FROM PLAY TO POLICY: CONDITIONAL BEHAVIOR GENERATION FROM UNCURATED ROBOT DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

While large-scale sequence modeling from offline data has led to impressive performance gains in natural language and image generation, directly translating such ideas to robotics has been challenging. One critical reason for this is that uncurated robot demonstration data, i.e. *play* data, collected from non-expert human demonstrators are often noisy, diverse, and distributionally multi-modal. This makes extracting useful, task-centric behaviors from such data a difficult generative modeling problem. In this work, we present Conditional Behavior Transformers (C-BeT), a method that combines the multi-modal generation ability of Behavior Transformer with future-conditioned goal specification. On a suite of simulated benchmark tasks, we find that C-BeT improves upon prior state-of-the-art work in learning from play data by an average of 45.7%. Further, we demonstrate for the first time that useful task-centric behaviors can be learned on a real-world robot purely from play data without any task labels or reward information. Robot videos are best viewed on our project website: https://cbet-anon.github.io.

1 INTRODUCTION

Machine Learning is undergoing a Cambrian explosion in large generative models for applications across vision (Ramesh et al., 2022) and language (Brown et al., 2020). A shared property across these models is that they are trained on large and uncurated data, often scraped from the internet. Interestingly, although these models are trained without explicit task-specific labels in a self-supervised manner, they demonstrate a preternatural ability to generalize by simply conditioning the model on desirable outputs (e.g. "prompts" in text or image generation). Yet, the success of conditional generation from uncurated data has remained elusive for decision making problems, particularly in robotic behavior generation.

To address this gap in behavior generation, several works (Lynch et al., 2019; Pertsch et al., 2020b) have studied the use of generative models on *play* data. Here, play data is a form of offline, uncurated data that comes from either humans or a set of expert policies interacting with the environment. However, once trained, many of these generative models require significant amounts of additional online training with task-specific rewards (Gupta et al., 2019; Singh et al., 2020). In order to obtain task-specific policies without online training, a new line of approaches employ offline RL to learn goal-conditioned policies (Levine et al., 2020; Ma et al., 2022). These methods often require rewards or reward functions to accompany the data, either specified during data collection or inferred through hand-crafted distance metrics, for compatibility with RL training. Unfortunately, for many real-world applications, data does not readily come with rewards. This prompts the question: *how do we learn conditional models for behavior generation from reward-free, play data?*

Table 1: Comparison between existing algorithms to learn from large, uncurated datasets: GCBC (Lynch et al., 2019), GCSL (Ghosh et al., 2019), Offline GCRL (Ma et al., 2022), Decision Transformer Chen et al. (2021)

	GCBC	GCSL	Offline RL	Decision Transformer	C-BeT (ours)
Reward-free	✓	✓	✗	✗	✓
Offline	✓	✗	✓	✓	✓
Multi-modal	✗	✗	✗	✗	✓

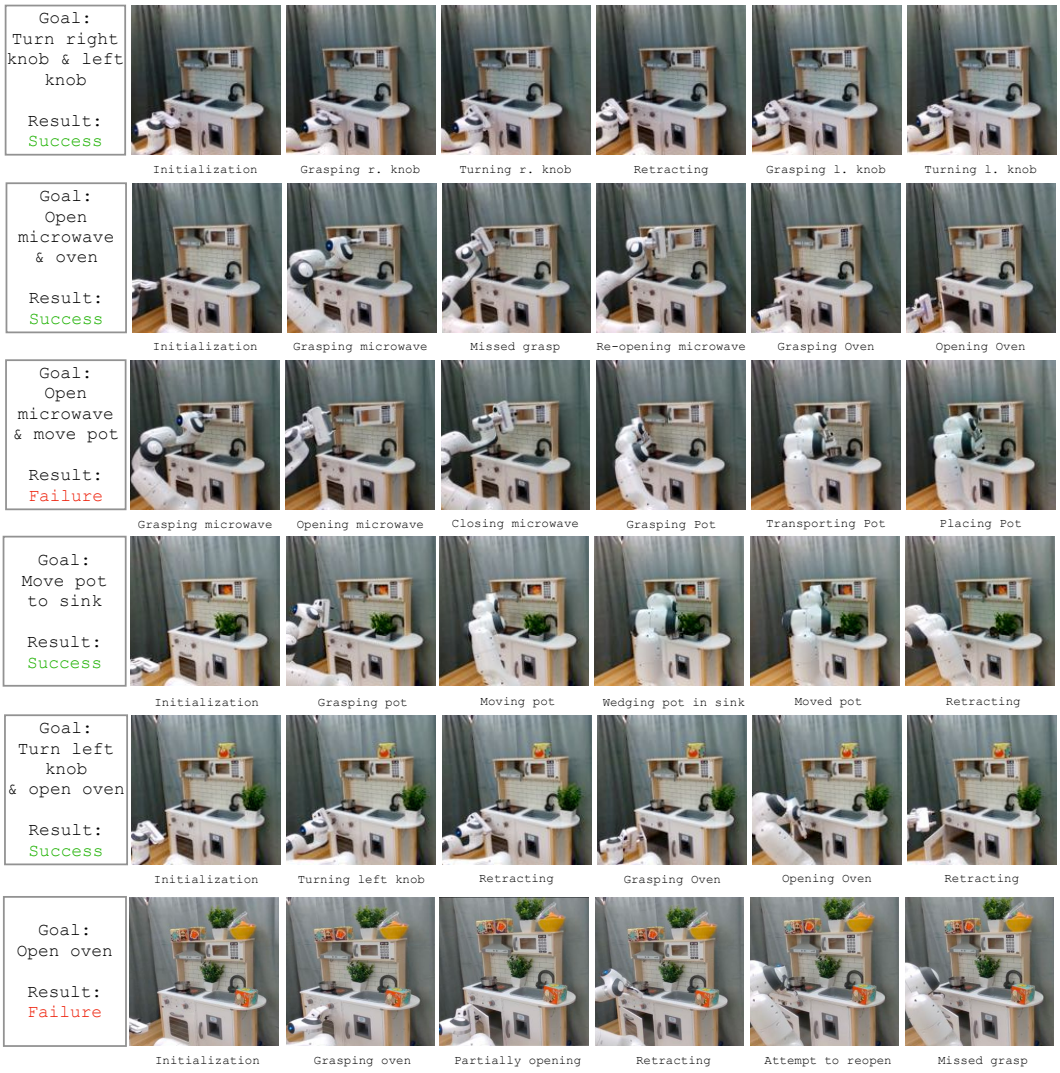

Figure 1: Multiple conditioned roll-outs of visual robot policies learned on our toy kitchen with only 4.5 hours of human play interactions. Our model learns purely from image and proprioception without human labeling or data curation. During evaluation, the policy can be conditioned either on a goal observation or a demonstration. Note that the last three rows contain distractor objects in the environment that were never seen during training.

