Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

RT-1: Robotics Transformer for Real-World Control at Scale

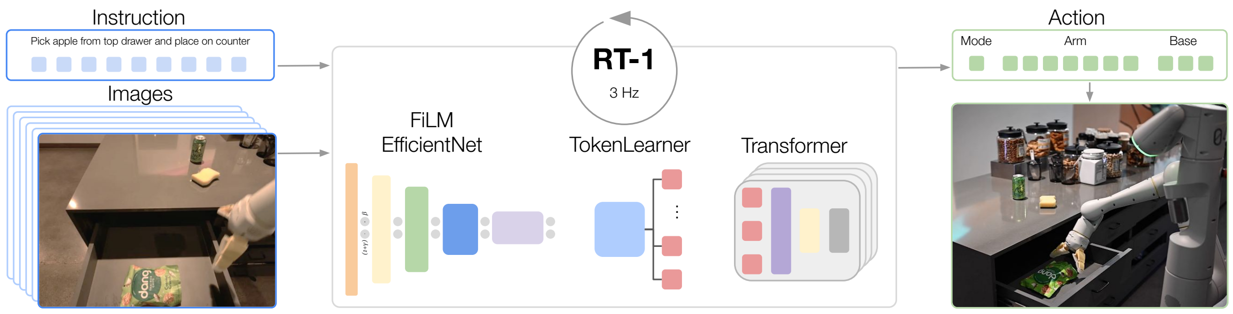
Anthony Brohan*, Noah Brown*, Justice Carbajal*, Yevgen Chebotar*, Joseph Dabis*, Chelsea Finn*, Keerthana Gopalakrishnan*, Karol Hausman*, Alex Herzog†, Jasmine Hsu*, Julian Ibarz*, Brian Ichter*, Alex Irpan*, Tomas Jackson*, Sally Jesmonth*, Nikhil J Joshi*, Ryan Julian*, Dmitry Kalashnikov*, Yuheng Kuang*, Isabel Leal*, Kuang-Huei Lee‡, Sergey Levine*, Yao Lu*, Utsav Malla*, Deeksha Manjunath*, Igor Mordatch‡, Ofir Nachum‡, Carolina Parada*, Jodilyn Peralta*, Emily Perez*, Karl Pertsch*, Jornell Quiambao*, Kanishka Rao*, Michael Ryoo*, Grecia Salazar*, Pannag Sanketi*, Kevin Sayed*, Jaspiar Singh*, Sumedh Sontakke‡, Austin Stone*, Clayton Tan*, Huong Tran*, Vincent Vanhoucke*, Steve Vega*, Quan Vuong*, Fei Xia*, Ted Xiao*, Peng Xu*, Sichun Xu*, Tianhe Yu*, Brianna Zitkovich*

*Robotics at Google, †Everyday Robots, ‡Google Research, Brain Team
Authors listed in alphabetical order. Contributions in Appendix A.
Corresponding emails: {keerthanapg,kanishkarao,karolhausman}@google.com
Website: robotics-transformer.github.io. Code: github.com/google-research/robotics_transformer

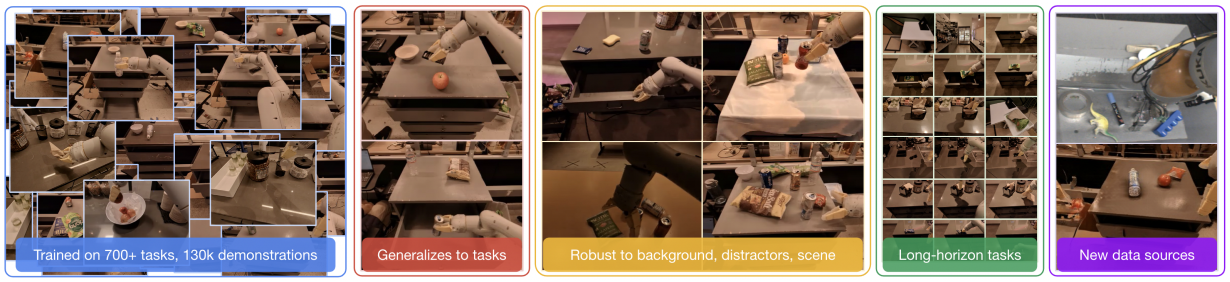
***Abstract*—By transferring knowledge from large, diverse, task-agnostic datasets, modern machine learning models can solve specific downstream tasks either zero-shot or with small task-specific datasets to a high level of performance. While this capability has been demonstrated in other fields such as computer vision, natural language processing or speech recognition, it remains to be shown in robotics, where the generalization capabilities of the models are particularly critical due to the difficulty of collecting real-world robotic data. We argue that one of the keys to the success of such general robotic models lies with open-ended task-agnostic training, combined with high-capacity architectures that can absorb all of the diverse, robotic data. In this paper, we present a model class, dubbed Robotics Transformer, that exhibits promising scalable model properties. We verify our conclusions in a study of different model classes and their ability to generalize as a function of the data size, model size, and data diversity based on a large-scale data collection on real robots performing real-world tasks.**

I. Introduction

End-to-end robotic learning, with either imitation or reinforcement, typically involves collecting task-specific data in either single-task [28, 71] or multi-task [30, 23] settings that are narrowly tailored to the tasks that the robot should perform. This workflow mirrors the classic approach to supervised learning in other domains, such as computer vision and NLP, where task-specific datasets would be collected, labeled, and deployed to solve individual tasks, with little interplay between the tasks themselves. Recent years have seen a transformation in vision, NLP, and other domains, away from siloed, small-scale datasets and models and towards large, general models pre-trained on broad, large datasets. The keys to the success of such models lie with open-ended task-agnostic training,



(a) RT-1 takes images and natural language instructions and outputs discretized base and arm actions. Despite its size (35M parameters), it does this at 3 Hz, due to its efficient yet high-capacity architecture: a FiLM [45] conditioned EfficientNet [61], a TokenLearner [52], and a Transformer [63].


Trained on 700+ tasks, 130k demonstrations
Generalizes to tasks
Robust to background, distractors, scene
Long-horizon tasks
New data sources

(b) RT-1's large-scale, real-world training and evaluation (3000 real-world trials) show generalization, robustness, and ability to learn from diverse data.

Fig. 1: A high-level overview of RT-1's architecture, dataset, and evaluation.

combined with high-capacity architectures that can absorb all of the knowledge present in large-scale datasets. If a model can "sponge up" experience to learn general patterns in language or perception, then it can bring them to bear on individual tasks more efficiently. While removing the need for large task-specific datasets is appealing generally in supervised learning, it is even more critical in robotics, where datasets might require engineering-heavy autonomous operation or expensive human demonstrations. We therefore ask: can we train a single, capable, large multi-task backbone model on data consisting of a wide variety of robotic tasks? And does such a model enjoy the benefits observed in other domains, exhibiting zero-shot generalization to new tasks, environments, and objects?

Building such models in robotics is not easy. Although recent years have seen several large multi-task robot policies proposed in the literature [51, 23], such models often have limited breadth of real-world tasks, as with Gato [51], or focus on training tasks rather than generalization to new tasks, as with recent instruction following methods [56, 57], or attain comparatively lower performance on new tasks [23].

The two main challenges lie in assembling the right dataset and designing the right model. While data collection and curation is often the "unsung hero" of many large-scale machine learning projects [48, 50], this is especially true in robotics, where datasets are often robot-specific and gathered manually [6, 10]. As we will show in our evaluations, good generalization requires datasets that combine both scale and breadth, covering a variety of tasks and settings. At the same time, the tasks in the dataset should be sufficiently well-connected to enable generalization, such that the model can discover the patterns between structural similar tasks and perform new tasks that combine those patterns in novel ways. We utilize a dataset that we gathered over the course of 17 months with a fleet of 13 robots, containing $\sim$130k episodes and over 700 tasks, and we ablate various aspects of this dataset in our evaluation.

The second challenge lies in the design of the model itself. Effective robotic multi-task learning requires a high capacity model, and Transformer [63] models excel in this regard, particularly when it is necessary to learn many tasks conditioned, as in our case, on language instructions. However, robotic controllers must also be efficient enough to run in real time, which presents a major challenge for Transformers in particular. We propose a novel architecture that we call RT-1 (Robotics Transformer 1), which by encoding high-dimensional inputs and outputs, including camera images, instructions and motor commands into compact token representations to be used by the Transformer, allows for efficient inference at runtime to make real-time control feasible.

Our contribution is the RT-1 model and experiments with this model on a large and broad dataset of real-world robotic tasks. Our experiments not only demonstrate that RT-1 can exhibit significantly improved generalization and robustness compared to prior techniques, but also evaluate and ablate many design choices in both the model and in the composition of the training set. Our results show that RT-1 can perform over 700 training instructions at 97% success rate, and can generalize to new tasks, distractors, and backgrounds 25%, 36% and 18% better than the next best baseline, respectively. This level of performance allows us to execute very long-horizon tasks in the SayCan [1] framework, with as many as 50 stages. We further show that RT-1 can incorporate data from simulation or even other robot types, retaining performance on the original tasks and improving generalization to new scenarios. Fig. 1b shows an overview of RT-1's capabilities.

II. Related Work

A number of recent works have proposed Transformer-based policies for robotic control. As in RT-1, several works use language commands processed with Transformers as a robust framework for specifying and generalizing to new tasks [72, 44, 58, 23, 1, 42]. Our work takes the application of Transformers a step further and treats the mapping of language and vision observations to robot actions as a sequence modelling problem, using a Transformer to learn this mapping. This idea is directly inspired by successes in game-playing [4, 34] as well as simulated robot navigation [12], locomotion [24, 15], and manipulation [25] environments. We note that several of these works go beyond only text conditioning and use Transformers to also generalize across robot morphologies (e.g., Gupta et al. [15]) and other modalities for task specifications (e.g., Jang et al. [23], Jiang et al. [25]). These extensions are promising future directions for RT-1.

Beyond Transformer-based policies, the focus of our work is on generalizable and robust real-world robotic manipulation at scale. Existing works on real-world Transformer-based robotic manipulation focus on efficiently learning tasks from a set of demonstrations per task [57]. Behavior Transformer [54] and Gato [51] advocate for training a single model on large-scale robotic and non-robotic datasets. However, these works are limited in their real-world robotic tasks; e.g., Gato learns effectively a single task (colored block stacking) without evaluating generalization to new tasks or a variety of real-world settings. On the technical side, our work examines how Transformer-based policies can be built so as to combine high capacity and generalization with the computational efficiency necessary for real-time control.

