latent $z \in \mathbb{R}^{10}$. The reconstruction loss $\mathcal{L}_{\text{rec,nav}}$ is the cross-entropy loss between predicted action probabilities and expert's actions. Before each rollout, a single latent z is sampled and then applied as the policy for all steps. The prior policy distribution is able to achieve 65.4% success in novel environments in simulation. Fig. 7(c) shows the diverse trajectories generated by the trained cVAE.

Posterior performance. The prior is "fine-tuned" using 500 and 1000 training environments. The resulting PAC-Bayes bound and empirical success rates across 2000 test environments are shown in Table 3. Fig. 7(d) shows that similar to grasping and pushing examples, the robot follows relatively the same trajectories in the same environment when executing "fine-tuned" posterior policies. In that environment, the robot now consistently chooses the bigger gap on the right to navigate and avoids the narrow one on the left chosen by some prior policies.

N (# of training environments)	$1 - C^*_{\text{bound}}$	True expected success (estimate)	
		Prior in simulation	Posterior in simulation
500	0.723	0.654	0.791
1000	0.741		0.799

Table 3: Prior performance, posterior performance, and generalization guarantees in indoor navigation example

We also investigate the benefit of "fine-tuning" a multi-modal prior distribution vs. a single-modal one. Fig. 7(b,c) show the differences in the policies sampled from these two priors in the same environment. We find that the multi-modality of the prior accelerates PAC-Bayes training and leads to better empirical performance of the posterior. Please refer to A4 for full discussions.

5 Conclusion

We have presented a framework for providing generalization guarantees for policies learned via imitation learning. Policies are trained through two stages: (i) a "prior" policy distribution is learned through multi-modal behavior cloning to mimic an expert's behavior, and (ii) the prior is then "fine-tuned" using PAC-Bayes Control framework by explicitly optimizing the generalization bound. The resulting "posterior" distribution P over policies achieves strong empirical performance and generalization guarantees in novel environments, which are verified in simulation of manipulation and navigation tasks, as well in hardware experiments of manipulation tasks.

Challenges and future work: we find training the cVAE requires significant tuning and can be difficult with embedding long sequences of high-dimensional images as input. This limits the framework from handling long-horizon tasks where longer sequences (if not whole trajectories) of observation/action pairs need to be embedded in the latent space and information from images varies significantly along the rollout. For example, tasks like pouring water involve multiple stages of manipulation — picking up the mug, rotating it to pour, and putting it back on the table. We are looking into learning separate latent distributions that encode different stages of the task. The other exciting direction is to apply the Dataset Augmentation ("DAgger") Algorithm [22] that constantly injects additional expert knowledge into training as the policy is refined.

Acknowledgments

The authors were partially supported by the Office of Naval Research [Award Number: N00014-18-1-2873], the Google Faculty Research Award, and the Amazon Research Award.

References

[1] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[2] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[3] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[4] D. A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[5] M. Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3(Oct):233–269, 2002.

[6] J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 439–446, 2003.

[7] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[8] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

[9] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

[10] J. Morton and M. J. Kochenderfer. Simultaneous policy learning and latent state inference for imitating driver behavior. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[11] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[12] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[13] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[15] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[16] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2018.

[17] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1235–1245, 2017.

[18] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5320–5329, 2017.

[19] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765, 2018.

[20] F.-I. Hsiao, J.-H. Kuo, and M. Sun. Learning a multi-modal policy via imitating demonstrations with mixed behaviors. *arXiv preprint arXiv:1903.10304*, 2019.

[21] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[22] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[23] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[24] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

[25] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[26] A. Majumdar and M. Goldstein. PAC-Bayes Control: synthesizing controllers that provably generalize to novel environments. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

[27] A. Majumdar, A. Farid, and A. Sonar. PAC-Bayes Control: Learning policies that provably generalize to novel environments. *arXiv preprint arXiv:1806.04225*, 2019.

[28] S. Veer and A. Majumdar. Probably approximately correct vision-based planning using motion primitives. *arXiv preprint arXiv:2002.12852*, 2020.

[29] A. Maurer. A note on the PAC-Bayesian theorem. *arXiv preprint cs/0411099*, 2004.

[30] H.-G. Beyer and H.-P. Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

[31] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(27):949–980, 2014.

[32] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] J. Langford and R. Caruana. (not) bounding the true error. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 809–816, 2002.