To answer this question, we turn towards transformer-based generative models that are commonplace in text generation. Here, given a prompt, models like GPT-3 (Brown et al., 2020) can generate text that coherently follow or satisfy the prompt. However, directly applying such models to behavior generation requires overcoming two significant challenges. First, unlike the discrete tokens used in text generation, behavior generation will need models that can output continuous actions while also modeling any multi-modality present in the underlying data. Second, unlike textual prompts that serve as conditioning for text generation, behavior generation may not have the condition and the operand be part of the same token set, and may instead require conditioning on future outcomes.

In this work, we present Conditional Behavior Transformers (C-BeT), a new model for learning conditional behaviors from offline data. To produce a distribution over continuous actions instead of discrete tokens, C-BeT augments standard text generation transformers with the action discretization introduced in Behavior Transformers (BeT) (Shafiullah et al., 2022). Conditioning in C-BeT is done by specifying desired future states as input similar to Play-Goal Conditioned Behavior Cloning (Play-GCBC) (Lynch et al., 2019). By combining these two ideas, C-BeT is able to leverage the multi-modal generation capabilities of transformer models with the future conditioning capabilities of conditional policy learning. Importantly, C-BeT does not require any online environment interactions during training, nor the specification of rewards or Q functions needed in offline RL.

We experimentally evaluate C-BeT on three simulated benchmarks (visual self-driving in CARLA (Dosovitskiy et al., 2017), multi-modal block pushing (Florence et al., 2021), and simulated kitchen (Gupta et al., 2019)), and on a real Franka robot trained with play data collected by human volunteers. The main findings from these experiments can be summarized as:

1. On future-conditioned tasks, C-BeT achieves significantly higher performance compared to prior work in learning from play.
2. C-BeT demonstrates that competent visual policies for real-world tasks can be learned from fully offline data (rollouts visualized in Figure 1). To the best of our knowledge, C-BeT represents the first work to demonstrate this ability from reward-free play data.

2 Background and Preliminaries

Play-like data: Learning from Demonstrations (Argall et al., 2009) is one of the earliest frameworks explored for behavior learning algorithms from offline data. Typically, the datasets used in these frameworks have a built in assumption that the demonstrations are collected from an expert repeatedly demonstrating a single task in exactly the same way. On the contrary, play datasets violate many of such assumptions, like those of expertise of the demonstrator, and the unimodality of the task and the demonstrations. Algorithms that learn from such datasets sometimes assume that the demonstrations collected are from a rational agent with possibly some latent intent in their behavior (Lynch et al., 2019). Note that, unlike standard offline-RL datasets (Fu et al., 2020), play-like behavior datasets neither contain fully random behaviors, nor have rewards associated with the demonstrations.

Behavior Transformers (BeT): BeT (Shafiullah et al., 2022) is a multi-modal behavior cloning model designed particularly for tackling play-like behavior datasets. BeT uses a GPT-like transformer architecture to model the probability distribution of action given a sequence of states $\pi(a_t \mid s_{t-h:t})$ from a given dataset. However, unlike previous behavior learning algorithms, BeT does not assume a unimodal prior for the action distribution. Instead, it uses a k-means discretization to bin the actions from the demonstration set into k bins, and then uses the bins to decompose each action into a discrete and continuous component. This support for multi-modal action distributions make BeT particularly suited for multi-modal, play-like behavior datasets where unimodal behavior cloning algorithms fail. However, vanilla BeT only supports unconditonal behavior rollouts, which means that it is not possible to choose a targeted mode of behavior during BeT policy execution.

Conditional behavior learning: Generally, the problem of behavior learning for an agent is considered the task of learning a *policy* $\pi : \mathcal{O} \to \mathcal{A}$ mapping from the environment observations to the agent's actions that elicit some desired behavior. Conditional behavior learning is concerned with learning a policy $\pi : \mathcal{O} \times \mathcal{G} \to \mathcal{A}$ conditioned additionally on a secondary variable g sampled from a distribution $p(g)$. This condition variable could be specific environment states, latents (such as one-hot vectors), or even image observations. The success of a conditioned policy can be evaluated either through pre-specified reward functions, distance function between achieved outcome g' and specified outcome g, or by discounted visitation probability $d_{\pi(\cdot|g)} = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t \delta(\phi(o_t) = g)]$ if a mapping ϕ between states and achieved outcome is defined (Eysenbach et al., 2022).

Goal Conditioned Behavior Cloning (GCBC): In GCBC (Lynch et al., 2019; Emmons et al., 2021), the agent is presented with a dataset of (observation, action, goal) tuples (o, a, g), or sequences of such tuples, and the objective of the agent is to learn a goal-conditioned behavior policy. The simplest way to achieve so is by training a policy $\pi(\cdot \mid o, g)$ that maximizes the probability of the seen data $\pi^* = \arg\max_\pi \prod_{(o,a,g)} \mathbb{P}[a \sim \pi(\cdot \mid o, g)]$. Assuming a unimodal Gaussian distribution for $\pi(a \mid o, g)$ and a model parametrized by θ, this comes down to finding the parameter θ minimizing the MSE loss, $\theta^* = \arg\min_\theta \sum_{(o,a,g)} ||a - \pi(o, g; \theta)||^2$. To make GCBC compatible with play data that inherently does not have goal labels, goal relabeling from future states is often necessary. A common form of data augmentation in training such models, useful when $\mathcal{G} \subset \mathcal{O}$, is hindsight data relabeling (Andrychowicz et al., 2017), where the dataset $\{(o, a, g)\}$ is augmented with $\{(o_t, a, o_{t'}) \mid t' > t\}$ by relabeling any reached state in a future timestep as a goal state and adding it to the dataset.

3 APPROACH

Given a dataset $\{(o, a)\} \in \mathcal{O} \times \mathcal{A}$ of sequences of (observation, action) pairs from a play dataset, our goal is to learn a behavior generation model that is capable of handling multiple tasks and multiple ways of accomplishing each task. At the same time, we wish to be able to extract desired behavior from the dataset in the form of a policy through our model, or, in terms of generative models, "controllably generate" our desired behavior (see Figure 2). Finally, in the process of learning this controllable, conditional generative model, we wish to minimize the amount of additional human annotation or curation required in preparing the dataset. The method we develop to address these needs is called Conditional Behavior Transformer.

