WHAT ENABLES IN-CONTEXT BEHAVIOR PROMPTING FOR MANIPULATION?

Austin Patel[1] **Ben Pekarek**[1] **Joel Enrique Castro Hernandez**[2] **Shuran Song**[1]
[1]Stanford University [2]University of California, Berkeley
behavior-prompting.github.io

ABSTRACT

Behavior prompting is a paradigm in which a sensorimotor robot demonstration, called a behavior prompt, serves as an in-context prompt for performing new tasks at test time. While prior work has shown that this capability is possible, the conditions that enable it remain poorly understood. We present an empirical study of when, how, and why behavior prompting works. To support this study, we introduce DrawAnything and LIBERO-Gen, benchmarks with up to 2000 procedurally generated tasks that evaluate test-time adaptation to unseen drawing and tabletop manipulation tasks. We also present Behavior Prompting Policy (BPP), an in-context visuomotor architecture, and iPhUMI, a handheld interface to demonstrate behavior prompts at test time. Our main finding is that task diversity, rather than demonstrations per task, is a key driver of prompting capability. Given sufficient diversity, a behavior prompt improves adaptation to unseen tasks, reducing drawing error by 80.7% over goal-image conditioning and improving success on chained manipulation tasks by up to 20.8% over language conditioning. Given insufficient diversity in a real-world laundry experiment, behavior prompting has weaker task conditioning than a language baseline. An attention analysis shows how the prompt is used: the policy follows it step by step as a source of dense sub-goals. Prompt ablations show that dense sensorimotor detail matters: removing actions or downsampling the prompt hurts fine-grained action adaptation. We have open-sourced all components to enable reproducible research on behavior prompting without needing industrial-scale data collection or compute.

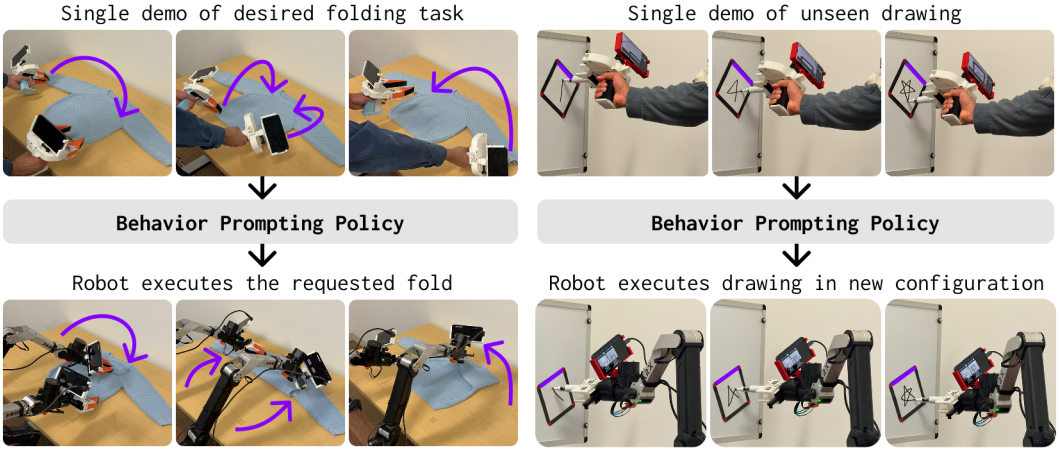


Figure 1: **Behavior prompting** conditions test-time execution on a single robot demonstration. This enables a user to specify a task via demonstration (left) or define new robot capabilities (right).

1 INTRODUCTION

Teaching a robot a new skill typically requires fine-tuning a policy (Black et al., 2024; Kim et al., 2025b). Large language models instead adapt to new tasks from a few examples placed in context, with no parameter updates (Brown et al., 2020). *Behavior prompting* brings this paradigm to manipulation (Fig. 1): at test time, the policy is conditioned in-context on a single demonstration of the

desired task, consisting of observations and actions in the robot's sensorimotor space (i.e., a *behavior prompt*). This prompt specifies *what* the task is and an example of *how* to complete it, whereas language instructions often leave the *how* underspecified (Gu et al., 2024; Jiang et al., 2023).

Adapting a policy from a demonstration has a long history in few-shot imitation learning (Duan et al., 2017; Finn et al., 2017; James et al., 2018; Mandi et al., 2022). More recently, ICRT (Fu et al., 2025) showed that an in-context policy can condition directly on raw sensorimotor demonstrations (i.e., behavior prompts) to execute new tasks at test time. ICRT establishes that behavior prompting is possible, but what remains missing is an understanding of the conditions that enable it.

Our goal is to understand what enables behavior prompting: *when* it works, *how* the policy uses the prompt, and *why* the prompt helps. Such a study needs a setting where training task diversity can be varied and where new *unseen* tasks, specified at test time, require the policy to rely on the prompt. It also needs ablations cheap enough to run systematically. ICRT's evaluation, a single real-world tabletop domain with six motion primitives, was designed to validate the approach rather than to support such a study, since both its task diversity and its ablations are limited by the cost of real-world data collection and evaluation. Existing simulation benchmarks remove this cost and enable reproducibility, but their scenes often reveal the task (Yu et al., 2020; James et al., 2020), making the prompt unnecessary, or, like LIBERO (Liu et al., 2023), only evaluate on training tasks.

We therefore study three manipulation domains across simulation and the real world. **DrawAnything** (simulation and real) trains on up to 2000 procedurally generated drawings and tests whether a policy can reproduce new human-drawn figures at test time, which requires fine-grained action adaptation. **LIBERO-Gen** (simulation) procedurally extends LIBERO to hundreds of tabletop manipulation tasks within a shared scene to test adaptation to new combinations and sequences of skills. **Laundry folding** (real) probes the low-diversity regime with only three tasks. To study them, we introduce Behavior Prompting Policy (BPP), a visuomotor architecture for behavior prompting that cross-attends to an encoded prompt and generates actions with diffusion (Chi et al., 2023). For the real-world experiments, we design iPhUMI, an iPhone-based handheld gripper building on UMI (Chi et al., 2024) that lets a human specify behavior prompts at test time without a robot.

Together, these let us answer three questions that were previously out of reach. **When does behavior prompting work?** Ablating the training data, we find that task diversity, rather than demonstrations per task, is a key driver of prompting capability. With sufficient diversity, a behavior prompt improves adaptation to unseen tasks, reducing drawing error by 80.7% over goal-image conditioning and improving success on chained tabletop manipulation by up to 20.8% over language conditioning. In our laundry folding experiment where task diversity is low, task conditioning weakens: the policy identifies the intended fold less reliably through demonstration than with language. **How does the policy use the prompt?** Analyzing the policy's attention, we find that it tracks the part of the prompt matching its current progress, using it as a source of dense sub-goals. **Why does the prompt help?** Ablating the prompt contents on DrawAnything, we find that the advantage comes from dense sensorimotor detail: observations anchor the lookup into the prompt, and removing actions or downsampling the prompt below 1 Hz degrades fine-grained action adaptation.

In summary, our contributions are: **1)** DrawAnything and LIBERO-Gen, benchmarks with up to 2000 procedurally generated tasks that evaluate behavior prompting on unseen tasks. **2)** An empirical study of when, how, and why behavior prompting works, with findings on task diversity, how the policy uses the prompt, and which prompt contents matter. **3)** Behavior Prompting Policy, a strong reference architecture for prompting. We have open-sourced all components to enable reproducible research on behavior prompting without needing industrial-scale data collection or compute.

2 Related Work

Test-time Adaptation to New Tasks. Language-conditioned multi-task policies (Jang et al., 2021; Octo Model Team et al., 2024) and VLA models (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025) achieve zero-shot adaptation to some unseen environments and objects, but rarely to new low-level actions or skills. Few-shot methods instead adapt to a new task from a small set of demonstrations, and differ in how they use the demonstration. *Weight-update* methods adapt parameters with a few gradient steps via meta-learning (Finn et al., 2017; Yu et al., 2018) or online via test-time training (Jiang et al., 2026). *Structured* methods transfer a demon-

stration to a new scene through explicit reference frames (e.g., task-parameterized models (Calinon, 2016) and align-then-replay (Valassakis et al., 2022)), learned correspondences (Zhang & Boularias, 2024), or task priors (Dreczkowski et al., 2025), trading generality for sample efficiency. *Conditioning* methods keep the weights fixed and feed the demonstration to the policy, either compressed into a task embedding (James et al., 2018) or attended to in context (Duan et al., 2017; Mandi et al., 2022; Jain et al., 2024; Xu et al., 2022; Shah et al., 2026), in some cases retrieved at test time from a larger set (Sridhar et al., 2025; Papagiannis et al., 2025).

In-context few-shot methods further differ in the prompt contents. Rather than raw observations and actions, many represent the demonstration as keypoints (Di Palo & Johns, 2024), graphs (Vosylius & Johns, 2025), multi-modal prompts (Jiang et al., 2023), reasoning traces (Nguyen et al., 2026), or structured rationales (Xu et al., 2026a). Others prompt with human video (Yu et al., 2018; Jain et al., 2024; Wang et al., 2023; Xu et al., 2023; Kim et al., 2025a; Zhu et al., 2026; Zhou et al., 2026; Jiang et al., 2026; Rhoda AI Team, 2026), which is easier to collect but crosses an embodiment gap.

Behavior Prompting. Behavior prompting is an in-context few-shot conditioning method in which the prompt is a raw demonstration in the robot's sensorimotor space. Compared to human video, *a behavior prompt has no embodiment gap and is already accessible from existing demonstration data during training*. However, performing a new task requires one test-time demo in the robot's space, e.g., with a handheld gripper. ICRT (Fu et al., 2025) introduces a behavior prompting architecture and validates the paradigm on a real robot in a single tabletop domain, training on 1098 trajectories spanning 29 tasks and 6 motion primitives. ICRT ablates model initialization, DROID pretraining, and the training loss, but does not study training task diversity, prompt modalities, or how the policy attends to the prompt, all of which we study. Concurrent industrial systems also report in-context demonstration prompting, where the capability either emerges from long-context training (Generalist Team, 2026) or is trained for explicitly (Skild AI, 2026), attributing it to pretraining scale and to task diversity, respectively. ***These works show that such prompting is possible but none isolates the enabling factors in a controlled, reproducible study, which is the focus of this work.***

Existing benchmarks do not support such a study out of the box. They evaluate only on training tasks (Liu et al., 2023; Zhou et al., 2025; Fei et al., 2025), in LIBERO's case language-sufficient and performance-saturated (Tab. 1); offer tasks identifiable from the scene alone, so the prompt is unnecessary (Yu et al., 2020; James et al., 2020); or provide no per-task demonstrations (Yang et al., 2026). Closest to our setting, VIMA-Bench (Jiang et al., 2023) holds out novel prompt-specified tasks, but over object rearrangements with fixed action primitives, and its prompts are object-centric rather than sensorimotor. Data generators such as RoboTwin 2.0 (Chen et al., 2025) could be extended to host our protocol, just as we extended LIBERO into LIBERO-Gen. Our benchmarks combine many demonstrable tasks per scene and support specifying new tasks at test time via a sensorimotor demonstration; they allow researchers to meaningfully study algorithm design of behavior prompting without the requirements of industrial-scale data and compute.

3 METHOD

3.1 BEHAVIOR PROMPT DEFINITION

A behavior prompt is a single demonstration of a desired task (at a different environment configuration) consisting of a sequence of observations, proprioception, and actions in the robot's sensorimotor execution space. At test time, the prompt is a training demonstration or, for new tasks, a human demonstration (e.g., with a handheld gripper). A demonstration is most useful when it clarifies what language leaves underspecified: spatial-temporal structure (e.g., a drawing), preferences easier to show than describe (e.g., a preferred fold), or a manipulation strategy (e.g., a grasp type). *Behavior prompting thus inherits the benefits of video conditioning (i.e., visual sub-goals for the task) while adding actions*, which we find improve fine-grained action adaptation (e.g., following drawings).

3.2 BEHAVIOR PROMPTING POLICY (BPP)

Behavior Prompting Policy is an in-context visuomotor architecture that takes a behavior prompt and current observation as input and outputs closed-loop actions (Fig. 2). BPP consists of a prompt encoder and an action decoder, like in Vid2Robot (Jain et al., 2024). We position BPP as a strong reference architecture for behavior prompting, with practical benefits, discussed below.

