Behavior Transformers: Cloning k modes with one stone

Anonymous Author(s)
Affiliation
Address
email

Abstract

While behavior learning has made impressive progress in recent times, it lags behind computer vision and natural language processing due to its inability to leverage large, human-generated datasets. Human behaviors have wide variance, multiple modes, and human demonstrations typically do not come with reward labels. These properties limit the applicability of current methods in Offline RL and Behavioral Cloning to learn from large, pre-collected datasets. In this work, we present Behavior Transformer (BeT), a new technique to model unlabeled demonstration data with multiple modes. BeT retrofits standard transformer architectures with action discretization coupled with a multi-task action correction inspired by offset prediction in object detection. This allows us to leverage the multi-modal modeling ability of modern transformers to predict multi-modal continuous actions. We experimentally evaluate BeT on a variety of robotic manipulation and self-driving behavior datasets. We show that BeT significantly improves over prior state-of-the-art work on solving demonstrated tasks while capturing the major modes present in the pre-collected datasets. Finally, through an extensive ablation study, we analyze the importance of every crucial component in BeT. Videos of behavior generated by BeT are available here: https://submission0.github.io.

1 Introduction

Creating agents that can behave intelligently in complex environments has been a longstanding problem in machine learning. Although Reinforcement Learning (RL) has made significant advances in behavior learning, its success comes at the cost of high sample complexity [49, 20, 1]. Without priors on how to behave, state-of-the-art RL methods require online interactions on the order of 1-10M 'reward-labeled' samples for benchmark control tasks [73]. This is in stark contrast to vision and language tasks, where pretrained models and data-driven priors are the norm [16, 10, 27, 5], which allows for efficient downstream task solving.

So how do we learn behavioral priors from pre-collected data? One option is offline RL [39], where offline datasets coupled with conservative policy optimization can learn task-specific behaviors. However, such methods have yet to tackle domains where task-specific reward labels are not present. Without explicit reward labels, imitation learning, particularly behavior cloning, is a more fitting option [59, 8, 69]. Here, given behavior data $\mathcal{D} \equiv \{s_t, a_t\}$, behavior models can be trained to predict actions $f_\theta(s_t) \rightarrow a_t$ through supervised learning. When demonstration data is plentiful, such approaches have found impressive success in a variety of domains from self-driving [59, 13] to robotic manipulation [76, 53]. Importantly, it requires neither online interactions nor reward labels to optimize behaviors.

However, state-of-the-art behavior cloning methods often make a fundamental assumption – that the data is drawn from a unimodal expert solving a single task. This assumption is often baked in to the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

