BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning

Eric Jang[1*], **Alex Irpan**[1*], **Mohi Khansari**[2], **Daniel Kappler**[2], **Frederik Ebert**[3†], **Corey Lynch**[1], **Sergey Levine**[1,3], **Chelsea Finn**[1,4]

[1]Robotics at Google [2]Everyday Robots [3]UC Berkeley [4]Stanford University

https://sites.google.com/view/bc-z/home

Abstract: In this paper, we study the problem of enabling a vision-based robotic manipulation system to generalize to novel tasks, a long-standing challenge in robot learning. We approach the challenge from an imitation learning perspective, aiming to study how scaling and broadening the data collected can facilitate such generalization. To that end, we develop an interactive and flexible imitation learning system that can learn from both demonstrations and interventions and can be conditioned on different forms of information that convey the task, including pre-trained embeddings of natural language or videos of humans performing the task. When scaling data collection on a real robot to more than 100 distinct tasks, we find that this system can perform 24 *unseen* manipulation tasks with an average success rate of 44%, without any robot demonstrations for those tasks.

Keywords: Zero-Shot Imitation Learning, Multi-Task Imitation, Deep Learning

1 Introduction

One of the grand challenges in robotics is to create a general-purpose robot capable of performing a multitude of tasks in unstructured environments based on arbitrary user commands. The key challenge in this endeavour is *generalization*: the robot must handle new environments, recognize and manipulate objects it has not seen before, and understand the intent of a command it has never been asked to execute. End-to-end learning from pixels is a flexible choice for modeling the behavior of such generalist robots, as it has minimal assumptions about the state representation of the world. With sufficient real-world data, these methods should in principle enable robots to generalize across new tasks, objects, and scenes without requiring hand-coded, task-specific representations. However, realizing this goal has generally remained elusive. In this paper, we study the problem of enabling a robot to generalize zero-shot or few-shot to new vision-based manipulation tasks.

We study this problem using the framework of imitation learning. Prior works on imitation learning have shown one-shot or zero-shot generalization to new objects [1, 2, 3, 4, 5] and to new object goal configurations [6, 7]. However, zero-shot generalization to new tasks remains a challenge, particularly when considering vision-based manipulation tasks that cover a breadth of skills (e.g., wiping, pushing, pick-and-place) with diverse objects. Achieving such generalization depends on solving challenges relating to scaling up data collection and learning algorithms for diverse data.

We develop an interactive imitation learning system with two key properties that enable high-quality data collection and generalization to entirely new tasks. First, our system incorporates shared autonomy into teleoperation to allow us to collect both raw demonstration data and human interventions to correct the robot's current policy. Second, our system flexibly conditions the policy on different forms of task specification, including a language instruction or a video of a person performing the task. Unlike discrete one-hot task identifiers [8], these continuous forms of task specification can in principle enable the robot to generalize zero-shot or few-shot to new tasks by providing a language or video command of the new task at test time. These properties have been explored previously; our aim is to empirically study whether these ideas scale to a broad range of real-world tasks.

[*]Equal Contribution

[†]Work done while author was at Google

5th Conference on Robot Learning (CoRL 2021), London, UK.

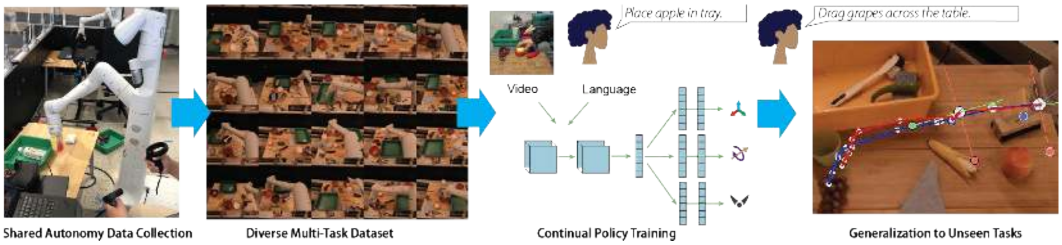

Place apple in tray.
Drag grapes across the table.
Video
Language
Shared Autonomy Data Collection
Diverse Multi-Task Dataset
Continual Policy Training
Generalization to Unseen Tasks

Figure 1: Overview of BC-Z. We collect a large-scale dataset (25,877 episodes) of 100 diverse manipulation tasks, and train a 7-DoF multi-task policy that conditions on task language strings or human video. We show this system produces a policy that is capable of generalizing zero-shot to new unseen tasks.

Our main contribution is an empirical study of a large-scale interactive imitation learning system that solves a breadth of tasks, including zero-shot and few-shot generalization to tasks *not seen* during training. Using this system, we collect a large dataset of 100 robotic manipulation tasks, through a combination of expert teleoperation and a shared autonomy process where the human operator "coaches" the learned policy by fixing its mistakes. Across 12 robots, 7 different operators collected 25,877 robot demonstrations that totaled 125 hours of robot time, as well as 18,726 human videos of the same tasks. At test time, the system is capable of performing 24 unseen manipulation tasks between objects that have never previously appeared together in the same scene. These closed-loop visuomotor policies perform asynchronous inference and control at 10Hz, amounting to well over 100 decisions per episode. We open-source the demonstrations used to train this policy at `https://www.kaggle.com/google/bc-z-robot`.

2 Related Work

Imitation learning has been successful in learning grasping and pick-place tasks from low-dimensional state [9, 10, 11, 12, 13, 14, 15]. Deep learning has enabled imitation learning directly from raw image observations [8, 16, 17]. In this work, we focus on enabling zero-shot and few-shot generalization to new tasks in an imitation learning framework.

Multiple prior imitation learning works have achieved different forms of generalization, including one-shot generalization to novel objects [1, 2, 3, 4, 18], to novel object configurations [19], and to novel goal configurations [6, 7, 20], as well as zero-shot generalization to new objects [5], scenes [21], and goal configurations [22]. Many of these works adapt to the new scenario by conditioning on a robot demonstration [1, 2], a video of a human [3, 4], a language instruction [23, 24], or a goal image [21]. Our system flexibly conditions on either a video of a human or a language instruction, and we focus on achieving zero-shot (language) and few-shot (video) generalization to *entirely new* 7-DoF manipulation tasks on a real robot, including scenarios without goal images and where task-relevant objects are never encountered together in the training data.

It is standard to collect demonstrations via teleoperation [25] or kinesthetic teaching [10], and active learning methods such as DAgger [26] help reduce distribution shift for the learner. Unfortunately, DAgger and some of its variants [27, 28] are notoriously difficult to apply to robotic manipulation because they necessitate an interface where the expert must annotate the correct action when not in control of the robot policy. Inspired by recent works in autonomous driving, HG-DAgger [29] and EIL [30], our system instead only requires the expert to intervene when they believe the policy is likely to make an error and allows the expert to temporarily take full control to put the policy back on track. The resulting data collection scheme is easy to use and helps address distribution shift. Furthermore, the rate of expert interventions during data collection can be used as a live evaluation metric, which we empirically find correlates with policy success.

Beyond imitation learning, generalization has been studied in a number of other robot learning works. This includes works that generalize skills to novel objects [31, 32, 33, 34, 35], to novel environments [36], from simulation to reality [37, 38, 39, 40, 41], and to new manipulation skills and objects [42, 43, 44, 45]. We focus on the last case of generalizing to novel tasks, but unlike these prior works, we tackle a large suite of 100 challenging tasks that involve 7 DoF control at 10 Hz and involve more than 100 decisions within an episode to solve the task.

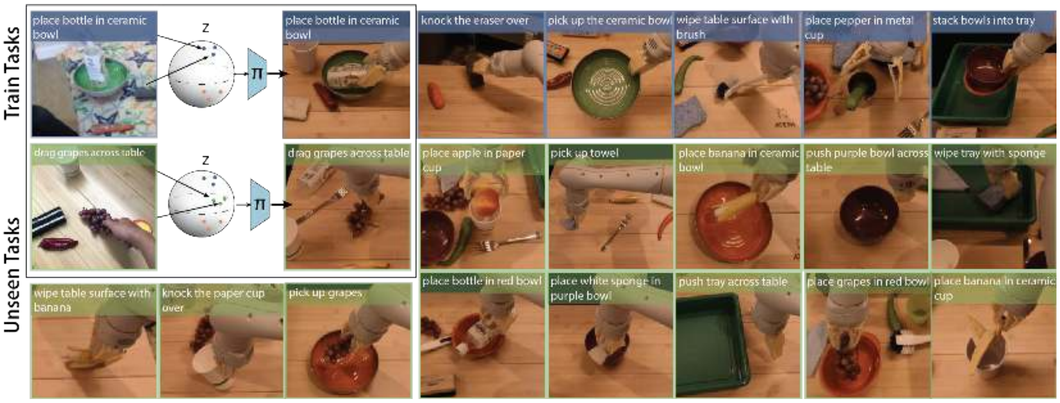


Figure 2: A subset of training tasks (top row), and a subset of held-out tasks (bottom two rows) used for evaluating zero shot task generalization. Top left: Given a pretrained task embedding computed from human videos or text, BC-Z acts as an "action decoder" for the task embedding.

3 Problem Setup and Method Overview

An overview of our imitation learning system is shown in Figure 1. Our goal is to train a conditional policy that can interpret RGB images, denoted $s \in \mathcal{S}$, together with a task command $w \in \mathcal{W}$, which might correspond to a language string or a video of a person. Different tasks correspond to completing distinct objectives; some example tasks and corresponding commands are shown in Figure 2. The policy is a mapping from images and commands to actions, and can be written as $\mu : \mathcal{S} \times \mathcal{W} \rightarrow \mathcal{A}$, where the action space $\mathcal{A}$ consists of the 6-DoF pose of the end effector as well as a 7th degree of freedom for continuous control of the parallel jaw gripper.