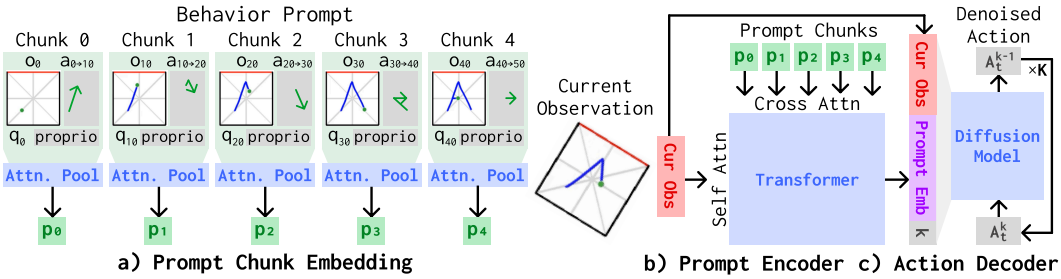


Figure 2: **Behavior Prompting Policy architecture.** a) Every Δt steps of the behavior prompt form a chunk that contains one step of observation and proprio along with Δt actions. Attention pooling merges $\{o, q, a\}$ into a chunk embedding p_i. The policy consists of: b) a prompt encoder, which extracts relevant prompt information given the current obs, and c) an action decoder, which reasons over the current obs and relevant prompt information to generate actions over K diffusion steps.

Prompt Encoder (Fig. 2b): To encode the prompt, we temporally downsample the observations and proprioception for computational efficiency (typically to 1Hz). Actions are not downsampled to retain the full behavior sequence. The prompt is a sequence of chunks each consisting of a single timestep of observation o, proprioception q, and the sequence of actions a leading up to the next chunk. We apply attention pooling per chunk to merge $\{o, q, a\}$ into a single chunk embedding p_i forming the sequence $P = [p_0, p_1, \ldots, p_n]$ where n varies with prompt length. The pooling temporally associates information from the same timestep and reduces the prompt sequence length.

The prompt encoder is a transformer decoder that extracts relevant prompt information given the current observation. We tokenize the current observation by representing each observation entry with a single token per timestep of history and then perform cross-attention with the prompt chunk embeddings P. We use learned positional embeddings for the prompt and the current observation.

Action Decoder (Fig. 2c): Given the relevant prompt information extracted by the prompt encoder, the policy now needs to generate actions. We concatenate the current observation, the extracted prompt information, and diffusion timestep k and pass it to a diffusion model to denoise actions. We use the CNN action diffusion architecture with FiLM (Perez et al., 2018) from Chi et al. (2023).

Training: We follow standard behavior cloning training practices for action diffusion policies with additional care taken to handle the behavior prompt. For each training step, we sample a single demonstration from the training data as the prompt. We then load a batch of receding horizon observations and future action chunks from other demonstrations of the *same task*. Each demonstration will have varying environment configurations, so the policy must reason about the *temporal similarities* and *spatial differences* between the prompt and the current observation to generate actions.

During training we require no explicit correspondence, spatially or temporally, between two demonstrations of the same task. The prompt is simply provided as an additional model input and the model learns end-to-end how to leverage it. Since prompts are demonstrations, ***BPP can be directly trained on existing multi-task imitation datasets with no additional data collection***.

Inference: We select a single prompt per rollout, encode it once, and cache the chunk embeddings.

Comparison to ICRT: BPP differs from ICRT (Fu et al., 2025) in three ways: (i) prompt handling, cross-attention from a separate encoder versus causal self-attention; (ii) the action decoder, diffusion versus L1; and (iii) history, a short window versus the full rollout retained in context. §4.1 isolates the benefit of (ii) and the effect of (iii) under disturbance. Two practical benefits follow: action diffusion accounts for most of BPP's improvement over ICRT, and BPP's short window never reaches a context limit during execution, unlike ICRT's retained history. Further comparisons are in §G.

3.3 iPhUMI

iPhUMI is a handheld gripper for collecting training data and specifying behavior prompts. It adapts the UMI gripper (Chi et al., 2024) by replacing the GoPro with an iPhone, following prior iPhone-based grippers (Shafiullah et al., 2023). ARKit provides real-time localization, removing the mapping stage, and our app wirelessly streams a demo to prompt BPP at test time. See §J for details.

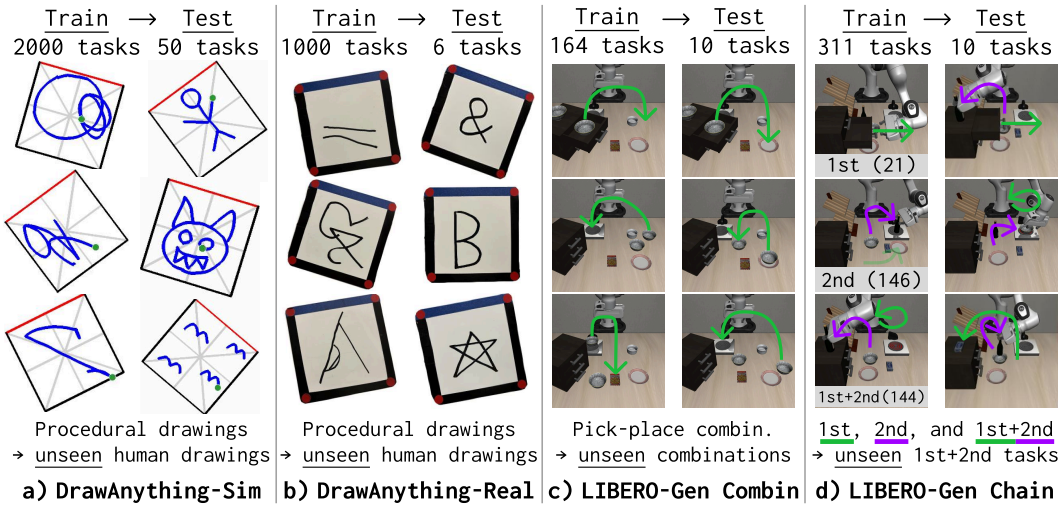


Figure 3: **Benchmarking Suite.** In DrawAnything (a, b), we evaluate whether a policy can recreate a previously unseen drawing at varying board poses given a single human demo. In LIBERO-Gen Combination (c) two identical bowls are randomly positioned, and the robot is given instructions for which one to grasp and where to place it. In LIBERO-Gen Chain (d) we explore the set of two step interactions. We have: 1) *first step tasks* include open middle/top drawer, push plate, turn on stove, and pick-place, 2) *second step tasks* just do second action (pick-place only) with first action already done, and 3) *chained tasks* where we do a first step task then a second step task in succession.

4 Evaluation

We introduce DrawAnything and LIBERO-Gen (Fig. 3), benchmarks with high task diversity, procedurally generated demonstrations, and test-time task specification of new tasks.

- **DrawAnything-Sim (Fig. 3a):** Drawing is well suited for behavior prompting as it requires the policy to continuously reference low-level drawing instructions in the prompt. We train on 2000 procedurally generated drawing tasks with 5 demos per task at randomized board orientations. We evaluate on 50 unseen drawing tasks collected by a human to ensure significant variation from the training tasks. During evaluation, the policy must reconstruct a previously unseen drawing given a single human demo. The rollout occurs at a different board orientation than the prompt.
- **DrawAnything-Real (Fig. 3b):** We create an analogous real-world drawing setup using an ARX robot arm with an iPhone wrist camera and marker attached to the end effector (Fig. 1). Within a large whiteboard we have a square drawing region that can be placed at any position and in a 90° rotation range. We collect 1000 training tasks (200 tasks at 5 demos/task from a human using iPhUMI and 800 tasks at 6 demos/task with a scripted policy). Evaluation is done on 10 tasks (4 training, 6 unseen) collected by a human using iPhUMI. This benchmark requires full 6DoF action (compared to 2D in sim) and robustness to visual occlusions caused by the marker.
- **LIBERO-Gen Combination (Fig. 3c):** We introduce LIBERO-Gen, a procedural generation framework for LIBERO (Liu et al., 2023). Compared to other LIBERO extensions that focus on visual or semantic robustness (Fei et al., 2025; Zhou et al., 2025), LIBERO-Gen evaluates *test-time instruction-following capability on unseen tasks* by generating new environments, tasks, and demonstrations. Using this tool, we create LIBERO-Gen Combination which extends the 10 tasks in LIBERO Spatial with 164 more tasks in the same environment. All tasks involve picking one of the two identical bowls and placing it on one of nine locations. For evaluation, we hold out 10 "combinations" of pick-place locations seen individually during training, but never jointly.
- **LIBERO-Gen Chain (Fig. 3d):** Using LIBERO-Gen, we create LIBERO-Gen Chain, which extends the 10 original tasks in LIBERO Goal with 311 additional tasks in the same environment. The tasks involve sequentially executing two single-step tasks in a "chain." To understand whether behavior prompting can perform unseen chains of manipulation skills at test time, we hold out 10 two-step tasks consisting of individually seen manipulation skills.

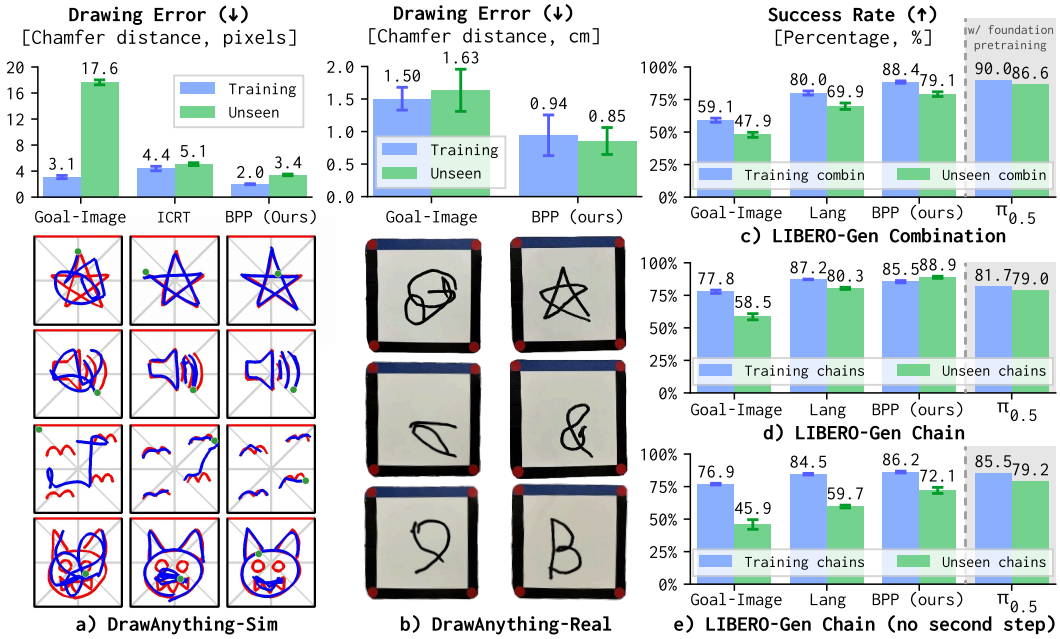


Figure 4: **Benchmark Results.** For DrawAnything (a,b) we find `Goal-Image` performs well only on training drawings, while BPP and ICRT perform well on unseen drawings. Side-by-side qualitative results in (a,b) are for *unseen* drawings. For unseen manipulation tasks in LIBERO-Gen (c,d,e), BPP outperforms baselines and rivals $\pi_{0.5}$ despite not having foundation pretraining. We report the ± 1 stdev error bar across three seeds for sim ($\pi_{0.5}$ is single seed) and across tasks for a single seed for the real-world drawing.

4.1 When does behavior prompting work?

We compare to `Goal-Image` and `Language` that match the BPP action decoder but replace the prompt encoder with ViT goal image encoding (Dosovitskiy et al., 2021) and finetuned CLIP language encoding (Radford et al., 2021), respectively. We also compare to ICRT (Fu et al., 2025), a prior behavior prompt model, and *finetuned* $\pi_{0.5}$ (Physical Intelligence et al., 2025), a VLA with foundation pretraining. $\pi_{0.5}$ has foundation pretraining so it is not directly comparable.

Behavior prompting improves test-time adaptation to unseen tasks. DrawAnything: We observe BPP can reconstruct new drawings from a single human demonstration via mouse in sim or iPhUMI in real (Fig. 4a,b). For unseen drawings, BPP reduces error compared to `Goal-Image` by 80.7% in sim and 47.9% in real. A single goal image only indicates *what* drawing to do, while a behavior prompt describes a step-by-step example for *how* to complete a drawing. Against ICRT (Fu et al., 2025), BPP reaches an error of 3.43 ± 0.10 versus 5.07 ± 0.22, a 32.3% reduction. To dissect this gain, replacing BPP's diffusion head with L1 regression to match ICRT raises its error to 4.84 ± 0.22, so the action decoder accounts for most of the gap. Because ICRT retains the full rollout history, a mid-rollout disturbance that reverts the canvas to an earlier state leaves its context out of distribution: its error increases by 10.47 ± 0.98, versus 5.48 ± 0.15 for BPP, whose fixed-length history is less affected (disturbance protocol in §G; recovery mechanism discussed in §4.2).