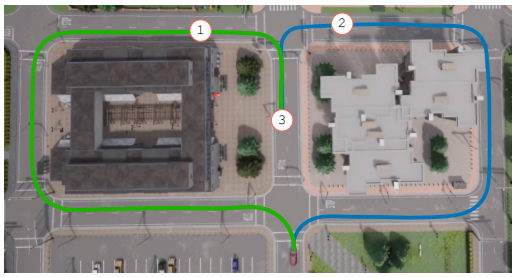

Figure 2: Conditional behavior learning from play demonstrations. Here, a policy conditioned on reaching ① or ② has only one possible course of action, but conditioned on reaching ③ there are two reasonable paths.

3.1 CONDITIONAL BEHAVIOR TRANSFORMERS (C-BET)

Conditional task formulation: First, we formulate the task of learning from a play dataset as learning a conditional behavior policy, i.e. given the current state, we need to model the distribution of actions that can lead to particular future states. For simplicity, our formulation can be expressed as $\pi : \mathcal{O} \times \mathcal{O} \to \mathcal{D}(\mathcal{A})$ where, given a current observation o_c and a future observation o_g, our policy π models the distribution of the possible actions that can take the agent from o_c to o_g. Mathematically, given a set of play trajectories T, we model the distribution $\pi(a \mid o_c, o_g) \triangleq \mathbb{P}_{\tau \in T}(a \mid o_c = \tau_t, o_g = \tau_{t'}, t' > t)$. Next, to make our policy more robust since we operate in the partially observable setting, we replace singular observations with a sequence of observations; namely replacing o_c and o_g with $\bar{o}_c = o_c^{(1:N)}$ and $\bar{o}_g = o_g^{(1:N)}$ for some integer N. Thus, the final task formulation becomes learning a generative model π with:

$$\pi\left(a \mid o_c^{(1:N)}, o_g^{(1:N)}\right) \triangleq \mathbb{P}_{\tau \in T}\left(a \mid o_c^{(1:N)} = \tau_{t:t+N}, o_g^{(1:N)} = \tau_{t':t'+N}, t' > t\right) \tag{1}$$

Architecture selection: Note that the model for our task described in the previous paragraph is necessarily multi-modal, since depending on the sequences $\bar{o}_c$ and $\bar{o}_g$, there could be multiple plausible sequences of actions with non-zero probability mass. As a result, we choose Behavior Transformers (BeT) (Shafiullah et al., 2022) as our generative architecture base as it can learn action generation with multiple modes. We modify the input to the BeT to be a concatenation of our future conditional observation sequence and current observation sequence. We choose to concatenate the inputs instead of stacking them, as this allows us to independently choose sequence lengths for the current and future conditional observations. Since BeT is a sequence-to-sequence model, we only consider the actions associated with the current observations as our actions. We show the detailed architecture of our model in Figure 3.

Dataset preparation: To train a C-BeT model on our play dataset $\{(o, a)\}$, we will need to appropriately prepare the dataset. We first convert the dataset to hold sequences of observations associated with actions, $\{(o_{t:t+N}, a_{t:t+N})\}$. Then, during training time, we dynamically augment each pair with a sequence of future observations, functionally converting our dataset into $\{(o_{t:t+N}, a_{t:t+N}, o_{t':t'+N'})\}$ for some $t' > t$, and treat the sequence $o_{t':t'+N'}$ as $\bar{o}_g$.

Training objective: We employ the same objective as BeT in training C-BeT. For each of the current observation and future conditional pair, we compute the BeT loss (see appendix B for details) between the ground truth actions and the predicted actions. We compute the focal loss (Lin et al., 2017) on the predicted action bins, and the MT-loss (Girshick, 2015) on the predicted action offsets corresponding to the action bins as described in BeT.

Test-time conditioning with C-BeT: During test time, we again concatenate our future conditional sequence with our current observations, and sample actions from our model according to the BeT framework. While in this work, we primarily condition C-BeT on future observations, we also study

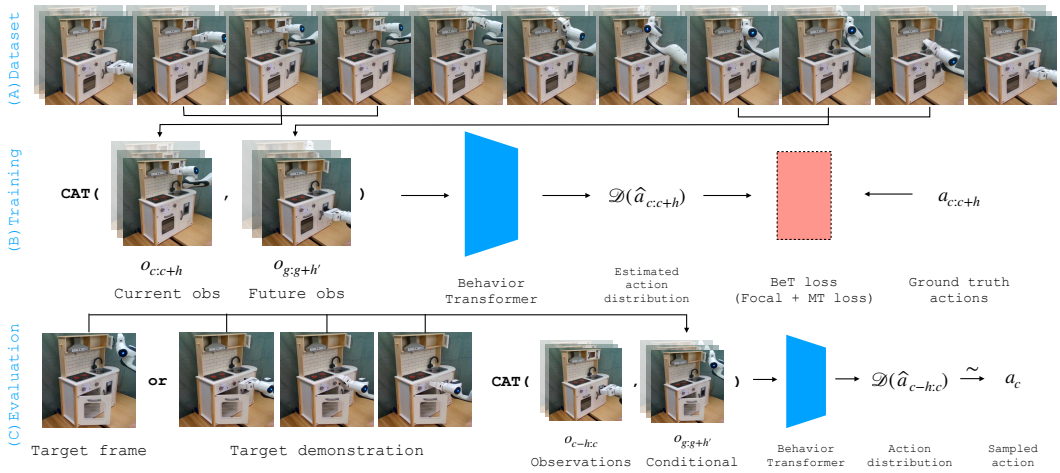

Figure 3: End-to-end training and evaluation of C-BeT. (A) Our dataset consists of play data in an environment, which may contain semi-optimal behavior, multi-modal demonstrations, and failures, and does not contain any annotations or task labels. (B) We train our C-BeT model by conditioning on current and future states using BeT (Section 2) (C) During evaluation, our algorithm can be conditioned by target observations or newly collected demonstrations to generate targeted behavior.

other ways of training and conditioning it, such as binary latent vectors denoting the modes in a trajectory in our experiments, and compare its performance to observation-conditioned C-BeT (see Section 4.5).

4 C-BeT on Simulated Benchmarks

In this section, we discuss our experiments in simulation that are designed to answer the following key questions: How well does C-BeT learn behaviors from play? How important is multi-modal action modeling? And finally, how does C-BeT compare to other forms of conditioning?