The policy is trained using a large-scale dataset collected via a VR-based teleoperation rig (see Figure 1, left) through a combination of direct demonstration and human-in-the-loop shared autonomy. In the latter, trained policies are deployed on the robot, and the human operator intervenes to provide corrections when the robot makes a mistake. This procedure resembles the human-gated DAgger (HG-DAgger) algorithm [26, 29], and provides iterative improvement for the learned policy, as well as a continuous signal that can be used to track the policy's performance.

The policy architecture is divided into an encoder $q(z|w)$, which processes the command w into an embedding $z \in \mathcal{Z}$, and a control layer π, which processes (s, z) to produce the action a, i.e. $\pi : \mathcal{S} \times \mathcal{Z} \rightarrow \mathcal{A}$. This decomposition is illustrated in Figure 2, with further details in Section 5. It provides our method with the ability to incorporate auxiliary supervision, such as pretrained language embeddings, which help to structure the latent task space and facilitate generalization. In our experiments, we will show that this enables generalization to tasks that were not seen during training, including novel compositions of verbs and objects.

4 Data Collection and Workflow

In order for an imitation learning system to generalize to new tasks with zero demonstrations of said task, we must be able to easily collect a diverse dataset, provide corrective feedback, and evaluate many tasks at scale. In this section, we discuss these components of our system.

System Setup. Our teleoperation system uses an Oculus VR headset which is attached to the robot's onboard computer via USB cable and tracks two handheld controllers. The teleoperator stands behind the robot and uses the controllers to operate the robot with a line-of-sight 3rd-person view. The robot responds to the operator's movement in a 10 Hz non-realtime control loop. The relatively fast closed-loop control allows the operator to demonstrate a wide range of tasks with ease and quickly intervene if the robot is about to enter an unsafe state during autonomous execution. Further details on the user interface and data collection are in Appendices A and B.

Environment and Tasks. We place each robot in front of a table with anywhere from 6 to 15 household objects with randomized poses. We collect demonstrations and videos of humans for 100 pre-specified tasks (listed in Tables 7 and 8), which span 9 underlying skills such as pushing and pick-and-place. The model is then evaluated on 29 *new* tasks using a new language description or video of that task. For the method to perform well on these held-out tasks, it must both correctly interpret the new task command and output actions that are consistent with that task.

Shared Autonomy Data Collection. Data collection begins with an initial expert-only phase, where the human provides the demonstration of the task from start-to-finish. After an initial multi-task policy is learned from expert-only data, we continue collecting demonstrations in “shared autonomy” mode, where the current policy attempts the task while the human supervises. At any point the human may take over by gripping an “override” switch, which allows them to briefly take full control of the robot and perform necessary corrections when the policy is about to enter an unsafe state, or if they believe the current policy will not successfully complete the task. This setup enables HG-DAgger [29], where intervention data is then aggregated with the existing data and used to re-train the policy. For the multi-task manipulation tasks, we collect 11,108 expert-only demonstrations for the initial policies, then collected an additional 14,769 HG-DAgger demonstrations covering 16 iterations of policy deployment, where each iteration deploys the most recent policy trained on the aggregated dataset. This gives a total of 25,877 robot demos. We find in Table 4 that when controlling for the same number of total episodes, HG-DAgger improves performance substantially.

Shared Autonomy Evaluation. When success rates are low, resources are best spent on collecting more data to improve the policy; but evaluation is also important to debug problems in the workflow. As the expected degree of generalization increases, we need more trials to evaluate the extent of policy generalization. This creates a resource trade-off: how should robot time be allocated between measuring policy success rates and collecting additional demonstrations to improve the policy? Fortunately, shared autonomy data collection confers an additional benefit: the *intervention rate*, measured as the average number of interventions required per episode, can be used as an indication for policy performance. In Figure 5, we find that the intervention rate correlates negatively with overall policy success rate.

5 Learning Algorithm

The data collection procedure above results in a large multi-task dataset. For each task i, this dataset contains expert data $(s, a) \in \mathcal{D}_e^i$, human video data $w_h \in \mathcal{D}_h^i$, and one language command w_ℓ^i. We now discuss how we use this data to train the encoder $q(z|w)$ and the control layer $\pi(a|s, z)$.

5.1 Language and Video Encoders

Our encoder $q(z|w)$ takes either a language command w_ℓ^i or a video of a human w_h as input and produces a task embedding z. If the command is a language command, we use a pretrained multilingual sentence encoder [46][1] as our encoder, producing a 512-dim language vector for each task. Despite the simplicity, we find that these encoders work well in our experiments.

When task commands are instead a video of a human performing the task, we use a convolutional neural network to produce z, specifically a ResNet-18 based model. Inspired by recent works [2, 3], we train this network in an end-to-end manner. We collected a dataset of 18,726 videos of humans doing each training task, in a variety of home and office locations, camera viewpoints, and object configurations. Using paired examples of a human video w_h^i and corresponding demonstration demo $\{(s, a)\}^i$, we encode the human video $z^i \sim q(\cdot \mid w_h^i)$, then pass the embedding to the control layer $\pi(a|s, z^i)$, and then backpropagate gradient of the behavior cloning loss to both the policy and encoder parameters.

Visualizations of learned embeddings in Appendix E indicate that by itself, this end-to-end approach tends to overfit to initial object scenes, learn poor embeddings, and show poor task generalization. To help align the video embeddings more semantically, we therefore further introduce an auxiliary *language regression* loss. Concretely, this auxiliary loss trains the video encoder to predict the embedding of the task’s language command with a cosine loss. The resulting video encoder objective is as follows:

$$\min \sum_{\text{task } i} \sum_{\substack{(s,a) \sim \mathcal{D}_e^i \\ w_h \sim \mathcal{D}_h^i \bigcup \mathcal{D}_e^i}} \underbrace{- \log \pi(a|s, z^i)}_{\text{behavior cloning}} + \underbrace{D_{\text{cos}}(z_h^i, z_\ell^i)}_{\text{language regression}} \text{, where } \underbrace{z_h^i \sim q(\cdot|w_h)}_{\text{video encoder}}, \underbrace{z_\ell^i \sim q(\cdot|w_\ell^i)}_{\text{language encoder}} \quad (1)$$

where D_{cos} denotes the cosine distance. Since robot demos double as videos of the task, we also train encoded robot videos to match to the language vector. This language loss is critical to learning a more organized embedding space. Additional architecture and training details are in Appendix E.

[1] Checkpoint from https://tfhub.dev/google/universal-sentence-encoder-multilingual/3

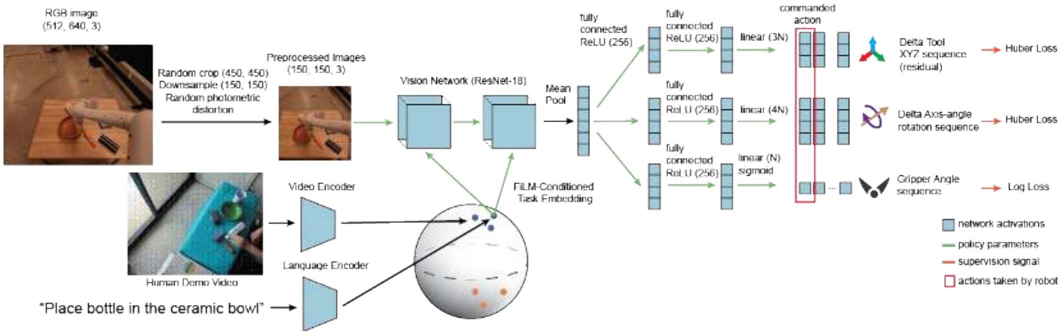


Figure 3: BC-Z network architecture. A monocular RGB image from the head-mounted camera is passed through a ResNet18 encoder, then through a two-layer MLP to predict each action modality (delta XYZ, delta axis-angle, and gripper angle). FiLM layers [47] condition the architecture on a task embedding z computed from language w_ℓ or video w_h.

5.2 Policy Training

Given a fixed task embedding, we train $\pi(a|s, z)$ via Huber loss on XYZ and axis-angle predictions, and log loss for the gripper angle. During training, images are randomly cropped, downsampled, and subjected to standard photometric augmentations. Below we describe two additional design choices that we found to be helpful. Additional training details such as learning rates, batch sizes, pseudocode, and further hyperparameters are discussed in Appendix D.

Open-Loop Auxiliary Predictions. The policy predicts the action the robot would take, as well as an open-loop trajectory of the next 10 actions the policy would take if it were operating in an open-loop manner. At inference time, the policy operates closed-loop, only executing the first action based on the current image. The open-loop prediction confers an auxiliary training objective, and provides a way to visually inspect the quality of a closed-loop plan in an offline manner (see Figure 1, right).

State Differences as Actions. In standard imitation learning implementations, actions taken at demonstration-time are used directly as target labels to be predicted from states. However, cloning expert actions at 10Hz resulted in the policy learning very small actions, as well as dithering behavior. To address this, we define actions as state differences to target poses $N > 1$ steps in the future, using an adaptive algorithm to choose N based on how much the arm and gripper move. We provide ablation studies for this design choice in Section 6.3 and further details in Appendix C

5.3 Network Architecture

We model the policy using a deep neural network, shown in Figure 3. The policy network processes the camera image with a ResNet18 "torso" [48], which branches from the last mean-pool layer into multiple "action heads". Each head is a multilayer perceptron with two hidden layers of size 256 each and ReLU activations, and models part of the end-effector action, specifically the delta XYZ, delta axis-angle, and normalized gripper angle. The policy is conditioned on a 512-dim task embedding z, through FiLM layers [47]. Following Perez et al. [47], the task conditioning is linearly projected to channel-wise scales and shifts for each channel of each of the 4 ResNet blocks.

6 Experimental Results

Our experiments aim to evaluate BC-Z in large-scale imitation learning settings. We start with an initial validation of BC-Z on single-task visual imitation learning. Then, our experiments will aim to answer the following questions: (1) Can BC-Z enable zero-shot and few-shot generalization to new tasks from a command in the form of language or a video of a human? (2) Is the performance of BC-Z bottlenecked by the task embedding or by the policy? (3) How important are different components of BC-Z, including HG-DAgger data collection and adaptive state diffs? We present experiments aimed at these questions in this section.