LIBERO-Gen: BPP improves test-time adaptation to *unseen* manipulation tasks for both LIBERO-Gen benchmarks. For LIBERO-Gen Combination there are two decisions (which bowl to pick and where to place it) and for LIBERO-Gen Chain there are up to four (ex1: *pick, place, pick, place*, ex2: *turn stove knob on, pick, place*). Language explicitly indicates these steps, while the goal image only provides the final state. Although language names each step, a behavior prompt additionally grounds each step in the robot's sensorimotor space, showing where to grasp and place and how the steps connect. In LIBERO-Gen Combination (Fig. 4c), BPP improves on unseen tasks by 13.2% over `Language`. In LIBERO-Gen Chain (Fig. 4d,e) we ablate including 2nd step tasks in the training data (1st,2nd,1st+2nd $\rightarrow$ 1st,1st+2nd). Removing 2nd step tasks makes adaptation to unseen chained tasks (1st+2nd) more challenging as the policy will have never seen the 2nd step after the 1st step has

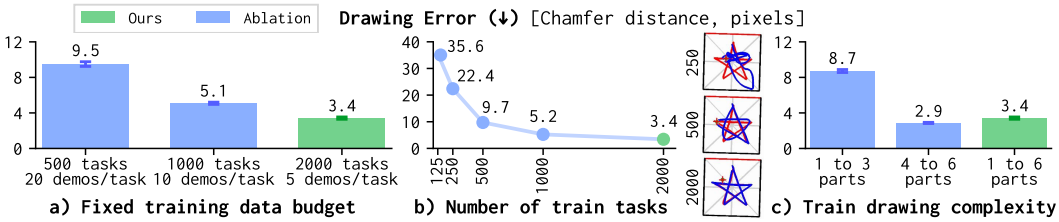


Figure 5: **Training data ablations on DrawAnything-Sim**. We report performance on *unseen* drawings (3 seeds). Panel (b) uses 5 demos/task and shows representative qualitative examples.

Task	Result	Language	BPP (Ours)
Left arm	Success	24 (96%)	19 (76%)
	Fail, dropped left sleeve	1 (4%)	4 (16%)
	Fail (a), did *fold right arm*		1 (4%)
	Fail (b), did *fold bottom up*		1 (4%)
Right arm	Success	25 (100%)	25 (100%)
Bottom up	Success	25 (100%)	15 (60%)
	Fail (c), did *fold right arm*		6 (24%)
	Fail (d), right: grabs bottom ✓, left: *fold left arm* ✗		4 (16%)

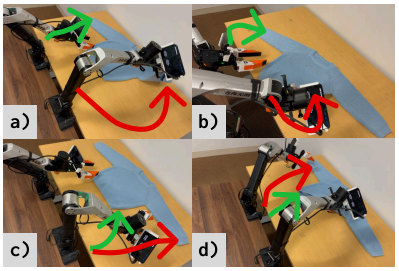


Figure 6: **Laundry folding (low task diversity study).** Left: Results over 25 rollouts per task (one seed). Language conditioning succeeds on all tasks, while BPP occasionally competently completes the wrong fold; this implies weak task conditioning at low task diversity. Evaluations use *unseen* prompts for *seen* tasks. Right: corresponding failures; green is desired action, red is executed one.

completed (ex: *never seen how to put the wine on the cabinet after a cabinet drawer is opened*). In this ablation, BPP achieves a 20.8% gain over `Language` in Fig. 4e (without 2nd step) compared to a 10.7% gain over `Language` in Fig. 4d (with 2nd step); in other words, a larger unseen adaptation gap yields greater BPP gains. Additionally, despite having no pretraining, BPP rivals finetuned $\pi_{0.5}$.

Task diversity is more important than quantity per task. All ablations in this section use DrawAnything-Sim and may vary for other domains (for LIBERO-Gen, Fig. 4e gives a data ablation and Fig. 7b,d provides the attention analysis). DATA COLLECTION STRATEGY. For a fixed demonstration budget, collecting a few demonstrations for many drawing tasks improves test-time adaptation compared to many demonstrations of few tasks (Fig. 5a). We also find that increasing the number of training tasks improves performance on unseen tasks and that this adaptation to unseen tasks only emerges at high task diversity (Fig. 5b). These trends are consistent with task-diversity effects in sequence models (Raventos et al., 2023) and in imitation learning (Lin et al., 2025), and with concurrent industrial reports that attribute demonstration prompted in-context manipulation to task diversity (Skild AI, 2026). INCLUDING COMPLEX TRAINING TASKS. We ablate the number of parts (line, circle, oval, Bézier curve) that are stitched together in the procedurally generated drawings that form the training set (Fig. 5c). We find that training only on low complexity drawings (1 to 3 parts) yields poor adaptation to complex, unseen drawings. Training on only complex drawings (4 to 6 parts) yields the best performance, but at the expense of a substantially larger demonstration dataset. The diversity required depends on the task space: for open-ended drawing, error keeps falling until roughly 1000 tasks (Fig. 5b), whereas LIBERO-Gen's discrete combinatorial tasks benefit from prompting with a few hundred. In either case, the ability to adapt to new tasks via prompting appears only above a domain-dependent level of task diversity.

With low task diversity, task conditioning weakens. LAUNDRY FOLDING: Behavior prompts are a far richer task representation than a fixed-length language embedding, so we ask whether that richness becomes a liability when task diversity is *low*. We train on three sweater folding tasks collected with bimanual iPhUMI (Fig. 1): *fold left arm*, *fold right arm*, and *fold bottom up*. All tasks start from a fully unfolded sweater, so the policy must identify the task from its descriptor alone. We compare against a language baseline which uses a CLIP encoding (Radford et al., 2021). Further task and evaluation details are in §H.

`Language` completes all three tasks at high success, while BPP succeeds on *fold right arm* but sometimes executes the wrong fold on the other two (Fig. 6). With only three tasks, the prompt's

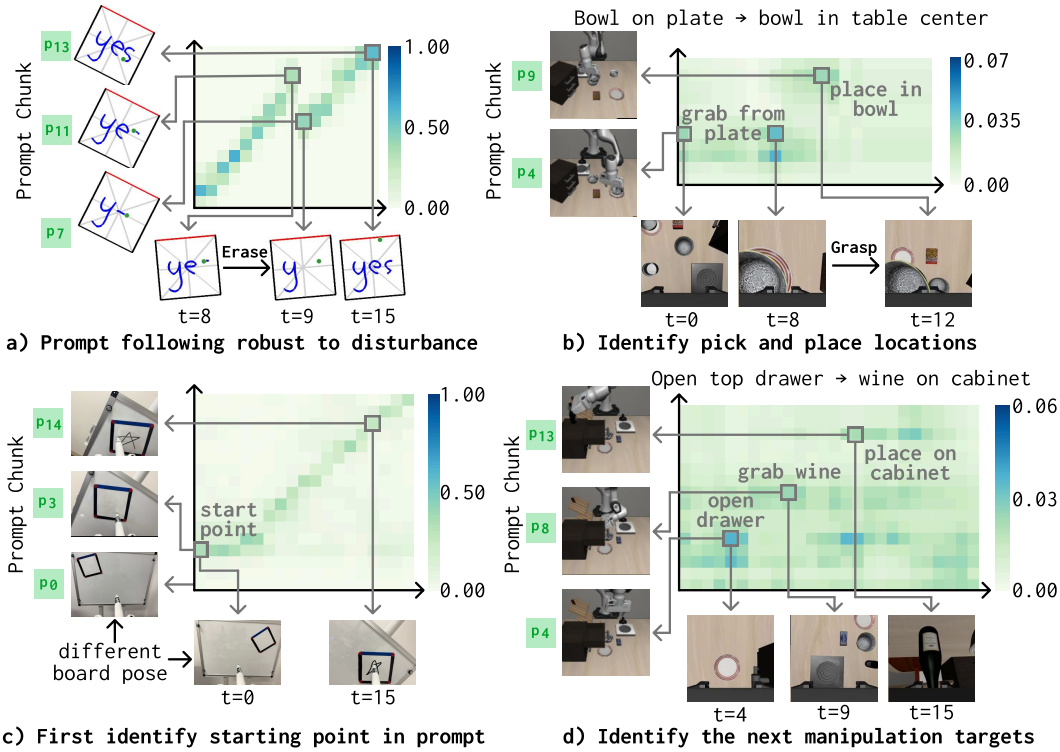


Figure 7: **Prompt encoder attention (unseen tasks).** We visualize the normalized attention scores in the BPP prompt encoder on *unseen* tasks during rollout (x-axis) to see how the model attends to the prompt (y-axis). For DrawAnything (a,c) the policy attention continuously tracks the portion of the prompt closest to the current observation, while in LIBERO-Gen (b: Combination, d: Chain) the attention tracks discrete task "milestones". The lower attention magnitude in LIBERO-Gen is due to learned attention sink tokens (Xiao et al., 2024) added to the start of the prompt (not shown).

spatial and temporal detail does not further disambiguate the task; it adds variation in prompt duration and object configuration, which we postulate weakens task conditioning. The failures are in task identification, not motor competence: BPP performs the wrong fold competently. In *fold bottom up*, for example, BPP grasps the bottom of the sweater correctly with the right arm, but the left arm then folds the left sleeve instead of also grasping the bottom, switching mid-execution to a different training task (Fig. 6d). This result is expected given the task diversity ablations that show low task diversity is the hardest for prompting.

4.2 How does the policy use the prompt?

BPP achieves test-time adaptation through dense sub-goal conditioning on the prompt. In Fig. 7, we visualize the attention in the prompt encoder. For DrawAnything, the attention closely follows the task progression, indicating that BPP identifies *temporal similarities* through a prompt lookup operation to find the section most similar to the current observation. As a result, the policy can extract upcoming states and actions that inform action generation after accounting for *spatial differences* between the prompt and rollout board pose. This *step-by-step guidance largely simplifies the learning complexity* compared to `Goal-Image`, which must reconstruct complex tasks from just the final state. This lookup also explains the robustness to disturbance in §4.1: when the canvas is erased mid-rollout (Fig. 7a, $t = 8 \rightarrow 9$), the attention jumps back from prompt chunk p_{11} to p_7, the chunk matching the reverted canvas, and then resumes advancing. This lookup is likely easier in BPP as the disturbance is not retained in the history, unlike ICRT. For LIBERO-Gen, we also find that the attention follows task progression, though in a more discrete manner to identify the next key event, such as a task transition, the next object to interact with, or where to place an object. These milestones are the steps a language command names, now grounded in the robot's observations.

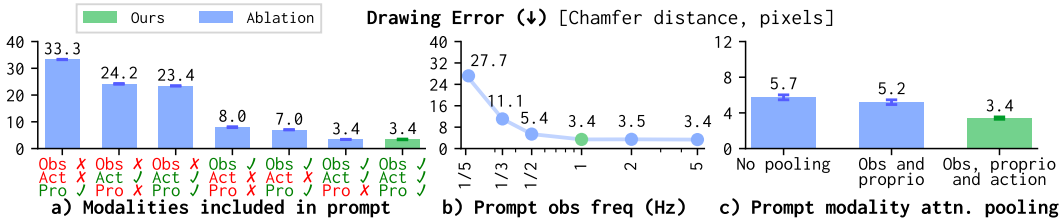


Figure 8: **Prompt ablations on DrawAnything-Sim**. We report performance on *unseen* drawing tasks (3 seeds) while ablating what information is included in the prompt and the impact of applying attention pooling to the prompt tokens. We find that dense sensorimotor information in the prompt is critical for test-time adaptation and that pooling provides further improvement.

4.3 Why does the prompt help?

The advantage comes from dense sensorimotor detail in the prompt. All ablations in this section use DrawAnything-Sim and results may vary for other domains. Including multi-modal sensorimotor information (Fig. 8a). For DrawAnything-Sim, we find that including observations in the prompt is necessary to anchor the prompt lookup and that including actions improves adaptation to unseen tasks. Proprioception (i.e., cursor position) is not useful as the cursor is already visually shown in the observation image. Including observations at a sufficiently high frequency (Fig. 8b). Aggressive downsampling of the prompt observations (less than 1Hz) makes BPP struggle to follow prompts for unseen drawings. Using attention pooling to aggregate the modalities (Fig. 8c). Attention pooling helps temporally associate modalities from the same prompt chunk and improves performance. It also reduces the sequence length by merging each prompt chunk into a single embedding, rather than having separate tokens per modality.

Robustness to prompt quality: Perturbing the prompt degrades performance only moderately: slowing the prompt by $1.25\times$ raises error from 3.43 ± 0.10 to 3.64 ± 0.12, adding trajectory jitter ($\sigma = 5$ px) to 3.69 ± 0.02, and inserting three pauses to 4.26 ± 0.12, a 24% increase (Tab. 2).

