

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
<i>Right Controller (Arm)</i>	
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.
<i>Left Controller (Base)</i>	
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.

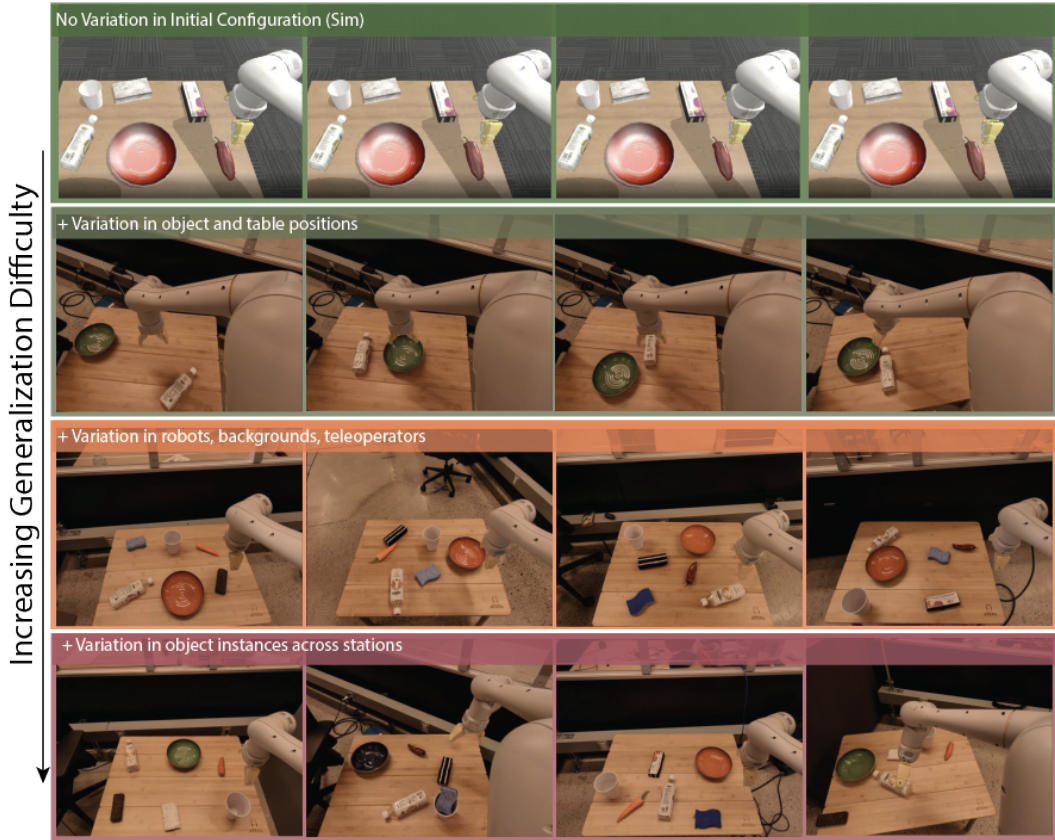


Figure 7: Within each manipulation sub-task (e.g. “place the bottle in the ceramic bowl”) the policy must generalize not only to variations in object positions, but also varying backgrounds, different robots, and different object instance variations. The top row shows a simulated setup used to confirm 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.

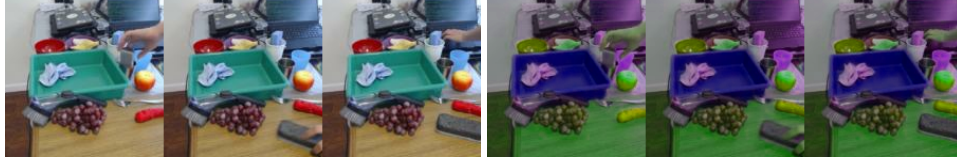


Figure 8: Human demonstrations of the task (left) are augmented with random distortions and reflections (right), then trained to match language features for the task. Robot videos from expert demos are embedded with the same network, along with an end-to-end policy loss.

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 ($\delta=1.0$) [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 framework². 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 $5e-3$.

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.