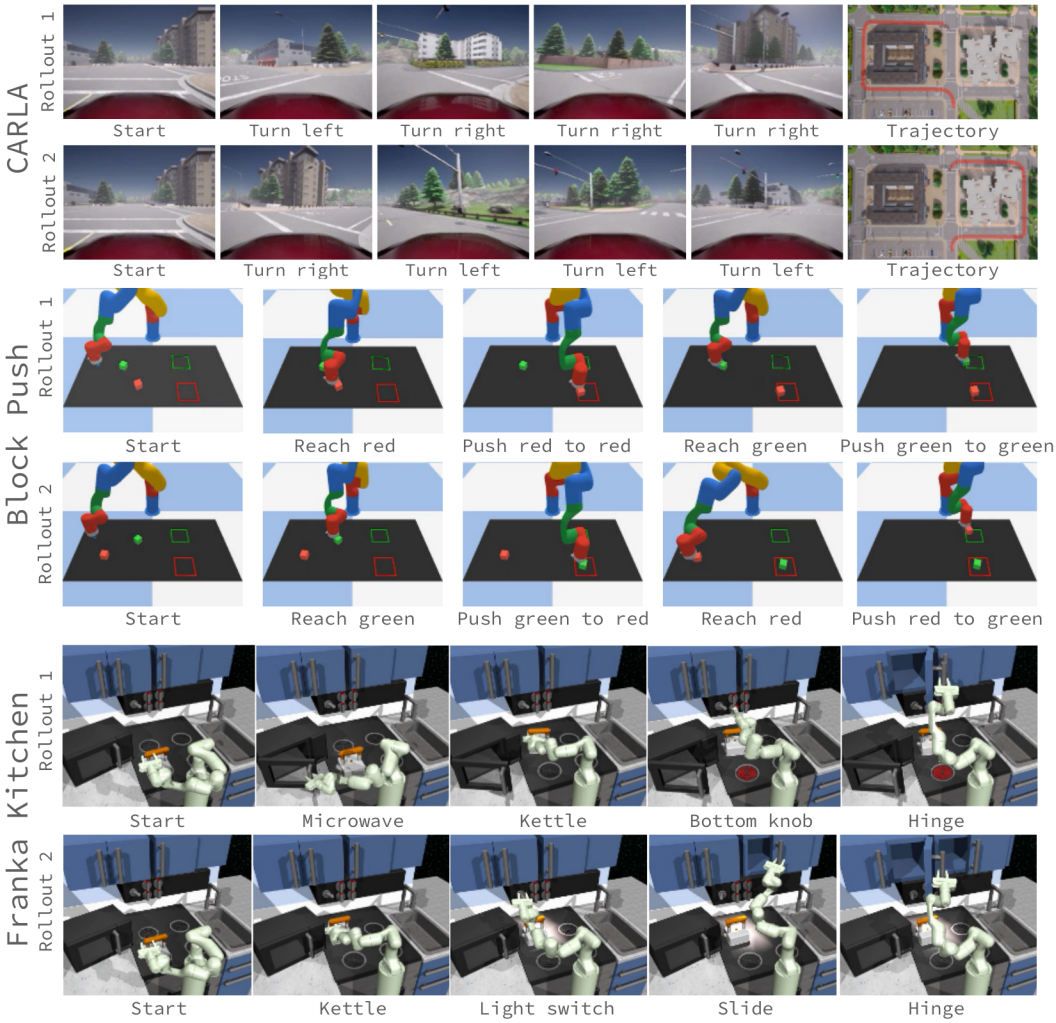

Figure 1: Unconditional rollouts from BeT models trained from multi-modal demonstartions on the CARLA, Block push, and Franka Kitchen environments. Due to the multi-modal architecture of BeT, even in the same environment successive rollouts can achieve different goals or the same goals in different ways.

architecture design, such as using a Gaussian prior. On the other hand, natural pre-collected data is sub-optimal, noisy, and contains multiple modes of behavior, all entangled in a single dataset. This distributionally multi-modal experience is most prominent in human demonstrations. Not only do we perform a large variety of behaviors every day, our personal biases result in significant multi-modality even for the same behavior [26, 44]. This raises an important question: How do we train models that can "clone" multi-modal behavior data?

In this work, we present Behavior Transformers (BeT), a new method for learning behaviors from rich, distributionally multi-modal data. BeT is based of three key insights. First, we leverage the context based multi-token prediction ability of transformer-based sequence models [70] to predict multi-modal actions. Second, since transformer-based sequence models are naturally suited to predicting discrete classes, we cluster continuous actions into discrete bins using k-means [45]. This allows us to model high-dimensional, continuous multi-modal action distributions as categorical distributions without learning complicated generative models [34, 17]. Third, to ensure that the actions sampled from BeT are useful for online rollouts, we concurrently learn a residual action corrector to produce continuous actions for a specific sampled action bin.

We experimentally evaluate BeT on five datasets ranging from simple diagnostic toy datasets to complex datasets that include simulated robotic pushing [22], sequential task solving in kitchen

environments [29], and self-driving with visual observations in CARLA [18]. The two main findings from these experiments can be summarized as:

1. On multi-modal datasets, BeT achieves significantly higher performance during online rollouts compared to prior behavior modelling methods.
2. Rather than collapsing or latching onto one mode, BeT is able to cover the major modes present in the training behavior datasets. Unconditional rollouts from this model can be seen in Fig. 1.

All of our datasets, code, and trained models will be made publicly available.

2 Behavior Transformers

Given a dataset of continuous observation and action pairs $\mathcal{D} \equiv \{(o, a)\} \subset \mathcal{O} \times \mathcal{A}$ that contains behaviors we are interested in, our goal is to learn a behavior policy $\pi : \mathcal{O} \mapsto \mathcal{A}$ that models this data without any online interactions with the environment or reward labels. This setup follows the Behavior Cloning formulation, where policies are trained to model demonstrations from expert rollouts. Often, such policies are chosen from a hypothesis class parametrized by parameter set θ. Following this convention, our objective is to find the parameter θ that maximizes the probability of the observed data

$$\theta^* := \arg\max_{\theta} \prod_{t} \mathbb{P}(a_t \mid o_t; \theta)$$

When the model class is restricted to unimodal isotropic Gaussians, this maximum likelihood estimation problem leads to minimizing the Mean Squared Error (MSE), $\sum_t \|a_t - \pi(o_t; \theta)\|^2$.

Limitations of traditional MSE-based BC: While MSE-based BC has been able to solve a variety of tasks [8, 69], it assumes that the data distribution is unimodal. Clean data from an expert demonstrator solving a particular task in a particular way satisfies this assumption, but pre-collected intelligent behavior often may not [44, 29]. While more recent behavior generation models have sought to address this problem, they often require complex generative models [68], an exponential number of bins for actions [46], complicated training schemes [57], or time-consuming test-time optimization [22]. An experimental analysis of some of these prior works is presented in Section 3.

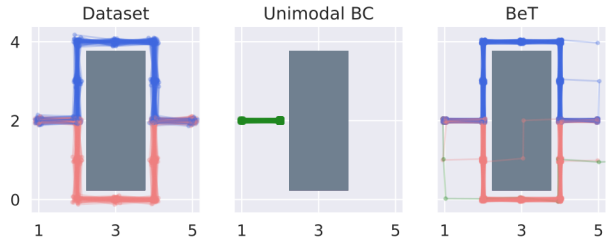

Figure 2: Comparison between a regular MSE-based BC model and a BeT models that can capture multi-modal distributions. The MSE-BC model takes 0 action to minimize MSE.