5 Limitations

Our study has several limitations. First, in our benchmarks prompts and rollouts differ in object positions and board rotations but not in scene or object identity. A general-purpose behavior prompting policy trained on large amounts of diverse manipulation data would help study cases where the prompt is more different from the target environment. Second, we identify various levels of task diversity requirements across our two benchmarks, DrawAnything-Sim and LIBERO-Gen. Future work should identify what task properties determine the task diversity requirements for behavior prompting across many domains. Third, our attention analysis characterizes BPP; applying it to other in-context architectures would test whether dense sub-goal attention lookup is a general mechanism of behavior prompting. Fourth, unseen tasks in LIBERO-Gen recombine seen skills; a benchmark that isolates adaptation to entirely new tabletop action primitives would test whether a single prompt can teach one. Finally, in many cases, goal images or language may be sufficient to achieve the desired level of test-time adaptation and do not require collecting a full behavior prompt. Future studies should thus evaluate policies that can flexibly select among task descriptors at test time.

6 Conclusion

We presented an empirical study of behavior prompting. We find that task diversity, rather than demonstrations per task, is a key driver of prompting capability; that the policy uses the prompt as a source of dense sub-goals through an attention lookup over the prompt; and that dense sensorimotor detail in the prompt matters, with observations anchoring the lookup and actions improving fine-grained action adaptation. The task diversity finding reproduces claims in concurrent industrial reports on in-context demonstration prompting, while the sub-goal attention analysis and prompt composition ablations go beyond what these reports describe. All findings were obtained in reproducible settings without requiring industrial-scale data or compute. We release all benchmarks, data, and code so that the conditions enabling in-context behavior prompting can be further studied.

Reproducibility Statement

We describe the BPP architecture, training procedure, and inference procedure in §3.2, and provide the training compute requirements and the prompt encoder and action decoder architecture specifications in Appendix §I (Tab. 4, 5, and 6). Implementation details for all baselines (`Goal-Image`, `Language`, ICRT, and $\pi_{0.5}$) are in Appendix §E, and the architectural comparison to ICRT is in Appendix §G. Our simulation benchmarks are procedurally generated: the DrawAnything-Sim training and evaluation tasks are shown in Appendix §B, and the LIBERO-Gen task generation, scripted demonstration generation, and the complete lists of held-out test tasks are in Appendix §D. For the real-world experiments, the DrawAnything-Real data generation and evaluation procedure are in Appendix §A, the laundry folding data and evaluation protocol are in §4.1, and the iPhUMI hardware, data collection, and deployment pipeline are in Appendix §J. Simulation results are reported across three seeds (except where stated) with standard deviations (Fig. 4 and 5 and 8). We have open-sourced all code and datasets, and all training hyperparameters are specified in the code release. The iPhUMI hardware and application software are open-sourced as well. All simulation results are directly reproducible with the open-sourced code and datasets.

Acknowledgments

We would like to thank the REAL lab for continuous project support, in particular Huy Ha for their insightful project guidance, Yihuai Gao for their help with the robot deployment setup, and Max Du for help with data collection. Regarding iPhUMI development, Yihuai Gao contributed USB streaming support and Xiaomeng Xu extended the bimanual data collection feature to support head-mounted iPhone data collection. We would also like to thank Chen Chen, Benoit Landry, Walter Talbott, and Jian Zhang for their feedback and discussions. This work was supported in part by the NSF Graduate Fellowship, NSF Award #2143601, #2037101, and #2132519, and Apple. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

S. Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. ISSN 1861-2776. doi: 10.1007/s11370-015-0187-9.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2402.10329.

Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Kamil Dreczkowski, Pietro Vitiello, Vitalis Vosylius, and Edward Johns. Learning a thousand tasks in a day. *Science Robotics*, 10(108):eadv7594, 2025. doi: 10.1126/scirobotics.adv7594. URL https://www.science.org/doi/abs/10.1126/scirobotics.adv7594.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Haritheja Etukuru, Norihito Naka, Zijin Hu, Seungjae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8275–8283. IEEE, 2025.

Junjie Fang, Wendi Chen, Han Xue, Fangyuan Zhou, Tian Le, Yi Wang, Yuting Zhang, Jun Lv, Chuan Wen, and Cewu Lu. RoboPocket: Improve robot policies instantly with your phone. *arXiv preprint arXiv:2603.05504*, 2026. URL https://arxiv.org/abs/2603.05504.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pp. 357–368. PMLR, 2017.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *International Conference on Robotics and Automation (ICRA)*, 2025. URL https://arxiv.org/abs/2408.15980.

Yihuai Gao, Jinyun Liu, Shuang Li, and Shuran Song. Gated memory policy. *arXiv preprint arXiv:2604.18933*, 2026. URL https://arxiv.org/abs/2604.18933.

Generalist Team. GEN-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. https://generalistai.com/blog/gen-1.5.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. RT-trajectory: Robotic task generalization via hindsight trajectory sketches. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F1TKzG8LJO.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Vidhi Jain, Maria Attarian, Nikhil J Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, Igor Gilitschenski, Yonatan Bisk, and Debidatta Dwibedi. Vid2Robot: End-to-end video-conditioned policy learning with cross-attention transformers. In *Proceedings of (RSS) Robotics Science and Systems*. Proceedings of Robotics: Science and Systems, May 2024.

Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pp. 783–795. PMLR, 2018.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, and Linxi "Jim" Fan. RoboTTT: Context scaling for robot policies. *arXiv preprint arXiv: 2607.15275*, 2026.

Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. UniSkill: Imitating human videos via cross-embodiment skill representations. In *9th Annual Conference on Robot Learning*, 2025a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025b.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pISLZG7ktL.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Zeyi Liu, Cheng Chi, Eric Cousineau, Naveen Kuppuswamy, Benjamin Burchfiel, and Shuran Song. ManiWAV: Learning robot manipulation from in-the-wild audio-visual data. In *Conference on Robot Learning*, pp. 947–962. PMLR, 2025.

Zhao Mandi, Fangchen Liu, Kimin Lee, and Pieter Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2434–2444, 2022. doi: 10.1109/ICRA46639.2022.9812450.

Toan Nguyen, Weiduo Yuan, Songlin Wei, Hui Li, Daniel Seita, and Yue Wang. ICLR: In-context imitation learning with visual reasoning. In *2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Georgios Papagiannis, Norman Di Palo, Pietro Vitiello, and Edward Johns. R+X: Retrieval and execution from everyday human videos. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Allan Raventos, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BtAz4a5xDg.

Rhoda AI Team. Causal video models are data-efficient robot policy learners. *Rhoda AI Blog*, 2026.

Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

Rutav Shah, Shuijing Liu, Qi Wang, Zhenyu Jiang, Sateesh Kumar, Mingyo Seo, Roberto Martín-Martín, and Yuke Zhu. MimicDroid: In-context learning for humanoid manipulation from human play videos. *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Skild AI. Introducing S1: In-context learning for robotics. *Skild AI Blog*, August 2026. URL https://skild.ai/blogs/s1.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding in-context adaptability to pre-trained vision-language-action models. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=6AASPlloSt.

Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. In *9th Annual Conference on Robot Learning*, 2025.

Eugene Valassakis, Georgios Papagiannis, Norman Di Palo, and Edward Johns. Demonstrate once, imitate immediately (DOME): Learning visual servoing for one-shot imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. MimicPlay: Long-horizon imitation learning by watching human play. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=hRZ1YjDZmTo.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Mengda Xu, Zhenjia Xu, Cheng Chi, Manuela Veloso, and Shuran Song. XSkill: Cross embodiment skill discovery. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=8L6pHd9aS6w.

Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, B. Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *Thirty-ninth International Conference on Machine Learning*, 2022.

Siyu Xu, Yunke Wang, Zijian Wang, Dihao Zhu, Chenghao Xia, Chengbin Du, Daochang Liu, Tao Huang, and Chang Xu. StellaVLA: In-context structured demonstration for generalizable vision-language-action models, 2026a. URL https://arxiv.org/abs/2608.11671.

Xiaomeng Xu, Jisang Park, Han Zhang, Eric Cousineau, Aditya Bhat, Jose Barreiros, Dian Wang, Jeannette Bohg, and Shuran Song. HoMMI: Learning whole-body mobile manipulation from human demonstrations. *arXiv preprint arXiv:2603.03243*, 2026b.

Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. RoboLab: A High-Fidelity Simulation Benchmark for Analysis of Task Generalist Policies. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. URL https://arxiv.org/abs/2604.09860.

Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.002.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Xinyu Zhang and Abdeslam Boularias. One-shot imitation learning with invariance matching for robotic manipulation. *arXiv preprint arXiv:2405.13178*, 2024.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, Junwei Liang, and Yinghao Xu. Zero-WAM: In-context world-action modeling from human videos for open-ended task generalization, 2026. URL https://arxiv.org/abs/2608.26103.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

Xiang Zhu, Yichen Liu, Hezhong Li, and Jianyu Chen. Learning generalizable robot policy with human demonstration video as a prompt, 2026. URL https://arxiv.org/abs/2505.20795.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

Appendix

A DrawAnything-Real implementation

A.1 Scripted data generation

For our real drawing task we train on a mixture of data collected via iPhUMI and a scripted policy. Specifically, we collect 200 tasks with 5 demonstrations per task from a human using iPhUMI. We also collect 800 tasks with 6 demonstrations per task using a scripted policy. The scripted policy maps procedurally generated drawings used in DrawAnything-Sim to open-loop commands in the 6DOF robot end-effector space. Using iPhUMI data during training enables the policy to work well with iPhUMI prompts coming from humans at test time and the scripted policy helps reach the task diversity required to enable adaptation to new tasks at test time.

For each scripted demonstration, a human operator places the drawing canvas at a random position and orientation on the whiteboard. We localize the four red corner markers on the drawing canvas with a segmentation model on an RGB image from the iPhone's main rear camera, yielding 2D keypoints in the camera frame that we use to estimate the canvas pose. We then transform the resulting pose into the robot base frame using a fixed camera-to-robot extrinsic calibration. The robot first moves to a random pose in front of the whiteboard, starts recording, then moves to the estimated canvas pose and aligns its end-effector orientation with the canvas. It executes the scripted drawing trajectory and finally retreats to a random pose. This is repeated across tasks and demonstrations.

A.2 Evaluation procedure

Qualitative results of BPP on some unseen drawings are shown in Fig. 9. Training and test tasks used for evaluation are shown in Fig. 10. For each evaluation trial, the drawing canvas is placed at a random position and orientation on the whiteboard. We sample a single demonstration to define a matched conditioning pair: the full demonstration is used as the behavior prompt for the prompting policy, and the corresponding goal image extracted from that same demonstration is used for the goal-image policy.

To control for initial conditions, we execute the prompting policy, reset the robot and environment to the same initial state, and then execute the goal-image policy with the matched goal image. We repeat this procedure after moving the canvas to a new random placement and sampling a new prompt/goal-image pair.

For evaluation on training tasks, conditioning pairs are sampled from the set of six demonstrations collected from four of the procedural training tasks. For evaluation on unseen test tasks, conditioning pairs are sampled from a set of three iPhUMI demonstrations collected for each of the six unseen evaluation tasks. Ten evaluation rollouts are done for each task.

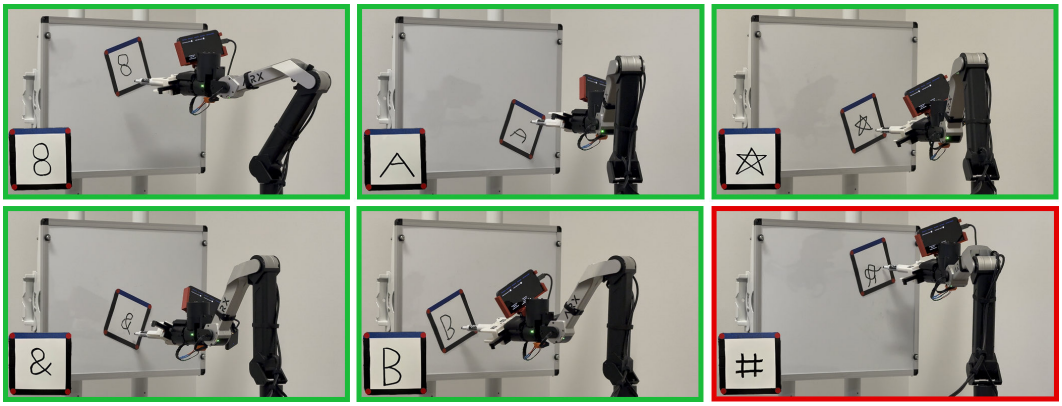


Figure 9: **BPP qualitative results on unseen tasks in DrawAnything-Real.** We find that BPP is able to reconstruct unseen drawings given a single iPhUMI demo. We show successful drawings (green) and a failure case (red).