²https://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 $i \in \text{batch}$ 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_ℓ^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 $t \in 1, \dots, T$

 Compute action $\pi(\hat{a}|s_t, z_h^i)$

 BC-loss $\leftarrow 100 \cdot \text{Huber}(xyz) + 10 \cdot \text{Huber}(\text{angle}) + 0.5 \cdot \text{LogLoss}(\text{gripper})$

 Minimize $\mathcal{L} \leftarrow \text{BC-loss} + D_{\cos}(z_h^i, z_\ell^i) + D_{\cos}(z_e^i, z_\ell^i)$

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-modal 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_ℓ^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}

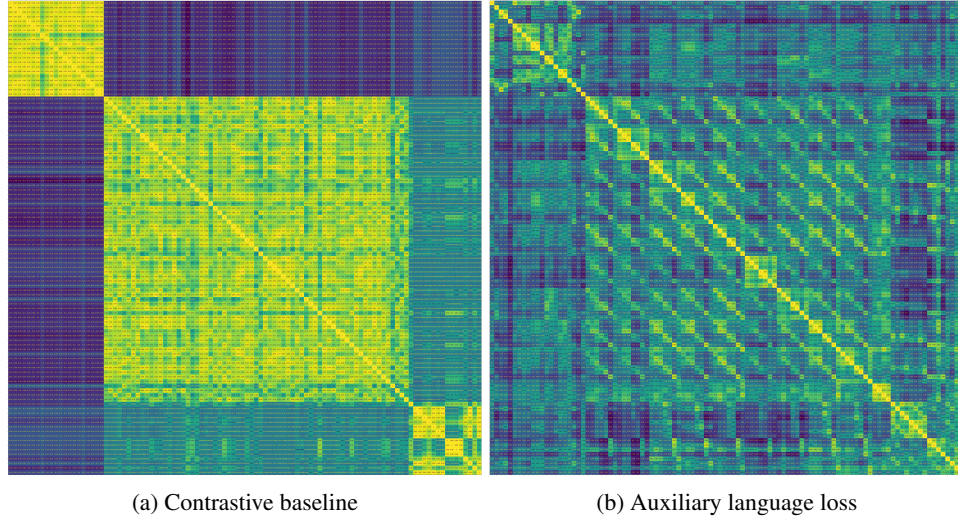


Figure 9: Visualizations of different video encoders. Each row and column indicates a different task, with the entry at (i, j) indicating the cosine similarity between video embeddings for task i and task j . The 21-task family are the first 21 rows/columns of the visualization. The contrastive baseline 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.