Overview of Behavior Transformers (BeT): We address two critical assumptions in regular BC. First, we relax the assumption that the behavior we are cloning is purely Markovian, and instead model $P(a_t \mid o_t, o_{t-1}, \cdots, o_{t-h+1})$ for some horizon h. Second, instead of assuming that actions are generated by a unimodal action distribution, we model our action distribution as a mixture of gaussians. However, unlike previous efforts similar to Mixture Density Networks (MDN) to do so, whose limitations have been explored in Florence et al. [22], we do not explicitly predict mode centers, which significantly improves our modeling capacity. To operationalize these two features in a single behavior model, we make use of transformers since (a) they are effective in utilizing prior observational history, and (b) they are naturally suited to output multi-modal tokens through their architecture.

2.1 Action discretization for distribution learning

Although transformers have become standard as a backbone for sequence-to-sequence models [16, 10], they are designed to process discrete tokens and not continuous values. In fact, modeling multi-modal distributions of high-dimensional continuous variables in a tractable manner is in itself a challenging problem, especially if we want the trained behavior model to cover the modes present in the dataset.

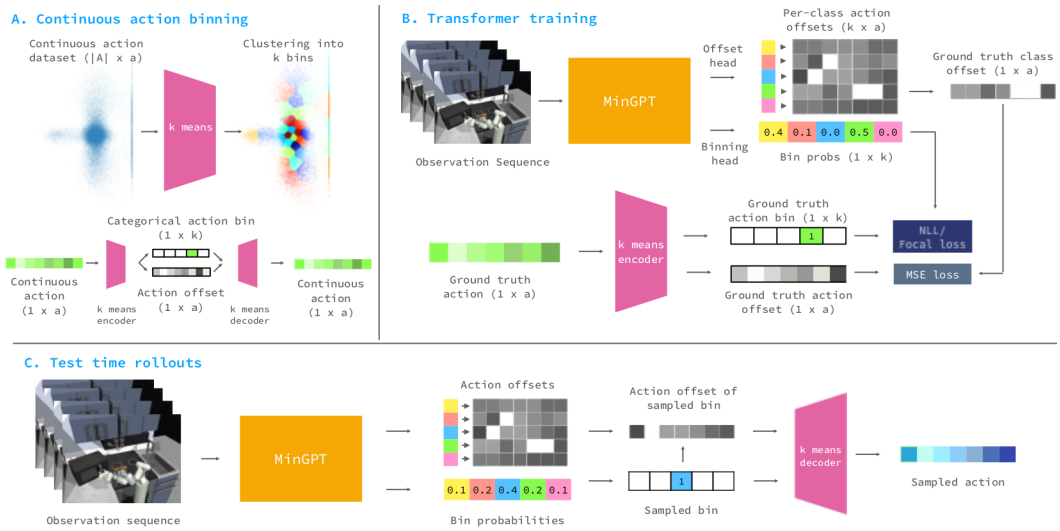

Figure 3: Architecture of Behavior Transformer. (A) The continuous action binning using k-means algorithm that lets BeT split every action into a discrete bin and a continuous offset, and later combine them into one full action. (B) Training BeT using demonstrations offline; each ground truth action provides a ground truth bin and residual action, which is used to train the minGPT trunk with its binning and action offset heads. (C) Rollouts from BeT in test time, where it first chooses a bin and then picks the corresponding offset to reconstruct a continuous action.

To address this, we propose a new factoring of the action prediction task by dividing each action in two parts: a categorical variable denoting an 'action center', and a corresponding 'residual action'.

To this end, given the actions in our dataset, we first optimize for a set of k action centers, $\{A_1, A_2, \cdots, A_k\} \subset \mathcal{A}$. We then decompose each action into two parts: a categorical variable representing the closest action bin, $\lfloor a \rfloor := \arg\min_i \|a - A_i\|_2$, and a continuous residual action $\langle a \rangle := a - A_{\lfloor a \rfloor}$. If we are given the set of action centers $\{A_i\}_{i=1}^k$, an action bin index $\lfloor a \rfloor$ and the residual action $\langle a \rangle$, we can deterministically reconstruct the true action $a := A_{\lfloor a \rfloor} + \langle a \rangle$. Once learned, these k-means based encoder and decoders for this action factorization process are fixed for the rest of the train and testing phases. The action factorization procedure is illustrated in Fig. 3 (A).

2.2 Attention-based behavior mode learning

Once we have the clustering based autoencoder learned from the actions in the dataset, we model our demonstration trajectories with BeT. We use a transformer decoder model, namely minGPT [10], with minor modifications, as our backbone. The transformer $\mathcal{T}$ takes in a sequence of continuous observations $(o_i, o_{i+1}, \cdots, o_{i+h-1})$ and learns a sequence-to-sequence model mapping each observation to a categorical distribution over k discrete action bins. The predicted probability sequence is then compared with the ground truth labels, $(\lfloor a_i \rfloor, \lfloor a_{i+1} \rfloor, \lfloor a_{i+2} \rfloor, \cdots, \lfloor a_{i+h-1} \rfloor)$. We use a negative log-likelihood-based Focal loss [41] between the predicted categorical distribution probabilities and the ground truth labels to train the transformer head. The model is illustrated in Fig. 3 (B).