[35] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[36] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[37] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

Appendix

A1 Natural Evolutionary Strategies

Below are the implementation details for training the posterior distribution with Natural Evolutionary Strategies (NES) (Sec. 3.2). Our objective is to minimize the upper bound C_{PAC} in Thm. 1. We will first express the gradient of C_{PAC} w.r.t. $\psi := (\mu, \sigma)$ as follows (using (5) and (6)):

$$
\begin{aligned}
\nabla_\psi C_{\text{PAC}}(P, P_0) &= \nabla_\psi C_S(P) + \nabla_\psi \sqrt{R(P, P_0)} \ , \\
&= \nabla_\psi C_S(P) + \frac{1}{2\sqrt{R}} \nabla_\psi R(P, P_0) \ , \\
&= \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{z \sim \mathcal{N}_\psi} [C(\pi_{\theta^*, z}; E) \nabla_\psi \ln \mathcal{N}_\psi(z)] + \frac{1}{4N\sqrt{R}} \nabla_\psi \mathop{\mathbb{E}}_{z \sim \mathcal{N}_\psi} \Big[\log \frac{\mathcal{N}_\psi(z)}{\mathcal{N}_{\psi_0}(z)} \Big] \\
&= \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{z \sim \mathcal{N}_\psi} [C(\pi_{\theta^*, z}; E) \nabla_\psi \ln \mathcal{N}_\psi(z)] + \frac{1}{4N\sqrt{R}} \mathop{\mathbb{E}}_{z \sim \mathcal{N}_\psi} \Big[\log \frac{\mathcal{N}_\psi(z)}{\mathcal{N}_{\psi_0}(z)} \nabla_\psi \ln \mathcal{N}_\psi(z) \Big] \\
&= \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{z \sim \mathcal{N}_\psi} \Big[\underbrace{\Big(C(\pi_{\theta^*, z}; E) + \frac{1}{4N\sqrt{R}} \log \frac{\mathcal{N}_\psi(z)}{\mathcal{N}_{\psi_0}(z)} \Big) \nabla_\psi \ln \mathcal{N}_\psi(z)}_{C_{ES}(z; E)} \Big]
\end{aligned}
$$

To estimate the expectation on the right-hand side (RHS) of the above equation, we sample a few z's from $\mathcal{N}_\psi$ and average C_{ES} over them. In particular, we perform antithetic sampling of z (i.e., for each z that we sample, we also evaluate C_{ES} for $-z$) to reduce the variance of the gradient estimate [1]. This gives us the following ES gradient estimate for m samples of z ($2m$ with antithetic sampling):

$$
\nabla_\psi C_{\text{PAC}}(P, P_0) \approx \frac{1}{N} \sum_{E \in S} \left[\frac{1}{2m} \sum_{i=1}^{m} (C_{ES}(z_i; E) + C_{ES}(-z_i; E)) \right]
$$

Finally, using the Fisher information matrix F_ψ (of the normal distribution $\mathcal{N}_\psi$ w.r.t. ψ) we compute an estimate of the natural gradient [2], denoted by $\tilde{\nabla}_\psi$, from the above equation:

$$
\tilde{\nabla}_\psi C_{\text{PAC}}(P, P_0) \approx F_\psi^{-1} \frac{1}{N} \sum_{E \in S} \left[\frac{1}{2m} \sum_{i=1}^{m} (C_{ES}(z_i; E) + C_{ES}(-z_i; E)) \right].
$$

The natural gradient estimate computed above is passed to the Adam optimizer [3] to update the belief distribution parameterized by ψ. In practice we use $\psi = (\mu, \log \sigma^2)$ instead of (μ, σ) to avoid imposing the strict positivity constraint on σ for the gradient update with Adam.