4.1 Baselines

We compare with the following state-of-the-art methods in learning from reward-free offline data:

- **Goal Conditioned Behavior Cloning (GCBC):** GCBC (Lynch et al., 2019; Emmons et al., 2021) learns a policy through optimizing the probability of the seen action given any state and the end state in a trajectory. It assumes that each trajectory is optimal for reaching its final state.
- **Weighted Goal Conditioned Supervised Learning (WGCSL) (Yang et al., 2022):** GCSL (Ghosh et al., 2019) is an online algorithm that operates through multiple rounds of collecting online data, relabeling, and training a policy on that data using GCBC. WGCSL (Yang et al., 2022) improves over GCSL by learning an additional value function used to weigh the GCSL loss. Here, we compare against an offline variant of WGCSL algorithm that trains for one round until convergence.
- **Offline Goal-Conditioned RL:** While offline RL is generally incompatible with play data without rewards, recently some offline goal-conditioned RL algorithms have emerged that optimize for a proxy reward defined through state occupancy. We compare against one such algorithm, GoFAR (Ma et al., 2022), which learns a value function through f-Advantage regression and then optimizes a policy that maximizes the goal-conditioned value. We choose GoFAR since to the best of our knowledge, it is the only offline GCRL algorithm with experiments on a real-robot.
- **Learning Motor Primitives from Play (Play-LMP):** Play-LMP (Lynch et al., 2019) is a behavior generation algorithm that focuses on learning short ($\sim$ 30 timesteps) motor primitives from play data. Play-LMP does so by using a variational-autoencoder (VAE) style architecture to encode action sequences into motor program latents and decoding actions from them.
- **Behavior Transformers (BeT):** We include unconditional BeT (Sec. 2) in our baseline to understand the improvements made by the C-BeT conditioning. In practice, it acts as a "random" baseline that performs the tasks without regard for the goal.
- **Unimodal C-BeT:** We use our method without the multi-modal head introduced in BeT. This also corresponds to a variant of Decision Transformer conditioning on outcomes instead of rewards.

Table 2: Results of future-conditioned algorithms on a set of simulated environments. The numbers reported correspond to the number of successes in completing the objective in each environment. In CARLA, success counts as reaching the location corresponding to the observation; for BlockPush, it is pushing the blocks into the target squares; and for Kitchen, success corresponds to the portion of conditioned tasks completed successfully.

	GCBC	WGCSL	GoFAR	Play-LMP	BeT	C-BeT (unimodal)	C-BeT (multimodal)
CARLA	0.04	0.0	0.31	0.0	0.72	0.62	**0.98**
BlockPush	0.06	0.06	0.04	0.02	0.34	0.31	**0.51**
Kitchen	0.74	1.46	1.24	0.04	1.77	2.74	**2.80**

Note that neither WGCSL nor GoFAR are directly compatible with image states and goals, since they require a proxy reward function $r : \mathcal{S} \times \mathcal{G} \to \mathbb{R}$. Thus, we had to design a proxy reward function on the image representations, $\exp\left(-(1/4||g - s||)^2\right)$ to apply them on image-based environments.

4.2 Simulated Environments and Datasets

We run our algorithms and baselines on a collection of simulated environments as a benchmark to select the best algorithms to run on our real robotic setup. The simulated environments are selected to cover a variety of properties that are necessary for the real world environment, such as pixel-based observations, diverse modes in the play dataset, and complex action spaces (see Figure. 4).

1. **CARLA self-driving:** CARLA (Dosovitskiy et al., 2017) is a simulated self-driving environment created using Unreal Engine. In this environment, the observations are RGB pixel values of dimension $(224, 224, 3)$, and actions are two-dimensional (accelerate/brake and steer). We use an environment with a fork in the road (see Figure 2) following two possible routes to the same goal, collecting 200 demonstrations in total. We condition on one of the two possible routes to the goal, and at the goal where choosing either of the two modes is valid.
2. **Multi-modal block-pushing:** We use the multi-modal block-pushing environment from Florence et al. (2021) for complicated multi-modal demonstrations. In this environment, an xArm robot pushes two blocks, red and green, into two square targets colored red and green. All positions are randomized with some noise at episode start. We use 1,000 demonstrations collected using a deterministic controller, and condition on just the future block positions on each baseline.
3. **Franka relay kitchen:** Originally introduced in Gupta et al. (2019), Relay Kitchen is a robotic environment in a simulated kitchen with seven possible tasks. A Franka Panda robot is used to manipulate the kitchen, and the associated dataset comes with 566 demonstrations collected by humans with VR controllers performing four of the seven tasks in some sequence.

4.3 How well does C-BeT learn behaviors from play?

On each of these environments, we train conditional behavior generation models and evaluate them on a set of conditions sampled from the dataset. The success is defined by the model performing the same tasks as conditioned by the future outcome. We see from Table. 2 that C-BeT performs significantly better compared to the baselines on all three tasks. BeT, as our unconditioned "random" baseline, shows the success rate of completing tasks unconditionally, and see that none of the baselines surpasses it consistently. Out of the MLP-based baselines, WGCSL performs best in the state-based tasks. However, GoFAR performs best on the CARLA vision based environment where the other two MLP-based baselines fail almost completely. We note that Play-LMP performs poorly because our tasks are long-horizon and quite far from its intended motor primitive regime, which may be challenging for Play-LMP's short-horizon auto-encoding architecture.

4.4 How important is multi-modal action modeling?

While we use a multi-modal behavior model in this work, it is not immediately obvious that it may be necessary. Specifically, some previous outcome-conditioned policy learning works (Chen et al., 2021; Emmons et al., 2021) implicitly assume that policies are unimodal once conditioned on an outcome. In Table 2 the comparison between C-BeT and unimodal C-BeT shows that this assumption may not be true for all environments, and all else being equal, having an explicitly multi-modal model helps learning an outcome conditioned policy when there may be multiple ways to achieve an outcome.

Table 4: Single-task success rate in a real world kitchen with conditional models. We present the success rate and number of trials on each task, with cumulative results presented on the last column.

	Knobs	Oven	Microwave	Pot	Cumulative
GoFAR	$0/10$	$0/5$	$0/5$	$0/5$	$0/25$
Unconditional BeT	$5/20$	$6/10$	$1/10$	$0/10$	$12/50$
Unimodal C-BeT	$1/20$	$8/10$	$4/10$	$0/10$	$13/50$
Multimodal C-BeT	$3/20$	$9/10$	$7/10$	$5/10$	$24/50$

4.5 How does C-BeT compare to other forms of conditioning?

We consider the question of how much comparative advantage there is in getting human labels for our tasks. We do so by adding manual one-hot (CARLA, BlockPush) or binary (Kitchen) labels to our tasks, and training and evaluating C-BeT with those labels. As we see on Table 3, while C-BeT is able to get close to peak performance without human labels on two out of the three environments, on BlockPush adding human labels significantly improves the performance.