While the use of high-capacity Transformer models to learn robotic control policies is a fairly recent innovation, robotics has a long history of multi-task and language-conditioned learning, and RT-1 builds on these foundations. A significant body of work deals with learning policies and predictive models for robotic grasping [53, 36, 46, 14, 64], with the aim of generalizing to new objects. Prior works have sought to address robotic language understanding through pipelined approaches that combine language parsing, vision, and robotic control [40, 31, 62] and with end-to-end approaches [41, 60, 39, 1]. Multi-task robotic learning has also been approached from the perspective of learning to reach goals [5, 49, 27, 19], as well as learning policies that can perform tasks in a discrete set or some other parameterized form [7, 8, 13, 29]. Mobile manipulation policy learning too has been studied, primarilary via reinforcement-learning [32, 26, 67, 37, 65, 20, 18, 11, 69], and recently through foundation models [3, 38, 66]. Herein, we add to this line of work, where we provide evidence that it is possible to learn simple mobile manipulation tasks in an end-to-end fashion with large-scale Transformer policies. A number of prior works in robotics have also focused on collecting datasets containing demonstrations or trials that illustrate a variety of different tasks [55, 6, 70, 59, 22]. Our work adds further evidence in support of the power of multi-task, language-conditioned robotic learning, presenting experimental results at a larger scale and with a greater variety of behaviors, objects, and scenes and proposing new architectures and design choices that enable robotic learning at a significantly larger scale.

III. RT-1: Robotics Transformer

The goal of this work is to build and demonstrate a general robot learning system that can absorb large amounts of data and generalize effectively. In the following, we overview the system that allows us to effectively learn how to execute over 700 instructions in the real world.

A. *Robots and environments*

We use mobile manipulators with a 7 dof arm, a two-fingered gripper, and a mobile base (see Fig. 3 (d)). To collect data and evaluate our method, we use three kitchen-based environments: two real office kitchens and a training environment modelled off these real kitchens. The training environment, shown in Fig. 3 (a), consists of partial counters and is constructed for large scale data collection. The two real environments, shown in Fig. 3 (b, c), have similar counter tops to the training environment, but vary in lighting, background, and full kitchen geometry (e.g., there may be a cabinet instead of a drawer or a sink may be visible). We evaluate the performance of our policies across these environments, measuring the policy's performance and ability to generalize.

B. *Model*

Our model is built on a Transformer architecture [63] and takes a history of images and task description as input and directly outputs tokenized actions, as shown in Fig. 1a and in detail in Fig. 2. In the following we describe the components of the model, following the top-to-bottom order in Fig. 2. More detail on model selection at scale are provided in Appendix F.

Instruction and image tokenization. The RT-1 architecture relies on a data-efficient and compact tokenization of images *and* language instruction. RT-1 tokenizes a history of 6 images by passing images through an ImageNet pretrained EfficientNet-B3 [61] model, which takes 6 images of resolution 300×300 as input and outputs a spatial feature map of shape $9 \times 9 \times 512$ from the final convolutional layer. Unlike Reed et al. [51], we do not patchify the images into visual tokens prior to feeding them to our Transformer backbone. We instead flatten the output feature map from the EfficientNet into 81 visual tokens which are passed on to the later layers of the network.

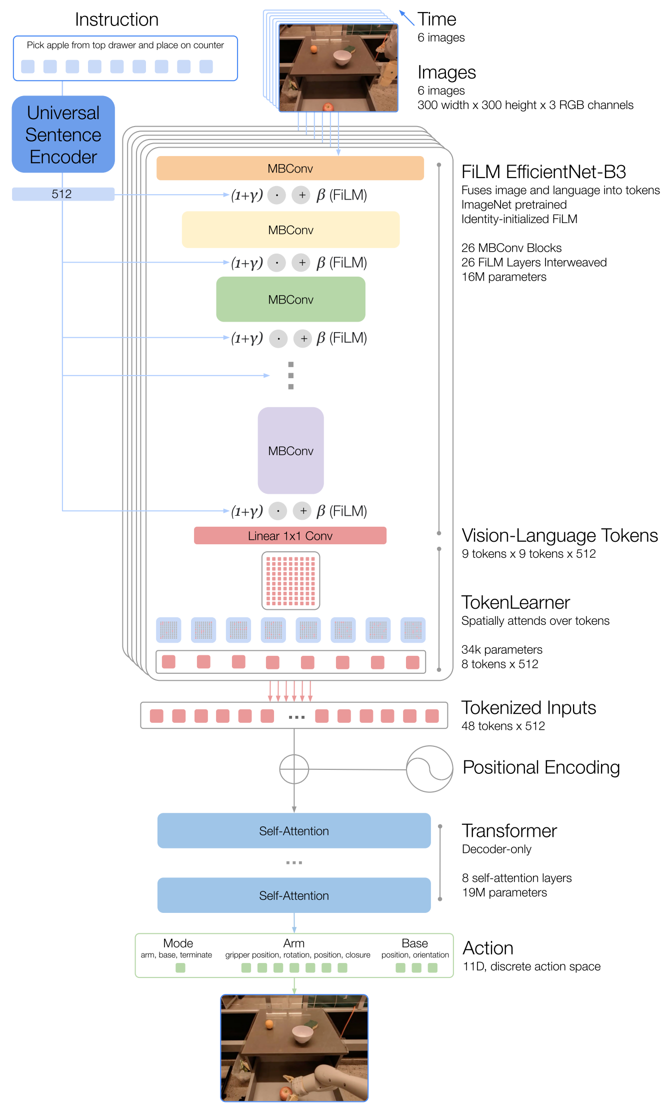


Fig. 2: The architecture diagram of RT-1. The instruction is transformed into a USE embedding and used to condition a pre-trained EfficientNet via FiLM layers. The resulting vision-language tokens are reduced by the TokenLearner and fed into a decoder-only Transformer, which outputs tokenized actions.

To include the language instruction, we condition the image tokenizer on the natural language instruction in the form of a pretrained language embedding, allowing extraction of task-relevant image features early on and improving performance of RT-1. The instruction is first embedded via Universal Sentence Encoder [2]. This embedding is then used as input to identity-initialized FiLM layers [45] added to the pretrained EfficientNet to condition the image encoder. Normally, inserting a FiLM layer into the interior of a pretrained network would disrupt the intermediate activations and negate the benefit of using pretrained weights. To overcome this, we initialize the weights of the dense layers (f_c and h_C) which produce the FiLM affine transformation to zero, allowing the FiLM layer to initially act as an identity and preserve the function of

Skill	Count	Description	Example Instruction
Pick `Object`	130	Lift the object off the surface	pick iced tea can
Move `Object` Near `Object`	337	Move the first object near the second	move pepsi can near rxbar blueberry
Place `Object` Upright	8	Place an elongated object upright	place water bottle upright
Knock `Object` Over	8	Knock an elongated object over	knock redbull can over
Open `Drawer`	3	Open any of the cabinet drawers	open the top drawer
Close `Drawer`	3	Close any of the cabinet drawers	close the middle drawer
Place `Object` into `Receptacle`	84	Place an object into a receptacle	place brown chip bag into white bowl
Pick `Object` from `Receptacle` and Place on the Counter	162	Pick an object up from a location and then place it on the counter	pick green jalapeno chip bag from paper bowl and place on counter
Section IV-C and IV-D tasks	9	Skills trained for realistic, long instructions	open the large glass jar of pistachios pull napkin out of dispenser grab scooper
Total	744		

TABLE I: The list of skills collected for RT-1 together with their descriptions and example instructions.

the pretrained weights. We find that identity-initialized FiLM also produces better results when training with an EfficientNet initialized from scratch, without ImageNet pretraining, but it does not surpass the initialization described above. The architecture of the image tokenizer is presented in Fig. 2.

RT-1's image and instruction tokenization via FiLM EfficientNet-B3 is a total of 16M parameters, with 26 layers of MBConv blocks and FiLM layers, which output 81 tokens.

TokenLearner. To further compress the number of tokens that RT-1 needs to attend over and thus speed up inference, RT-1 uses TokenLearner [52]. TokenLearner is an element-wise attention module that learns to map a large number of tokens into a much smaller number of tokens. This allows us to soft-select image tokens based on their information, passing only the important token combinations to the subsequent Transformer layers. The inclusion of TokenLearner subsamples the 81 visual tokens that come out of the pre-trained FiLM-EfficientNet layers to just 8 final tokens that are then passed on to our Transformer layers.

Transformer. These 8 tokens per-image are then concatenated with the other images in the history, forming 48 total tokens (with added position encoding) to be fed into the Transformer backbone of RT-1. The Transformer is a decoder-only sequence model with 8 self-attention layers and 19M total parameters that outputs action tokens.

Action tokenization. To tokenize actions, each action dimension in RT-1 is discretized into 256 bins. As mentioned previously, the action dimensions we consider include seven for the arm movement (x, y, z, roll, pitch, yaw, opening of the gripper), three for base movement (x, y, yaw) and a discrete variable to switch between three modes: controlling arm, base or terminating the episode. We specify the low-level robot control details in Appendix Section C. For each variable, we map the target to one of the 256 bins, where the bins are uniformly distributed within the bounds of each variable.

Loss. We use a standard categorical cross-entropy entropy objective and causal masking that was utilized in prior Transformer-based controllers [51, 34] and a standard behavior cloning loss used in imitation learning. We provide more details on preliminaries in the appendix Sec. A.

Inference speed. In contrast to many applications of large models, such as natural language or image generation, one of the unique requirements for a model that needs to run on real robots in real time is fast and consistent inference speed. Given our measurements of how long it takes a human to accomplish the instructions considered in this work (which are in the $2-4$ secs range), we want the model to be not significantly slower than that. Based on our experiments this requirement corresponds to at least 3Hz control frequency and the resulting inference time budget for the model, given other latencies in the system, to be less than 100ms.

This requirement limits the size of the model that we can use. We further explore the impact of model size on inference speed in the experiments. We employ two techniques to speed up inference: (i) reduce the number of tokens generated by a pre-trained EfficientNet model by using TokenLearner [52], (ii) compute these tokens only once and reuse them for the following windows that overlap for the future inferences. Both of these allow us to speed up the model inference by 2.4 and 1.7 times, respectively. Additional details on model inference are in Appendix B.

C. Data

Our goal is to build a system that exhibits high performance, generalization to new tasks, and robustness to distractors and backgrounds. We therefore aim to collect a large, diverse dataset of robot trajectories that includes multiple tasks, objects and environments. Our primary dataset consists of ~130k robot demonstrations, collected with a fleet of 13 robots over the course of 17 months. We conducted this large-scale data collection in a series of office kitchen segments, which we refer to as *robot classrooms*, shown in Fig. 3. More details on data collection are in Appendix E.

Skills and instructions. While the definition of a task remains inconsistent in the literature, in this work we count the number of language instructions that the system can perform, where an instruction corresponds to a verb surrounded by one or multiple nouns, such as "*place* water bottle *upright*", "*move* the coke can *to* the green chip bag" or "*open* the drawer". RT-1 is able to perform over 700 language instructions in multiple realistic office kitchen environments that we evaluate and describe in detail in the experiments. In order to group the evaluations and draw conclusions on the performance of the system, we group the instructions by the verbs used in them, which we refer to as *skills*. Unfortunately, there are no agreed upon definitions of what constitutes a new task, and thus we strive to define our version as clearly as possible. For our definition, there is a substantial body of work in the machine learning and robotics literature (e.g. [7, 70, 33]) that consider different tasks to be MDPs with different dynamics or reward functions. Specifically, different tasks correspond to different objective criteria being met: picking up and placing the apple into the top drawer will not be accomplished if the robot picks up the wrong object from the table or places the apple into the wrong drawer. A more detailed list of instructions is shown in Table I, with examples and the number of instructions per skill.