A2 Derivations of the final bound

The derivations follow [4, 5]. Following Sec. 3.2, first the empirical training cost is estimated by sampling a large number of policies $z_1, ..., z_L$ from the optimized posterior distribution $\mathcal{N}_{\psi^*}$, and averaging over all N training environments in S:

$$
\hat{C}_S(\mathcal{N}_\psi^*) := \frac{1}{NL} \sum_{E \in S} \sum_{i=1}^{L} C(\pi_{\theta^*, z_i}; E). \tag{A1}
$$

Next, the error between $\hat{C}_S(\mathcal{N}_{\psi^*})$ and $C_S(\mathcal{N}_{\psi^*})$ can be bounded using a sample convergence bound [6] $\bar{C}_S$, which is an application of the relative entropy verision of the Chernoff bound for random variables (i.e., costs) bounded in $[0, 1]$ and holds with probability $1 - \delta'$:

$$
C_S(\mathcal{N}_{\psi^*}) \leq \bar{C}_S(\mathcal{N}_{\psi^*}; L, \delta') := \text{KL}^{-1}(\hat{C}_S(\mathcal{N}_{\psi^*}) || \frac{1}{L} \log(\frac{2}{\delta'})). \tag{A2}
$$

where KL^{-1} refers to the KL-inverse function and can be computed using a Relative Entropy Program (REP) [4]. $\text{KL}^{-1} : [0, 1] \times [0, \infty) \to [0, 1]$ is defined as:

$$
\text{KL}^{-1}(p||c) := \sup\{q \in [0, 1] \mid \text{KL}(p||q) \leq c\}. \tag{A3}
$$

The KL-inverse function can also provide the following bound on the true expected cost $C_{\mathcal{D}}(P)$ (Theorem 2 from [5]):

$$C_{\mathcal{D}}(P) \leq \mathrm{KL}^{-1}\Big(C_S(P)||\frac{\mathrm{KL}(P||P_0) + \log\frac{2\sqrt{N}}{\delta}}{N}\Big). \tag{A4}$$

Now combining inequalities (A2) and (A4) using the union bound, the following final bound C_{bound} holds with probability at least $1-\delta-\delta'$:

$$C_{\mathcal{D}}(\mathcal{N}_{\psi^*}) \leq C_{\text{bound}}(\mathcal{N}_{\psi^*}) := \mathrm{KL}^{-1}\left(\bar{C}_S(\mathcal{N}_{\psi^*}; L, \delta')||\frac{\mathrm{KL}(\mathcal{N}_{\psi^*}||P_0) + \log\frac{2\sqrt{N}}{\delta}}{N}\right). \tag{A5}$$

A3 Algorithms for the two training stages

Algorithm 1 Multi-modal Behavioral Cloning

Require: $q_\phi, \pi_{\theta,z}, \{\zeta_i\}_{i=1}^M$ ▷ Encoder network, decoder network, demonstrations

for $i = 1, \cdots, n_{\text{iter}}$ **do**

  $\phi, \theta \leftarrow \arg\min_{\phi,\theta} \sum_{\zeta_i}[(\sum_{t=1}^T \mathcal{L}_{\text{rec}}(\pi_{\theta,z}(o_t), a_t)) - \lambda D_{\text{KL}}((q_\phi(z|o_{1:T}, a_{1:T})||p(z))]$

  ▷ cVAE training

end for

return $\pi_{\theta^*,z}, P_0 = \mathcal{N}_{\psi_0} := \mathcal{N}(0, I)$ ▷ Optimized decoder network, prior over policies

Algorithm 2 PAC-Bayes Policy "Fine-tuning"

Require: $\pi_{\theta^*,z}, P_0, S = \{E_1, ..., E_N\}, \delta, \delta'$ ▷ Policy (decoder) network, prior over policies

▷ training environments, probability thresholds

for $i = 1, \cdots, n_{\text{iter}}$ **do**

  Sample $z_{j,k} \sim P$ and $-z_{j,k}$ for $j = 1, ..., N, k = 1, ..., m$

  Perform rollouts to get cost $C(\pi_{\theta^*,z_{j,k}}; E_j)$

  Compute $\tilde{\nabla}_\psi C_{\text{PAC}}(P, P_0)$, and update $P = \mathcal{N}_\psi$

end for

Sample $z_{j,l} \sim P$ for $j = 1, ...N, l = 1, ..., L$

Perform rollouts to get the estimate empirical training cost $\hat{C}_S(P)$ using (A1)

Compute the sample convergence bound $\bar{C}_S(P)$ using (A2)

Compute the final bound $C_{\text{bound}}(P)$ using (A5)

return $P = \mathcal{N}_{\psi^*}, C_{\text{bound}}(P)$ ▷ Posterior over policies, PAC-Bayes generalization bound

A4 Benefit of using a multi-modal prior policy distribution

As shown in Fig. 7, the prior policy distribution for indoor navigation can exhibit either uni-modal or multi-modal behavior. Uni-modal prior has low entropy while multi-modal prior has higher entropy. We expect that a prior with high entropy can benefit PAC-Bayes "fine-tuning" as opposed to a low-entropy prior; we investigate this further in the indoor navigation example and report the training curves with the low- and high-entropy priors in Fig. A1. While the two priors achieve similar empirical performance before being "fine-tuned", the prior with higher entropy trains faster and achieves better empirical performance on test environments at the end of "fine-tuning" (0.799 vs. 0.706 in success rate).