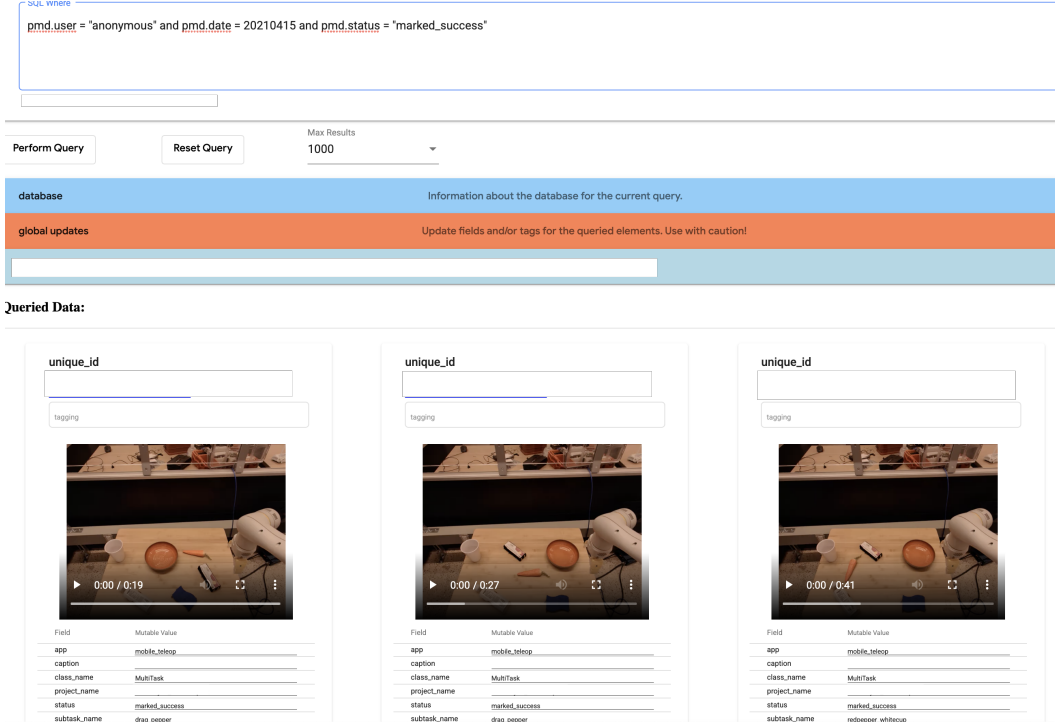


Figure 10: A data browsing and reannotation web interface is used to manually curate episodes and check for low-quality demonstrations. Episodes are searchable via SQL, and their metadata can be edited in-place through the web interface.



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

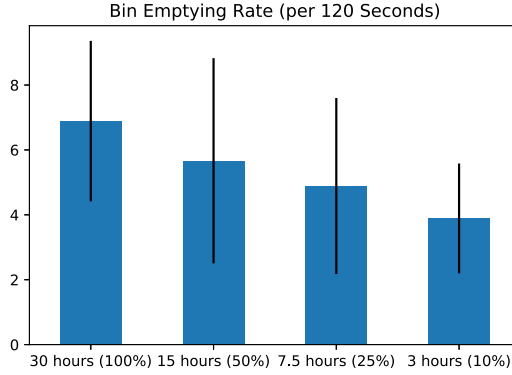


Figure 12: Objects picked in 120 seconds vs. dataset size for bin emptying task



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 comparison 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)
<i>Overall</i>	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	Name
'wipe purple bowl with sponge'	'place eraser in plastic cup'
'wipe red bowl with sponge'	'place eraser in paper cup'
'wipe table surface with sponge'	'place grapes in purple bowl'
'wipe purple bowl with towel'	'place grapes in tray'
'wipe red bowl with towel'	'place grapes in table surface'
'wipe tray with towel'	'place grapes in metal cup'
'wipe table surface with towel'	'place grapes in plastic cup'
'wipe purple bowl with brush'	'place grapes in paper cup'
'wipe red bowl with brush'	'place banana in purple bowl'
'wipe tray with brush'	'place banana in red bowl'
'wipe table surface with brush'	'place banana in tray'
'wipe purple bowl with eraser'	'place banana in table surface'
'wipe red bowl with eraser'	'place banana in metal cup'
'wipe tray with eraser'	'place banana in plastic cup'
'wipe table surface with eraser'	'place banana in paper cup'
'place sponge in purple bowl'	'place apple in purple bowl'
'place sponge in red bowl'	'place apple in red bowl'
'place sponge in table surface'	'place apple in tray'
'place sponge in metal cup'	'place apple in table surface'
'place sponge in plastic cup'	'place apple in metal cup'
'place sponge in paper cup'	'place apple in plastic cup'
'place towel in purple bowl'	'place pepper in purple bowl'
'place towel in red bowl'	'place pepper in red bowl'
'place towel in tray'	'place pepper in tray'
'place towel in table surface'	'place pepper in table surface'
'place towel in metal cup'	'place pepper in metal cup'
'place towel in plastic cup'	'place pepper in plastic cup'
'place towel in paper cup'	'place pepper in paper cup'
'place brush in purple bowl'	'place fork in purple bowl'
'place brush in red bowl'	'place fork in red bowl'
'place brush in tray'	'place fork in tray'
'place brush in table surface'	'place fork in table surface'
'place brush in metal cup'	'place fork in metal cup'
'place brush in plastic cup'	'place fork in plastic cup'
'place brush in paper cup'	'place fork in paper cup'
'place eraser in purple bowl'	'stack cups on top of each other'
'place eraser in red bowl'	'stack bowls on top of each other'
'place eraser in tray'	'stack cups into tray'
'place eraser in table surface'	'stack bowls into tray'
'place eraser in metal cup'	

- 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)
<i>Average (held-out)</i>	14.4%	9.3%	8.6%