The current set of skills includes picking, placing, opening and closing drawers, getting items in and out drawers, placing elongated items up-right, knocking them over, pulling napkins and opening jars. The skills were chosen to demonstrate multiple behaviors with many objects (seen in Fig. 3(e)) to test aspects of RT-1 such as generalization to new instructions and ability to perform many tasks. We then greatly expanded the object diversity for the "pick" skill to make sure that the skills generalize to varied objects (see the expanded set of objects in Fig. 3(f)). The skills were further expanded while we conducted the ablations to include instructions added in the last row of Table I, which were used for the experiments described in Sec. IV-D and IV-C. These additional skills focused on realistic, long-horizon instructions in an office kitchen. The entire process of adding tasks and data is described in the Appendix G. Since we do not make any assumptions about particular skills when adding new instructions, the system is easily extendable, and we can continuously provide more diverse data to improve its capabilities.

IV. Experiments

Our experiments seek to answer the following questions: (1) Can an RT-1 learn to perform a large number of instructions, as well as to generalize in zero shot to new tasks, objects and environments? (Section IV-B) (2) Can we push the resulting model even further by incorporating heterogeneous data sources, such as simulated data or data from different robots? (Section IV-C) (3) How do various methods generalize to long-horizon robotic scenarios? (Section IV-D) (4) How do generalization metrics change with varying amounts of data quantity and data diversity? (Section IV-E) (5) What are the important and practical decisions in the design of the model and how do they affect performance and generalization? (Appendix Section M)

Throughout this section we will compare to two baseline state of the art architectures, Gato [51] and BC-Z [23]. Importantly both of these are trained on our data described in detail in Sec. III-C (which is an important part of our system) since the original models in these publications would not exhibit generalization properties required for our evaluation tasks. We provide details on the baselines in the appendix Section H.

We evaluate the success rate in experiments to measure performance on training instructions, generalization to unseen instructions, robustness to backgrounds and distractors, and performance in long-horizon scenarios, as detailed below. Throughout this section, we evaluate our approach and baselines with over 3000 real-world trials, making one of the largest scale evaluation of a robot learning system to-date.

A. Experimental Setup

We evaluate RT-1 with a set of mobile manipulators in three environments: two real office kitchens and a training environment modelled off these real kitchens. The training environment, shown in Fig. 3 (a), consists of partial counters while the two real environments, shown in Fig. 3 (b, c), have similar counter tops to the training environment, but vary in lighting, background, and full kitchen geometry (e.g., there may be a cabinet instead of a drawer or a sink may be visible). The policies are evaluated for performance on training tasks as well as generalization to new tasks, robustness to unseen environments, and performance when chained together for long-horizon tasks, as detailed below.

Success Metric. The key metric used throughout our experimental evaluation is success rate of a given skill. An episode is marked as failure if either the robot does not perform the requested skill (as judged by a human rater) or if it violates any of the following constraints: i) the robot collides with the environment, ii) the robot touches irrelevant objects iii) the robot behaves in an unsafe manner iv) the robot does not complete the task within 100 actions ($\sim$ 33secs), roughly half a minute. For example, if the robot gripper is broken while opening a drawer or if objects are knocked to the floor those are considered failures even though the task might be accomplished.

Seen task performance. To evaluate performance on seen instructions, we evaluate performance on instructions sampled from the training set. Note, however, that this evaluation still involves varying the placement of objects and other factors of the setup (e.g., time of day, robot position), requiring the skills to generalize to realistic variability in the environment. In all, we test over 200 tasks in this evaluation: 36 for picking objects, 35 for knocking objects, 35 for placing things upright, 48 for moving objects, 18 for opening and closing various drawers, and 36 for picking out of and placing objects into drawers.

Unseen tasks generalization. To evaluate generalization to unseen tasks, we test 21 novel, unseen instructions. These instructions are distributed across skills and objects. This ensures that at least some instances of each object and skill were present in the training set but they will be combined in novel ways. For example, if "pick up the apple" is held out, then there are other training instructions that include the apple. The list of all unseen instructions can be found in the Appendix I.

Robustness. To evaluate robustness, we perform 30 real-world tasks for distractor robustness and 22 tasks for back-

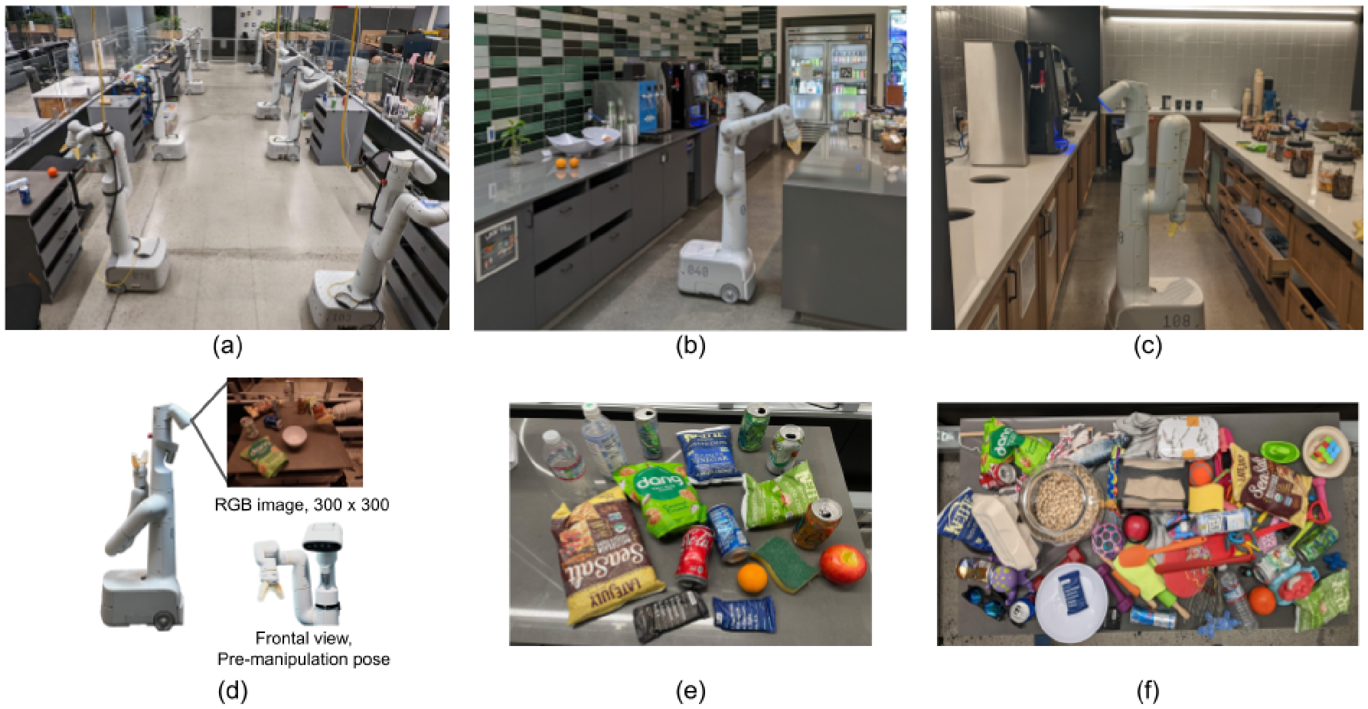


Fig. 3: (a) Robot classroom where we collect data at scale; (b) a real office kitchen, one of the two realistic environments used for evaluation (named Kitchen1 in the rest of the paper); (c) a different office kitchen used for evaluation (named Kitchen2 in the rest of the paper); (d) mobile manipulator used throughout the paper; (e) a set of objects used for most of the skills to expand skill diversity; (f) a more diverse set of objects used mostly to expand object diversity of the picking skill.

ground robustness. The background robustness was tested by evaluating in new kitchens (which have different lighting and background visuals) and with different counter surfaces (e.g., a patterned table cloth). Example configurations of the robustness evaluation scenarios are depicted in Fig. 4.

Long-horizon scenarios. We also evaluate generalization to more realistic long-horizon scenarios, which each require executing a sequence of skills. These evaluations consist of 15 long-horizon instructions in two real kitchens, which require executing sequences of skills consisting of $\sim$ 10 distinct steps, with each step of roughly comparable scope as the training instructions. These steps are obtained automatically from higher level instructions, such as "how would you throw away all the items on the table?" by using the SayCan system [1], as described in detail in Section IV-D and Appendix K.

B. Can RT-1 learn to perform a large number of instructions, and to generalize to new tasks, objects and environments?

To answer our first question, we analyze the overall performance, generalization, and robustness capabilities of RT-1 compared to previously proposed models. Specifically, we compare to the model architectures used by Gato [51] and BC-Z [23], as well as a larger version of BC-Z, which we refer to as BC-Z XL. Note, however, that all models are trained on the same data as RT-1, and the evaluation only compares the model architectures, not the task sets, datasets, or overall robotic systems. The capabilities of RT-1 are determined to a large extent by the dataset and task set, which we believe

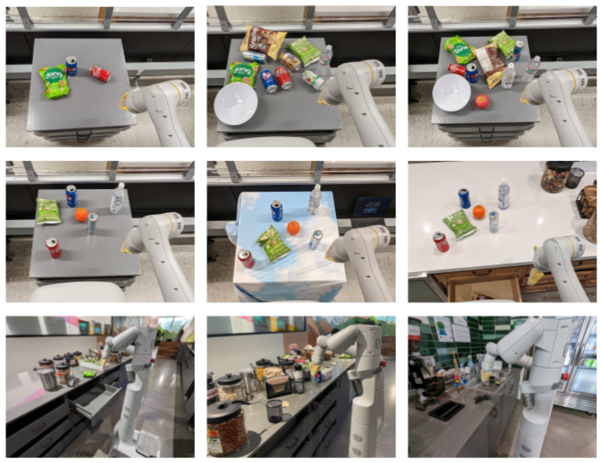


Fig. 4: Evaluation scenarios for distractors (first row), from left to right: easy (0-5 distractors), medium (9 distractors), hard (9 distractors and occluded object); background (second row), from left to right: original environment, patterned table cloth, new kitchen; and realistic scenarios in the real kitchen (third row), generalization levels from left to right: $L1$, $L2$ and $L3$.

improves significantly over prior works (e.g. BC-Z uses 100 tasks and the original Gato model trains a stacking task with various shapes), and thus this comparison should be viewed as rather favorable to the prior models, which also benefit from the large and diverse dataset and task set that we collected.

Model	Seen Tasks	Unseen Tasks	Distractors	Backgrounds
Gato [51]	65	52	43	35
BC-Z [23]	72	19	47	41
BC-Z XL	56	43	23	35
RT-1 (ours)	**97**	**76**	**83**	**59**

TABLE II: Overall performance of RT-1 and baselines across seen tasks, generalization to unseen tasks, and robustness to distractors and backgrounds.