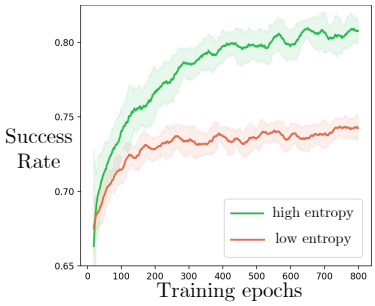


Figure A1: Empirical success rates over all training environments when "fine-tuning" priors with low entropy vs. higher entropy.

As illustrated in Fig. 7, the prior distribution with low entropy always picks the small gap on the left to navigate. Although the policies were successful in behavior cloning environments, they might fail in the "fine-tuning" training environments where the gap can shrink or there can be an occluded piece of furniture behind the gap. Hence "fine-tuning" can be difficult as the prior always chooses that specific route. Instead, the prior with higher entropy "encourages" the robot to try different directions, and is intuitively easier to "adapt" to new environments. "Fine-tuning" then picks the better policy among all available ones for each environment.

A5 Synthetic training details

All behavioral cloning training is run on a desktop machine with Intel i9-9820X CPU and a Nvidia Titan RTX GPU. PAC-Bayes training for manipulation tasks is performed on an Amazon Web Services (AWS) c5.24xlarge instance that has 96 threads. PAC-Bayes training for the navigation task is done using an AWS g4dn.metal instance that has 64 threads and 8 Nvidia Tesla T4 GPUs. It took about 3 hours to fine-tune the grasping policy with 500 training environments, and about 8 hours for pushing with 500 environments (a GPU instance could have been used to accelerate model inferences). The navigation task is more computationally intensive as it requires GPUs to render the indoor scene - it took about 30 hours with 500 training environments.

Choice of latent dimensions We find that a relatively small latent dimension (i.e. less than 10) is sufficient to encode multi-modality of the demonstrations. We use 10 for both grasping and indoor navigation, and 5 for pushing as the demonstrations are less multi-modal. It is possible to use an even smaller dimension and achieve similar empirical performance of the prior, but posterior performance could suffer as the small dimension constrains fine-tuning. We also find difficulty in learning a structured latent space of higher dimension (i.e. more than 20) for behavioral cloning.

A5.1 Grasping mugs

cVAE architecture.

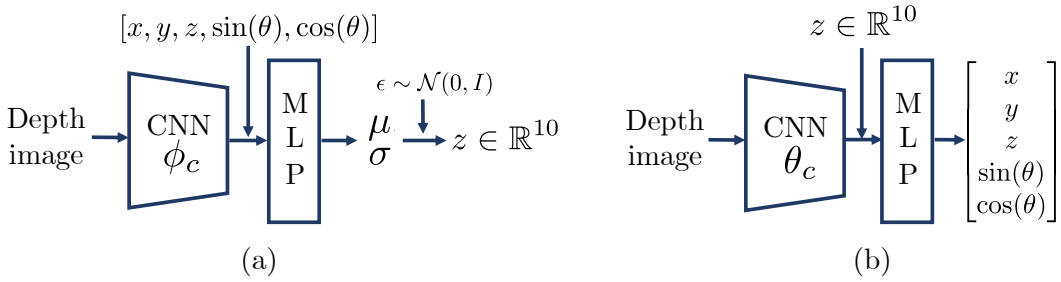


Figure A2: (a) Encoder, and (b) decoder of the cVAE for grasping mugs example.

In both the encoder and decoder, image features are generated through a spatial-softmax layer after the convolutional layers (CNN). In the encoder, the action is appended to image features before being passed into a multi-layer perceptron (MLP). In the decoder, the sampled latent z is appended to the image features before being passed into an MLP. Sine and cosine encodings are used for yaw angle action. A learning rate of 1e-3 and a weight decay rate of 1e-5 are used for cVAE training.