Table 3: Comparison between C-BeT with no supervised labels and labels acquired with human supervision.

	No labels	Labels
CARLA	0.98	1.0
BlockPush	0.51	0.89
Kitchen	2.80	2.75

5 C-BeT on Real-World Robotic Manipulation

We now discuss our robot experiments, which are geared towards understanding the usefulness of C-BeT on real-world play data.

5.1 Robotic Environment and Dataset

Robot setup: Our environment consists of a Franka Emika Panda robot, similar to the simulated Franka Kitchen environment, set up with a children's toy kitchen set (see Figure 1). The toy kitchen has an oven, a microwave, a pot, and two stove knobs that are relevant to our play dataset. The action space in this environment contains the seven joint angle deltas normalized within the $[-1, 1]$ range, and a binary gripper control.

Play dataset: We collected 460 sequences totaling to 265 minutes (about 4.5 hours) of play data on the toy kitchen with volunteers using a Vive VR controller to move the Franka. While collecting the play data, we did not give the volunteers any explicit instructions about doing any particular tasks, or number of tasks, beyond specifying the interactable items, and stipulating that the pot only goes on the left stove or in the sink, to prevent dropping the pot and reaching an unresettable state. As the observations, we save the RGB observations from two cameras on the left and right of the setup, as well as the robot's proprioceptive joint angles. Overall, the dataset contains 45 287 frames of play interactions and their associated actions.

Representation learning To simplify the task of learning policies on image space, we decouple the task of image representation learning from policy learning following Pari et al. (2021). For each camera, we first fine-tune a pretrained ResNet-18 (He et al., 2016) encoder on the acquired frames with BYOL self-supervision (Grill et al., 2020). Then, during policy learning and evaluation, instead of the image from the cameras, we pass the two 512-dimensional BYOL embeddings as part of the observation. For the proprioceptive part of the observation, we repeat the $(\sin, \cos)$ of seven joint states 74 times to get a 1036-dimensional proprioceptive representation, making our overall observation representation 2060-dimensional.

5.2 Conditional Behavior Generation on Real Robot

Behavior generation on single tasks: Our first experiment in the real robot is about extracting single-task policies from the play dataset. We define our tasks as manipulating the four types of interactable objects one at a time: opening the oven door, opening the microwave door, moving the pot from the stove to the sink, and rotating a knob 90 degrees to the right. We use appropriate conditioning

Table 5: Task success rate in a real world kitchen with conditional models evaluated on a long-horizon goal. We present the success rate and number of trials on each task, with cumulative result presented on the last column.

	Oven $\rightarrow$ Pot	Microwave $\rightarrow$ Oven	Pot $\rightarrow$ Microwave	Avg. Tasks/Run
Unconditional BeT	$(6, 0)/10$	$(1, 6)/10$	$(0, 1)/10$	0.47
Unimodal C-BeT	$(1, 1)/10$	$(2, 0)/10$	$(8, 0)/10$	0.37
Multimodal C-BeT	$(5, 4)/10$	$(8, 8)/10$	$(4, 4)/10$	1.1

frames from our observation dataset, and start the robot from the neutral state to complete the four tasks. The result of this experiment is presented in Table 4. We see that on single task conditionals, C-BeT is able to complete all tasks except the knobs consistently, outperforming all our baselines, showing that C-BeT is able to extract single-task policies out of uncurated, real-world play data. We discuss failures of C-BeT on the knob tasks in Section 5.3. While our GoFAR baseline was able to move towards the task targets, it was unable to successfully grasp or interact with any of the target objects. We believe it may be the case because unlike the robot experiment in Ma et al. (2022), we do not have the underlying environment state, the tasks are much more complicated, and our dataset is an order of magnitude smaller (400 K vs 45 K).

Behavior generation for longer horizons: Next, we ask how well our models work for longer-horizon conditioning with multiple tasks. We choose play sequences from the dataset with multiple tasks completed and use their associated states as the conditions for our models. In our roll-outs, we calculate how many tasks completed in the original sequence were also completed in the conditional roll-outs. We calculate this metric over 3 conditioning sequences, and report the results in Table 5. We see that even without any high level controller, C-BeT is able to stitch together multiple tasks from play demonstrations to complete long-horizon goals.

Generalization to prompt and environment perturbations: A major requirement from any robot system deployed in the real world is to generalize to novel scenarios. We evaluate the generalizability of our learned policies in two different ways. In the first set of experiments, we collect fresh demonstrations that were not in the training set, and we condition our policies on such trajectories. We find that across the different tasks, even with unseen conditionings, C-BeT retains 67% of the single-task performance, with $16/50$ task successes in total. In the second set of experiments, we add environmental distractors in the setup (Figure 1, bottom three rows) and run the single- and multi-task conditions on the modified environments. We see once again that the performance drops to around 67% of original with two distractors on the scene, but if we keep adding (four or more) distractors, the robot is unable to complete any tasks.

5.3 Analysis of Failure Modes

We see a few failure modes in our experiments that may provide additional insights into learning from real-world play data. We discuss the most salient ones in this section.

Failure in knob operation in the real world: We see that in all of our real world experiments, the accuracy in operating the knob is consistently lower than all other tasks. This is due to the failure of the learned representations. Upon inspection of the dataset images' nearest neighbors in the representation space, we see that the BYOL-trained representation cannot identify the knob state better than random chance: the returned nearest neighbor differs in knob status often. Since the representation cannot identify the knob status properly, conditioning on it naturally fails.

Importance of a multi-modal policy architecture: One of our motivations behind incorporating the BeT architecture in our work is its ability to learn multi-modal action distributions. In our experiments, we show that for some single-task conditions such as opening the oven door, having no multi-modality is sufficient (Table 4), but for more complicated tasks and learning from a more interconnected form of play data, it is always the case that a multi-modal architecture prevents our policies from collapsing to sub-optimal solutions (Table 5).

6 Related Work

Outcome-conditioned behavior learning: Behavior learning conditioned on particular outcomes, such as reward or goals, is a long studied problem (Kaelbling, 1993; Schaul et al., 2015; Veeriah et al., 2018; Zhao et al., 2019). Compared to standard behavior learning, learning conditioned behavior can generally be more demanding since the same model can be expected to learn a multitude of behaviors depending on the outcome, which can make learning long-term behavior harder (Levy et al., 2017; Nachum et al., 2018). As a result, a common line of work in outcome-conditioned learning is to use some form of relabeling of demonstrations or experience buffer as a form of data augmentation (Kaelbling, 1993; Andrychowicz et al., 2017; Ghosh et al., 2019) similar to what we do in the paper. As opposed to goal or state conditioned learning, which we focus on in this paper, recently reward conditioned learning using a transformer (Chen et al., 2021) was introduced. However, later work found that it may not work as expected in all environments (Paster et al., 2022; Brandfonbrener et al., 2022) and large transformer models may not be necessary (Emmons et al., 2021) for reward conditioned learning. In this work, we find that using transformers is crucial, particularly when dealing with high dimensional visual observation and multi-modal actions.