2.3 Action correction: from coarse to finer-grained predictions

Using a transformer allows us to model multi-modal actions. However, discretizing the continuous action space in any way invariably causes loss of fidelity [33]. Discretization error may cause online rollouts of the behavior policy to go out of distribution from the original dataset [65], which can in turn cause critical failures. To predict the complete continuous action, we add an extra head to the transformer decoder that offsets the discretized action centers based on the observations.

For each observation o_i in the sequence, the head produces a $k \times \dim(A)$ matrix with k proposed residual action vectors, $\left(\langle a_i^{(j)} \rangle\right)_{j=1}^k = (\langle \hat{a}_i^{(1)} \rangle, \langle \hat{a}_i^{(2)} \rangle, \langle \hat{a}_i^{(3)} \rangle, \cdots, \langle \hat{a}_i^{(k)} \rangle)$, where the residual actions correspond to bin centers $A_1, A_2, A_3, \cdots, A_k$. These residual actions are trained with a loss akin to

the *masked multi-task loss* [25] from object detection. In our case, if the ground truth action is $\mathbf{a}$, the loss is:

$$\text{MT-Loss}\left(\mathbf{a}, \left(\langle \hat{a}_i^{(j)} \rangle\right)_{j=1}^{k}\right) = \sum_{j=1}^{k} \mathbb{I}[\lfloor \mathbf{a} \rfloor = j] \cdot \|\langle \mathbf{a} \rangle - \langle \hat{a}^{(j)} \rangle\|_2^2$$

Where $\mathbb{I}[]$ denotes the Iverson bracket, ensuring the offset head of BeT only incurs loss from the ground truth class of action $\mathbf{a}$. This mechanism prevents the model from trying to fit the ground truth action using the offset at every index.

2.4 Test-time sampling from BeT

During test time, at timestep t we input the latest h observations $(o_t, o_{t-1}, \cdots, o_{t-h+1})$ to the transformer, combining the present observation o_t with $h-1$ previous observations. Our trained MinGPT model gives us $h \times 1 \times k$ bin center probability vectors, and $h \times k \times \dim(A)$ offset matrix. To sample an action at timestep t, we first sample an action center according to the predicted bin center probabilities on the tth index. Once we have chosen an action center $A_{t,j}$, we add the corresponding residual action $\langle \hat{a}_t^{(j)} \rangle$ to it to recover a predicted continuous action $\hat{\mathbf{a}}_\mathbf{t} = A_{t,j} + \langle \hat{a}_t^{(j)} \rangle$. This sampling procedure is illustrated in Fig. 3 (C).

3 Experiments

We now study the empirical performance of BeT on a variety of behavior learning tasks. Our experiments are designed to answer the following questions: (a) Is BeT able to imitate multi-modal demonstrations? (b) How well does BeT capture the modes present in behavior data? (c) How important are the individual components of BeT?

3.1 Environments and datasets

We experiment with five broad environments. While full descriptions of these environments, dataset creation procedure, and overall statistics are in Appendix A, a brief description of them are as follows.

(a) **Point mass environment #1:** Our first set of experiments in Fig. 2, used to get a qualitative understanding of BeT, were performed in a simple Pointmass environment with a 2D observation and action space with two hundred demonstrations. The pre-collected demonstrations start at a fixed point, and then make their way to another point while avoiding a block in the middle. The two primary modes in this dataset are taking a left turn versus a right turn.

(b) **Point mass environment #2:** The setup is similar to the previous environment with the exception of one straight line and two complicated prolonged 'Z' shaped modes of demonstration (Fig. 5.)

(c) **CARLA self-driving environment:** CARLA [18] uses the Unreal Engine to provide a simulated driving environment in a visually realistic landscape. The agent action space is 2D (accelerate/brake and left/right steer), while the observation space is (224,224,3)-dimensional RGB image from the car. A hundred total demonstrations drive around a building block in two distinct modes. This environment highlights the challenge of behavior learning from high-dimensional observations as shown in Fig. 1 (a). For visual observations with BeT, we use a frozen ResNet-18 [30] pretrained on ImageNet [15] as an encoder.

(d) **Multi-modal block-pushing environment:** For more complicated interaction data, we use the multi-modal block-pushing environment from Implicit Behavioral Cloning (IBC) [22], where an XArm robot needs to push two blocks into two squares in any order. The blocks and target squares are colored red and green. The positions of the blocks are randomized at episode start. We collect 1,000 demonstrations using a deterministic controller with two independent axes of multi-modality: (a) it starts by reaching either the red or the green block, with 50% probability, and (b) it pushes the blocks to (red, green) or (green, red) squares respectively with 50% probability.