6.1 BC-Z on Single-Task Imitation Learning

We first aim to verify that BC-Z can learn individual vision-based tasks before considering the more challenging multi-task setting. We choose two tasks: a *bin-emptying* task where the robot must grasp objects from a bin and drop them into an adjacent bin, and a *door opening* task where the robot must

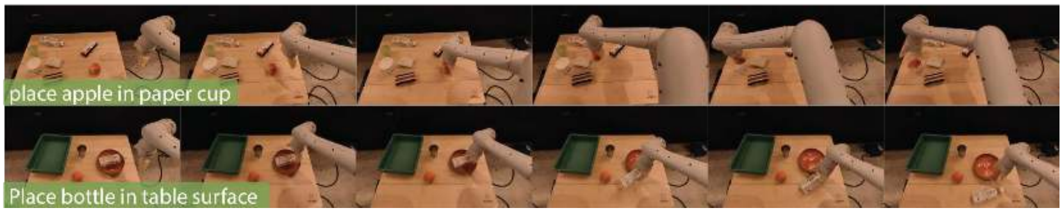


Figure 4: Qualitative examples of BC-Z successfully performing held-out tasks.

push open a door while avoiding collisions. Both tasks use the architecture in Figure 3, except that the door opening task involves predicting the forward and yaw velocity of the base instead of controlling the arm. The bin-emptying dataset has 2,759 demonstrations, while the door opening dataset has 12,000 demonstrations collected across 24 meeting rooms and 36,000 demonstrations across 5 meeting rooms in simulation. Further task and dataset details are in Appendix I.

Table 1: Single-task bin and door performance, average and standard deviation across runs.

Bin-Emptying	**Picks / Minute**	**# Runs**
Human Expert	6.3 (2.1)	2759
BC-Z (2759 demos)	3.4 (1.2)	9
Door Opening	**Success Rate**	**# Runs**
BC-Z (24 Train Doors)	87% (2.2)	480
BC-Z (4 Holdout Doors)	94% (2.7)	80

In Table 1, we see that the BC-Z model is able to reach a pick-rate of 3.4 picks per minute, over half the speed of a human teleoperator. Further, we see that BC-Z reaches a success rate of 87% on the training door scenes and 94% on held-out door scenes. These results validate that the BC-Z model and data collect system can achieve good performance on both training and held-out scenes in the single-task setting. Additional analysis is provided in Appendix H.

6.2 Evaluating Zero-Shot and Few-Shot Task Generalization

Next, we aim to test whether BC-Z can achieve generalization to new tasks. Demonstrations are collected across 100 different manipulation tasks, comprising two disjoint sets of objects. Using disjoint sets of objects allows us to specifically test generalization to combinations of object-object pairs and object-action pairs that are not seen together during training. For the first set of objects, demonstrations are collected across 21 different tasks, listed in Table 7, which cover a wide range of skills, from pick-and-place tasks to skills that require positioning the object in a certain way, like "stand the bottle upright". For the second set of objects, demonstrations are collected for 79 different tasks, including pick-and-place, surface wiping, and object stacking. The latter family has a smaller variety of manipulation behaviors, but is defined over a larger object set with more clutter. Object sets are shown in Appendix B and a full list of train task sentences are in Appendix J.

We evaluate BC-Z on 29 held-out tasks. Language conditioned policies are given a novel sentence, while video conditioned policies are given the average embedding of a few human videos of the new task. Four held-out tasks use objects in the 79-task family, whereas 25 tasks are generated by mixing objects between the 21-task family and 79-task family. Thus, the first 4 held-out tasks do not require cross-object set generalization, so they are easier to generalize to. Even so, we find that each of these 4 tasks are sufficiently challenging that training single-task policies on 300+ held-out demos with DAgger interventions completely fails, achieving 0% task success. This provides a degree of calibration on the difficulty of these tasks. We hypothesize that a major contributing factor to this challenge is the wide range of locations, objects, and distractors that the skills must generalize to in our settings, as well as the wide range of these factors in the training data.

In Table 2, we see that language-conditioned BC-Z is able to generalize zero-shot to both kinds of held-out tasks, averaging at 32% success and showing non-zero success on 24 held-out tasks. Among the 24 hold-out tasks with non-zero success rates, BC-Z achieves an average success of 44% when conditioned on language embeddings it has never seen. When conditioning on videos of humans, we find that generalization is much more difficult, but that BC-Z is still able to generalize to nine novel tasks with a non-zero success rate, particularly when the task does not involve novel object combinations. Qualitatively, we observe that the language-conditioned policy usually moves towards the correct objects, clearly indicating that the task embedding is reflective of the correct task, as we further illustrate in the supplementary video. The most common source of failures are

Table 2: Success rates for zero-shot (language) and few-shot (video) generalization to tasks not in the training dataset. The first 4 tasks only use objects from the 79-task family. The remaining tasks mix objects between the 21-task and 79-task families, requiring further generalization. Numbers in parentheses are 1 unit standard deviation. The language conditioning generalizes to several holdout tasks, whereas the video conditioning shows promise on tasks that do not mix objects between task families. Overall performance improves slightly with fewer distractor objects.

Skill	**Held-out tasks (no demos during training)**	**Lang-conditioned (1 distractor)**	**Lang-conditioned (4-5 distractors)**	**Video-conditioned (4-5 distractors)**
pick-place	'place sponge in tray'	83% (6.8)	82% (9.2)	22% (2.2)
	'place grapes in red bowl'	87% (6.2)	75% (10.8)	12% (7.8)
	'place apple in paper cup'	30% (8.4)	33% (12.2)	14% (7.8)
pick-wipe	'wipe tray with sponge'	40% (8.9)	0% (0)	28% (10.6)
pick-place	'place banana in ceramic bowl'	50% (15.8)	75% (9.7)	7.5% (4.2)
	'place bottle in red bowl'	50% (15.8)	75% (9.7)	0% (0)
	'place grapes in ceramic bowl'	70% (14.5)	70% (10.3)	0% (0)
	'place bottle in table surface'	0	50% (11.2)	5% (3.5)
	'place white sponge in purple bowl'	70% (14.9)	45% (11.2)	0% (0)
	'place white sponge in tray'	50% (15.8)	40% (11.0)	0% (0)
	'place apple in ceramic bowl'	30% (14.5)	20% (8.9)	0% (0)
	'place bottle in purple bowl'	30% (14.5)	20% (8.9)	0% (0)
	'place banana in ceramic cup'	10% (9.5)	0% (0)	0% (0)
	'place banana on white sponge'	40% (15.5)	0% (0)	0% (0)
	'place metal cup in red bowl'	0% (0)	0% (0)	0% (0)
grasp	'pick up grapes'	70% (14.5)	65% (10.7)	0% (0)
	'pick up apple'	20% (12.7)	55% (11.2)	5% (3.5)
	'pick up towel'	50% (15.8)	42.8% (18.7)	0% (0)
	'pick up pepper'	50% (15.8)	35% (10.7)	12.5% (5.2)
	'pick up bottle'	40% (15.5)	30% (10.3)	17.5% (6.0)
	'pick up the red bowl'	30% (14.5)	0% (0)	0% (0)
pick-drag	'drag grapes across the table'	0% (0)	14% (13.2)	0% (0)
pick-wipe	'wipe table surface with banana'	0% (0)	10% (6.7)	0% (0)
	'wipe tray with white sponge'	20% (12.7)	0% (0)	0% (0)
	'wipe ceramic bowl with brush'	10% (9.49)	0% (0)	0% (0)
push	'push purple bowl across the table'	50% (15.8)	30% (10.3)	0% (0)
	'push tray across the table'	30% (14.5)	25% (9.7)	0% (0)
	'push red bowl across the table'	60% (15.5)	0% (0)	0% (0)
	Holdout Task Overall	38%	32%	4%

"last-centimeter" errors: failing to close the gripper, failing to let go of objects, or a near miss of the target object when letting go of an object in the gripper.

Is Performance Bottlenecked on the Encoder or the Policy? Now that we see that BC-Z can generalize to a substantial number of held-out tasks to some degree, we ask whether the performance is limited more by the generalization of the encoder $q(z|w)$, the control layer $\pi(a|s, z)$, or both. To disentangle these factors, we measure the policy success rate on the training tasks conditioned in three ways: a one-hot task identifier, language embeddings of the training task commands, and video embeddings of *held-out* human videos of the training tasks. This comparison is in Table 3. The similar performance between one-hot and language suggests the latent language space is sufficient, and that language-conditioned performance on held-out tasks is bottlenecked on the control layer more than the embedding. The more significant drop in performance of video-conditioned policies suggests inferring tasks from videos is much more difficult, particularly for held-out tasks.

Table 3: Training vs. generalization performance, averaged across 21 of the training tasks and all 28 held-out tasks.

Setting	**Task Conditioning**	**Success**
Train	One-hot	42%
	Language	40%
	Video	24%
Held-Out	Language	32%
	Video	4%

6.3 Ablation Studies and Comparisons

We validate the importance of several BC-Z design decisions using the (training) 21-task family. Our first set of ablations evaluate on the "place the bottle in ceramic bowl" command, which has the most demos (1000) of any task. We first test whether multi-task training is helpful for performance: we compare the multi-task system trained on 25,877 demos across all tasks, to a single-task policy trained on just the 1000 demos for the target task. In Table 4 (left), the single-task baseline achieves

Table 4: Ablation Studies. Left: Multi-task vs. single task models on the 'place the bottle in the ceramic bowl' task. Training across tasks and with adaptive state-diffs is important for good training performance. Right: DAgger comparison on 'place the bottle in the ceramic bowl' (1-Task) and the 8-Task subset from Table 7. Controlled for the same amount of data, DAgger reaches higher success numbers significantly more quickly.