The results are shown in Table II. Across each category, we find that RT-1 outperforms the prior models significantly. On seen tasks, RT-1 is able to perform 97% of the more than 200 instructions successfully, which is 25% more than BC-Z and 32% more than Gato. On unseen tasks, RT-1 shows it is capable of generalizing to novel instructions, performing 76% of the never-before-seen instructions, 24% more than the next best baseline. While such generalization to novel instructions is made possible due to natural language conditioning of the policy, as the policy is able to understand new combinations of previously seen concepts, all of the baselines are also conditioned on natural language and in principle enjoy the same benefits. We further ablate different components of RT-1 in the next section to better understand what aspects of our method contribute the most to this difference. On distractors and backgrounds, we find that RT-1 is quite robust, successfully executing 83% of the distractor robustness tasks and 59% of the background robustness tasks (36% and 18% higher than the next best alternative, respectively). Overall, we find that RT-1 has high general performance, while exhibiting impressive degrees of generalization and robustness. We show example trajectories of the RT-1 agent including instructions that cover different skills, environments and objects in Fig. 5. We also present additional trajectory examples for different generalization tests in the Appendix, which include backgrounds (Fig. 9), and distractors (Fig. 11).

Generalization to realistic instructions. Next, we test whether our method generalizes enough across all the different axes that we evaluated previously to be deployed in a real kitchen, which poses multiple distribution shifts all at once such as new tasks combinations, object distractors as well as a novel environment.

To evaluate our algorithm in realistic scenarios in a real kitchen, we construct task sequences to accomplish a number of realistic goals. The robot restocks several snacks in drawers, tidies up knocked over condiment bottles and closes drawers left open by humans, prepares a snack with an orange and a napkin and fetches lost sunglasses and an octopus toy from several places in the kitchen. The detailed instructions used in these scenarios are listed in the Appendix I. The office kitchen involves a dramatic shift from the training environment and we categorize tasks across these scenarios with varying levels of generalization: $L1$ for generalization to the new counter-top layout and lighting conditions, $L2$ for additionally generalization to unseen distractor objects, $L3$ for additional generalization to drastically new task settings, new task objects or objects in unseen locations such as near a sink. The three levels that correspond to the three tasks of restocking, preparing a snack and fetching a lost object in the real kitchen are depicted in the last row of Fig. 4. Example trajectories for each level are presented in the Appendix in Fig. 10.

We report the per-task success rate in these realistic scenarios along with the varying generalization levels in Table III and find RT-1 to be the most robust on all levels. Gato generalizes fairly well at the first level but it performs significantly drops for the more difficult generalization scenarios. BC-Z and its XL equivalent perform fairly well at $L2$ level and better than Gato at $L3$ but still not at the generalization level of RT-1.

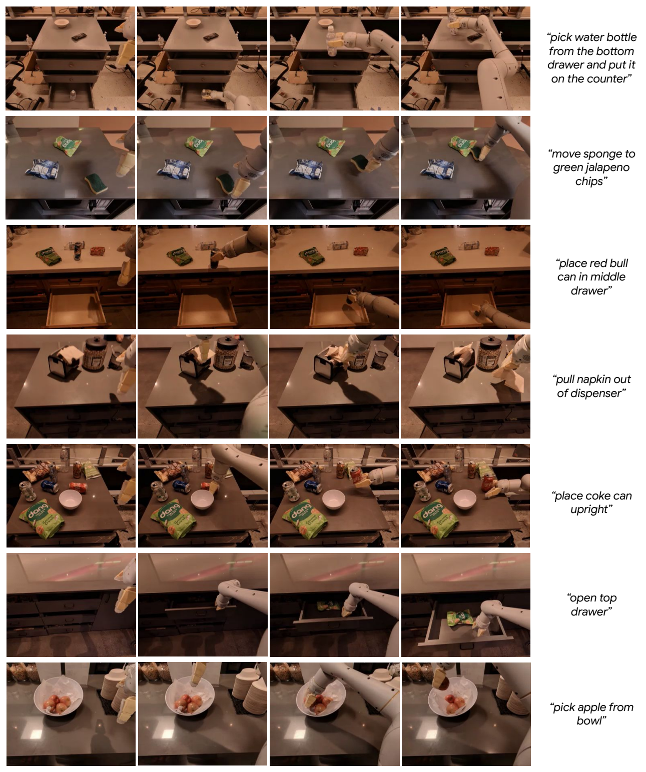

"pick water bottle from the bottom drawer and put it on the counter"
"move sponge to green jalapeno chips"
"place red bull can in middle drawer"
"pull napkin out of dispenser"
"place coke can upright"
"open top drawer"
"pick apple from bowl"

Fig. 5: Example evaluation trajectories for RT-1 across various instructions.

		Generalization Scenario Levels		
Models	All	L1	L2	L3
Gato [51]	30	63	25	0
BC-Z [23]	45	38	50	**50**
BC-Z XL	55	63	**75**	38
RT-1 (ours)	**70**	**88**	**75**	**50**

TABLE III: Realistic generalization scenarios: we compare model success rate in a realistic kitchen scenarios across three levels of generalization: $L1$ for generalization to the new counter-top layout and lighting conditions, $L2$ for additionally generalization to unseen distractor objects, $L3$ for additionally generalization to drastically new task settings, new task objects or in unseen locations like near a sink.

		Real Objects	Sim Objects (not seen in real)	
Models	Training Data	Seen Skill w/ Objects	Seen Skill w/ Objects	Unseen Skill w/ Objects
RT-1	Real Only	92	23	7
RT-1	Real + Sim	90(-2)	**87(+64)**	**33(+26)**

TABLE IV: Experimental results for incorporating simulation data in RT-1. Adding simulation data does not impact the performance on real objects, while significantly improving real performance on objects that were only introduced in simulation (+64%). It also improves real-world generalization on simulated objects used with skills seen only in the real world (+26%), e.g. "move X to Y" where X only appeared in simulated "pick X" task.

C. Can we push the resulting model further by incorporating heterogeneous data sources such as simulation or data from different robots?

Next, we explore the limits of RT-1 for utilizing highly heterogeneous data. We demonstrate how RT-1 can incorporate and learn from vastly different data sources and improve from such data without sacrificing its original-tasks performance across the varied tasks inherent in this data. To this end, we conduct two experiments: (1) RT-1 trained and tested on both real data and simulation data and (2) RT-1 trained across large datasets of different tasks, originally collected by different robots. More information on each is provided in Appendix J.

Absorbing simulation data. Table IV shows the ability of RT-1, and baselines, to absorb both real and simulation data. To test this, we take all of the real demonstration data but we also provide additional simulation data that includes objects that the robot has never seen in the real world. Specifically, we specify different generalization scenarios: for *seen skills with real objects* the training data has real data of that instruction (i.e., performance on seen tasks), for *seen skills with sim objects* the training data has sim data of that instruction (e.g. "pick up a sim object", which was present in sim), and for *unseen skills with sim objects* the training data has sim data of that object but there are no examples of the instruction describing the skill with that object either in sim or in real (e.g., "move a sim object to apple", even though the robot has only practiced in picking that sim object and not moving it near other objects). All evaluations are done in the real world but to limit the number of instructions evaluated, we focus on pick and move-to skills.

We find in Table IV that for RT-1, we do not lose performance adding simulation data compared to the Real Only dataset. We do however, see a significant increase in performance (from 23% to 87%) on objects and tasks seen only in simulation, to approximately the performance of the those in real, demonstrating an impressive degree of domain transfer. We also see a significant increase in performance on unseen instructions from 7% to 33%; impressive given the object in question has never been seen in real and the instruction never seen at all. Overall, we find that RT-1 is able to efficiently absorb new data, even from a very different domain.

Absorbing data from different robots. To push the data absorption limits of RT-1, we conduct an additional set of experiments where we combine two data sources that originate from different robots: Kuka IIWA as well as the mobile manipulators used in the experiments so far. The Kuka data contains all the successful examples collected in QT-Opt [28], which corresponds to 209k episodes, where the robot was indiscriminately grasping objects in a bin (see an example of a Kuka episode in Table. V). To test whether RT-1 can effectively absorb these two very different datasets, which we refer to as the standard "Classroom eval", as well as the performance on the newly constructed tasks that reflect the bin-picking setup present in the Kuka data, which we refer to as the "Bin-picking eval" (see Fig. 12).

We would like to emphasize the difficulty of this setting by noting the major differences between the datasets. Not only are the robots that collected the data different in appearance and action space, but also the environment they were deployed in has different appearance and dynamics. In addition the QT-Opt data presents a completely different action distribution – it was collected by an RL agent as opposed to human demonstrations present in our dataset.

The results are presented in Table V. We observe that the model that mixes the RT-1 data and the Kuka data has only a minimal decrease in the original tasks' performance (i.e. Classroom eval), i.e. 2%. Even more importantly, in the Bin-picking eval, we observe that the model trained on multi-robot data performs at 39% compared to the 22% of the model that was trained only on the RT-1 data. This is a 17% performance difference (almost 2x). Additionally, RT-1 trained on Kuka bin-picking data and evaluated on the bin-picking tasks with the mobile manipulator (MM) achieves 0% performance, confirming that it is difficult to transfer a behavior from another robot morphology. However, mixing the data from both robots allows RT-1 to infer the correct actions of the MM robot even when faced with the states observed by Kuka robots. This is achieved without explicit demonstrations of bin-picking on MM robot and by taking advantage of past experiences collected by Kuka robots. These results indicate that RT-1's absorption properties also include the ability to acquire new skills through observing other robots' experiences and present an exciting avenue of future work where we combine many more multi-robot datasets to enhance the robot capabilities.

D. How do various methods generalize long-horizon robotic scenarios?

In the next set of experiments we evaluate whether our method generalizes enough to be used in long-horizon realistic kitchen settings. To answer this question, we execute RT-1 and various baselines within the SayCan [1] framework in two different real kitchens. Since SayCan combines many low-level instructions to perform high-level instructions, the number of possible high-level instructions increases combinatorially with skills, so the skill-breadth of RT-1 can be fully seen (for more details on the SayCan algorithm please refer to [1]). The

Models	Training Data	Classroom eval	Bin-picking eval
RT-1	Kuka bin-picking data + MM data	90(-2)	**39(+17)**
RT-1	MM only data	92	22
RT-1	Kuka bin-picking only data	0	0

TABLE V: Experimental results for mixing data from two different robots. Incorporating Kuka bin-picking data from QT-Opt [28] in RT-1 minimally impacts the standard classroom evaluation performance and results in almost a 2x improvement in generalization to the Bin-picking evaluation (that is similar to the setup in the Kuka data) on the mobile manipulator (MM). This demonstrates an effective transfer across two different robot morphologies.

success rate of long-horizon tasks also decreases exponentially with the length of the task, so high success rates in manipulation skills are particularly important. Furthermore, as mobile manipulation tasks require both navigation and manipulation, the policies ability to be robust to base position is crucial. More detail is provided in Appendix K.