(e) **Franka kitchen environment:** To highlight the complexity of performing long sequences of actions, we use the Relay Kitchen Environment [29] where a Franka robot manipulates a virtual kitchen environment. We use the relay policy learning dataset with 566 demonstrations collected by human participants wearing VR headsets. The participants completed a sequence of four

Table 1: Performance of BeT compared with different baselines in learning from demonstrations. For CARLA, we measure the probability of the car reaching the goal successfully. For Block push, we measure the probability of reaching one and two blocks, and the probabilities of pushing one and two blocks to respective squares. For Kitchen, we measure the probability of n tasks being completed by the model within the allotted 280 times. Evaluations are over 100 rollouts in CARLA and 1,000 rollouts in Block push and Kitchen environments.

	CARLA	Block push				Kitchen				
	Driving	Reach		Push		# Tasks completed				
Baselines	Success	R1	R2	P1	P2	1	2	3	4	5
RBC	0.98	0.67	0	0	0	0	0	0	0	0
1-NN	0.99	0.49	0.05	0.01	0	0.90	0.72	0.44	0.17	0
LWR	**1**	0.50	0.06	0	0	**1**	0.83	0.52	0.21	0
VAE	0	0.60	0.05	0	0	**1**	0	0	0	0
Flow	0.03	0.59	0.02	0	0	0.04	0	0	0	0
IBC	0.25	0.98	0.04	0.01	0	0.99	0.87	0.61	0.24	0
BeT (Ours)	0.98	**1**	**0.99**	**0.96**	**0.71**	0.99	**0.93**	**0.71**	**0.44**	**0.02**

object-interaction tasks in each episode [29]. There are a total of seven interactable objects in the kitchen: a microwave, a kettle, a slide cabinet, a hinge cabinet, a light switch, and two burner knobs. This dataset contains two different kinds of multi-modality: one from the inherent noise in human demonstrations, and another from the demonstrators' intent.

3.2 Baseline behavior learning methods

While a full description of our baselines are in Appendix B.1, a brief description of them is here:

(a) **Multi-layer Perceptron with MSE (RBC):** We use MLP networks trained with MSE loss as our first baseline, since this is the standard way of performing behavioral cloning for a new task [69]. A comparison with transformer-based behavior cloning is discussed in Section 3.5.

(b) **Nearest neighbor (NN):** Nearest neighbor based algorithms are easy to implement, and has recently shown to have strong performance on complicated behavioral cloning tasks [3].

(c) **Locally Weighted Regression (LWR):** This non-parametric approach provides better regularization compared to NN and is a strong alternative to parametric BC [4, 53].

(d) **Variational auto-encoders (VAE):** Inspired by SPiRL [57], where behavioral priors are learned through a VAE [34], we compare with continuous actions generated from the VAE and the prior.

(e) **Normalizing Flow (Flow):** Inspired by PARROT [68], where state-conditioned action priors are learned through a Flow model [17], we compare with actions generated from the Flow model.

(f) **Implicit Behavioral Cloning (IBC):** Instead of modeling the conditional distribution $P(a \mid o)$, IBC models the joint probability distribution $P(a, o)$ using energy-based models [22]. While IBC is slower than explicit BC models because of their sampling requirements, they have been shown to learn well on multi-modal data, and outperform earlier work such as MDNs [7].

3.3 Is BeT able to imitate multi-modal demonstrations?

The first question we ask is whether BeT can actually clone behaviors given a mixed dataset of unlabeled, multi-modal behaviors. To examine that, we look at the performance of our model in CARLA, Block push, and Kitchen environments compared with our baselines in Table 1.

We see that BeT outperforms all other methods in all environments except CARLA, where it is narrowly outperformed by LWR. Since the models are all behavioral cloning algorithms, they share the failure mode of failing once the observations go out of distribution (OOD). However, they vary in the tolerance. For example, BeT shines in the Block push environment, where alongside extreme environment randomness and multi-modality, the models also have to learn significant long-term behaviors and commit to a single mode over a long period. While all baselines can somewhat successfully reach one block, they fail to complete the long-horizon, multi-modal task of pushing two blocks into two different bins. On the other hand, we observe that BeT's primary failure mode is not realizing a block has not completely entered the target yet, while other methods either go OOD quickly, or keep switching between modes. We also observe that BeT performs well even in complex