Method	**1-Task**
Multi-task, language conditioned	52% (6.3)
Multi-task, one-hot conditioned	45% (5.3)
Single-task baseline (1000 demos)	5% (2.8)
Multi-task, one-hot, no adaptive state-diff	3% (2.3)

Dataset	**1-Task**	**8-Task**
100% Manual	27% (5.2)	23% (4.2)
50% Manual + 50% HG-DAgger	53% (5.8)	47% (5.2)

just 5% success. The low number is consistent with the low single-task performance on holdout tasks from Section 6.2: collecting data over several robots and operators likely makes the task harder to learn. Only when pooling data across many tasks does BC-Z learn to solve the task. We ablate the adaptive state diff scheme described in Section 5.3 and find that it is important; when naively choosing the $N = 1$ future state to compute the expert actions, the policy fits the noise and moves too slowly, resulting in state drift away from good trajectories.

We next ablate the use of HG-DAgger while keeping the total amount of data fixed. Specifically, we compare performance of policies trained using 50% expert demos and 50% HG-DAgger interventions, versus using 100% expert demos. In Table 4 (right), we find that HG-DAgger significantly improves task performance over cloning expert demonstrations on both the 'place bottle in ceramic bowl' task and 7 other training tasks. Further details on this comparison are in Appendix K. Finally, in Figure 5, we evaluate whether measuring HG-DAgger interventions can give us a live proxy of policy performance. We see that intervention frequency is inversely correlated with policy success, as measured by the fraction of successful episodes not requiring intervention. This result suggests that we can indeed use this metric with HG-DAgger for development purposes.

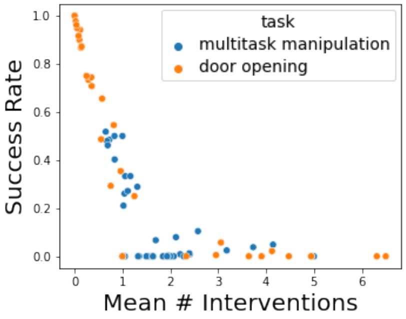


Figure 5: Mean number of interventions vs. task success rate. Each point represents a policy evaluated during HG-DAgger data collection. There is a clear correlation between the mean number of interventions and success rate, suggesting that interventions can be used as a live proxy for performance.

7 Discussion

We presented a multi-task imitation learning system that combines flexible task embeddings with large-scale training on a 100-task demonstration dataset, enabling it to generalize to entirely new tasks that were not seen in training based on user-provided language or video commands. Our evaluation covered 29 unseen vision-based manipulation tasks with a variety of objects and scenes. The key conclusion of our empirical study is that simple imitation learning approaches can be scaled in a way that facilitates generalization to new tasks with zero additional robot data of those tasks. That is, we learn that we do not need more complex approaches to attain task-level generalization. Through the experiments, we also learn that 100 training tasks is sufficient for enabling generalization to new tasks, that HG-DAgger is important for good performance, and that frozen, pre-trained language embeddings make for excellent task conditioners without any additional training.

Our system does have a number of limitations. First, the performance on novel tasks varies significantly. However, even for tasks that are less successful, the robot often exhibits behavior suggesting that it understands at least part of the task, reaching for the right object or performing a semantically related motion. This suggests that an exciting direction for future work is to use our policies as a general-purpose initialization for finetuning of downstream tasks, where additional training, perhaps with autonomous RL, could lead to significantly better performance. The structure of our language commands follows a simple "(verb) (noun)" structure. A direction to address this limitation is to relabel the dataset with a variety of human-provided annotations [24], which could enable the system to handle more variability in the language structure. Another limitation is the lower performance of the video-conditioned policy, which encourages future research on improving the generalization of video-based task representations and enhancing the performance of imitation learning algorithms as a whole, as low-level control errors are also a major bottleneck.

Acknowledgments

Eric Jang, Alex Irpan, and Frederik Ebert ran experiments on different forms of task conditioning in the task generalization setup. Mohi Khansari helped build the HG-DAgger interface and ran experiments in the bin-emptying and door opening tasks. Daniel Kappler built the data annotation visualizer. Corey Lynch advised Frederik's internship and gave pointers on language models. Sergey Levine and Chelsea Finn supervised the project.

We would like to give special thanks to Noah Brown, Omar Cortes, Armando Fuentes, Kyle Jeffrey, Linda Luu, Sphurti Kirit More, Jornell Quiambao, Jarek Rettinghouse, Diego Reyes, Rosario Jauregui Ruano, and Clayton Tan for overseeing robot operations and collecting human videos of the tasks, as well as Jeffrey Bingham and Kanishka Rao for valuable discussions.

References

[1] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[2] S. James, M. Bloesch, and A. J. Davison. Task-embedded control networks for few-shot imitation learning. In *Conference on Robot Learning*, pages 783–795. PMLR, 2018.

[3] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *Robotics: Science and Systems*, 2018.

[4] A. Bonardi, S. James, and A. J. Davison. Learning one-shot imitation from humans without humans. *IEEE Robotics and Automation Letters*, 5(2):3533–3539, 2020.

[5] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *arXiv e-prints*, pages arXiv–2008, 2020.

[6] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

[7] S. Dasari and A. Gupta. Transformers for one-shot visual imitation. *Conference on Robot Learning (CoRL)*, 2020.

[8] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

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

[10] S. Khansari-Zadeh and A. Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *IEEE Transaction on Robotics*, 27(5):943–957, 2011.

[11] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. *Robotics and autonomous systems*, 47(2-3):69–77, 2004.

[12] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005.

[13] R. Chalodhorn, D. B. Grimes, K. Grochow, and R. P. Rao. Learning to walk through imitation. In *IJCAI*, volume 7, pages 2084–2090, 2007.

[14] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[15] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[16] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, Carnegie-Mellon University, 1989.

[17] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[18] A. Zhou, E. Jang, D. Kappler, A. Herzog, M. Khansari, P. Wohlhart, Y. Bai, M. Kalakrishnan, S. Levine, and C. Finn. Watch, try, learn: Meta-learning from demonstrations and reward. *International Conference on Learning Representations (ICLR)*, 2020.

[19] T. L. Paine, S. G. Colmenarejo, Z. Wang, S. Reed, Y. Aytar, T. Pfaff, M. W. Hoffman, G. Barth-Maron, S. Cabi, D. Budden, et al. One-shot high-fidelity imitation: Training large-scale deep nets with rl. *arXiv preprint arXiv:1810.05017*, 2018.

[20] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8565–8574, 2019.

[21] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2050–2053, 2018.

[22] P. Goyal, R. J. Mooney, and S. Niekum. Zero-shot task adaptation using natural language. *arXiv preprint arXiv:2106.02972*, 2021.

[23] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks. *arXiv preprint arXiv:2010.12083*, 2020.

[24] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[25] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. In *2009 International Conference on Advanced Robotics*, pages 1–6. IEEE, 2009.

[26] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[27] S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[28] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

[29] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[30] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. 07 2020. doi:10.15607/RSS.2020.XVI.055.

[31] M. Khansari, D. Kappler, J. Luo, J. Bingham, and M. Kalakrishnan. Action image representation: Learning scalable deep grasping policies with zero real world data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3597–3603, 2020.

[32] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[33] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[34] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[35] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan. Interactively picking real-world objects with unconstrained spoken language instructions. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[36] A. Gupta, A. Murali, D. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *arXiv preprint arXiv:1807.07049*, 2018.

[37] F. Sadeghi, A. Toshev, E. Jang, and S. Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4691–4699, 2018.

[38] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[39] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[40] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[41] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2):1148–1155, 2019.

[42] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[43] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[44] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.

[45] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[46] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung, et al. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*, 2019.

[47] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[48] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[49] P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[50] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[51] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 International Conference on Robotics and Automation (ICRA)*, 2021.

[52] P. de Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. *arXiv preprint arXiv:1905.11979*, 2019.

[53] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR (Poster)*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

[54] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889, 2020.

Appendix

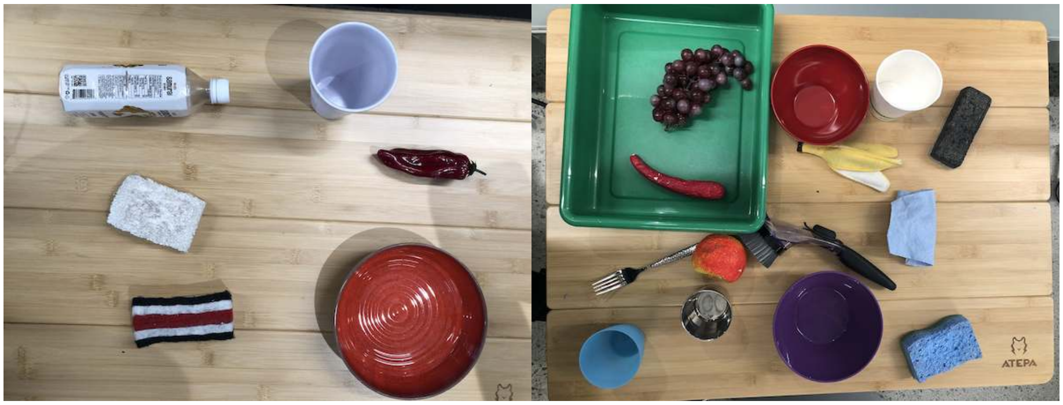
A Teleoperation Interface

The human teleoperator holds two wireless Oculus Quest controllers and uses the same interface to perform demonstrations for all tasks. When the override button is held (the clutch on the right controller), the robot arm tracks the controller's position and orientation. The robot can be toggled between autonomous and manual mode. In manual mode, the robot stays still unless the operator moves it. In autonomous mode, the robot follows a learned policy, unless the operator overrides it.

Table 5: Teleoperation buttons and controls.