Learning from play data: Our work is most closely related to previous works such as Lynch et al. (2019); Gupta et al. (2019), which also focus on learning from play demonstrations that may not be strictly optimal and uniformly curated for a single task. Learning policies capable of multiple tasks from play data allows knowledge sharing, which is why it may be more efficient compared to learning from demonstrations directly (Zhang et al., 2018; Rahmatizadeh et al., 2018; Duan et al., 2017; Pari et al., 2021). Gupta et al. (2022) attempts reset-free learning with play data, but requires human annotation and instrumentation in the environment for goal labels. Beyond learning policies from play data, learning self-supervised visual representations that are not entirely task-specific may also be a fruitful endeavor, as shown by Young et al. (2021); Nair et al. (2022).

Generative modeling of behavior: Our method of learning a generative model for behavior learning follows a long line of work, including Inverse Reinforcement Learning or IRL (Russell, 1998; Ng et al., 2000; Ho & Ermon, 2016), where given expert demonstrations, a model tries to construct the reward function, which is then used to generate desirable behavior. Another class of algorithms learn a generative action decoder (Pertsch et al., 2020a; Singh et al., 2020) from interaction data to make downstream reinforcement learning faster and easier, nominally making multi-modal action distribution easier. Finally, a class of algorithms, most notably Liu et al. (2020); Florence et al. (2021); Kostrikov et al. (2021); Nachum & Yang (2021) do not directly learn a generative model, but instead learn energy based models that need to be sampled to generate behavior, although they do not primarily focus on goal-conditioning.

Transformers for behavior learning: Our work follows earlier notable works in using transformers to learn a behavior model from an offline dataset, such as Chen et al. (2021); Janner et al. (2021); Shafiullah et al. (2022). Our work is most closely related to Shafiullah et al. (2022) as we build on their transformer architecture, while our unimodal baseline is a variant of Chen et al. (2021) that learns outcome conditioned instead of reward conditioned policy. Beyond these, Dasari & Gupta (2020); Mandi et al. (2021) summarizes historical visual context using transformers, and Clever et al. (2021) relies on the long-term extrapolation abilities of transformers as sequence models. The goal of C-BeT is orthogonal to these use cases, but can be combined with them for future applications.

7 Discussion and Limitations

In this work, we have presented C-BeT, a new approach for conditional behavior generation that can learn from offline play data. Across a variety of benchmarks, both simulated and real, we find that C-BeT significantly improves upon prior state-of-the-art work. However, we have noticed two limitations in C-BeT, particularly for real-robot behavior learning. First, if the features provided to C-BeT do not appropriately capture relevant objects in the scene, the robot execution often fails to interact with that object in its environment. Second, some tasks, like opening the oven door, have simpler underlying data that is not multimodal, which renders only meager gains with C-BeT. A more detailed analysis of these limitations are presented in Section 5.3. We believe that future work in visual representation learning can address poor environment features, while the collection of even larger play datasets will provide more realistic offline data for large-scale behavior learning models.

REPRODUCIBILITY STATEMENT

As a part of our commitment to reproducibility, we plan to release all our code, data, and roll-out videos on the project website, https://cbet-anon.github.io.

We have already referred to the codebases that we build upon in Section C, and included the hyperparameters used to train the models in our paper. Finally, our simulated experiments are built on free environments and thus our experiments should be easily replicable.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 3, 9

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 3

David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv: Arxiv-2206.01079*, 2022. 9

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1, 2

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 1, 6, 9

Henry M Clever, Ankur Handa, Hammad Mazhar, Kevin Parker, Omer Shapira, Qian Wan, Yashraj Narang, Iretiayo Akinola, Maya Cakmak, and Dieter Fox. Assistive tele-op: Leveraging transformers to collect robotic task demonstrations. *arXiv preprint arXiv:2112.05129*, 2021. 9

Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020. 9

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017. 3, 6

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017. 9

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021. 3, 5, 6, 9

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. *arXiv preprint arXiv: Arxiv-2206.07568*, 2022. 3

Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021. 3, 6, 9

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 3

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019. 1, 5, 9

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015. 4, 13

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 7

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 1, 3, 6, 9

Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Bootstrapped autonomous practicing via multi-task reinforcement learning. *arXiv preprint arXiv:2203.15755*, 2022. 9

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 7

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, volume 29, pp. 4565–4573, 2016. 9

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021. 9

Leslie Pack Kaelbling. Learning to achieve goals. In *IN PROC. OF IJCAI-93*, pp. 1094–1098. Morgan Kaufmann, 1993. 9

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021. 9

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017. 9

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017. 4, 13

Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 33, 2020. 9

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, S. Levine, and Pierre Sermanet. Learning latent plans from play. *Corl*, 2019. 1, 2, 3, 5, 9

Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. *arXiv preprint arXiv:2206.03023*, 2022. 1, 5, 8

Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. Towards more generalizable one-shot visual imitation learning. *arXiv preprint arXiv:2110.13423*, 2021. 9

Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. *arXiv preprint arXiv:2105.12272*, 2021. 9

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 9

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv: Arxiv-2203.12601*, 2022. 9

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 663–670, 2000. 9

Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, Lerrel Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 7, 9

Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers fail in stochastic environments. *arXiv preprint arXiv: Arxiv-2205.15967*, 2022. 9

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020a. 9

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020b. 1

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey 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)*, pp. 3758–3765. IEEE, 2018. 9

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1

Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998. 9

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html. 9

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022. 2, 3, 4, 9, 13, 14

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020. 1, 9

Vivek Veeriah, Junhyuk Oh, and Satinder Singh. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*, 2018. 9

Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *Iclr*, 2022. 5

Sarah Young, Jyothish Pari, Pieter Abbeel, and Lerrel Pinto. Playful interactions for representation learning. *arXiv preprint arXiv: Arxiv-2107.09046*, 2021. 9

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635. IEEE, 2018. 9

Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7553–7562. PMLR, 2019. 9

APPENDIX

A SIMULATED ENVIRONMENT VISUALIZATIONS

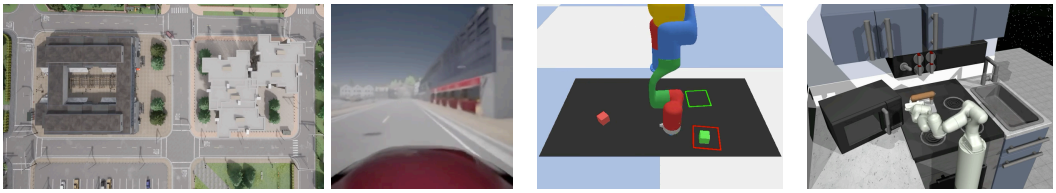

Figure 4: Visualizations of simulated environments that we evaluate our methods on, from left to right: CARLA self-driving (top down view and agent POV), BlockPush, and Franka Kitchen.