Table VI shows our results (on instructions in Appendix Table XI). Except for original SayCan, all methods get 87% as planning success rate, and RT-1 performs the best, with 67% execution success rate in Kitchen1. Kitchen2 constitutes a much more challenging generalization scene, since the Robot Classroom training scenes are modeled after Kitchen1 (see the pictures of the kitchens in Fig. 3). Due to this generalization difficulty, SayCan with Gato is not able to finish any long horizon task, and SayCan with BC-Z is able to achieve a success rate of 13%. The original SayCan paper did not evaluate performance in a new kitchen. Surprisingly, the manipulation performance does not see a visible drop from Kitchen1 to Kitchen2 for our method. This enables us to operate unseen drawers in Kitchen2, and that we can use SayCan-RT1 to plan and execute ultra-long horizon tasks, with as many as 50 steps.

	SayCan tasks in Kitchen1		SayCan tasks in Kitchen2	
	Planning	Execution	Planning	Execution
Original SayCan [1]*	73	47	-	-
SayCan w/ Gato [51]	87	33	87	0
SayCan w/ BC-Z [23]	87	53	87	13
SayCan w/ RT-1 (ours)	87	**67**	87	**67**

TABLE VI: SayCan style long horizon tasks in Kitchen1 and Kitchen2. (*Original SayCan eval uses a slightly different prompt so the planning success rate is lower.)

E. How do generalization metrics change with varying amounts of data quantity and data diversity?

While previous works have shown the scaling abilities of Transformer-based models [34, 51, 25] with the number of model parameters, in many robotics works the model size is often not the primary bottleneck, and the maximum size is limited by the latency requirement for running such models on real robots. Instead, in this study we focus on ablating the influence of dataset size and diversity, as they play an important role in the traditionally data-limited robot learning field. Since data collection is particularly expensive for real robots, it is important to quantify what kind of data our models need to achieve a certain performance and generalization. Thus, our last question focuses on the scaling properties of RT-1 with different data properties.

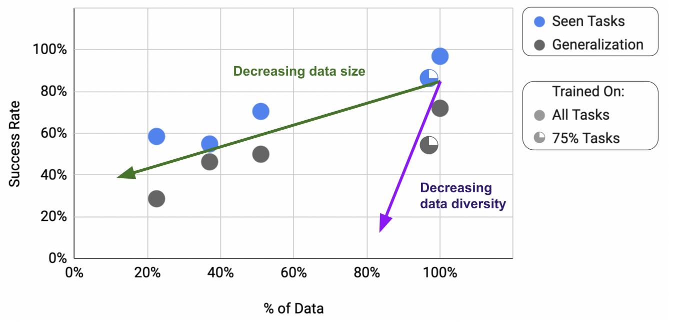


Fig. 6: Various data ablations of RT-1 across seen tasks, generalization to unseen tasks, and robustness to distractors and backgrounds. Data diversity has a higher impact on the performance and generalization than data quantity. Appendix Table XII shows the values for each ablation.

In Figure 6 we show the performance, generalization, and robustness of RT-1 as we decrease the dataset size (% data) and the dataset diversity (% tasks). To separate the axes of dataset size and diversity, we create smaller datasets with the same task diversity by removing data from the tasks with the largest data, capping the number of examples per task at 200 (resulting in 51% of the data), 100 (37% of the data), and 50 (22.5% of the data). To create a narrow dataset, we remove the tasks with the least data, thus keeping 97% of the overall data but only 75% of the tasks. As we decrease dataset size, we see a general trend of decreasing performance and a steeper trend of decreasing generalization. As we make the dataset more narrow, we see much steeper performance reductions, particularly in terms of generalization. In fact, removing 25% of the tasks while keeping 97% of the data achieves an equivalent generalization performance to reducing the dataset size by as much as 49%. Our key takeaway is thus that data diversity is more essential than data quantity.

V. Conclusions and Limitations

We presented Robotics Transformer 1, RT-1, a robot learning method that can effectively absorb large amounts of data and scales with data quantity and diversity. We trained RT-1 on a large dataset of demonstrations containing over 130k episodes collected over the course of 17 months with 13 robots. In our broad set of experiments, we demonstrated that our method that can perform over 700 instructions at 97% success rate and effectively generalize to new tasks, objects and environments better than previously published baselines. We also demonstrated that RT-1 can successfully absorb heterogeneous data from simulation and other robot morphologies without sacrificing original-tasks performance and while improving generalization to new scenarios. Lastly, we showed how this level of performance and generalization allowed us to execute very long-horizon tasks in the SayCan [1] framework, with as many as 50 steps.

While RT-1 presents a promising step towards large-scale robot learning with an data-absorbent model, it comes with a number of limitations. First, it is an imitation learning method, which inherits the challenges of that class of approaches such as the fact that it may not be able to surpass the performance of the demonstrators. Second, the generalization to new instructions is limited to the combinations of previously seen concepts and RT-1 is not yet able to generalize to a completely new motion that has not been seen before. Third, our generalization experiments are conducted in a new kitchen setting that is still similar to the training setup. More work is needed to expand the generalization capabilities of these models to generalize to much more diverse environments. Lastly, our method is presented on a large but not very dexterous set of manipulation tasks. We plan to continue extending the set of instructions that RT-1 enables to address this challenge.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[3] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022.

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

[5] Michael Jae-Yoon Chung, Abram L Friesen, Dieter Fox, Andrew N Meltzoff, and Rajesh PN Rao. A bayesian developmental approach to robotic goal-based imitation learning. *PloS one*, 10(11):e0141965, 2015.

[6] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, 2019.

[7] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 3876–3881. IEEE, 2014.

[8] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[9] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

[10] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[11] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4497–4506, 2021.

[12] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 538–547, 2019.

[13] Roy Fox, Ron Berenstein, Ion Stoica, and Ken Goldberg. Multi-task hierarchical imitation learning for home automation. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2019.

[14] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[15] Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.

[16] Josiah P Hanna, Peter Stone, and Scott Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. RetinaGAN: An object-aware approach to sim-to-real transfer, 2020. URL https://arxiv.org/abs/2011.03148.

[18] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):6289–6296, 2021.

[19] De-An Huang, Yu-Wei Chao, Chris Paxton, Xinke Deng, Li Fei-Fei, Juan Carlos Niebles, Animesh Garg, and Dieter Fox. Motion reasoning for goal-based imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4878–4884. IEEE, 2020.

[20] Ander Iriondo, Elena Lazkano, Loreto Susperregi, Julen

Urain, Ane Fernandez, and Jorge Molina. Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning. *Applied Sciences*, 9(2):348, 2019.

[21] Alexander Irpan, Kanishka Rao, Konstantinos Bousmalis, Chris Harris, Julian Ibarz, and Sergey Levine. Off-policy evaluation via off-policy classification. *Advances in Neural Information Processing Systems*, 32, 2019.

[22] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[23] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2021.

[24] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

[25] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

[26] Yuqian Jiang, Fangkai Yang, Shiqi Zhang, and Peter Stone. Integrating task-motion planning with reinforcement learning for robust decision making in mobile robots. *arXiv preprint arXiv:1811.08955*, 2018.

[27] Tom Jurgenson, Or Avner, Edward Groshev, and Aviv Tamar. Sub-goal trees a framework for goal-based reinforcement learning. In *International Conference on Machine Learning*, pages 5020–5030. PMLR, 2020.

[28] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[29] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[30] Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Ben Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. MT-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv*, 2021.

[31] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

[32] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1468–1473, 2011.

[33] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML-27th international conference on machine learning*, pages 599–606. Omnipress, 2010.

[34] Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Winnie Xu, Sergio Guadarrama, Ian Fischer, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*, 2022.

[35] Kuang-Huei Lee, Ted Xiao, Adrian Li, Paul Wohlhart, Ian Fischer, and Yao Lu. PI-QT-Opt: Predictive information improves multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2210.08217*, 2022.

[36] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[37] Chengshu Li, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[38] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[39] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[40] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4, 2006.

[41] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[42] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[43] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.

[44] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021.

[45] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32 (1), Apr. 2018. doi: 10.1609/aaai.v32i1.11671. URL

https://ojs.aaai.org/index.php/AAAI/article/view/11671.

[46] Lerrel Pinto and Abhinav 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.

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

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

[49] Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. *arXiv preprint arXiv:1901.08651*, 2019.

[50] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[51] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[52] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in Neural Information Processing Systems*, 34:12786–12797, 2021.

[53] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Ng. Robotic grasping of novel objects. *Advances in neural information processing systems*, 19, 2006.

[54] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

[55] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[56] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[57] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022.

[58] Andrew Silva, Nina Moorman, William Silva, Zulfiqar Zaidi, Nakul Gopalan, and Matthew Gombolay. Lanconlearn: Learning with language to enable generalization in multi-task manipulation. *IEEE Robotics and Automation Letters*, 7(2):1635–1642, 2021.

[59] Avi Singh, Eric Jang, Alexander Irpan, Daniel Kappler, Murtaza Dalal, Sergey Levinev, Mohi Khansari, and Chelsea Finn. Scalable multi-task imitation learning with autonomous improvement. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2167–2173. IEEE, 2020.

[60] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[61] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/tan19a.html.

[62] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1507–1514, 2011.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[64] Ulrich Viereck, Andreas Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Conference on robot learning*, pages 291–300. PMLR, 2017.

[65] Cong Wang, Qifeng Zhang, Qiyan Tian, Shuo Li, Xiaohui Wang, David Lane, Yvan Petillot, and Sen Wang. Learning mobile manipulation through deep reinforcement learning. *Sensors*, 20(3):939, 2020.

[66] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[67] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020.

[68] Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. *arXiv preprint arXiv:2004.06089*, 2020.

[69] Naoki Yokoyama, Alexander William Clegg, Eric Undersander, Sehoon Ha, Dhruv Batra, and Akshara Rai.

Adaptive skill coordination for robotic mobile manipulation. *arXiv preprint arXiv:2304.00410*, 2023.

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

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

[72] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *arXiv preprint arXiv:2106.03427*, 2021.

Appendix

- **Evaluations (ablations, designing procedures, implementations, and running ablations)**: Yevgen Chebotar, Keerthana Gopalakrishnan, Karol Hausman, Julian Ibarz, Brian Ichter, Alex Irpan, Isabel Leal, Kuang-Huei Lee, Yao Lu, Ofir Nachum, Kanishka Rao, Sumedh Sontakke, Austin Stone, Quan Vuong, Fei Xia, Ted Xiao, and Tianhe Yu.
- **Network Architecture (tokenizer, training, inference)**: Yevgen Chebotar, Keerthana Gopalakrishnan, Julian Ibarz, Alex Irpan, Kuang-Huei Lee, Yao Lu, Karl Pertsch, Kanishka Rao, Michael Ryoo, Sumedh Sontakke, Austin Stone, and Quan Vuong.
- **Developed Infrastructure (data, training, collect, simulation, evaluations, storage, and operations)**: Anthony Brohan, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Yao Lu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Tianhe Yu.
- **Leadership (managed or advised on the project)**: Chelsea Finn, Karol Hausman, Julian Ibarz, Sally Jesmonth, Sergey Levine, Yao Lu, Igor Mordatch, Carolina Parada, Kanishka Rao, Pannag Sanketi, Vincent Vanhoucke.
- **Paper (figures, vizualizations, writing)**: Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Sergey Levine, Ofir Nachum, Karl Pertsch, Kanishka Rao, Austin Stone, Fei Xia, and Ted Xiao.
- **Data collection and evaluations**: Noah Brown, Justice Carbajal, Joseph Dabis, Tomas Jackson, Utsav Malla, Deeksha Manjunath, Jodily Peralta, Emily Perez, Jornell Quiambao, Grecia Salazar, Kevin Sayed, Jaspiar Singh, Clayton Tan, Huong Tran, Steve Vega, and Brianna Zitkovich.