Control	**Function**
Right Controller (Arm)	
A	Start recording, or mark demo as success if already recording
B	Stops current recording marking as failure (if applicable), then bring robot to reset pose
Clutch	Override policy and engage manual arm teleop until clutch is released
Trigger	Continuous gripper closure. Pressing the trigger all the way closes the gripper fully, and letting go of the trigger opens the gripper.
Left Controller (Base)	
X	Stop recording demonstration and mark as failure
Y	Engage / disengage autonomous policy
Joystick	Control base forward and yaw velocity (for door opening)

B Data Collection Details



(a) Object Set 1

(b) Object Set 2

Figure 6: Objects used for the 100-task manipulation tasks. Object Set 1 (left) was used to collect data for 21 train tasks, and Object Set 2 (right) was used to collect data for 79 train tasks. For evaluation, 4 tasks were generated between objects in Object Set 2, and 24 holdout tasks used objects across both sets. We used several instances of these objects, with occasional slight differences. For example, the ceramic bowl in Object Set 1 is red in the picture above, but data was also collected with ceramic bowls that were painted green or blue instead.

B.1 Inter-task Variability

The data collection protocol is distributed across multiple robots in 1-4 physical locations, resulting in a policy that handles variations across robot hardware, different backgrounds, and scene configurations. Furthermore, each data collection station uses a set of objects with slight physical variations. For instance, the ceramic bowls come in different colors, the sponges can be blue or white with differences in shape, and the erasers and peppers come in different sizes and materials. This variability, illustrated in Figure 7, results in a higher sample complexity needed to achieve a desired level of performance on a given set of objects.

The low performance of the single-task baseline in Table 4 trained on 1000 demonstrations begs the question of whether this is due to sample-inefficiency of the behavior cloning implementation, or whether a large number of demos are needed to generalize across the variability in the training data. To study this question, we verify single-task policy performance on "place the bottle in the ceramic bowl" in a simulated version of the task, where we can minimize variability across evals and perform exhaustive evaluation of all training checkpoints. When the scene is initialized deterministically with no randomization in initial object positions, 37 expert demos are sufficient to learn the single-task policy with a 97.2% (0.7) success rate. However, when objects in the scene are randomized, a single-task BC baseline only learns a success rate of 56% (2.2) when trained on 40 expert demos. These results suggest that the low success rate of the single-task policies in the real setup are indeed caused by the increased diversity in the environment, instead of other factors.

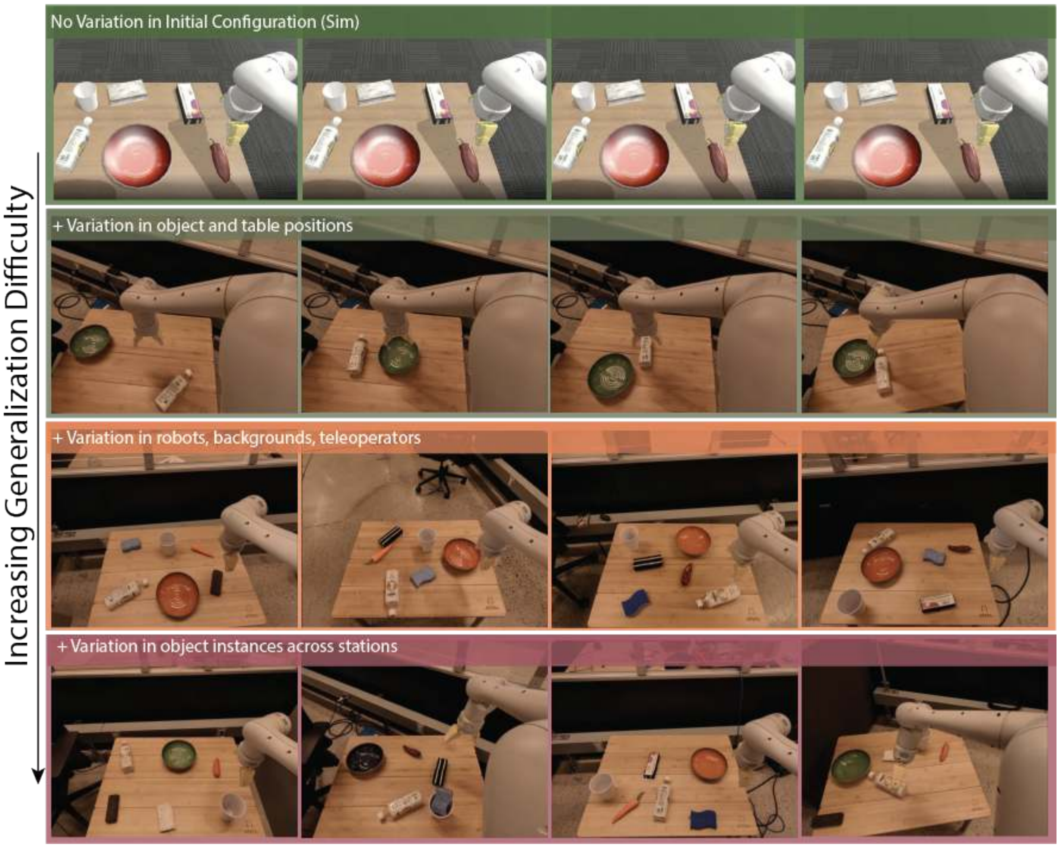

No Variation in Initial Configuration (Sim)
+ Variation in object and table positions
+ Variation in robots, backgrounds, teleoperators
+ Variation in object instances across stations
Increasing Generalization Difficulty

Figure 7: Within each manipulation sub-task (e.g. "place the bottle in the ceramic bowl") the policy must generalize not only to variations in object positions, but also varying backgrounds, different robots, and different object instance variations. The top row shows a simulated setup used to confirm that in absence of scene variation, only 37 expert demonstrations are required to solve the task with a 97% success rate.

B.2 Robotic Teleoperation

Data for the experiments reported in the paper were collected by 7 operators over the course of 5 months. During data collection, tasks are sampled randomly at the beginning of each episode. Objects were occasionally shuffled between episodes, but usually were not, meaning the final state of a demo for one task would often be the start state of a demo for the next task. We found that sampling tasks uniformly was important to performance, since asking teleoperators to pick tasks themselves biased task sampling towards demos that would be easy to perform, creating spurious correlations between initial scene and task demonstrated.

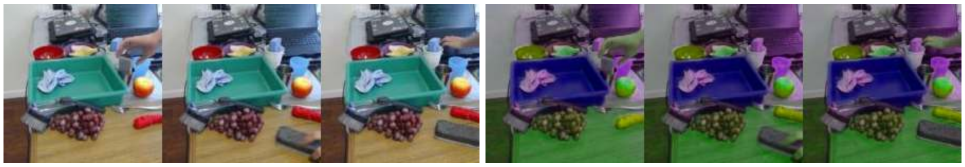


Figure 8: Human demonstrations of the task (left) are augmented with random distortions and reflections (right), then trained to match language features for the task. Robot videos from expert demos are embedded with the same network, along with an end-to-end policy loss.

The camera view is kept fixed over the episode. Automatically moving the head to keep the gripper centered in camera frame affords a larger workspace for performing manipulation tasks, but made learning substantially harder.

We experimented with injecting noise into expert actions, following methods like DART [28], but found that this compromised the ease of providing demonstrations.

B.3 Human Video Data Collection

Human videos were collected in a variety of home and office environments. In each environment, a copy of the scene was set up, and videos were recorded simultaneously by several webcams from different viewpoints. Each viewpoint is treated as a different example of the task, allowing us to collect several videos at once. In general, we found that human videos could be demonstrated 5x-7x faster than a teleoperated robot, so collecting these videos took much less time than collecting the robot dataset.

C Featurization Details

In this work, actions are defined as the difference between states. We found using directly adjacent states ($N = 1$ apart) led to poor performance. Increasing N made action magnitude larger, making the action easier to learn, but introduces bias in the robot trajectory depending on N. Through experiments, we found this bias was most problematic when the robot was changing its gripper between open and closed. We therefore used an *adaptive* algorithm to choose N. For each state, we initialize $N = 1$, then increment it until the change in gripper value of the N'th future state exceeds 0.01 or the L2 norm of the arm joint deltas exceeds 0.05. Intuitively, this speeds up the arm when moving the tool across the workspace (higher bias) and slows it down when it is about to come into or out of contact. We also discard labels in which neither the end effector or the gripper are moving.

D Policy Training Details

A common approach in behavior cloning is to use a Gaussian policy $\pi(a|s, z) = \mathcal{N}(\mu, \sigma)$ to maximize log likelihood of expert actions. We found using a deterministic policy was sufficient to achieve generalization. The network $\pi(a|s, z)$ predicts the delta XYZ, delta axis-angle, and gripper angle (0 to 1). A Huber loss (δ=1.0) [49] is used for XYZ and axis angle, and a log loss is used for gripper angle.

We scaled the XYZ delta losses, axis-angle delta losses, and gripper angle losses by weights of 100, 10, and 0.5, respectively, in order to keep the losses of comparable magnitude for each part of the action.

We implement the model using the FiLM-conditioned ResNet implementation from the open-source Tensor2Robot framework[2]. All models are trained using the Adam optimizer with default TensorFlow momentum parameters. When conditioning the model, we add $\mathcal{N}(0, 0.1)$ Gaussian noise to embedding of the task command, which was critical to getting the model to predict actions based on the task embedding (as opposed to spurious correlations in the camera image). Multitask manipulation policies were trained with a batch size of 4096 on a TPUv3 pod with a learning rate of $5\text{e}{-3}$.

For video task commands, we found policy performance was stronger if we averaged several video embeddings prior to conditioning. When evaluating a novel task, we average the embeddings of 10 new human videos collected for the task.

[2] https://github.com/google-research/tensor2robot/blob/master/layers/film_resnet_model.py

Bin emptying and door opening policies were trained with asynchronous SGD over 10 GPUs with a batch size of 32 per worker and a learning rate of $2.5\text{e}{-4}$. In these tasks, we do not add FiLM layers to condition the model, since there is no task command.