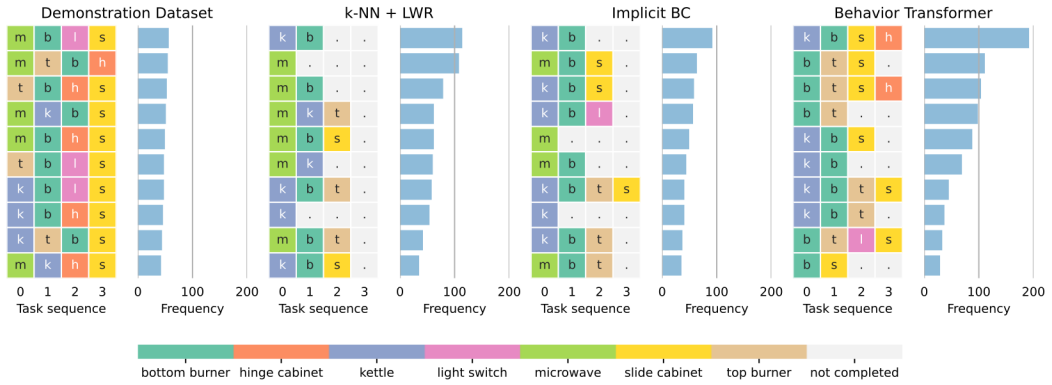

Figure 4: Distribution of most frequent tasks completed in sequence in the Kitchen environment. Each task is colored differently, and frequency is shown out of a 1,000 unconditional rollouts from the models.

Table 2: Multimodality learned from the multimodal demonstrations by different algorithms. In CARLA, we consider the probability of turning left vs. right at the intersection, ignoring OOD rollouts. In Block push, we consider two set of probabilities, (a) which block was reached first, and (b) what was the pushing target for each block. Finally, in Franka Kitchen, we consider the empirical entropy for the task sequences, considered as strings, sampled from the model. We highlight the values closest to the corresponding demonstration values.

	CARLA		Block: first block reached		Push: red block target		Push: green block target		Kitchen
Baselines	Left	Right	Red	Green	Red	Green	Red	Green	Task entropy
RBC	0	0.98	0.41	0.25	0	0	0	0	0
1-NN	0	0.99	0.24	0.25	0	0	0	0.01	2.12
LWR	0	1	0.26	0.26	0.01	0	0.01	0.01	2.29
VAE	0	0	0.27	0.33	0	0	0	0	0.72
Flow	0	0	0.31	0.29	0	0	0	0	0.08
IBC	0.12	0.13	**0.48**	**0.50**	0	0	0.01	0.01	2.41
BeT (Ours)	**0.34**	**0.64**	0.54	0.46	**0.43**	**0.44**	**0.41**	**0.40**	**2.47**
Demonstrations	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	2.96

observation and action spaces. In the CARLA environment, the model takes in visual observations, while in the Franka Kitchen environment, the action space corresponds to a 9-DOF torque controlled robot. BeT handles both cases with the same ease as it does environments with lower-dimensional observation or action spaces.

3.4 Does BeT capture the modes present in behavior data?

Next, we examine the question of whether, given a dataset where multi-modal behavior exists, our model learns behavior that is also multi-modal. Here, we are interested in seeing the variance of the behavior of the model over different rollouts. In each of our environments, the demonstrations contain different types of multi-modality. As a result, we show a comprehensive analysis of multi-modality seen in our agent behaviors.

We see in Table 2 that in CARLA and Block push, BeT covers all the modes of the demonstration data, even in the few cases where it does not perfectly match the demonstrated task probabilities. For the Kitchen environment, we see in Fig. 4 that BeT visits certain strings of tasks more frequently than in the original demonstrations. However, compared to other strong baselines, BeT generates longer task strings more often while maintaining diversity and not collapsing to a single mode.

3.5 How important are the individual components of BeT?

There are four key differences between BeT architecture and standard BC: (a) binning actions into discrete clusters, (b) using offsets to faithfully reconstruct actions later, (c) learning sequentially to

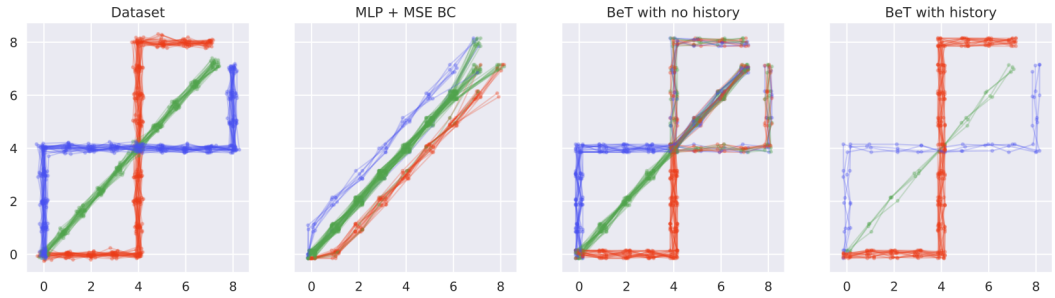

Figure 5: Comparison between an RBC model and two BeT models, trained with and without historical context on a dataset with three distinct modes. BeT with history is better able to capture the context-dependant behavior in the demonstrations.

use historical context, and (d) using an attention-based MinGPT trunk. In this section, we discuss the impacts they have in BeT performance.

Impact of discrete binning: Intuitively, having discrete options for bin centers is what enables BeT to express multi-modal behavior even when starting from an identical starting state. Indeed, if there is no binning, we see from Table 3 that the performance of BeT drops significantly. More tellingly, in the Franka Kitchen environment, the model only ever completed a subsequence of (kettle, top/bottom burner, light switch, slide cabinet) tasks after 100 random rollouts. This result shows us that having discrete bins helps BeT achieve multi-modality. To find the best value of the number of bins, k, we fit a Bayesian GMM [64] to our action dataset and use the estimated number of clusters as a guide. We show a plot of k vs. performance on the tasks in the Appendix C.2.