In the interest of rigor and reproducibility, we include extensive implementation details and and additional evaluations in the appendix. While these details are not essential for understanding the core ideas behind our approach, since RT-1 is situated in the context of a large and relatively complex robotic system, reproducibility and a complete evaluation requires additional discussion that is difficult to fit into the main paper.

A. Preliminaries

Robot learning. We aim to learn robot policies to solve language-conditioned tasks from vision. Formally, we consider a sequential decision-making environment. At timestep $t = 0$, the policy π is presented with a language instruction i and an initial image observation x_0. The policy produces an action distribution $\pi(\cdot \mid i, x_0)$ from which an action a_0 is sampled and applied to the robot. This process continues, with the policy iteratively producing actions a_t by sampling from a learned distribution $\pi(\cdot \mid i, \{x_j\}_{j=0}^{t})$ and applying those actions to the robot. The interaction ends when a termination condition is achieved. The full interaction $i, \{(x_j, a_j)\}_{j=0}^{T}$ from the starting step $t = 0$ to terminating step T is referred to as an *episode*. At the end of an episode, the agent will be given a binary reward $r \in \{0, 1\}$ indicating whether the robot performed the instruction i. The goal is to learn a policy π that maximizes the average reward, in expectation over a distribution of instructions, starting states x_0, and transition dynamics.

Transformers. RT-1 uses a Transformer [63] to parameterize the policy π. Generally speaking, a Transformer is a sequence model mapping an input sequence $\{\xi_h\}_{h=0}^{H}$ to an output sequence $\{y_k\}_{k=0}^{K}$ using combinations of self-attention layers and fully-connected neural networks. While Transformers were originally designed for text sequences, where each input ξ_j and output y_k represents a text token, they have been extended to images [43] as well as other modalities [34, 51]. As detailed in the next section, we parameterize π by first mapping inputs $i, \{x_j\}_{j=0}^{t}$ to a sequence $\{\xi_h\}_{h=0}^{H}$ and action outputs a_t to a sequence $\{y_k\}_{k=0}^{K}$ before using a Transformer to learn the mapping $\{\xi_h\}_{h=0}^{H} \to \{y_k\}_{k=0}^{K}$.

Imitation learning. Imitation learning methods train the policy π on a dataset $\mathcal{D}$ of demonstrations [47, 71, 23]. Specifically, we assume access to a dataset $\mathcal{D} = \{(i^{(n)}, \{(x_t^{(n)}, a_t^{(n)})\}_{t=0}^{T^{(n)}})\}_{n=0}^{N}$ of episodes, all of which are successful (i.e., have a final reward of 1). We learn π using *behavioral cloning* [47], which optimizes π by minimizing the negative log-likelihood of actions a_t given the images and language instructions.

B. Model inference

In addition to the inference speed requirement, we need to ensure that our system outputs actions at a consistent frequency, avoiding jitter. To accomplish this, we introduce a fixed-time waiting mechanism that waits a certain amount of time (280ms, the max observed latency of all components) after the state, that was used to compute the next action, has been captured, but before applying the action, similarly to the procedure described by Xiao et al. [68].

C. Low-level control details.

We run position controllers tracking the 6D Cartesian tool pose, 1D gripper aperture and 2D position and orientation of the mobile base. Tool and gripper controllers are run simultaneously in a non-blocking mode, meaning the policy can update the targets before they are achieved by the controllers. The base controller is run in blocking mode, meaning if the policy moves the base, the controller reaches the target before it hands execution back to the policy.

Next, we describe how targets are converted to trajectories. Cartesian tool poses are converted to straight line moves from the current to the target pose. We use inverse kinematics to convert cartesian trajectories to joint space and track them with high-gain position control. We check the joint trajectories for constraints such as joint limits and self-collision. We execute trajectories only up to and excluding the first violation of a constraint. This allows us to command targets that are not necessarily feasible, which results in an improved computational efficiency.

Base trajectories are executed by a non-holonomic base from the current to the target base pose. We check constraint satisfaction of the trajectory and execute up until and excluding the first constraint violation.

D. Data details.

We specify details of the dataset used together with example instructions in Table XI. We also provide pictures of the entire setup including the environments and objects used as shown in Fig. 3.

E. Data collection at scale.

Each of the robots autonomously approaches its station at the beginning of the episode and communicates to the operator the instruction that they should demonstrate to the robot. To ensure a balanced dataset as well as randomization of the scene, we created a software module responsible for sampling the instructions to be demonstrated as well as the randomization of the background configuration. Each of the robots tells the demonstrator how to randomize the scene and which instruction to demonstrate.

Demonstrations are collected with direct line-of-sight between operator and robot using 2 virtual reality remotes. We map remote controls onto our policy action space to preserve consistency of the transition-dynamics. 3D position and rotational displacements of the remote are mapped to 6d displacements of the robot tool. The x, y position of the joystick is mapped to a turning angle and driving distance of the mobile base. We compute and track trajectories to the target poses that we obtain from the joystick commands.

F. Model Selection at Scale

As robot learning systems become more capable and the number of instructions they can handle increases, evaluation of these models becomes difficult [29, 23]. This is an important consideration not only for evaluating different model classes and data distributions during the development process, but also for selecting the most performant model checkpoints for a particular training run. While there have been a number of proposed solutions to this problem [9, 21, 16], mostly known in the offline reinforcement learning literature as "off-policy evaluation", it still remains an open research challenge to evaluate multi-task robot learning systems at scale.

In this work, we propose leveraging simulation for "real to sim" transfer as a scalable tool that provides an approximate estimate of model performance during training across many real tasks. We run policies trained from real data in a simulator to test the full rollout performance. Note that all of our training data comes from the real world (except the experiment in Section IV-C), and the simulator is used only for model selection. To accomplish this, we expand the simulation environment proposed by Lee et al. [35] to support 551 of the tasks described in Section III-C. For each of these tasks, we define a set of scene setup randomizations, robot pose randomizations, and success detection criteria. To bridge the visual distribution shift between the real world and the simulation, we train a RetinaGAN [17] model that transforms simulated images into realistic looking images. Then, we deploy policies trained on real data directly into these simulation environments by applying RetinaGAN visual transformations at each timestep and measuring rollout simulated task success rates.

While models trained only on real world data perform better in the real world than they do in simulation, we find that the simulation success rates of high-performing real world policies are higher than the simulation success rates of low-performing real world policies. In other words, the *ordering* of simulation policy success rates are informative for predicting the ordering of real world policy success rates. We note that in this real-to-sim evaluation setting, we have a less strict requirement for simulation accuracy compared to sim-to-real settings; as long as simulation success rates are directionally correlated with real success rates, we can accept a moderate or even high gap between real and simulation success rates.

We present example camera images from simulation as well as their RetinaGAN-based transformations in Fig. 7.

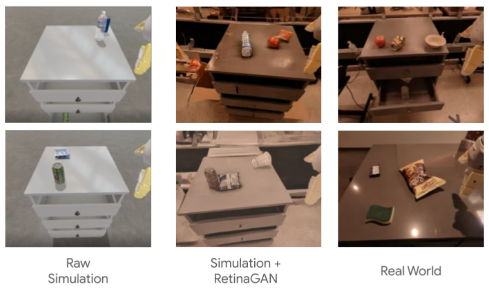


Fig. 7: Example camera images showcasing raw simulation, simulation with RetinaGAN applied, and the real world.

G. Data collection process

Figure 8 shows the growth of data, number of tasks, and the success rate of the policy over time. The number of tasks/instructions that our system is capable of grows over time as more data is collected. The same is true with the performance of seen tasks. One of the important aspects of the future work is develop techniques that allow us to grow the data as well as the robots performance and general capabilities at a faster rate.

H. Baselines

Gato is, similarly to RT-1, based on a Transformer architecture, but varies from RT-1 in multiple aspects. First, it computes image tokens without the notion of language and each image token embedding is computed separately for each image patch, as opposed to early language fusion and global image embedding in our model. Second, it does not use a

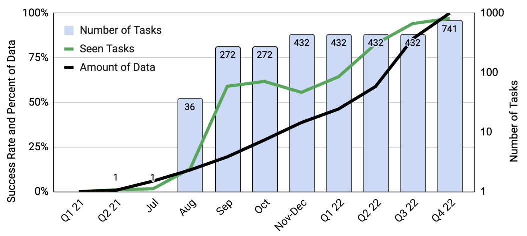


Fig. 8: The growth of data, number of tasks, and seen instruction performance over time.

pre-trained text embedding to encode the language string. It also does not include inference time considerations that are necessary for real robots as discussed in Sec. III-B such as TokenLearner and the removal of auto-regressive actions. In order to run Gato on real robots at a high enough frequency, we also limit the size of the model compared to the original publication, which was 1.2B parameters (resulting in on robot inference time of 1.9s), to be of similar size to RT-1 (37M parameters for Gato vs. 35M for RT-1). BC-Z is based on a ResNet architecture, and was used in SayCan [1]. BC-Z differs from RT-1 in that it is a feedforward model that does not use previous timesteps, and it uses continuous actions rather than discrete action tokens. In addition to the original BC-Z model size, we also compare our method to a larger version of BC-Z that has a similar number of parameters to RT-1 and refer to it as BC-Z XL. We study and analyze how each of these design decisions changes performance in Appendix Sections M and N.

I. Evaluation Details

In Section IV-B, we study the zero-shot generalization capabilities of RT-1 to difficult scenarios not present in the training dataset. To fairly evaluate different ablations of RT-1 as well as baseline policies, we design standardized evaluation procedures that cover a range of incremental difficulty levels.

Seen tasks. We evaluate on 744 tasks present in the training dataset. The breakdown between 12 skills is shown in Table I. For all "Seen" evaluations, we use the same classroom setting used for data collection as described in Section III-C. For each policy, we report a single representative metric that takes a skill-weighted average across individual skill evaluations.

Unseen tasks. We evaluate policy performance on 53 tasks that are held out during training. While the unseen instructions' specific combinations of skills and objects are not seen during training, other combinations of the same skills and objects are present in the training set. We evaluate these unseen tasks in the same environment and the same randomization procedure as the **Seen tasks**. A full list of these unseen tasks is shown in Table VII.