E Video Conditioning Details

For video conditioning, we initially applied a method similar to Task-Embedded Control [2]. A contrastive loss (cosine similarity or InfoNCE [50]) was applied to the videos, along with the end-to-end BC loss. Embedding visualizations (see Figure 9) revealed that the embeddings learned were subpar, collapsing to two main clusters, one for the 21-task family and one for the 79-task family. We hypothesize that because the 21-task family and 79-task family each used a distinct set of objects, a purely unsupervised objective learned to identify the set of visible objects first, rather than the task performed. Adding the language regression loss helped align the videos more semantically.

E.1 Video Preprocessing and Architecture

During training, videos w are randomly subsampled to be 20 frames long, done such that the first and last frame of the video remain in the subsampled w. The same image augmentations used for training the policy are applied to the video, but we additionally apply random reflections along the x-axis and y-axis. The augmentation is sampled once per video and applied identically to all 20 frames. At inference time, no augmentations are applied, and the subsampling of 20 frames is done at a uniform frame rate.

The 20 frames are arranged into a 4x5 array of images that are processed by a 2D ResNet-18 network. Arranging the images in this manner allows the 2D convolutions in the ResNet to perform several layers of temporal convolution without needing to tune a new architecture. This was more memory efficient and more performant than a previously tried temporal convolution architecture. After mean-pooling the final visual features, we add a fully-connected layer with 32 units and ReLU non-linearity, then a linear layer with 512 units and no non-linearity, to match the final size of the language output. The final embedding is normalized to have unit L2 norm. The intermediate 32 unit layer is used to restrict the expressivity of the embedding.

To implement the end-to-end behavior cloning loss, every batch of data must be made of *paired* examples: one human video $w_h \in \mathcal{D}_h^i$ to generate embedding z, and one expert demo $\{(s_t, a_t)_{t=1}^T\} \in \mathcal{D}_e^i$ of matching task to act as labels for $\pi(a|s,z)$. We first sample a batch of tasks, with replacement. For each task, we sample 1 human video and 1 robot demo, combining them into one overall batch for the model. The model is trained using 18 V100 GPUs with a batch size of 28 per GPU. Since each example is 1 pair of videos, every GPU effectively trains on 56 videos at a time. The model was trained with async gradient descent, with a learning rate picked via random search ($2.45\text{e}{-4}$).

At train time, the sequence of images s_t in the robot demo are treated as a video of the task, and preprocessed in the same way as the human video. Human videos and robot videos are encoded with the same q, and both are trained to match the language embedding with a cosine loss, define as $D_{cos}(v_1, v_2) = 1 - v_1 \cdot v_2$. Embedding noise is not added to z_h^i for the end-to-end loss.

E.2 Ablation on Video Encoder Batch

The sampling strategy for the video encoder batch is non-standard. By sampling tasks, then 1 human video and 1 robot video per task, every task will appear equally often, and every batch is guaranteed to be 50% human 50% robot. To examine how this affected model performance, we ran ablations where we first removed the end-to-end behavioral cloning loss, leaving just the language regression objective. Controlling for the same architecture, dataset, hyperparameters, and training time, we change the batch sampling strategy, to either directly sample human and robot videos from all tasks (maintaining a 50-50 batch), or sampling the entire batch uniformly at random over all videos.

Ablations in Table 6 indicate the task-based sampling scheme gives best performance. We hypothesize this sampling does better because it implicitly balances data across both tasks and modalities. The ResNet-18 model also includes batch norm, so it is possible the task-based batch construction affects batch norm in a positive way.

F Video Embedding Visualization

Figure 9 is a similarity matrix across different task embeddings. For each task, we first average the embeddings for all holdout human videos for that task. Entry (i, j) of the matrix is then the

Algorithm 1: Pseudocode for training the video encoder

```
Input: Task commands $\mathcal{W}$, per-task robot dataset $\mathcal{D}_e^i$, per-task human video data $\mathcal{D}_h^i$,
       language encoder $q(\cdot|w_\ell^i)$, video encoder $q(\cdot|w_h)$
while not done training do
    Sample a batch of tasks $i$, with replacement.
    for each task $i \in$ batch do
        Sample human video $w_h \in \mathcal{D}_h^i$
        Sample robot demo $\{(s_t, a_t)\}_{t=1}^T \in \mathcal{D}_e^i$
        Retrieve language command $w_\ell^i$
        $z_h^i \sim q(\cdot|w_h)$    // embed human video
        $z_e^i \sim q(\cdot|\{s_t\}_{t=1}^T)$    // embed robot video
        $z_\ell^i \sim q(\cdot|w_\ell^i)$    // get language vector
        Sample $t \in 1, \cdots, T$
        Compute action $\pi(\hat{a}|s_t, z_h^i)$
        BC-loss $\leftarrow 100 \cdot Huber(xyz) + 10 \cdot Huber(angle) + 0.5 \cdot LogLoss(gripper)$
        Minimize $\mathcal{L} \leftarrow$ BC-loss $+ D_{cos}(z_h^i, z_\ell^i) + D_{cos}(z_e^i, z_\ell^i)$
    end
end
```

Table 6: Ablations of video encoder batch composition. In the ablations below, we control for the same architecture, dataset, hyperparameters, and training time, changing only the sampling strategy for each batch. The end-to-end behavioral cloning loss is removed, leaving just the language regression loss. Accuracy is measured over held-out videos of training tasks, by checking whether the video embedding is closest to the true language embedding from all 100 train tasks.

Video Encoder Batch Makeup	**Video Accuracy**
Baseline (sample 28 tasks, pick 1 human + 1 robot video per task)	84%
Sample 28 human videos + 28 robot videos	80%
Sample 56 videos from entire dataset	74%

cosine similarity between that mean embedding for task i and task j. The first 21 rows/columns of the visualization are the 21-task family, and the remaining entries are sorted alphabetically, which groups tasks with the same leading verbs together (i.e. all ”place grapes in X” are adjacent to one another.) Adding a language loss term helps prevent the model from grouping tasks based on task family rather than semantic meaning.

G Data Annotation Visualization

During data collection, teleoperators would occasionally record an unsuccessful demonstration as a success, or vice versa. To fix these errors, we built a data visualizer where demos could be retrieved via SQL queries over their metadata, then reannotated. In addition to fixing incorrect labels, we used this tool to perform general data cleaning, such as flagging demonstrations where the robot hardware was faulty, or the arm occluded a target object for the entire demo. The interface is shown in Figure 10.

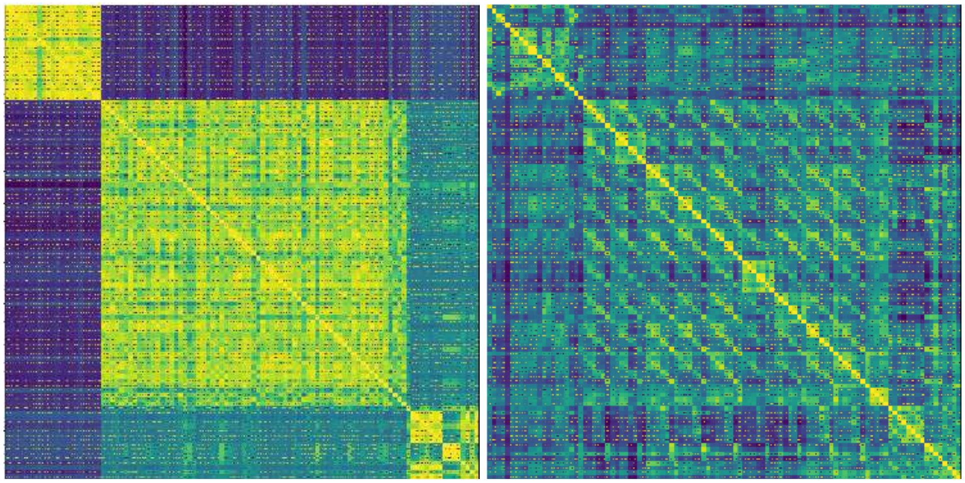
H Single-Task Validation on Bin-Emptying

Figure 11 illustrates the bin emptying and door opening environments. In Figure 12, we plot the bin emptying rate as a function of the amount of data used to train the policy. The policy trained on 30 hours of expert demonstrations can clear 3-4 objects a minute. By comparison, a human teleoperator requires about 43 seconds to demonstrate the task.

An interesting observation to note here is that the bin-emptying task is inherently multimodal, as the objects are grasped in any order during teleoperator demonstrations. The policy learned is a deterministic unimodal policy, and should in principle struggle to learn this multi-model task. but our model architecture is still able to solve the task. One hypothesis for why this worked in practice is that the noise and variety within the dataset enabled the model to break symmetry. For instance, the model may have learned to servo towards the nearest object in cases of ambiguity, or it may

Algorithm 2: Pseudocode for training BC-Z

```
Input: Task commands $\mathcal{W}$, per-task robot dataset $\mathcal{D}_e^i$, per-task human video data $\mathcal{D}_h^i$, language
       encoder $q(\cdot|w_\ell^i)$, video encoder $q(\cdot|w_h)$, number videos to average $N$
// Get task vectors, computed once at start of training
$taskToVec \leftarrow \{\}$
for every task $i$ do
    if video-conditioned then
        $z \leftarrow 0$
        for $c \leftarrow 1$ to $N$ do
            Sample video $w_h$ from $\mathcal{D}_{video}$ matching task $i$
            $z_h^i \sim q(\cdot|w_h)$
            $z \leftarrow z + z_h^i$
        $taskToVec[i] \leftarrow L2Normalize(z)$
    else
        Retrieve language $w_\ell^i$ for task $i$
        $z \leftarrow q(\cdot|w_\ell^i)$
        $taskToVec[i] \leftarrow L2Normalize(z)$
// Train policy
for every epoch do
    $(i, (s_t, a_t)) \sim \mathcal{D}$
    $z^i \leftarrow taskToVec[i] + \mathcal{N}(0, 0.1)$
    $\mathcal{L} = 100 \cdot Huber(xyz) + 10 \cdot Huber(angle) + 0.5 \cdot LogLoss(gripper)$
    Update $\pi(a|s, z^i)$ to minimize $\mathcal{L}$
```