B BEHAVIOR TRANSFORMERS

We use Behavior Transformers from Shafiullah et al. (2022) as our backbone architecture, building our conditional algorithm on top of it. In this section, we describe the BeT architecture and the training objective to help the readers understand the details of our algorithm.

B.1 BeT ARCHITECTURE

BeT uses a repurposed MinGPT architecture to model multi-modal behavior. It uses the MinGPT trunk as a sequence-to-sequence model that tries to predict a sequence of actions $a_{t:t+h}$ given a sequence of states or observations $o_{t:t+h}$. Beyond just prediction the actions, however, BeT tries to model the multi-modal action distribtuion given the observations. To create a multi-modal model over the continuous action distribution, BeT uses an action encoder-decoder architecture that can encode each action vector into a discrete latent and a smaller-norm continuous offset. BeT does so by using an offline action dataset to create a k-means model of the actions. Then, an action is encoded into its associated bin out of the k-bins (binning), and a small continuous offset from the associated bin.

The BeT model, given a sequence of observations $o_{t:t+h}$, predicts a k-dimensional multinomial distribution over the k-bins, as well as a $k \times |A|$ dimensional matrix for offsets associated with each action bins. Sampling from the BeT action distribution is done via sampling a discrete bin first, taking its associated action offset, and then adding the bin center with the action offset.

B.2 BeT TRAINING OBJECTIVE

Given an observation o and its associated ground truth action a, we will now present the simplified version of how the BeT loss is calculated.

Let us assume the BeT model prediction is $\pi(o)_d \in \mathbb{R}^k, \pi(o)_c \in \mathbb{R}^{k \times |A|}$ for the discrete and the continuous parts of the predictions. Let us also assume that $\lfloor a \rfloor$ is the discrete bin out of the k bins that a belongs to, and $\langle a \rangle = a - \text{BinCenter}(\lfloor a \rfloor)$. Then, the BeT loss becomes

$$\mathcal{L}_{\text{BeT}} = L_{focal}(\pi(o)_d, \lfloor a \rfloor) + \lambda \cdot L_{MT}(\langle a \rangle, \pi(o)_c)$$

Where L_{focal} is the Focal loss (Lin et al., 2017), a special case of the negative log likelihood loss defined as

$$\mathcal{L}_{focal}(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

and L_{MT} is the multi-task loss (Girshick, 2015) defined as

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

C IMPLEMENTATION DETAILS

C.1 IMPLEMENTATION USED

In our work, we base our C-BeT implementation off of the official repo published at https://github.com/notmahi/bet. For the GCBC, WGCSL, and GoFAR baselines, we use the official repo released by the GoFAR authors https://github.com/JasonMa2016/GoFAR/.

C.2 HYPERPARAMETERS LIST:

We present the C-BeT hyperparameters in Table 6 below, which were mostly using the default hyperparameters in the original Shafiullah et al. (2022) paper:

Table 6: Environment-dependent hyperparameters in BeT.

Hyperparameter	CARLA	Block-push	Kitchen
Layers	3	4	6
Attention heads	4	4	6
Embedding width	256	72	120
Dropout probability	0.6	0.1	0.1
Context size	10	5	10
Training epochs	40	350	50
Batch size	128	64	64
Number of bins k	32	24	64
Future conditional frames	10	3	10

The shared hyperparameters are in Table 7.

Table 7: Shared hyperparameters for BeT training

Name	Value
Optimizer	Adam
Learning rate	1e-4
Weight decay	0.1
Betas	(0.9, 0.95)
Gradient clip norm	1.0