Distractor robustness. We test three tasks ("pick coke can", "place coke can upright", "move coke can near green rice chip bag") with incrementally more distractor objects added to the scene. The easy setting includes 0, 2, or 5 distractor objects. The medium setting includes 9 distractor objects, but the coke can is never obscured. The hard setting includes 9 distractor objects, but the scene is more crowded and the coke can is partially occluded. Both the medium are hard setting are more difficult than scenarios in the training dataset, which contained between 0 and 4 distractors. Examples of these difficulty settings and policy evaluation rollouts are shown in Figure 11.

Background robustness. We test six tasks ("pick coke can", "move blue chip bag near orange", "knock redbull can over", "pick green jalapeno chip bag", "move sponge near brown chip bag","place redbull can upright") with incrementally more challenging backgrounds and counter textures. In the easy setting, we utilize the same background environments and counter textures as the training dataset. In the medium setting, we utilize the same background environment but add a patterned tablecloth to change the counter texture. In the hard setting, we utilize a brand new kitchen environment with a new countertop; this changes the counter texture, drawer material and color, and background visuals. Examples of these difficulty settings and policy evaluation rollouts are shown in Figure 9.

Realistic instructions. To study how RT-1 performs in more realistic scenarios, we propose an evaluation setting in a real office kitchen that is a dramatic shift from the original training classroom environment. We propose a variety of skills that combine aspects of the previous zero-shot evaluations, including adding new distractors, including new backgrounds, and new combinations of objects with skills. We refer to the easiest scenario as $L1$ generalization, which introduces a new countertop and lighting condition but keeps the skills and objects the same. Next, $L2$ generalization additionally adds novel distractor objects such as kitchen jar containers. Finally, $L3$ generalization adds new objects or new locations such as near a sink. While some of these distribution shifts are tested in Section IV-B, these realistic instructions aim to test multiple dimensions simultaneously. Examples of these instructions are presented in Fig. 10.

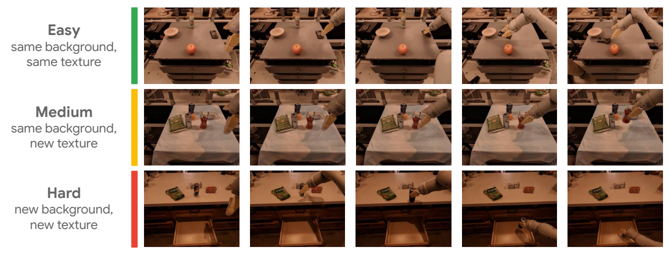

Easy
same background, same texture
Medium
same background, new texture
Hard
new background, new texture

Fig. 9: "Backgrounds" evaluations focus on testing the performance of RT-1 on settings with different table textures and different backgrounds, such as those found in kitchens never trained on. These visual differences are quite pronounced, which in the most challenging case entails a new kitchen with different counter texture, different lighting conditions, different counter material, and a different background.

Instruction
pick coke can from top drawer and place on counter
pick green can from top drawer and place on counter
pick green rice chip bag from middle drawer and place on counter
pick redbull can from top drawer and place on counter
place 7up can into bottom drawer
place brown chip bag into top drawer
place green can into middle drawer
move 7up can near redbull can
move apple near green rice chip bag
move apple near paper bowl
move apple near redbull can
move blue chip bag near blue plastic bottle
move blue chip bag near pepsi can
move blue chip bag near sponge
move brown chip bag near apple
move brown chip bag near green rice chip bag
move brown chip bag near redbull can
move coke can near green jalapeno chip bag
move coke can near water bottle
move green can near 7up can
move green can near apple
move green can near coke can
move green jalapeno chip bag near blue chip bag
move green rice chip bag near orange
move green rice chip bag near orange can
move green rice chip bag near paper bowl
move orange can near brown chip bag
move pepsi can near orange can
move redbull can near coke can
move rxbar blueberry near blue plastic bottle
move rxbar blueberry near orange can
move rxbar chocolate near paper bowl
move rxbar chocolate near rxbar blueberry
move sponge near apple
move water bottle near 7up can
move water bottle near sponge
move white bowl near orange can
pick blue plastic bottle
pick green rice chip bag
pick orange
pick rxbar chocolate
pick sponge
place pepsi can upright
knock orange can over
pick blue plastic bottle from paper bowl and place on counter
pick brown chip bag from white bowl and place on counter
pick green can from paper bowl and place on counter
pick green jalapeno chip bag from white bowl and place on counter
pick orange can from white bowl and place on counter
pick redbull can from white bowl and place on counter
place blue plastic bottle into paper bowl
place coke can into paper bowl
place orange can into paper bowl

TABLE VII: **List of Unseen Instructions in Sec. IV-B**. For the "Unseen Tasks" evaluation, we exclude a total of 53 tasks during training. While these exact instructions were not present in the training set, the objects and skills contained in these instructions were still present in the training set.

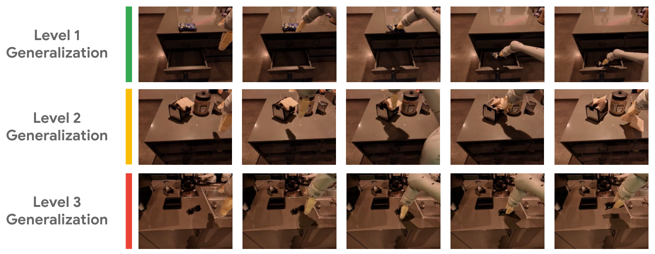


Fig. 10: “Realistic instructions” evaluations propose realistic scenarios multiple distribution shifts that incrementally increase in difficulty. $L1$ generalization introduces a new real office kitchen with new lighting conditions. $L2$ generalization additionally adds unseen distractor objects. Finally, $L3$ generalization includes new objects or objects in new locations, such as next to a sink.

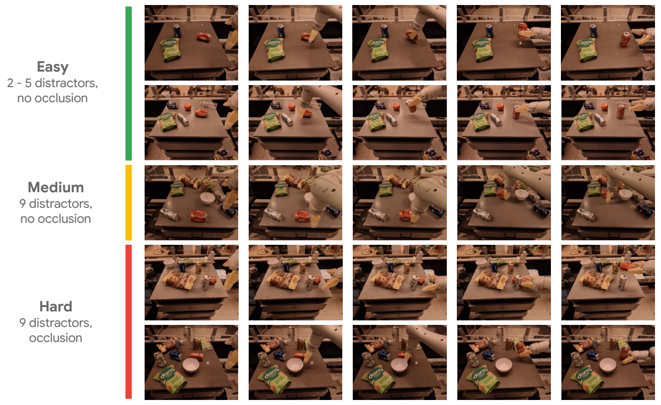


Fig. 11: “Distractors” evaluations focus on diversifying initial scene configurations well beyond the distributions contained in the training dataset, which contain between 2 and 4 distractor objects. In the most challenging scenarios, the scene is extremely cluttered and contains occlusions for the objects of interest.

J. Heterogeneous Data

We also explore the limits of RT-1 for utilizing highly heterogeneous data. We demonstrate how RT-1 can incorporate and learn from vastly different data sources and improve from such data without sacrificing its original-tasks performance across the varied tasks inherent in this data. To this end, we conduct two experiments: (1) RT-1 trained and tested on both real data and simulation data and (2) RT-1 trained across large datasets of different tasks, originally collected by different robots.

Absorbing simulation data. Table VIII shows the ability of RT-1, and baselines, to absorb both real and simulation data. To test this, we take all of the real demonstration data but we also provide additional simulation data that includes objects that the robot has never seen in the real world. We add a set of sim objects and only show them on a subset of tasks, specifically the picking tasks, in simulation. To accomplish this, we run our real2sim method described in Sec. F to bootstrap a simulation policy from the real world policy that is then trained with multi-task RL [29] with additional objects in simulation. From this process, we extract 518k successful trajectories of picking new objects and mix them with the real data that was used in the previous experiments. The goal of this experiment is to demonstrate that by expanding the dataset of simulation trajectories, we can benefit RT-1’s generalization capabilities without sacrificing the original training performance – a desired property of an absorbent model.

		Real Objects	Sim Objects (not seen in real)	
Models	Training Data	Seen Skill w/ Objects	Seen Skill w/ Objects	Unseen Skill w/ Objects
RT-1	Real Only	92	23	7
RT-1	Real + Sim	90	**87**	**33**

TABLE VIII: Experimental results for incorporating simulation data in RT-1. Adding simulation data does not impact the performance on real objects, while significantly improving real performance on objects that were only introduced in simulation.

To evaluate the properties of this model, we specify different generalization scenarios: for *seen skills with real objects* the training data has real data of that instruction (i.e., performance on seen tasks), for *seen skills with sim objects* the training data has sim data of that instruction (e.g. “pick up a sim object”, which was present in sim), and for *unseen skills with sim objects* the training data has sim data of that object but there are no examples of the instruction describing the skill with that object either in sim or in real (e.g., “move a sim object to apple”, even though the robot has only practiced in picking that sim object and not moving it near other objects). All evaluations are done in the real world but to limit the number of instructions evaluated, we focus on pick and move-to skills.

We find in Table VIII that for RT-1, we do not lose performance adding simulation data compared to the Real Only dataset. We do however, see a significant increase in performance (from 23% to 87%) on objects and tasks seen only in simulation, to approximately the performance of the those in real, demonstrating an impressive degree of domain transfer. We also see a significant increase in performance on unseen instructions from 7% to 33%; impressive given the object in question has never been seen in real and the instruction never seen at all. Overall, we find that RT-1 is able to efficiently “sponge up” new data, even from a very different domain.

Absorbing data from different robots. To push the data absorption limits of RT-1, we conduct an additional set of experiments where we combine two data sources that originate from different robots: Kuka IIWA as well as the mobile manipulators used in the experiments so far. The Kuka data contains all the successful examples collected in QT-Opt [28], which corresponds to 209k episodes, where the robot was indiscriminately grasping objects in a bin (see an example of a Kuka episode in Table. IX). Our goal in this experiment is to analyze whether the performance on the RT-1 tasks drops when adding the additional data and, more importantly, whether we can observe any transfer from data collected by a different robot morphology.

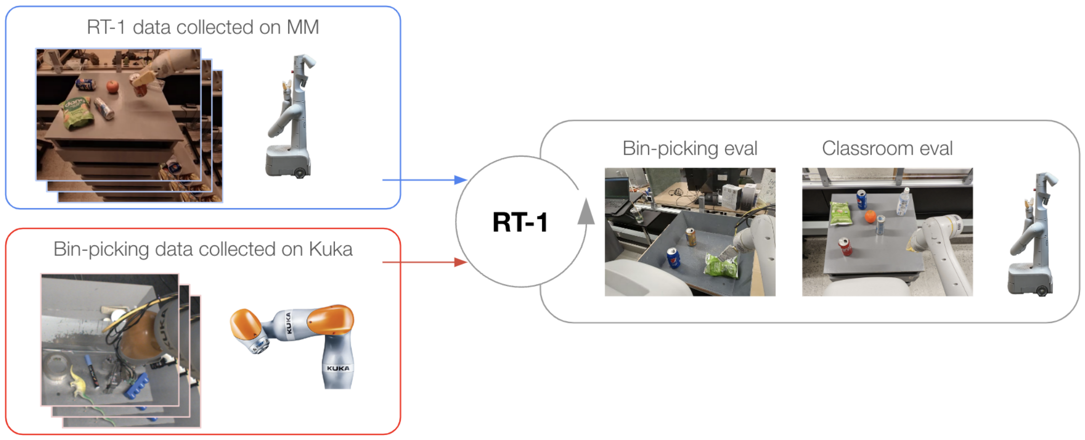


Fig. 12: In Table V, RT-1 is trained with data from two robotics platforms and learns to generalize across them.