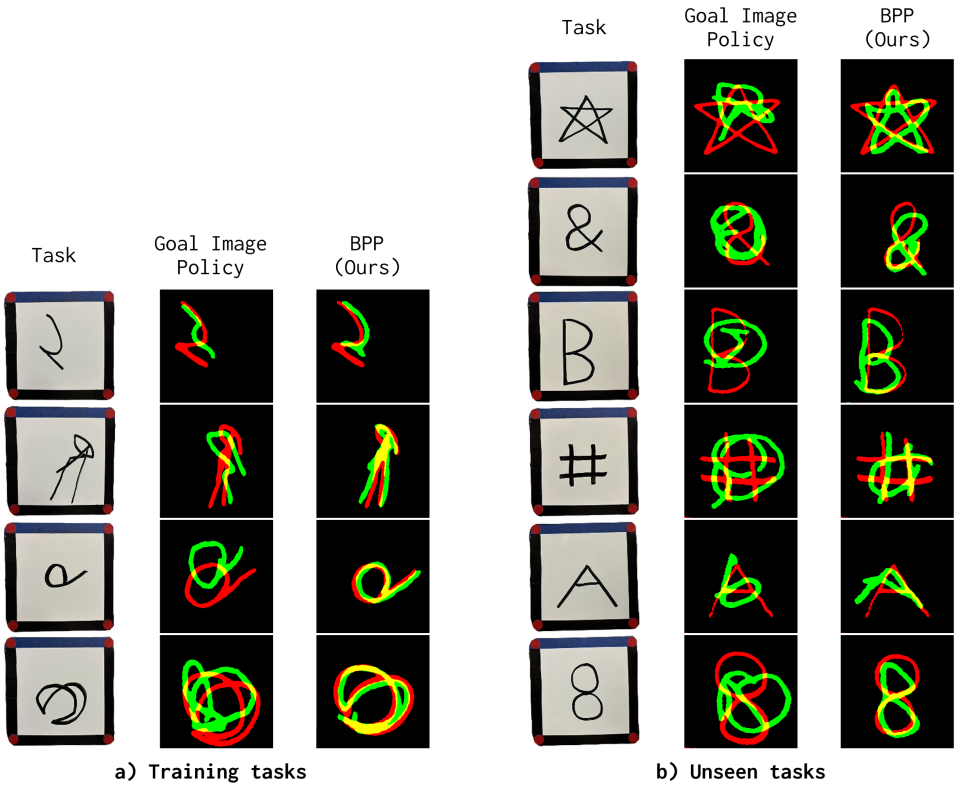


Figure 10: **DrawAnything-Real evaluation tasks with representative examples of robot executions.** Green: executed drawing by policy. Red: goal image input rendered as a reference overlay. We observe that `Goal-Image` can roughly match the structure of the training drawings, but fails to replicate unseen drawings. BPP is able to reconstruct both training and unseen drawings given a single demonstration.

A.3 Qualitative comparison: goal image vs. behavior prompting

Fig. 10 shows representative examples comparing BPP to `Goal-Image` on training and unseen drawing tasks. In DrawAnything-Sim, we observed that `Goal-Image` performs well on training tasks (Fig. 4a). In DrawAnything-Real, we find however that `Goal-Image` underperforms quantitatively on training tasks (Fig. 4b). However, qualitative inspection of Fig. 10a suggests a consistent failure mode: the policy often produces a recognizable rendition (or partial rendition) of the intended shape, but with spatial misalignment or local geometric errors that cause large Chamfer distances.

For unseen tasks, `Goal-Image` frequently degenerates into near-random strokes that do not resemble the target shape, whereas BPP produces coherent drawings aligned with the demonstrated behavior. Interestingly, we find that `Goal-Image` has similar quantitative error rates between training and unseen tasks (Fig. 4b). However, we see that unlike on training tasks, `Goal-Image` has poor qualitative fidelity on these test tasks (Fig. 10b). We believe this is because the unintelligible drawings produced by `Goal-Image` on test tasks can achieve moderate levels of Chamfer error by roughly covering the target drawing region without reproducing the intended structure.

B DRAWANYTHING-SIM TASKS

Metric. We report the Chamfer distance in pixels between the drawn canvas and the target drawing, taking the lowest distance reached during the rollout, averaged over 32 environment seeds and 50 unseen tasks.

We visualize the training (Fig. 11) and evaluation dataset (Fig. 12) for DrawAnything-Sim.

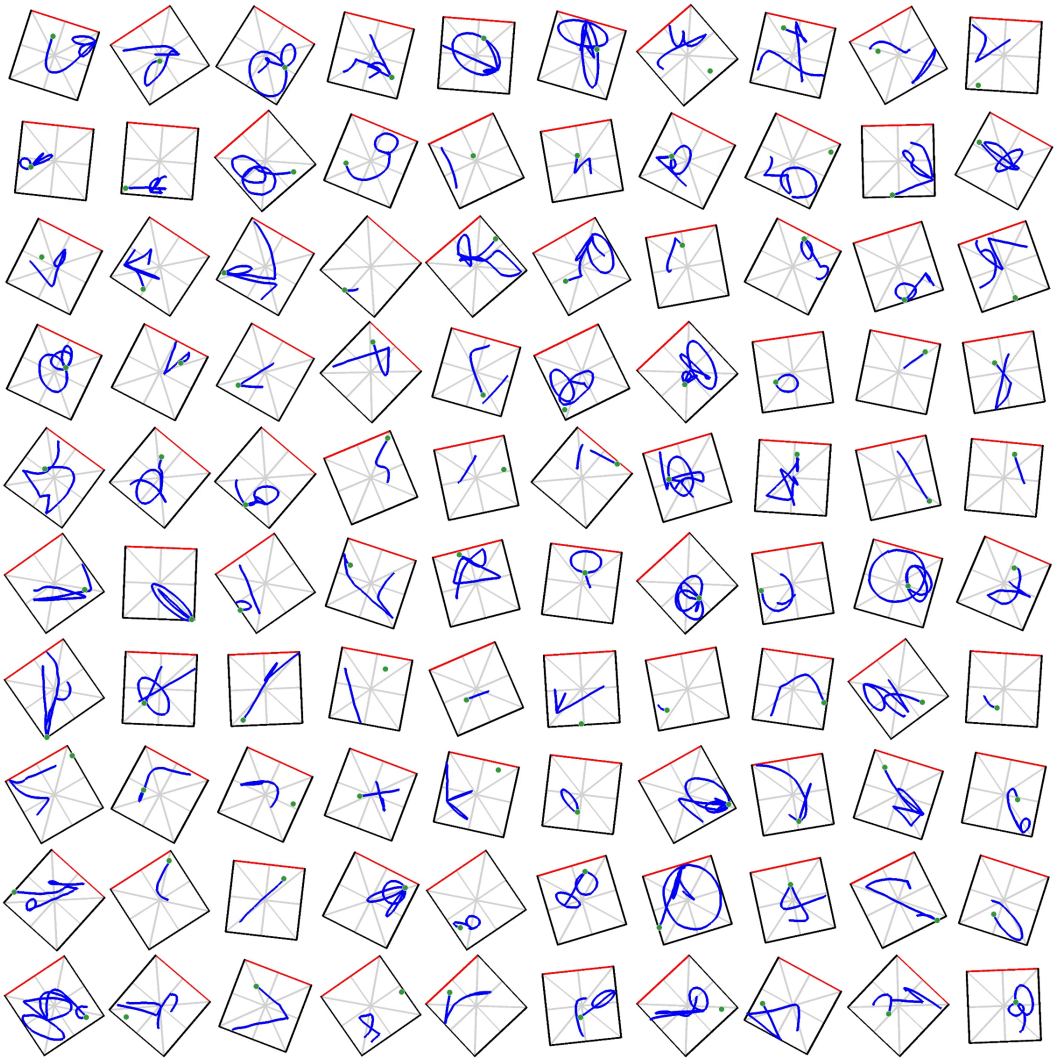


Figure 11: **DrawAnything-Sim training tasks.** We show a subset of 100 of the 2000 procedurally generated tasks (each having 5 demonstrations per task) using a combination of 1 to 6 parts. Parts include lines, Bézier curves, partial/full ovals, and free space (pen up) movement. The parameters for each part (such as start/end position, Bézier control points, proportion of oval, clockwise/counter-clockwise direction, etc.) are randomly sampled. With a specified probability, parts will connect back to the start/end position of a previous part. Each demonstration includes varied whiteboard rotation (from $-\frac{\pi}{4} to \frac{\pi}{4}$), has varied speed, and includes noise inserted into the trajectories.

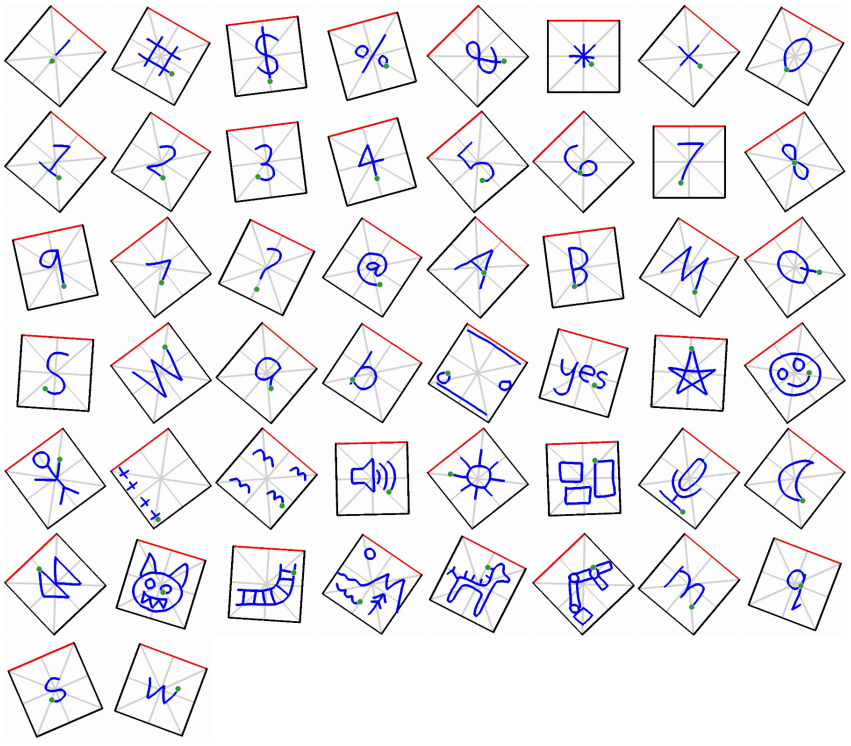


Figure 12: **DrawAnything-Sim evaluation tasks.** We hand-collect 50 evaluation tasks that were not seen during training with 5 demonstrations per task at varying board rotations. Tasks have varied complexity and duration.

C Results on original LIBERO dataset

We report results on the original LIBERO (Liu et al., 2023) dataset. The original LIBERO tasks are sufficiently described by language and the original benchmark splits do not evaluate test-time adaptation to unseen tasks. As expected, we find that using behavior prompting achieves similar performance to the language-conditioned diffusion policy. We empirically found that finetuning a CLIP language encoder gives substantially better performance compared to the frozen language embeddings used in some prior diffusion policy LIBERO evaluations (Kim et al., 2024). Modern policies have also started to saturate performance of the original LIBERO benchmark (Black et al., 2024; Kim et al., 2025b), emphasizing the importance of LIBERO-Gen which extends the dataset to evaluate test-time adaptation to unseen tasks.

Split	Language Diffusion Policy	Behavior Prompting Policy (Ours)	$\pi_{0.5}$
LIBERO Spatial	97.80 ± 0.71	97.40 ± 1.02	97.8
LIBERO Object	97.80 ± 0.49	98.93 ± 0.52	97.4
LIBERO Goal	95.60 ± 1.07	98.33 ± 0.38	97.8
LIBERO 10	93.73 ± 2.07	95.27 ± 0.66	95.6
Average	96.23 ± 0.55	97.48 ± 0.33	97.15

Table 1: **Original LIBERO dataset results.** Model success rate (%) on the original LIBERO (Liu et al., 2023) dataset by suite across 3 seeds (mean $\pm$ stdev). `Language` is detailed in §E. We finetuned $\pi_{0.5}$ (Physical Intelligence et al., 2025) for 30K LoRA (Hu et al., 2022) steps using openpi and report results for one seed. The $\pi_{0.5}$ training does not include LIBERO-90 in the finetuning mix, while the other models presented include LIBERO-90 split in the training data. Due to this difference, we put $\pi_{0.5}$ results in a separated column. For each of the three policy architectures listed we train one checkpoint per seed across all of the listed LIBERO splits rather than training and evaluating separate checkpoints per split.

D LIBERO-Gen implementation details

LIBERO-Gen procedurally generates new task definitions consisting of an environment definition and execution steps (§D.1). It also generates corresponding demonstrations for those tasks (§D.2). We then discuss details for LIBERO-Gen Combination (§D.3) and LIBERO-Gen Chain (§D.4), two benchmarks created using this tool.