Table 3: Relative performance of ablated variants of BeT, normalized by average BeT performance at the task

Ablations	CARLA	Block push	Kitchen
No offsets	0.94	0.95	0.78
No binning	0.94	0.25	0.68
No history	0.65	0.95	0.88
No MinGPT	0.90	0	0.05

Necessity of action offsets: An important feature of BeT is the residual action offset that corrects the discrete actions coming from the bins. While the bin centers may be quite expressive, Table 3 shows that the inability to correct them causes a performance degrade. Interestingly, the largest degradation comes in the Kitchen environment, which also has the highest dimensional action space. Intuitively, we can understand how in higher dimension the loss of fidelity from discretizing would be higher, and the relative performance loss across three environments support that hypothesis.

Importance of historical context: While RL algorithms traditionally assume environments are Markovian, human behavior in an open-ended environment is rarely so. Thus, using historical context helps BeT to perform well. We show a simple experiment in Fig. 5 on the second point mass environment. Here, training and evaluating with some historical context allows BeT to follow the demonstrations better. We experience the same in the CARLA, Block push, and Kitchen environments, where training with some historical context raises performance across the board as seen in Table 3. Since transformer-based models are generally able to learn from long-term historical context, we believe BeT should also be able to model real-world long-term behavior patterns.

Importance of transformer architecture: Despite transformers' success in other fields of machine learning, it is natural to wonder whether the tasks BeT solves here really requires one. We ablated BeT by replacing the MinGPT trunk with an MLP, and found that they have lower performance while also being difficult to train stably. This performance reduction remains even if the MLP is given some historical context by concatenating h observations before passing it to the MLP. See Table. 3 for results and Appendix C.3 for further details.

Computation considerations: While transformers in usual contexts are large models, we downscale them for our application in BeT (See Appendix B.4). Our models contain on the order of $10^4 - 10^6$ parameters, and even with a small batch size and on a single desktop GPU trains within an hour for our largest datasets (Block push). In contrast, for the same task, our strongest baseline IBC takes about 14 hours. On the same environment, 100 evaluation rollouts take about 165 seconds with BeT, as opposed to 1770 seconds with IBC.

4 Related Work

This paper builds upon a rich literature in imitation learning, offline learning, generative models, and transformer architectures. The most relevant ones to our work are discussed here.

Learning from offline data: Since Pomerleau [58] showed the possibility of driving an autonomous vehicle using offline data and a neural network, learning behavior from offline data has been a continuous topic of research for scalable behavior learning [2, 6, 67]. The approaches can be divided into two broad classes: Offline RL [24, 36, 37, 72, 39, 23], focusing on learning from datasets of a mixed quality that also have reward labels; and imitation learning [52, 54, 55, 31], focusing on learning behavior from a dataset of expert behavior without reward labels. BeT falls under the second category, as it is a behavior cloning model. Behavior cloning is a form of imitation learning that tries to model the action of the expert given the observation which is often used in real-world applications [76, 77, 76, 60, 21, 75]. As behavior cloning algorithms are generally solving a fully supervised learning problem, they tend to be faster and simpler than reinforcement learning or offline RL algorithms and in some cases show competitive results [23, 28].

Generative models for behavior learning: One approach for imitation learning is Inverse Reinforcement Learning or IRL [66, 51], where given expert demonstrations, a model tries to construct the reward function. This reward function is then used to generate desirable behavior. GAIL [31], an IRL algorithm, connects generative adversarial models with imitation learning to construct a model that can generate expert-like behavior. Under this IRL framework, previous works have tried to predict multi-modal, multi-human trajectories [38, 32]. Similarly, other works have tried Gaussian Processes [62] for creating dynamical models for human motion [71]. Another class of algorithms learn a generative action decoder [56, 44, 68] from interaction data to make downstream reinforcement learning faster and easier, which inspired BeT's action factorization. Finally, a class of algorithms, most notably [42, 22, 35, 50] do not directly learn a generative model but instead learn energy based models. These energy based models can then be sampled to generate desired behavior. Since [22] is a BC model capable of multi-modality, we compare against it as a baseline in Sec. 3.

Transformers for control: With the stellar success of transformer models [70] in natural language processing [16, 10] and computer vision [19], there has been significant interest in using transformer models to learn behavior and control. Among those, [11, 33] applies them to Reinforcement Learning and Offline Reinforcement Learning, respectively, while [12, 14, 46] use them for imitation learning. [14, 46] use transformers mostly to summarize historical visual context, while [12] relies on their long-term extrapolation abilities. BeT is inspired by both of these use cases, as we use a transformer to summarize historical context while leveraging its generative abilities.