(a) Contrastive baseline

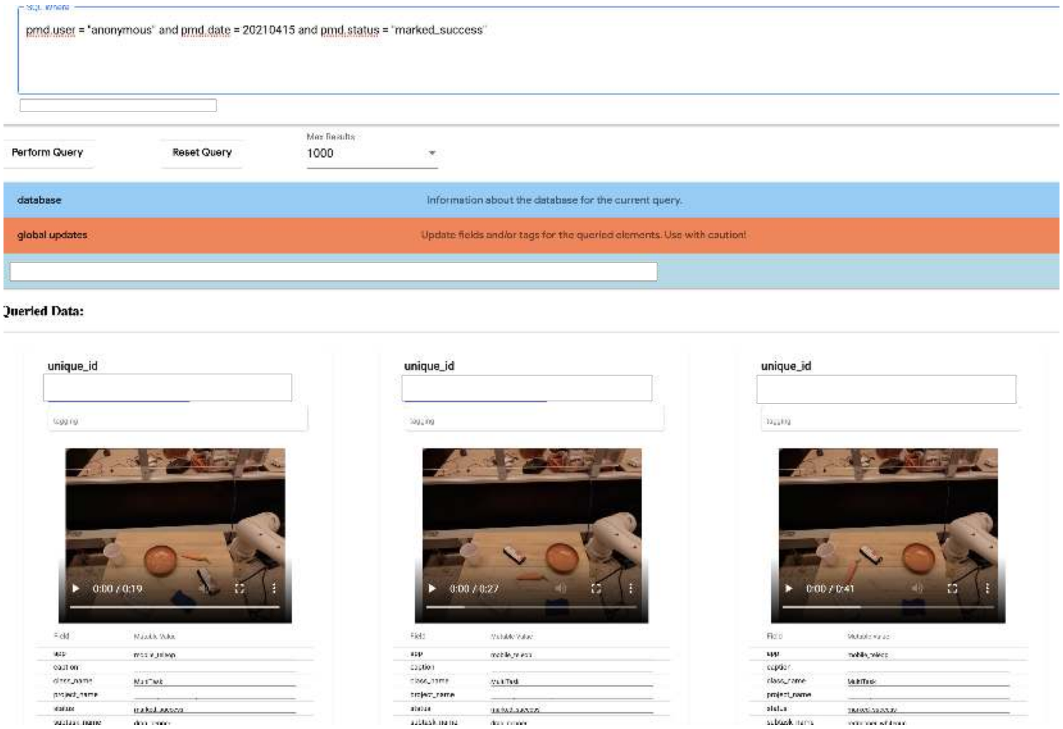
(b) Auxiliary language loss

Figure 9: Visualizations of different video encoders. Each row and column indicates a different task, with the entry at (i, j) indicating the cosine similarity between video embeddings for task i and task j. The 21-task family are the first 21 rows/columns of the visualization. The contrastive baseline learns to primarily group by objects in the scene (by task family), rather than task performed.

have used spurious correlations in the background to commit to a specific object. Whatever the mechanism, it suggests that simple architectures can be sufficient to learn complex tasks, as long as they are trained with appropriate data.

I Single-Task Validation on Door Opening

When validating BC-Z on a door opening task, the policy is trained using 12,000 demonstrations collected across 24 meeting rooms, and 36,000 demonstrations across 5 meeting rooms in simulation. For the 24 real meeting rooms, 10 of them swing open from the right, and 14 swing open from the left. For the 5 sim meeting rooms, 3 are right swing, and 2 are left swing.



SQL where

pmd.user = "anonymous" and pmd.date = 20210415 and pmd.status = "marked_success"

Perform Query

Reset Query

Max Results: 1000

database	Information about the database for the current query.
global updates	Update fields and/or tags for the queried elements. Use with caution!

Queried Data:

unique_id

tagging

0:00 / 0:19

Field	Mutable Value
app	[illegible]
caption	
class_name	MultiTask
project_name	
status	[illegible]
subtask_name	[illegible]

unique_id

tagging

0:00 / 0:27

Field	Mutable Value
app	[illegible]
caption	
class_name	MultiTask
project_name	
status	[illegible]
subtask_name	[illegible]

unique_id

tagging

0:00 / 0:41

Field	Mutable Value
app	[illegible]
caption	
class_name	MultiTask
project_name	
status	[illegible]
subtask_name	[illegible]

Figure 10: A data browsing and reannotation web interface is used to manually curate episodes and check for low-quality demonstrations. Episodes are searchable via SQL, and their metadata can be edited in-place through the web interface.

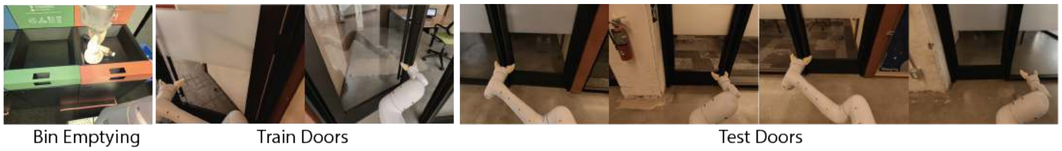


Figure 11: Bin Emptying and Door Opening tasks are used to validate that BC-Z can achieve a high level of task success and generalize to unseen scenes in a single-task imitation learning setting.

For each of these demos, the initial arm configuration, as well as the robot pose with respect to the meeting room's door, were randomized at the start of each run. Instead of controlling the arm, the policy predicts the forward and yaw velocity of the base. This is an easier control problem, but the policy must still determine the position of the randomized arm relative to the door. The policy also predicts a binary termination action to decide when to stop executing actions, which is predicted with a log loss. Real-world demonstrations were collected by 13 operators using a fleet of 30 robots. Sim demonstrations were collected using the same interface, with a virtual robot and door rendered on the computer monitor.

To utilize the simulated door opening demonstrations, we used *sim-to-real* transfer to lower the number of required real world demonstrations. A RetinaGAN model [51] adapted simulated images to look like real images. The RetinaGAN model was trained separately on both sim and real door opening images in an unsupervised fashion by enforcing object-detection consistency between the two domains. Using the RetinaGAN model, we created a third dataset which consists of adapted sim images. We trained the final policy on a mixture of three datasets: sim, real, and adapted sim2real.

Policies were evaluated over 4 holdout meeting rooms, made of 2 right swing doors and 2 left swing doors. For consistency during the evaluation, the operator placed the robot in a certain marked location in front of the meeting room. Then the robot drives to a random pose sampled in an area of 25x25cm from the marked location, and afterwards the trained policy takes control of the robot and performs the task. An episode is marked success if the door is sufficiently open for the robot to enter the room and at the time the policy terminates the task the robot is not in contact with the

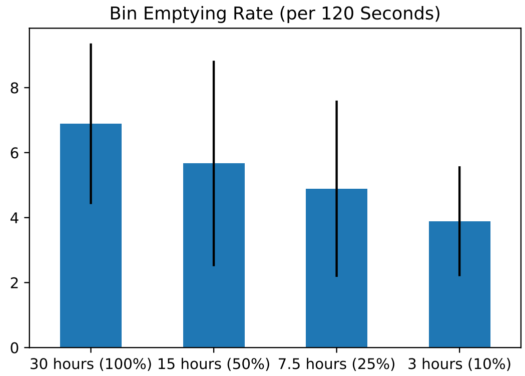


Figure 12: Objects picked in 120 seconds vs. dataset size for bin emptying task

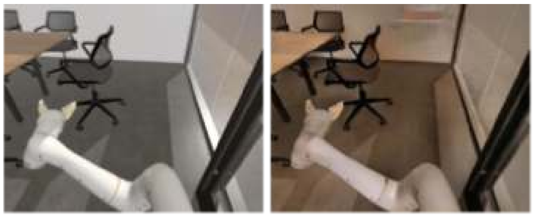


Figure 13: An example of adapting a sim image (left) to look real (right) using RetinaGAN [51].

environment (including the door). Further, any collision of the robot base and arm (not including the gripper) with the environment counted as the task failure by the operator.

J Multi-Task Manipulation Training Tasks

Table 7 lists full results of the different task conditioning ablation from Table 3, over the 21-task family. The other train tasks in the 79-task family are in Table 8.

K Details on HG-DAgger Ablation

Two models were trained, one using all expert-only demonstrations, and one using a 50-50 mixture of expert-only demonstrations and HG-DAgger demonstrations. The 50-50 dataset has the same number of episodes as the expert-only dataset. To reduce evaluation time, the HG-DAgger comparison was evaluated on a smaller subset of 8 tasks from the 21-task family, indicated in Table 7 by the ^{8} symbol.

L Negative Results

Below is an incomplete list of alternative methods tried during the course of the research project. These are empirical observations and are presented as-is. These ideas were generally tried once or twice, then set aside when they did not improve performance. We did not do a detailed investigation of the negative results, so it is possible they did not perform better due to incorrect implementation, the presence of a different performance bottleneck in the system, or improperly tuned hyperparameters, rather than deficiencies in their ideas. We hope this anecdotal experience will be helpful to researchers building on top of this work.

- The policy is trained with a Huber loss ($\delta = 1.0$)), that in practice is usually just an MSE loss, since predictions almost always lie within $(-1, 1)$. A deterministic policy trained with mean-squared error can be viewed as max likelihood with a unimodal Gaussian policy, with learned mean μ and fixed σ. We tried a stochastic policy, with a learned σ based on the current state, but found it did not help and seemed to make training less stable.
- Similarly, a mixture density network (mixture of 10 Gaussians) did not improve performance.