D ROBOT ENVIRONMENT DEMONSTRATION TRAJECTORIES

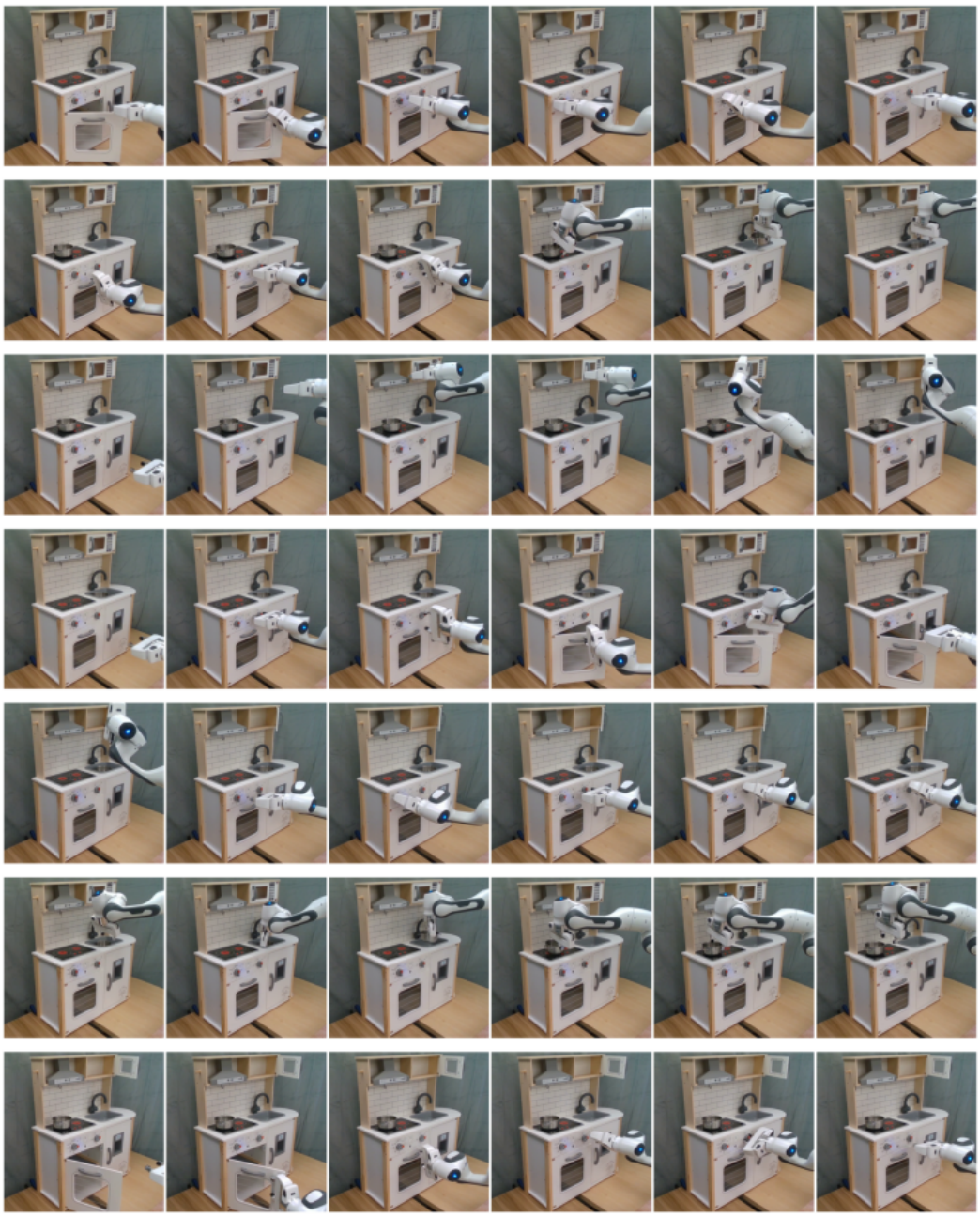

Figure 5: Sample demonstration trajectories for the real kitchen environment.

E SIMULATED ENVIRONMENT ROLLOUT TRAJECTORIES

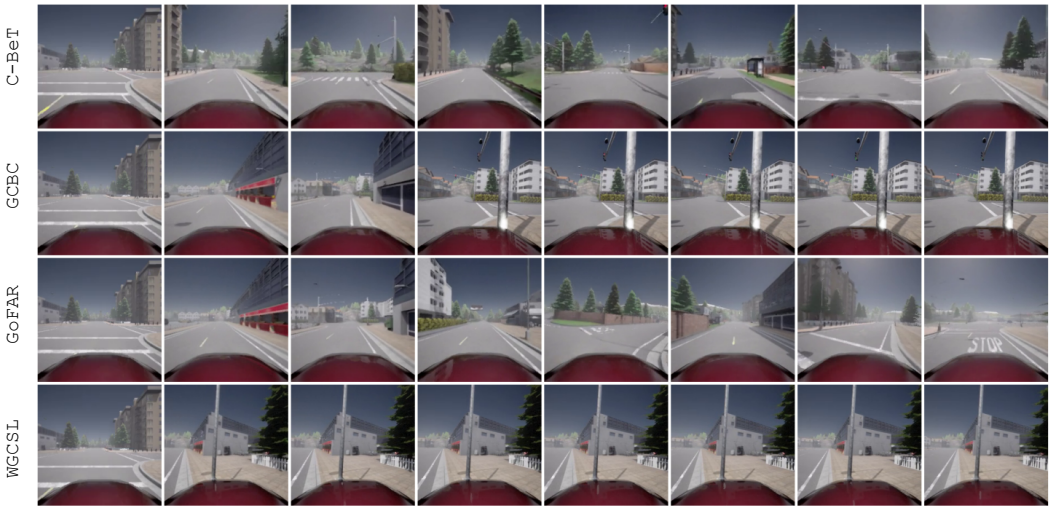

Figure 6: Sample demonstration trajectories for the CARLA self driving environment, conditioning on going to the right path.

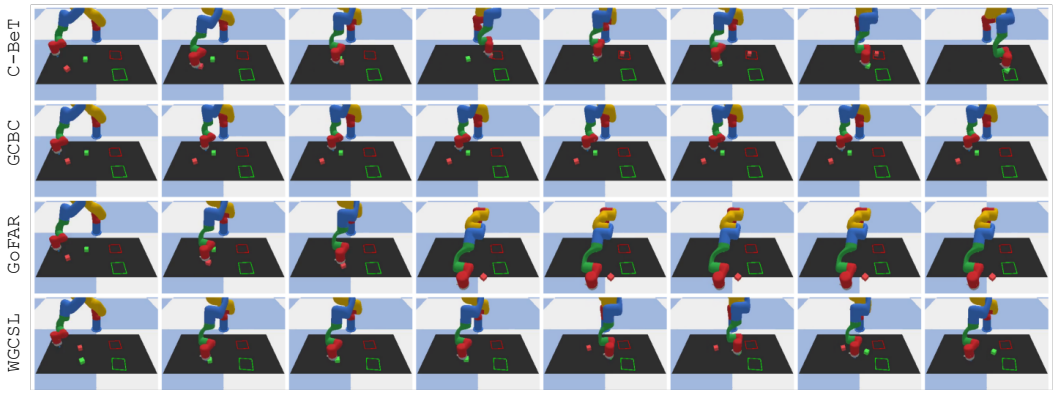

Figure 7: Sample demonstration trajectories for the multi-modal block pushing environment, conditioning on pushing the green block to green square and red block to red square.

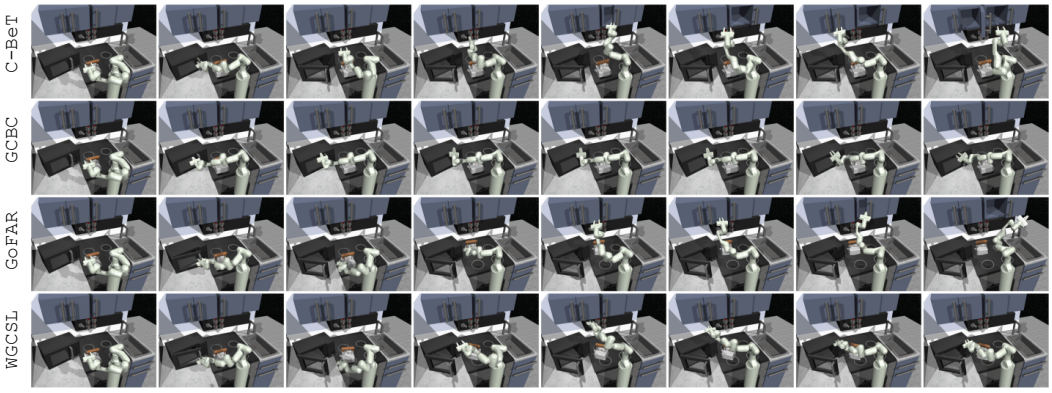

Figure 8: Sample demonstration trajectories for the Franka Kitchen environment, conditioning on completing the microwave, bottom knob, slide cabinet, hinge cabinet tasks.