D.1 Procedural task generation

The 130 original tasks in the LIBERO dataset are defined using BDDL, which defines both the environment and the intended robot task. The environment is defined by a base scene, objects present in that scene, and the initial locations of those objects. The task is defined through a set of predicates that specify the desired final states of the objects in the scene. LIBERO-Gen takes these BDDL files and enumerates the possible initial object locations and the desired final object states to generate new task definitions. The user specifies the set of procedural operations they would like to occur to generate a desired set of task variation, such as: 1) *generate environments where the goal is the same, but the plate is initialized at all of the possible starting locations* or 2) *generate environments where the initial object configuration stays the same, but the goal states are all the places the cream cheese could be placed*. LIBERO-Gen also supports the generation of chained tasks where the goal states consist of multiple action primitives applied sequentially (e.g., *turn on the stove and then put the bowl on the stove*).

In practice, we manually define a set of valid operators for each object (e.g., bowls can be grasped or you can put an object in them). We also start with a base set of environments from existing LIBERO tasks for which we apply the variations on top of. Doing so typically creates a large combinatorial space of tasks which either 1) may extend beyond the desired set of tasks for a targeted scientific experiment or 2) may want to be partitioned into a training and test set. We address this through the use of *views* which include filters that select desired portions of the generated tasks to put into training or testing splits.

D.2 PROCEDURAL DEMONSTRATION GENERATION

Given the procedurally generated task definitions, we need to procedurally generate demonstration data for these tasks. We do so through a scripted policy that leverages ground simulator object states and the existing demonstration data for the 130 original LIBERO tasks. From the existing teleoperation data, we extract object-relative grasp poses for each of the target objects. Many tasks also involve replaying a portion of teleoperation data and then switching to a scripted policy to complete the new part of the task. We introduce variations at multiple points to simulate differences present within teleoperated demonstrations: grasp location, placement position, intermediate poses on the way to or from each action primitive, and initial robot and object states. We validate the success of each demonstration using the predicate checking mechanism present in LIBERO. For chained tasks, the BDDL does not encode the ordering of each action primitive, so we encode the desired sequential order in a separate configuration file.

D.3 LIBERO-GEN COMBINATION TEST TASKS

Here we extend the discussion of LIBERO-Gen Combination from §4. We modify the LIBERO Spatial environments to have additional starting bowl locations as pictured in Fig. 13.

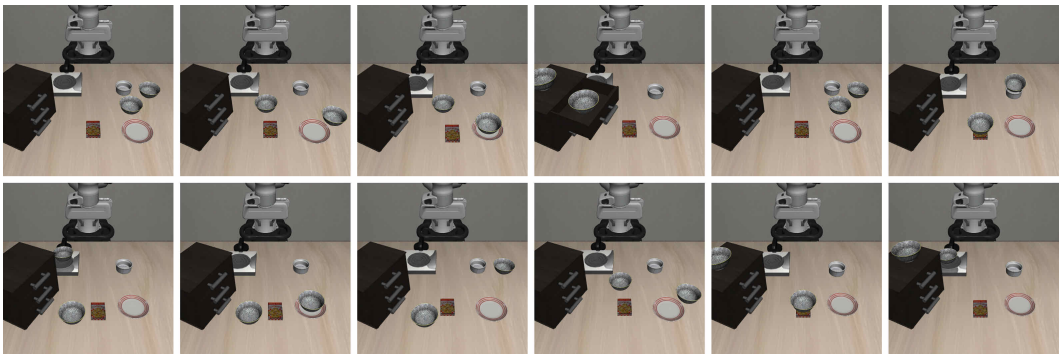


Figure 13: **Initial environments for LIBERO-Gen Combination.** There are two identical bowls in each environment. The task involves moving a specified bowl to a specified target location.

From each starting initialization, we generate the combinatorial space of all possible bowl pick locations and place locations to generate a task distribution. We withhold ten tasks (combinations of pick-place locations) to use as evaluation tasks. We document each of these unseen tasks below including the count of training tasks that have the same pick location and training tasks that have the same place location; this indicates that the pick and place locations have individually been seen during training, but never jointly. This experiment design evaluates the capability of the policy to adapt to unseen instructions at test time without evaluating unseen action primitives.

Unseen training tasks for LIBERO-Gen Combination and relevant training tasks:

```
Unseen: Pick the bowl from table center and place it on the cookies box
  Training: Same pick (bowl table center), different place [9 tasks]
  Training: Different pick, same place (bowl on the cookies box) [8 tasks]

Unseen: Pick the bowl in the top layer of the wooden cabinet and place it on the plate
  Training: Same pick (bowl wooden cabinet top), different place [7 tasks]
  Training: Different pick, same place (bowl on the plate) [9 tasks]

Unseen: Pick the bowl next to the cookies box and place it on the bowl next to the ramekin
  Training: Same pick (bowl next to box), different place [9 tasks]
  Training: Different pick, same place (bowl on the bowl next to the ramekin) [4 tasks]

Unseen: Pick the bowl next to the plate and place it on the stove
  Training: Same pick (bowl next to plate), different place [9 tasks]
  Training: Different pick, same place (bowl on the stove) [9 tasks]

Unseen: Pick the bowl on the cookies box and place it on the bowl on the wooden cabinet
  Training: Same pick (bowl cookies), different place [9 tasks]
```

```
  Training: Different pick, same place (bowl on the bowl on the wooden cabinet) [8 tasks]

Unseen: Pick the bowl on the cookies box and place it on the wooden cabinet
  Training: Same pick (bowl cookies), different place [9 tasks]
  Training: Different pick, same place (bowl on the wooden cabinet) [8 tasks]

Unseen: Pick the bowl on the plate and place it on the bowl on the table center
  Training: Same pick (bowl plate), different place [9 tasks]
  Training: Different pick, same place (bowl on the bowl on the table center) [4 tasks]

Unseen: Pick the bowl on the plate and place it on the table center
  Training: Same pick (bowl plate), different place [9 tasks]
  Training: Different pick, same place (bowl on the table center) [7 tasks]

Unseen: Pick the bowl on the stove and place it on the ramekin
  Training: Same pick (bowl stove), different place [9 tasks]
  Training: Different pick, same place (bowl on the ramekin) [9 tasks]

Unseen: Pick the bowl on the wooden cabinet and place it between the plate and the ramekin
  Training: Same pick (bowl wooden cabinet top side), different place [8 tasks]
  Training: Different pick, same place (bowl between the plate and the ramekin) [9 tasks]
```

D.4 LIBERO-GEN CHAIN TEST TASKS

Here we extend the discussion of LIBERO-Gen Chain from §4. In LIBERO-Gen Chain we have one environment initialization and enumerate the set of two-step sequences where we chain together two action primitives sequentially (see Fig. 14).

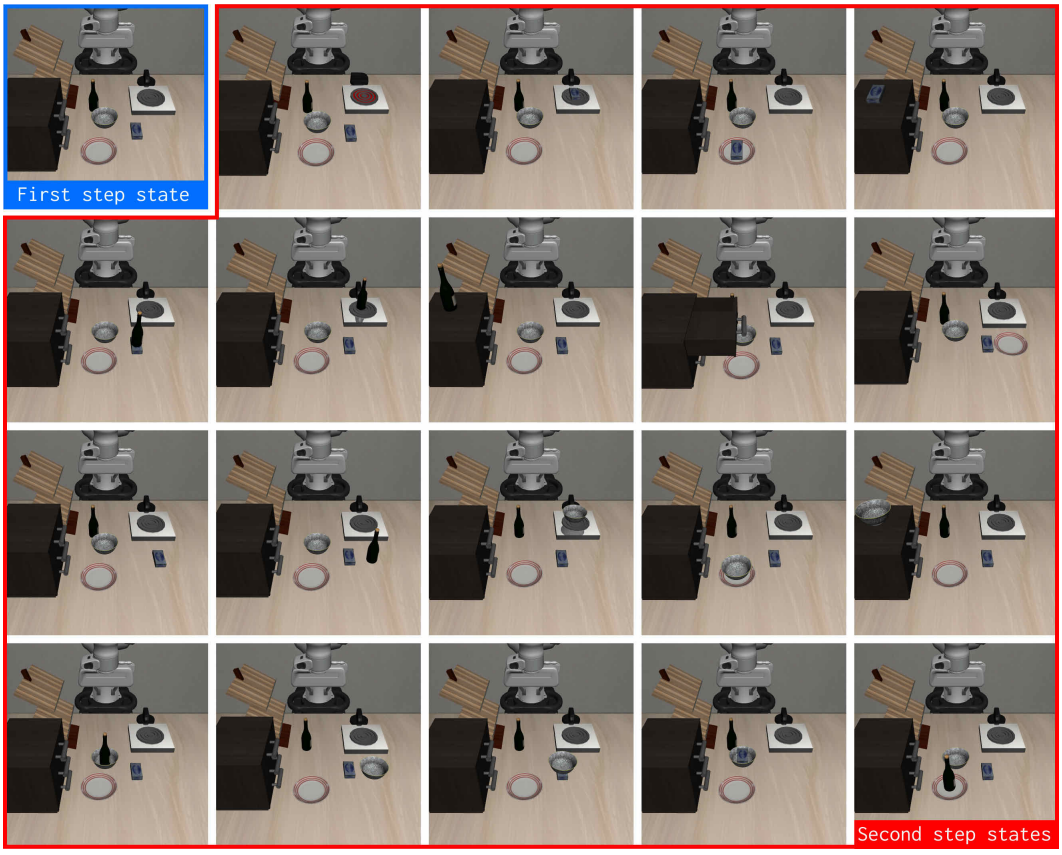


Figure 14: **Initial environment for LIBERO-Gen Chain.** All two-step chained tasks in this experiment start from the single first step state (blue). We also include one step actions starting from the first step state (blue) as well as second step actions which start assuming one action primitive has already been completed (red). The ablation for no second step excludes the single-step tasks that start from the second step initializations (red).

At test time, we evaluate adaptation to *unseen* sequential chains of individually *known* primitives. The purpose is not to adapt to an entirely new action primitive at test time that has not been seen during training, but rather to adapt to a new sequence of these primitives. This setup evaluates the capability of a model to follow previously unseen instructions. Here we enumerate all ten unseen evaluation tasks and, for each one, include the counts of the similar tasks that the model has seen during training. In short, this list demonstrates that the primitives for each unseen task have been seen individually during training, but never sequentially chained. Our ablation where we remove the second step tasks from the training mixture removes the guarantee that the second step task in the chain has been seen with the exact same environment initialization (see Fig. 14).