Table 7: Performance comparsion one-hot, language, or video conditioning over 21 training tasks. Video policies are conditioned on held-out videos of the training tasks. Tasks are ordered by increasing average "Demo Length", the number of states observed per expert demonstration, or roughly how many actions the learning policy must learn per episode (both executing at 10 Hz). Length is treated as an estimate of difficulty. A subset of eight tasks denoted by ^{8} are used for comparing additional ablations. Numbers in (parentheses) are 1 unit standard deviation. *See Table 8 for remaining training tasks.*

Tasks From 21-Task Family	**Demo Length**	**One-Hot**	**Language**	**Video**
'knock the eraser over'	54 (20)	65% (7.0)	91% (8.7)	40% (21.9)
'knock the bottle over'	61 (22)	58% (7.4)	71% (12.1)	83% (15.2)
'pick up the ceramic cup'^{8}	65 (25)	67% (4.7)	56% (7.2)	83% (15.2)
'pick up the ceramic bowl'	73 (32)	65% (7.3)	77% (11.7)	35% (12.8)
'push the ceramic bowl across the table'	89 (42)	67% (8.2)	58% (14.2)	50% (35.4)
'place the pepper in the ceramic bowl'	98 (30)	33% (6.5)	39% (11.5)	12.5% (11.7)
'place the ceramic cup in the ceramic bowl'	103 (26)	22% (6.5)	33% (13.6)	66% (27.2)
'place the white sponge in the ceramic bowl'^{8}	106 (32)	43% (5.1)	38% (6.7)	14% (13.2)
'place the bottle in the ceramic bowl'^{8}	110 (32)	45% (5.3)	52% (6.3)	43% (18.7)
'drag the pepper across the table'^{8}	115 (43)	55% (5.0)	33% (8.2)	20% (12.6)
'push the eraser across the table'^{8}	117 (50)	71% (4.7)	45% (7.9)	0% (0)
'place the eraser on the white sponge'^{8}	121 (37)	33% (4.9)	30% (7.2)	25% (15.3)
'place the pepper on the white sponge'	123 (44)	36% (7.1)	83% (10.8)	0% (0)
'place the pepper in the ceramic cup'^{8}	128 (44)	30% (4.6)	49% (7.6)	0% (0)
'place the ceramic cup over the eraser'	130 (38)	9% (5.0)	0% (0)	0% (0)
'place the white sponge in the ceramic cup'	131 (38)	21% (6.9)	25% (10.8)	14% (13.2)
'place the eraser in the ceramic cup'^{8}	135 (43)	37% (4.8)	33% (7.5)	20% (17.9)
'move the arm in a circular motion'	144 (69)	69% (6.7)	12% (6.8)	0% (0)
'drag the ceramic bowl in a circle'	164 (65)	32% (6.8)	0% (0)	0% (0)
'wipe the white sponge on the table'	177 (59)	29% (7.8)	18% (9.2)	0% (0)
'stand the bottle upright'	186 (55)	5% (3.5)	0% (0)	0% (0)
Overall	115	42%	40%	24%

- Using a larger model architecture than ResNet-18 (ResNet-34 and larger) also did not improve performance.

- To address the small action problem identified in Section 5.2, we tried decomposing XYZ prediction into direction and magnitude. The hypothesis was that by making it easier for the model to predict small actions, it would prevent the model from predicting small actions at every state. This did not outperform predicting XYZ directly, and eventually led to the adaptive algorithm used in the final results.

- We initially used a spatial softmax layer in our policy and video encoder. Visualizing those spatial softmax layers made it easier to interpret policy predictions, but performance increased when the spatial softmaxes were removed.

- Conditioning the policy on proprioceptive information, as well as previous robot poses, did not improve performance. It is possible this was due to causal confusion between that information and the expert actions [52].

- Using more video frames (40 instead of 20) did not improve performance of the video encoder, and slowed down training.

- We experimented with including human videos that did not correspond to any of the robot tasks, using them as negative examples for a contrastive loss, to encourage the task embeddings to be more continuous. We found this did not help, and the negative examples were too easy to embed far away from all other videos.

- Pre-training the ResNet on the ILSVRC2012 object classification dataset did not improve performance.

- We obtained better results on manipulation tasks by representing angles as delta axis-angle, rather than absolute axis-angle, absolute quaternions, or delta quaternions.

Table 8: Additional multitask manipulation training task sentences

Name
’wipe purple bowl with sponge’
‘wipe red bowl with sponge’
‘wipe table surface with sponge’
‘wipe purple bowl with towel’
‘wipe red bowl with towel’
‘wipe tray with towel’
‘wipe table surface with towel’
‘wipe purple bowl with brush’
‘wipe red bowl with brush’
‘wipe tray with brush’
‘wipe table surface with brush’
‘wipe purple bowl with eraser’
‘wipe red bowl with eraser’
‘wipe tray with eraser’
‘wipe table surface with eraser’
‘place sponge in purple bowl’
‘place sponge in red bowl’
‘place sponge in table surface’
‘place sponge in metal cup’
‘place sponge in plastic cup’
‘place sponge in paper cup’
‘place towel in purple bowl’
‘place towel in red bowl’
‘place towel in tray’
‘place towel in table surface’
‘place towel in metal cup’
‘place towel in plastic cup’
‘place towel in paper cup’
‘place brush in purple bowl’
‘place brush in red bowl’
‘place brush in tray’
‘place brush in table surface’
‘place brush in metal cup’
‘place brush in plastic cup’
‘place brush in paper cup’
‘place eraser in purple bowl’
‘place eraser in red bowl’
‘place eraser in tray’
‘place eraser in table surface’
‘place eraser in metal cup’
‘place eraser in plastic cup’
‘place eraser in paper cup’
‘place grapes in purple bowl’
‘place grapes in tray’
‘place grapes in table surface’
‘place grapes in metal cup’
‘place grapes in plastic cup’
‘place grapes in paper cup’
‘place banana in purple bowl’
‘place banana in red bowl’
‘place banana in tray’
‘place banana in table surface’
‘place banana in metal cup’
‘place banana in plastic cup’
‘place banana in paper cup’
‘place apple in purple bowl’
‘place apple in red bowl’
‘place apple in tray’
‘place apple in table surface’
‘place apple in metal cup’
‘place apple in plastic cup’
‘place pepper in purple bowl’
‘place pepper in red bowl’
‘place pepper in tray’
‘place pepper in table surface’
‘place pepper in metal cup’
‘place pepper in plastic cup’
‘place pepper in paper cup’
‘place fork in purple bowl’
‘place fork in red bowl’
‘place fork in tray’
‘place fork in table surface’
‘place fork in metal cup’
‘place fork in plastic cup’
‘place fork in paper cup’
‘stack cups on top of each other’
‘stack bowls on top of each other’
‘stack cups into tray’
‘stack bowls into tray’

- At each state, the policy predicts a 10 action long open-loop trajectory, then only executes the first action. Stopping the gradient from the 2nd to 10th predicted actions of the open-loop trajectory was inconclusive but generally did not help.
- Applying mixup regularization [53] to the images and robot poses did not help. We suspected it played poorly with the continuous outputs, and might work better if actions were discretized.
- Predicting gripper residuals instead of the absolute open/close angle.
- We found that while validation error on predicting future poses was correlated with task success, different models with similar validation errors could have wildly varying levels of task success. This made selecting the right checkpoint for evaluation challenging; it is quite possible that performance numbers would be higher with additional evaluation budget for specific checkpoints. This is likely because validation accuracy is most critical on specific states (especially near contact), while there is more tolerance for error on other states.

M Further Video Embedding Comparisons

We perform further experiments on alternative video embedding methods. Each method is trained on the same HG-DAgger dataset that was collected using the BC-Z system and policies. We consider two alternative embedding methods. First, since the pre-trained language embeddings showed considerable success, we consider a pre-trained video embedding [54] that was trained on a large set of instructional videos. Second, we also compare to the embedding approach introduced by [2], referred to as TecNets, which trains the embedding using the end-to-end policy objective and a contrastive loss between different videos (with no language component). In this comparison, we retrain the policy on top of the frozen embedding after it is learned, as in BC-Z, as we find this to improve training stability.

The results are shown in Table 9. Pre-trained video embeddings enable some generalization to held-out tasks, but do not perform quite as well as the learned video embedding in BC-Z. In our visualizations, we find that pretrained video embeddings are quite similar across different tasks, which may have hurt multitask learning from those embeddings. We theorize that these pre-trained models tend to group videos more based on the background scene than based on actions taken. In the TecNet comparison, we find that the TecNets video embedding leads to worse overall performance on held-out tasks. In particular, the results show that TecNets has similar performance to video-conditioned BC-Z on the 4 held-out tasks that only use objects from the 79-task family, but that performance is worse on 3 held-out tasks that mixed objects between the 21-task family and 79-task family. Anecdotally, we also found that the TecNet training runs were less stable: in two runs only differing by random seed, one embedding collapsed and the other did not (Table 9 reports performance of the training run that did not collapse).

Table 9: Performance comparison between different video embeddings on selected tasks. All tasks are held-out unless otherwise indicated. Numbers in (parentheses) are 1 unit standard deviation

Task	**Video-conditioned BC-Z**	**BC-Z, pretrained video embedding**	**BC-Z, TecNet embedding**
‘place sponge in tray’	22% (2.2)	10% (6.7)	0% (0)
‘place grapes in red bowl’	12% (7.8)	30% (10.2)	20% (8.9)
‘place apple in paper cup’	14% (7.8)	0% (0)	25% (9.7)
‘wipe tray with sponge’	28% (10.6)	15% (8.0)	15% (8.0)
‘place banana in ceramic bowl’	7.5% (4.2)	0% (0)	0% (0)
‘pick up bottle’	17.5% (6.0)	10% (6.7)	0% (0)
‘push purple bowl across the table’	0% (0)	0% (0)	0% (0)
Average (held-out)	14.4%	9.3%	8.6%