Reconstruction loss function of the CVAE. $\mathcal{L}_{\text{rec,grasp}}$ is a combination of l_1 and l_2 losses between the predicted actions and the expert's:

$$\mathcal{L}_{\text{rec,grasp}} = ||a_{\text{pred}} - a_{\text{expert}}||_1 + 0.1||a_{\text{pred}} - a_{\text{expert}}||_2$$

Environment setup.

- The diameter of the mugs is sampled uniformly from [8.5 cm, 13 cm].
- The mass of the mugs is sampled uniformly from [0.1 kg, 0.5 kg].
- The friction coefficients for the environment are 0.3 for lateral and 0.01 for torsional.
- The moment of inertia of the mugs is determined by the simulator assuming uniform density.
- The initial SE(2) pose of the mugs is sampled uniformly from [0.45 cm, 0.55 cm] in x, [-0.05 cm, 0.05 cm] in y, and $[-\pi, \pi]$ in yaw (all relative to robot base).

Posterior distribution (for $N = 1000$ training environments).

$\mu = [0.033, 0.319, 0.197, -0.846, -1.040, 0.052, -0.507, 0.273, -0.657, -0.357]$.

$\sigma = [1.062, 0.997, 0.923, 0.522, 0.229, 0.966, 1.012, 0.960, 0.236, 0.866]$.

Final bound. C_{bound} is computed using $\delta = 0.009, \delta' = 0.001$, and $L = 10000$.

A5.2 Pushing boxes

cVAE architecture.

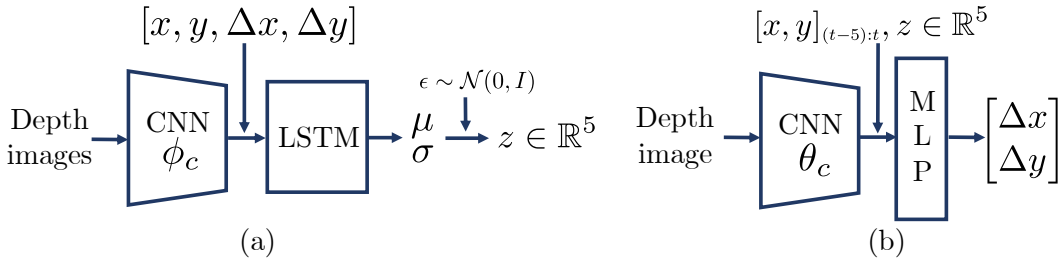


Figure A3: (a) Encoder, and (b) decoder of the cVAE for pushing boxes example.

In both the encoder and decoder, image features are generated through a spatial-softmax layer after the CNN. In the encoder, the action $(\Delta x, \Delta y)$ and the proprioceptive state (x, y) are appended to image features before being passed into an MLP. In the decoder, the sampled latent z and a history (5 steps) of the proprioceptive state are appended to the image features before being passed into a MLP. A learning rate of 1e-3 and a weight decay rate of 1e-5 are used for cVAE training.

Reconstruction loss function of the CVAE. $\mathcal{L}_{\text{rec,push}}$ is again a combination of l_1 and l_2 losses between the predicted actions and the expert's.

$$\mathcal{L}_{\text{rec,push}} = ||a_{\text{pred}} - a_{\text{expert}}||_1 + 3||a_{\text{pred}} - a_{\text{expert}}||_2$$

Environment setup.

- The dimensions of the rectangular boxes are sampled uniformly from [4 cm, 8 cm] in length, [6 cm, 10 cm] in width, and [5 cm, 8 cm] in height.
- The mass of the boxes is sampled uniformly from [0.1 kg, 0.2 kg].
- The friction coefficients for the environment are 0.3 for lateral and 0.01 for torsional.
- The moment of inertia of the boxes is determined by the simulator assuming uniform density.
- For the "Easy" task, the initial SE(2) pose of the boxes is sampled uniformly from [0.55 cm, 0.65 cm] in x, [-0.10 cm, 0.10 cm] in y, and $[-\pi/4, \pi/4]$ in yaw (all relative to robot base). The dimensions of the target region are [0.75 cm, 0.80 cm] in x and [-0.12 cm, 0.12 cm] in y.
- For the "Hard" task, the initial SE(2) pose of the boxes is sampled uniformly from [0.50 cm, 0.65 cm] in x, [-0.15 cm, 0.15 cm] in y, and $[-\pi/4, \pi/4]$ in yaw (all relative to robot base). The dimensions of the target region are [0.75 cm, 0.80 cm] in x and [-0.15 cm, 0.15 cm] in y.