Unseen training tasks for LIBERO-Gen Chain and relevant training tasks:

```
Unseen: [1st] Open the middle layer of the drawer and then [2nd] put the cream cheese on the
↪  stove
  Training: similar chain (same 1st, different 2nd) [3 tasks]:
  -  Open the middle layer of the drawer and put the cream cheese inside
  -  Open the middle layer of the drawer and then put the cream cheese in front of the stove
  -  Open the middle layer of the drawer and then put the cream cheese on the top of the
  ↪  drawer
  Training: similar single-step (2nd step match with wooden cabinet middle open): not present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Open the top layer of the drawer and then [2nd] put the wine bottle on the top
↪  of the drawer
  Training: similar chain (same 1st, different 2nd) [5 tasks]:
  -  Open the top layer of the drawer and put the wine bottle inside
  -  Open the top layer of the drawer and then put the wine bottle in front of the stove
  -  Open the top layer of the drawer and then put the wine bottle on the bowl
  -  Open the top layer of the drawer and then put the wine bottle on the cream cheese
  -  Open the top layer of the drawer and then put the wine bottle on the stove
  Training: similar single-step (2nd step match with wooden cabinet top open): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Push the plate to the front of the stove and then [2nd] put the bowl on the
↪  plate
  Training: similar chain (same 1st, different 2nd) [3 tasks]:
  -  Push the plate to the front of the stove and then put the bowl on the cream cheese
  -  Push the plate to the front of the stove and then put the bowl on the stove
  -  Push the plate to the front of the stove and then put the bowl on the top of the drawer
  Training: similar single-step (2nd step match with plate on stove front): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the bowl on the plate and then [2nd] put the cream cheese on the bowl
  Training: similar chain (same 1st, different 2nd) [4 tasks]:
  -  Put the bowl on the plate and then put the cream cheese in front of the stove
  -  Put the bowl on the plate and then put the cream cheese on the bowl region
  -  Put the bowl on the plate and then put the cream cheese on the stove
  -  Put the bowl on the plate and then put the cream cheese on the top of the drawer
  Training: similar single-step (2nd step match with bowl on plate): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the bowl on the top of the drawer and then [2nd] put the cream cheese on the
↪  plate
  Training: similar chain (same 1st, different 2nd) [4 tasks]:
  -  Put the bowl on the top of the drawer and then put the cream cheese in front of the stove
  -  Put the bowl on the top of the drawer and then put the cream cheese on the bowl
  -  Put the bowl on the top of the drawer and then put the cream cheese on the bowl region
  -  Put the bowl on the top of the drawer and then put the cream cheese on the stove
  Training: similar single-step (2nd step match with bowl on wooden cabinet top side): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the cream cheese on the plate and then [2nd] put the bowl on the cream
↪  cheese
  Training: similar chain (same 1st, different 2nd) [4 tasks]:
  -  Put the cream cheese on the plate and then put the bowl in front of the stove
  -  Put the cream cheese on the plate and then put the bowl on the cream cheese region
  -  Put the cream cheese on the plate and then put the bowl on the stove
  -  Put the cream cheese on the plate and then put the bowl on the top of the drawer
  Training: similar single-step (2nd step match with cream cheese on plate): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the cream cheese on the top of the drawer and then [2nd] put the wine bottle
↪  on the bowl
  Training: similar chain (same 1st, different 2nd) [5 tasks]:
  -  Put the cream cheese on the top of the drawer and then put the wine bottle in front of
  ↪  the stove
  -  Put the cream cheese on the top of the drawer and then put the wine bottle on the cream
  ↪  cheese
```

```
  -  Put the cream cheese on the top of the drawer and then put the wine bottle on the cream
  ↪  cheese region
  -  Put the cream cheese on the top of the drawer and then put the wine bottle on the plate
  -  Put the cream cheese on the top of the drawer and then put the wine bottle on the stove
  Training: similar single-step (2nd step match with cream cheese on wooden cabinet top side):
  ↪  present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the wine bottle in front of the stove and then [2nd] put the bowl on the
↪  stove
  Training: similar chain (same 1st, different 2nd) [3 tasks]:
  -  Put the wine bottle in front of the stove and then put the bowl on the plate
  -  Put the wine bottle in front of the stove and then put the bowl on the wine bottle region
  -  Put the wine bottle in front of the stove and then put the bowl on the top of the drawer
  Training: similar single-step (2nd step match with wine bottle on stove front): present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Put the wine bottle on the rack and then [2nd] put the bowl on the top of the
↪  drawer
  Training: similar chain (same 1st, different 2nd) [5 tasks]:
  -  Put the wine bottle on the rack and then put the bowl in front of the stove
  -  Put the wine bottle on the rack and then put the bowl on the cream cheese
  -  Put the wine bottle on the rack and then put the bowl on the plate
  -  Put the wine bottle on the rack and then put the bowl on the stove
  -  Put the wine bottle on the rack and then put the bowl on the wine bottle region
  Training: similar single-step (2nd step match with wine bottle on wine rack top): not
  ↪  present
  Training: similar single-step (2nd step match with diff background object config): present

Unseen: [1st] Turn on the stove and then [2nd] put the bowl on the stove
  Training: similar chain (same 1st, different 2nd) [4 tasks]:
  -  Turn on the stove and then put the bowl in front of the stove
  -  Turn on the stove and then put the bowl on the cream cheese
  -  Turn on the stove and then put the bowl on the plate
  -  Turn on the stove and then put the bowl on the top of the drawer
  Training: similar single-step (2nd step match with flat stove turned on): present
  Training: similar single-step (2nd step match with diff background object config): present
```

E BASELINE DETAILS

Our `Language` and `Goal-Image` baseline models follow the same diffusion CNN U-Net architecture from Chi et al. (2023) which also matches the architecture of the BPP action decoder. In other words, BPP is a standard diffusion policy with a prompt encoder attached, and the same action decoder serves all three task descriptors, enabling controlled comparisons across them. The difference between BPP and the baselines is the observation encoding; in place of a prompt encoder, `Language` uses a finetuned CLIP (Radford et al., 2021) language encoder and `Goal-Image` shares a finetuned CLIP vision encoder with the corresponding image input.

LIBERO:

- `Language` uses a fine-tuned CLIP encoding which we found performed substantially better than frozen language embeddings. For specific LIBERO splits where it's not possible to use the initial environment state to infer the task (such as LIBERO Goal), we observed that freezing the language encoding would often cause the policy to complete the wrong task.
- `Goal-Image` shares the same fine-tuned vision encoder used by the 3rd person camera. The goal image is the final frame of a demonstration from the 3rd person camera. Using goal images has ambiguity for some tasks within the original LIBERO dataset, so we only provide results for this baseline for LIBERO-Gen Combination and LIBERO-Gen Chain which do not contain tasks for which the goal image is ambiguous about the final state. For LIBERO-Gen Chain, goal images have ambiguity as they do not indicate the ordering in which the two chained operations should occur.
- $\pi_{0.5}$ is finetuned using LoRA (Hu et al., 2022). For training $\pi_{0.5}$ we follow the LIBERO preprocessing steps from (Kim et al., 2024) to regenerate our LIBERO-Gen datasets at a higher resolution, but do not filter out no-op operations. We finetune for 100K steps for LIBERO-Gen experiments (Fig. 4) and finetune for 30K steps for the results on the original LIBERO dataset (Tab. 1).

DrawAnything-Sim:

- `Goal-Image` shares the fine-tuned vision encoder with the main image encoder.
- ICRT extends the provided implementation by Fu et al. (2025) to support action chunking at the token input level; this means that 1) each input action token contains N sequential action steps rather than a single step and 2) we no longer encode observation tokens after every step, only after every N action steps. Additionally, at test time we only provide a single, full demonstration in the context at test-time. Both changes enable a more direct comparison to BPP, which chunks actions together in the prompt and uses a single behavior prompt during deployment.

DrawAnything-Real:

- `Goal-Image` shares the vision encoder with the iPhone main RGB camera in real. The drawing is not fully visible in all frames of the demonstrations due to the moving, wrist-mount camera and gripper occlusion. Thus, to ensure fair comparison, we select the goal image as the first frame searching backwards from the end of demonstration that fully shows the drawing region.

Laundry Folding:

- `Language` uses a finetuned CLIP language encoding.

F PROMPT-QUALITY SENSITIVITY

We perturb the behavior prompt at test time on DrawAnything-Sim (50 unseen tasks, 3 seeds) and report the drawing error of BPP (Tab. 2). Slowing the prompt, jittering its trajectory, and inserting pauses each degrade performance only moderately relative to the clean prompt.

Prompt perturbation	**Error (↓, mean ± std over 3 seeds)**	**Δ vs. clean**
Clean prompt	3.43 ± 0.10	–
$1.25\times$ slowdown	3.64 ± 0.12	$+5.9\%$
Trajectory jitter ($\sigma = 5$ px)	3.69 ± 0.02	$+7.4\%$
Three inserted pauses (10 steps each)	4.26 ± 0.12	$+24.0\%$

Table 2: **Prompt-quality sensitivity on DrawAnything-Sim.** Drawing error on 50 unseen tasks (3 seeds) when the test-time prompt is perturbed.

G BPP ARCHITECTURAL COMPARISON TO ICRT

Model architecture. ICRT (Fu et al., 2025) is a prior behavior prompting architecture that we baseline against. ICRT is a causal transformer decoder, while BPP has separate modules for prompt understanding and action generation (detailed in §3.2). BPP leverages cross-attention to reason over the prompt contents while ICRT leverages causal self-attention. Furthermore, BPP leverages action diffusion (Chi et al., 2023), while ICRT uses L1 action loss.

Training. Each step of training, ICRT samples a sequence of prompts and entire rollouts concatenated along the time dimension into a single sequence to fill the transformer context length. The loss is computed by predicting action loss for each step of the rollout sequence using teacher forcing. In contrast, for each training step of BPP, we sample a single behavior prompt along with a batch of observation-action pairs randomly sampled across many demonstrations. Action loss is computed for each of these randomly sampled observation-action pairs. Both models use prompts and rollouts from a single task per batch. In practice, due to DDP training across multiple GPUs or the use of gradient accumulation, each gradient update will take into account losses across multiple tasks.

Inference. BPP has a fixed length history, common in other visuomotor policies (Chi et al., 2023), whereas ICRT retains the full rollout history in the model context. As a result, ICRT's maximum execution duration is tied to its training-time context length, while BPP can be rolled out indefinitely. §4.1 reports the behavior of both models under a mid-rollout disturbance; the protocol is given at the end of this section.

BPP selects a single prompt per rollout, encodes it once, and caches the chunk embeddings; the prompt encoder then extracts relevant prompt information once per inference call, after which action denoising proceeds without referencing the prompt. Each forward pass of ICRT attends over the prompt and the full rollout history, with KV caching. Per inference call, ICRT is faster (6.5 ms with a single L1 regression pass versus 49 ms for BPP, which uses 16 denoising steps), and ICRT's per-call cost stays flat over a rollout thanks to KV caching at our scale. We therefore make no efficiency claim for BPP; the practical difference is the unbounded rollout duration and the use of action diffusion.

Full-history policies can also overfit to spurious correlations in training rollouts (Torne et al., 2025). We find little evidence of this here: replacing BPP's diffusion head with L1 regression accounts for most of the gap to ICRT on unseen DrawAnything-Sim drawings (error of 4.84 for BPP with an L1 head vs. 5.07 for ICRT, compared to 3.43 for BPP), leaving little to attribute to history.

Disturbance protocol. On the 50 unseen DrawAnything-Sim tasks (3 seeds), we snapshot the canvas at environment step 30 and reset the environment to that snapshot at step 60, erasing 3 s of drawing progress. Only the environment is reset; the prompt and each model's context are untouched, so ICRT's retained history now contains states that no longer match the current canvas progress. Both models execute actions with a horizon of 10 so that the disturbance lands at the same time step (the default model horizon of 12 used for the main experiments does not divide 30, which is why BPP's undisturbed error differs slightly from Fig. 4a). Tab. 3 reports the results; the Δ error bars are standard deviations of the per-seed paired differences.

Model	Undisturbed	Disturbed	Δ error
BPP	3.35 ± 0.09	8.83 ± 0.25	$+5.48 \pm 0.15$
ICRT	5.07 ± 0.22	15.54 ± 0.90	$+10.47 \pm 0.98$

Table 3: **Mid-rollout disturbance on DrawAnything-Sim.** Drawing error on 50 unseen tasks (mean $\pm$ std over 3 seeds) without and with the canvas reverted mid-rollout, and the resulting increase. For disturbed rollouts the error is the lowest reached after the revert.

H LAUNDRY FOLDING DETAILS

This section provides further details for the laundry folding experiment in §4.1.

Tasks. The three tasks are *fold left arm* (requires only the left robot arm), *fold right arm* (requires only the right robot arm), and *fold bottom up* (requires both arms to individually grab the bottom of the sweater and then jointly fold it to the top). All tasks start with the sweater fully unfolded.

Data. Using bimanual iPhUMI, we collect $\sim$150 full demonstrations per task and $\sim$1000 total error correction demonstrations. We collect five new behavior prompts per task for evaluation, so the evaluations in Fig. 6 use *unseen* prompts for *seen* tasks. The `Language` baseline is detailed in §E.

Failure modes. The *fold bottom up* task first grabs the bottom with the right arm and then with the left arm; we often observe failures where BPP correctly grasps the bottom with the right arm, but the left arm instead decides to fold the left sleeve instead of also grasping the bottom. Given the low task diversity, the indirectness of the prompt as a task descriptor might make BPP sensitive to overfitting to spurious task cues in the training data (e.g., background variations more often seen for a particular task).

Outlook. As expected, behavior prompting performs poorly in this low task diversity regime. In a high task diversity setting, however, we envision behavior prompting having substantial advantages with training data spanning *many* folding styles across *many* garments. In particular, a behavior prompt captures the *temporal* steps in the folding process as well as the *spatial* interaction points, both of which clarify and inform the desired execution in a high task diversity setting. This information could also help with visual adaptation to unseen garments and temporal adaptation to unseen folding orders. Practically, a user might also find it more natural to describe their desired folding preferences via a single demonstration, rather than through language.

I BPP TRAINING DETAILS

Task groupings. During training, the prompt and the target observation-action pairs are sampled from the same task grouping. These groupings define the granularity at which behavior prompts can influence robot execution: for example, a pick-place task with two distinct grasping strategies for one object would require two separate task groupings for the robot to adhere to the grasping strategy shown in the prompt.

Benchmark	Duration (hrs)	GPUs
LIBERO-Gen Combination	28	4$\times$ NVIDIA RTX A6000
LIBERO-Gen Chain	34	4$\times$ NVIDIA L40S
DrawAnything-Sim	27	4$\times$ NVIDIA L40S
DrawAnything-Real	42	4$\times$ NVIDIA L40S
Laundry Folding	24	4$\times$ NVIDIA L40S

Table 4: **Compute requirements for training BPP for each training benchmark.** Simulation experiments include time for intermediary and final rollout evaluations. We train using DDP.