Datasets for distributionally multi-modal data: Similar to computer vision [15, 40, 43] and natural language processing [9, 61], there has been a recent interest in collecting behavior datasets that may aid in downstream behavior learning. Some of them are labeled with agent goals or rewards for downstream tasks [47, 23, 48], while others are more open ended [29, 44, 74] and come without reward or task labels. In our work, we focus towards the latter class. The lack of labeled goal or reward labels in the second category implies that there is more multi-modality in the action distributions compared to action distributions of goal or reward conditioned datasets, which is the same reason a lot of work learning from multi-modal datasets try to learn a goal-conditioned model [29, 44, 14]. Finally, the lack of labelling requirements mean that the unlabelled datasets are cheaper to obtain, which should help BeT scale further in the future.

5 Discussion, limitations, and scope for future work

In this work, we introduce Behavior Transformers (BeT), which uses a transformer-decoder based backbone with a discrete action mode predictor coupled with a continuous action offset corrector to model continuous actions sequences from open-ended, multi-modal demonstrations. While BeT shows promise, the truly exciting use of it would be to learn diverse behavior from human demonstrations or interactions in the real world. In parallel, extracting a particular, unimodal behavior policy from BeT during online interactions, either by distilling the model or by generating the right 'prompts' [63], would make BeT tremendously useful as a prior for online Reinforcement Learning.

References

[1] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 1

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 9

[3] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022. 6

[4] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Lazy learning*, pages 11–73, 1997. 6

[5] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 1

[6] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008. 9

[7] C. M. Bishop. Mixture density networks. *Neural Computing Research Group Report*, 1994. 6

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1, 3

[9] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015. 9

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1, 3, 4, 9

[11] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *CoRR*, abs/2106.01345, 2021. URL https://arxiv.org/abs/2106.01345. 9

[12] H. M. Clever, A. Handa, H. Mazhar, K. Parker, O. Shapira, Q. Wan, Y. Narang, I. Akinola, M. Cakmak, and D. Fox. Assistive tele-op: Leveraging transformers to collect robotic task demonstrations. *arXiv preprint arXiv:2112.05129*, 2021. 9

[13] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019. 1

[14] S. Dasari and A. Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020. 9

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 5, 9

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, pages 4171–4186, 2018. doi: 10.18653/v1/N19-1423. 1, 3, 9

[17] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2, 6

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 3, 5

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 9

[20] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1329–1338. JMLR.org, 2016. 1

[21] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019. 9

[22] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021. 2, 3, 5, 6, 9

[23] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 9

[24] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, PMLR, 2018. 9

[25] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 5

[26] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. *arXiv preprint arXiv:2110.07058*, 3, 2021. 2

[27] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 1

[28] C. Gulcehre, Z. Wang, A. Novikov, T. Le Paine, S. Gomez Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv e-prints*, pages arXiv–2006, 2020. 9

[29] 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. 3, 5, 6, 9

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. 5

[31] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, volume 29, pages 4565–4573, 2016. 9

[32] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone. Generative modeling of multimodal multi-human behavior. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3088–3095. IEEE, 2018. 9

[33] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021. 4, 9

[34] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 6

[35] I. Kostrikov, J. Tompson, R. Fergus, and O. Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021. 9

[36] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019. 9

[37] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. 9

[38] N. Lee and K. M. Kitani. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016. 9

[39] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1, 9

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 9

[41] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4

[42] M. Liu, T. He, M. Xu, and W. Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 33, 2020. 9

[43] Z. Liu, P. Luo, X. Wang, and X. Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018. 9

[44] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020. 2, 3, 9

[45] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 2

[46] Z. Mandi, F. Liu, K. Lee, and P. Abbeel. Towards more generalizable one-shot visual imitation learning. *arXiv preprint arXiv:2110.13423*, 2021. 3, 9

[47] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018. 9

[48] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. 9

[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 1

[50] O. Nachum and M. Yang. Provable representation learning for imitation with contrastive fourier features. *arXiv preprint arXiv:2105.12272*, 2021. 9

[51] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000. 9

[52] 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. 9

[53] J. Pari, N. Muhammad, S. P. Arunachalam, L. Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 1, 6

[54] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 9

[55] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021. 9

[56] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020. 9

[57] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020. 3, 6

[58] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 9

[59] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 1

[60] 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 *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018. 9

[61] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 9

[62] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010. 9

[63] L. Reynolds and K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021. 9

[64] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian approaches to gaussian mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1133–1142, 1998. 8

[65] S. Ross, G. Gordan, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *arXiv preprint arXiv:1011.0686*, 2010. 4

[66] S. Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998. 9

[67] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999. 9

[68] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020. 3, 6, 9

[69] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 1, 3, 6

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 2, 9

[71] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007. 9

[72] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. 9

[73] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. 1

[74] S. Young, J. Pari, P. Abbeel, and L. Pinto. Playful interactions for representation learning. *arXiv preprint arXiv:2107.09046*, 2021. 9

[75] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 2020. 9

[76] 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 *ICRA*, pages 5628–5635. IEEE, 2018. 1, 9

[77] 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. 9

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See abstract.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec 3.5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See linked code.
   (b) Did you mention the license of the assets? [Yes] See linked code.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]