Posterior distribution (for $N = 1000$ training environments).

For the "Easy" task:

$\mu = [0.690, -0.098, -0.733, 0.361, 1.947]$, and $\sigma = [0.829, 0.373, 0.346, 0.281, 0.255]$.

For the "Hard" task:

$\mu = [0.868, 0.531, -0.877, 0.172, 2.114]$, and $\sigma = [0.892, 0.517, 0.491, 0.445, 0.421]$.

Final bound. C_{bound} is computed using $\delta = 0.009, \delta' = 0.001$, and $L = 25000$.

A5.3 Indoor navigation

cVAE architecture.

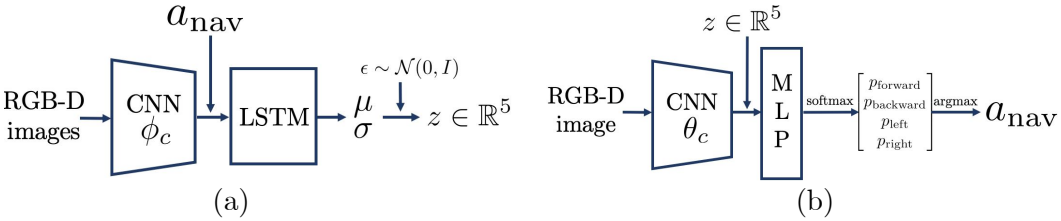


Figure A4: (a) Encoder, and (b) decoder of the cVAE for indoor navigation example.

The CNNs in the encoder and decoder (ϕ_c and θ_c) share weights. Two separate CNNs are used for RGB and depth channels of the images. In both the encoder and decoder, image features are generated through a spatial-softmax layer after the CNN. In the encoder, the action (one-hot encoding of the motion primitives) is appended to image features before being passed into a single LSTM layer. In the decoder, the sampled latent z is appended to the image features before being passed into an MLP. The output of the MLP is passed through a softmax layer to get normalized probabilities for the four motion primitives. Finally the action is chosen as the $\arg\max$ of the four. A learning rate of 1e-3 and a weight decay rate of 1e-5 are used for cVAE training.

Reconstruction loss function of the CVAE. $\mathcal{L}_{\text{rec,nav}}$ is the cross entropy loss between the predicted action probabilities and the expert's. One-hot encoding is used for expert's actions.

Environment setup.

- The step length for "move forward/backward" is fixed as 20 cm, and the turning angle for "turn left/right" is fixed as 0.20 radians.
- Instead of physically simulating the robot movement, we change the base position and orientation of the robot at each step and check its collisions with the furniture and the wall.
- The arm on the Fetch robot is removed from the robot URDF file to save computations.
- The initial SE(2) pose of the furniture is sampled uniformly from [-3.0 m, -1.0 m] in x, [-1.0 m, 1.5 m] in y, and $[-\pi/2, \pi/2]$ in yaw (all relative to the world origin in the Sodaville scene from iGibson [7]).
- A red snack box from the YCB object dataset [8] is placed at the target region in each environment.

Posterior distribution (for $N = 1000$ training environments).

$\mu = [-1.600, 0.704, 1.330, 0.233, 0.004, 0.735, 1.447, 0.630, 1.843, 1.132]$.

$\sigma = [0.166, 0.885, 0.947, 0.973, 0.830, 0.816, 0.962, 0.860, 0.848, 0.965]$.

Final bound. The final bound C_{bound} is computed using $\delta = 0.009, \delta' = 0.001$, and $L = 25000$.

A6 Hardware experiment details

All hardware experiments are performed using a Franka Panda arm and a Microsoft Azure Kinect RGB-D camera. Robot Operating System (ROS) Melodic package (on Ubuntu 18.04) is used to integrate robot arm control and perception.

A6.1 Grasping mugs

Figure A5: All 25 mugs used in hardware experiments. Some of these mugs are very small or thin at the rim, raising challenges in perception using the real camera and requiring precise grasps.