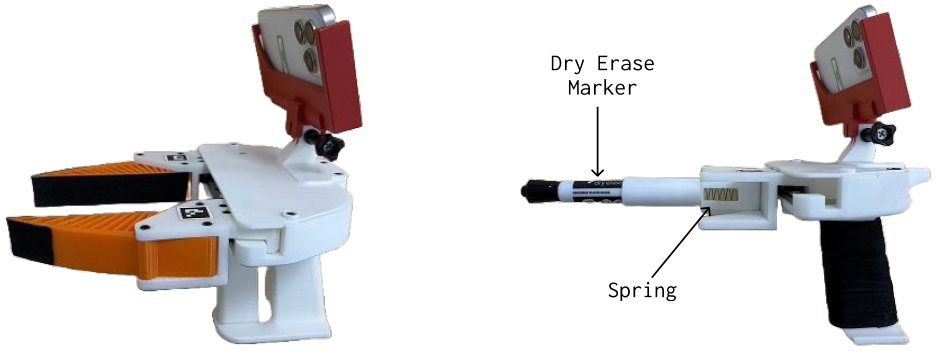
Layers	Hidden size D	MLP size	Heads	Params
6	768	3072	8	57M

Table 5: **BPP prompt encoder architecture (transformer decoder).**

Structure	Step emb dim	Down dims	Kernel	Params
U-Net	128	[256, 512, 1024]	5	151M

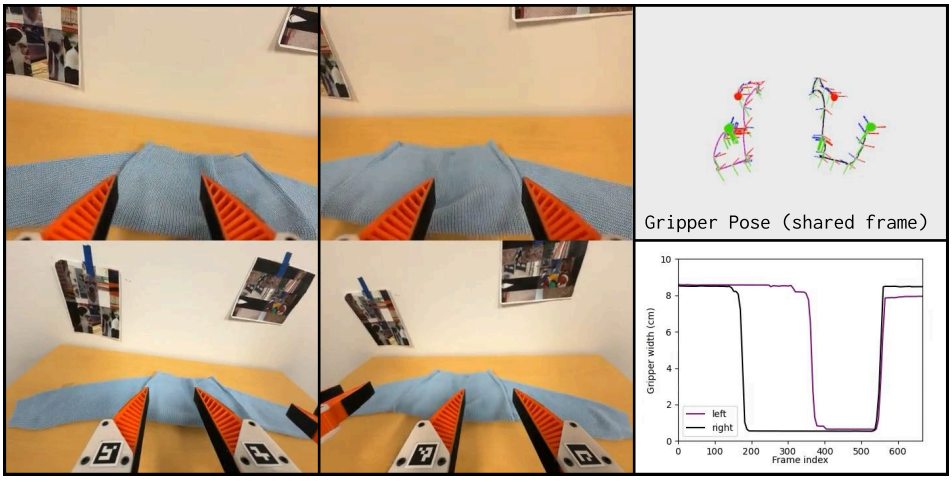
Table 6: **Action decoder model architecture details.** We use the CNN U-Net action diffusion architecture from Chi et al. (2023). BPP, `Language`, and `Goal-Image` models use this for action diffusion.

J iPhUMI

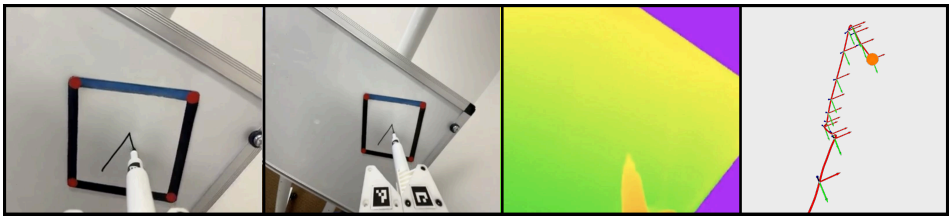


a) iPhUMI b) iPhUMI for drawing

Figure 15: **The iPhUMI handheld data collection gripper.** iPhUMI enables real-time localization in new environments, which dramatically reduces the setup time required for collecting demonstration data compared to the original UMI (Chi et al., 2024). It features a custom-built application that facilitates data collection and policy deployment. With iPhUMI, a user can also specify behavior prompts at test-time to immediately condition behavior prompting policies. We present two different instantiations of iPhUMI: a) with fingers for standard manipulation tasks and b) with a marker attachment and spring for compliant drawing.



a) Bimanual iPhUMI used for laundry folding task



b) iPhUMI with drawing attachment for DrawAnything-Real

Figure 16: **iPhUMI collected modalities.** Data collected with bimanual iPhUMI for a) laundry folding and b) drawing visualized part-way through a user drawing the letter A. For DrawAnything-Real we do not use the ultrawide or depth camera as policy inputs, but include them for reference.

iPhUMI is a handheld data collection interface designed to enable rapid data collection across new tasks and environments. We modify the original UMI gripper design (Chi et al., 2024) by replacing the GoPro with a custom 3D-printed mount to hold an iPhone (Fig. 15a). For drawing tasks, the UMI gripper is further modified to attach a marker with a spring to allow compliance for drawing without the need for a force-torque sensor (Fig. 15b). We open source a custom iOS application for iPhUMI that provides an intuitive application interface for recording demonstration data and for using the iPhone as a camera on the robot during policy deployment.

J.1 iPhUMI DATA COLLECTION

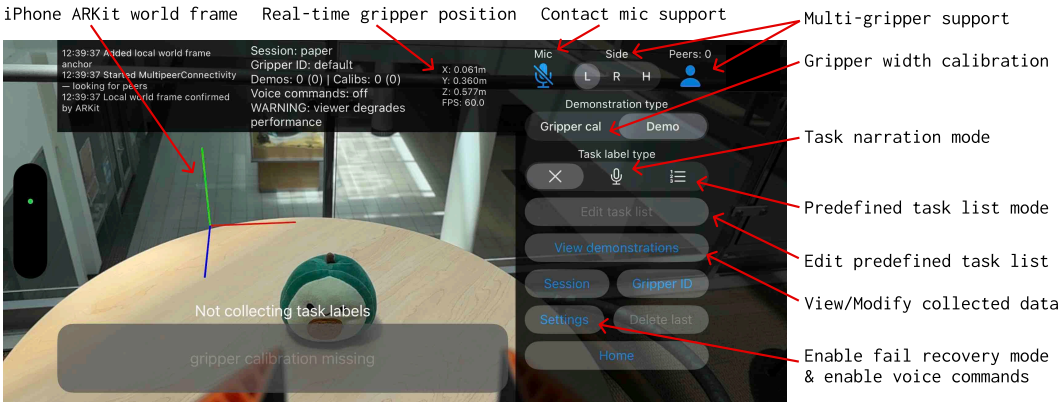


Figure 17: **iPhUMI data collection interface.** We provide an interface to perform gripper calibration, collect demonstrations, and set appropriate settings for data collection.

iPhUMI is capable of collecting five types of data from the iPhone simultaneously: main camera (1920x1440 at 60Hz), ultrawide camera (640x480 at 10 Hz), LiDAR depth (256x192 at 60Hz), gripper pose (60Hz), and gripper width (10Hz) (Fig. 16). Note that depth data is not used in any models or experiments presented in this paper. Unlike previous iPhone-based grippers that either rely on an external fisheye camera (Fang et al., 2026) or do not capture the ultrawide camera (Etukuru et al., 2025), our iPhUMI application is capable of capturing the ultrawide camera input. This provides increased visual context to the policy, which has been found to improve performance (Chi et al., 2024).

The primary data collection interface is shown in Fig. 17. Upon launching the app, ARKit initializes a world coordinate frame and then continually estimates the iPhone pose using SLAM. The resulting 6-DoF pose of the iPhone in this world frame is logged in real time throughout each demonstration. The app also supports simultaneous recording from multiple grippers. Up to three devices (bimanual iPhUMI and head-mounted iPhone) can be connected into a shared ARKit session to share a single world coordinate frame among them. The triple iPhone setup was co-developed with Xu et al. (2026b). The iPhones will connect into a shared session once common world features have been detected between them.

Before collecting demonstrations, the user performs a one-time gripper width calibration by recording a $\sim$10 s clip while opening and closing the gripper multiple times in calibration mode. The iPhUMI uses the ultrawide camera to detect ArUco tags on the fingers to determine the minimum and maximum gripper width for the specific hardware.

When collecting data, the user can specify the task name(s) associated with the demonstration and switch labels online as the task changes. Alternatively, the user may narrate the task while demonstrating the task, and iPhUMI uses speech-to-text to label segments with the active task. We also have support for connecting the contact microphone used by (Liu et al., 2025) through the USB-C connector on the iPhone to capture synchronized audio data.

Recorded demonstrations can be reviewed using the interface shown in Fig 18. Selecting a demonstration displays the main, ultrawide, and depth videos, while a left swipe allows the user to delete

a demonstration. Tapping the `Export` button saves the demonstration data to an external location such as an SD card connected via a USB C adapter.

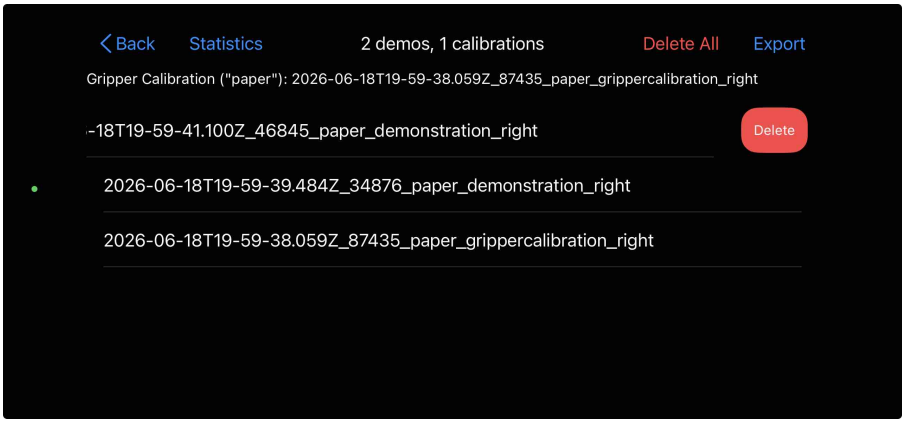


Figure 18: **iPhUMI Demonstration management interface.** This interface lets the user view and manage collected demonstration data. The data can also be exported to an external SD card connected with a USB C adapter.

J.2 iPhUMI POLICY DEPLOYMENT

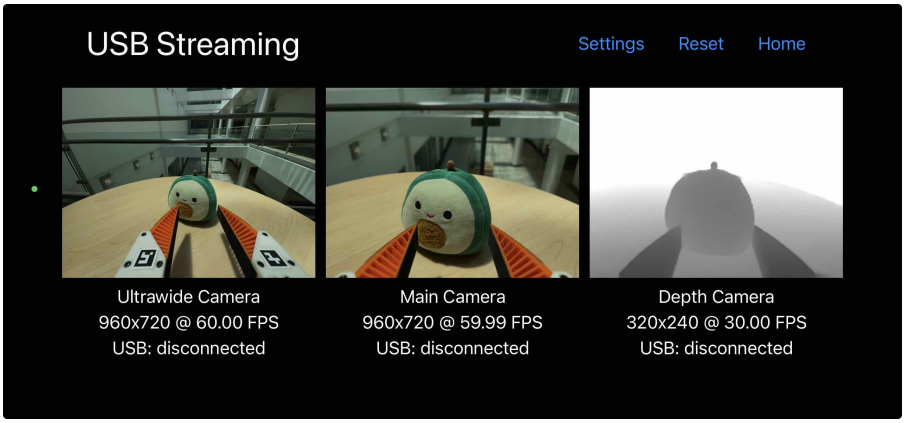


Figure 19: **iPhUMI deployment interface**. Both USB and Ethernet streaming are supported to stream main camera, ultrawide camera, and LiDAR depth for use during robot deployment.

The deployment interface (Fig. 19) can be used to deploy a policy on the robot using an iPhone as the camera. The iPhone streams its camera feeds and depth through a USB or Ethernet connection. The main camera and ultrawide camera feeds are each streamed 960x720 at 60Hz. Depth is streamed at 320x240 at 30Hz. Our robot deployment infrastructure is built off the code from (Gao et al., 2026).

J.3 iPhUMI FOR TEST-TIME BEHAVIOR PROMPTING

The iPhUMI app supports wireless transfer of behavior prompts from the iPhone to a deployment desktop connected to a robot. This enables rapid behavior prompt collection and conditioning of robot execution with BPP. When in behavior prompting mode, a user collects a single demonstration using iPhUMI that is then wirelessly transmitted to a desktop connected to a robot. The desktop processes the raw iPhone data into a behavior prompt that is used to condition BPP. This enables a rapid process for practically leveraging behavior prompting at test-time to specify new tasks.