Figure A6: The only mug failed to be grasped during one set of trials. It has a unique shape with a larger diameter at the bottom than at the rim.

No.	Material	Dimensions (diameter and height, cm)	Weight (g)
1	Ceramic	8.5, 11.6	402.5
2	Ceramic	9.6, 11.9	606.1
3	Ceramic	8.7, 8.8	194.1
4	Ceramic	8.3, 9.8	302.1
5	Ceramic	8.3, 9.6	337.5
6	Ceramic	9.7, 8.9	423.6
7	Ceramic	9.6, 10.3	530.3
8	Ceramic	8.5, 10.5	349.0
9	Ceramic	9.6, 10.6	453.8
10	Ceramic	8.7, 11.9	426.6
11	Ceramic	9.7, 11.1	360.3
12	Ceramic	11.2, 10.2	443.6
13	Ceramic	8.3, 9.7	350.3
14	Ceramic	8.2, 9,5	351.1
15	Ceramic	12.7, 9.8	426.6
16	Ceramic	9.0, 10.7	354.4
17	Ceramic	7.7, 11.9	400.5
18	Ceramic	9.8, 11.5	342.6
19	Ceramic	8.6, 11.7	409.4
20	Ceramic	9.7, 10.9	373.3
21	Rubber	6.8, 7.1	85.7
22	Rubber	7.8, 7.2	100.8
23	Rubber	9.3, 10.8	124.0
24	Stainless steel	9.2, 8.1	130.8
25	Plastic	10.6, 9.7	171.4

Table A1: Materials, dimensions, and weights of all 25 mugs used in hardware experiments

A6.2 Pushing boxes

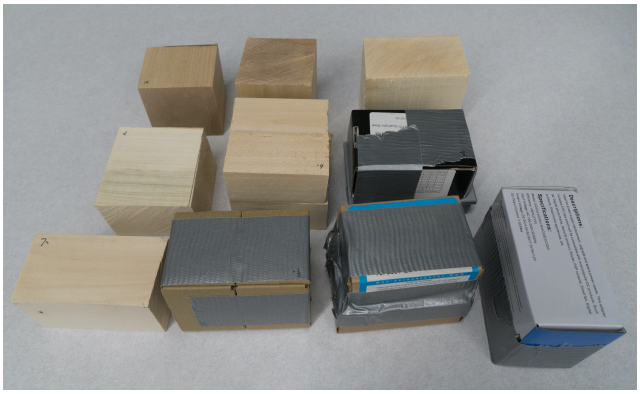


Figure A7: All boxes used in hardware experiments. Some wood blocks are used in more than one trial as they can be placed in different orientations.

No.	Material	Dimensions (cm)	Weight (g)
1	Wood	7.2, 6.4, 6.4	195.6
2	Wood	6.4, 7.2, 6.4	195.6
3	Wood	6.4, 6.4, 7.2	195.6
4	Wood	5.6, 6.4, 6.4	145.2
5	Wood	6.4, 6.4, 5.6	145.2
6	Wood	5.0, 8.0, 7.0	120.5
7	Wood	5.0, 7.0, 8.0	120.5
8	Wood	5.0, 9.0, 7.5	143.3
9	Wood	5.0, 7.5, 9.0	143.3
10	Wood	7.5, 8.0, 7.5	167.8
11	Wood	5.0, 8.0, 5.0	126.7
12	Cardboard	10.0, 6.0, 6.0	105.5
13	Cardboard	5.8, 9.0, 7.3	331.8
14	Cardboard	6.5, 9.2, 7.5	382.1
15	Cardboard	5.4, 10.4, 7.4	226.1

Table A2: Materials, dimensions, and weights of all 15 rectangular boxes used in hardware experiments. Cardboard boxes are filled with weights. Some of the entries have the same weight since they refer to the same box with different orientations.

References

[1] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[2] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(27):949–980, 2014.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] A. Majumdar, A. Farid, and A. Sonar. PAC-Bayes Control: Learning policies that provably generalize to novel environments. *arXiv preprint arXiv:1806.04225*, 2019.

[5] S. Veer and A. Majumdar. Probably approximately correct vision-based planning using motion primitives. *arXiv preprint arXiv:2002.12852*, 2020.

[6] J. Langford and R. Caruana. (not) bounding the true error. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 809–816, 2002.

[7] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[8] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *Proceedings of International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.