We would like to emphasize the difficulty of this setting by noting the major differences between the datasets. Not only are the robots that collected the data different in appearance and action space, but also the environment they were deployed in has different appearance and dynamics. In addition the QT-Opt data presents a completely different action distribution – it was collected by an RL agent as opposed to human demonstrations present in our dataset.

To mix the Kuka data together with the RT-1 data, we first transform the original Kuka 4-DOF action space into the same action space as RT-1, namely we set the roll and pitch to 0, while keeping the yaw values that were present in the original Kuka data. In addition, we transform the binary *gripper-close* command into a continuous gripper-closedness command that is present in the RT-1 data. We also need text instructions corresponding to the task performed and since the Kuka data does not contain the name of the object that was grasped, we relabel all the data to the “pick anything” instruction. With these modifications, we mix both datasets with the 2:1 (RT-1 data : Kuka data) ratio and train RT-1 to obtain the final model.

To test whether RT-1 can effectively absorb these two very different datasets, we evaluate the performance on the original RT-1 tasks (in this case, we also focus on “pick” and “move to” skills), which we refer to as the standard “Classroom eval”, as well as the performance on the newly constructed tasks that reflect the bin-picking setup present in the Kuka data, which we refer to as the “Bin-picking eval”. For the Bin-picking eval to be close to the original dataset, we put in the same looking bin for the objects as well as modify the robot to be similar to the Kuka manipulators by adding extra wires and coloring the gripper gray. For all of the evaluations we use the mobile manipulator robot with the picking commands and evaluate it based on 72 grasping trials.

The results are presented in Table IX. We observe that the model that mixes the RT-1 data and the Kuka data has only a minimal decrease in the original tasks’ performance (i.e. Classroom eval), i.e. 2%. Even more importantly, in the Bin-picking eval, we observe that the model trained on multi-robot data performs at 39% compared to the 22% of the model that was trained only on the RT-1 data. This is a 17% performance difference (almost 2x). Additionally, RT-1 trained on Kuka bin-picking data and evaluated on the bin-picking tasks with the mobile manipulator (MM) robot achieves 0% performance, confirming that it is difficult to transfer a behavior from another robot morphology. However, mixing the data from both robots allows RT-1 to infer the correct actions of the MM robot even when faced with the states observed by Kuka robots. This is achieved without explicit demonstrations of bin-picking on MM robot and by taking advantage of past experiences collected by Kuka robots. These results indicate that RT-1’s absorption properties also include the ability to acquire new skills through observing other robots’ experiences and present an exciting avenue of future work where we combine many more multi-robot datasets to enhance the robot capabilities.

Models	Training Data	Classroom eval	Bin-picking eval
RT-1	Kuka bin-picking data + MM data	90	**39**
RT-1	MM only data	92	22
RT-1	Kuka bin-picking only data	0	0

TABLE IX: Experimental results for mixing data from two different robots. Incorporating Kuka bin-picking data from QT-Opt [28] in RT-1 minimally impacts the standard classroom evaluation performance and results in almost a 2x improvement in generalization to the Bin-picking evaluation (that is similar to the setup in the Kuka data) on the mobile manipulator (MM). This demonstrates an effective transfer across two different robot morphologies.

K. Long-horizon Evaluation Details

In addition to short-horizon individual skill evaluations shown in previous sections, we also evaluate how RT-1 performs in a long-horizon realistic kitchen setting that chains multiple manipulation and navigation skills to accomplish natural language instructions within the SayCan framework [1]. A list of long-horizon instructions used for these evaluations is listed in Table XI.

The success rate of long-horizon tasks decreases exponentially with the length of the task, so high success rates in manipulation skills are particularly important. Furthermore, as mobile manipulation tasks require both navigation and manipulation, the policies ability to be robust to base position is crucial. Since SayCan combines many low-level instructions to perform high-level instructions, the number of possible high-level instructions increases combinatorially with instructions, so the skill-breadth of RT-1 can be fully seen.

SayCan works by grounding language models in robotic affordances and it leverages few-shot prompting to break down a long horizon task expressed in natural language to a sequence of low level skills. An example of long horizon task would be "Bring me two different sodas", and one feasible plan would be "1. find a coke, 2. pick up the coke, 3. bring it to you, 4. put down the coke, 5. find a pepsi, 6. pick up the pepsi, 7. bring it to you, 8. put down the pepsi, 9. done." To obtain the affordance function we use value functions trained with MT-OPT [29]. For a detailed description of SayCan algorithm please refer to [1].

Since the focus of this paper is acquisition of many generalizable skills, we focus our evaluation on one subset of tasks presented in [1]. It is the long-horizon family of tasks, involving 15 instructions, each instruction requires an average of 9.6 steps to complete, and involves an average of 2.4 manipulation skills per instruction. A full list of the instructions can be found in Table XI.

We compare against 3 baselines. 1) SayCan with BC-Z, which uses SayCan planning algorithm with BC-Z as manipulation policy, 2) SayCan with Gato, which uses SayCan planning algorithm with Gato as manipulation policy, 3) Originally reported SayCan results, which use SayCan planning algorithm with BC-Z, but since it uses a slightly different prompt, the planning success rate is lower. We reimplemented 3) in 1) for a fair comparison.

As shown in Table X, except for original SayCan, all methods get 87% as planning success rate, and RT-1 performs the best, with 67% execution success rate in Kitchen1. Kitchen2 constitutes a much more challenging generalization scene, since the Robot Classroom training scenes are modeled after Kitchen1 (see the pictures of the kitchens in Fig. 3). Due to this generalization difficulty, SayCan with Gato is not able to finish any long horizon task, and SayCan with BC-Z is able to achieve a success rate of 13%. The original SayCan paper did not evaluate performance in a new kitchen. Surprisingly, the manipulation performance does not see a visible drop from Kitchen1 to Kitchen2 for our method. This enables us to operate unseen drawers in Kitchen2, and that we can use SayCan-RT1 to plan and execute ultra-long horizon tasks, with as many as 50 steps.

L. Data ablations

While previous works have shown the scaling abilities of Transformer-based models [34, 51, 25] with the number of model parameters, in many robotics works the model size is often not the primary bottleneck, and the maximum size is limited by the latency requirement for running such models on real robots. Instead, in this study we focus on ablating the influence of dataset size and diversity, as they play an important role in the traditionally data-limited robot learning field. Since data collection is particularly expensive for real robots, it is important to quantify what kind of data our models need to achieve a certain performance and generalization. Thus, our last question focuses on the scaling properties of RT-1 with different data properties.

In Table XII we show the performance, generalization, and robustness of RT-1 as we decrease the dataset size (% data) and the dataset diversity (% tasks). To separate the axes of dataset size and diversity, we create smaller datasets with the same task diversity by removing data from the tasks with the largest data, capping the number of examples per task at 200 (resulting in 51% of the data), 100 (37% of the data), and 50 (22.5% of the data). To create a narrow dataset, we remove the tasks with the least data, thus keeping 97% of the overall data but only 75% of the tasks. As we decrease dataset size, we see a general trend of decreasing performance and a steeper trend of decreasing generalization. As we make

	SayCan tasks in Kitchen1		SayCan tasks in Kitchen2	
	Planning	Execution	Planning	Execution
Original SayCan [1]*	73	47	-	-
SayCan w/ Gato [51]	87	33	87	0
SayCan w/ BC-Z [23]	87	53	87	13
SayCan w/ RT-1 (ours)	87	**67**	87	**67**

TABLE X: SayCan style long horizon tasks in Kitchen1 and Kitchen2. (*Original SayCan eval uses a slightly different prompt so the planning success rate is lower.)

Instruction
How would you put an energy bar and water bottle on the table
How would you bring me a lime soda and a bag of chips
Can you throw away the apple and bring me a coke
How would you bring me a 7up can and a tea?
How would throw away all the items on the table?
How would you move an multigrain chips to the table and an apple to the far counter?
How would you move the lime soda, the sponge, and the water bottle to the table?
How would you bring me two sodas?
How would you move three cokes to the trash can?
How would you throw away two cokes?
How would you bring me two different sodas?
How would you bring me an apple, a coke, and water bottle?
I spilled my coke on the table, how would you throw it away and then bring me something to help clean?
I just worked out, can you bring me a drink and a snack to recover?
How would you bring me a fruit, a soda, and a bag of chips for lunch

TABLE XI: **List of SayCan instructions evaluated in Sec. IV-D**

Models	% Tasks	% Data	Seen Tasks	Generalization			
				All	Unseen Tasks	Distractors	Backgrounds
Smaller Data							
RT-1 (ours)	100	100	97	73	76	83	59
RT-1	100	51	71	50	52	39	59
RT-1	100	37	55	46	57	35	47
RT-1	100	22	59	29	14	31	41
Narrower Data							
RT-1 (ours)	100	100	97	73	76	83	59
RT-1	75	97	86	54	67	42	53

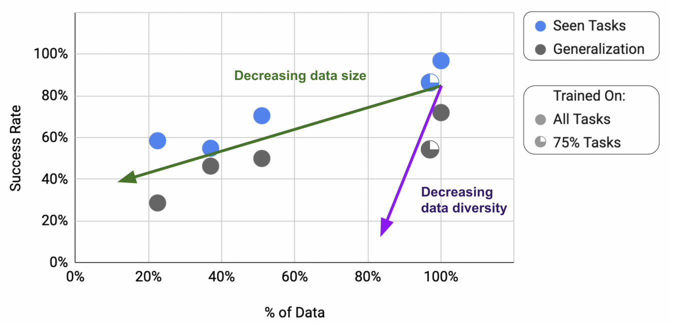


TABLE XII: Various data ablations of RT-1 across seen tasks, generalization to unseen tasks, and robustness to distractors and backgrounds. Data diversity has a higher impact on the performance and generalization than data quantity.

the dataset more narrow, we see much steeper performance reductions, particularly in terms of generalization. In fact, removing 25% of the tasks while keeping 97% of the data achieves an equivalent generalization performance to reducing the dataset size by as much as 49%. Our key takeaway is thus that data diversity is more essential than data quantity.

M. Model Ablations

What are the important and practical decisions in the design of the model and how do they affect performance and generalization?

To answer this question, we perform a set of ablations over different design decisions in RT-1. We aim to test a number of hypotheses that will help us disambiguate where the benefits of our method come from. Possible hypotheses about the source of improvement include: (i) the capacity and expressiveness of our model, which we verify by ablating the model size, trying other architectures (e.g., by removing the Transformer component); (ii) the particular action representation, which makes it easy to represent complex multi-modal action distributions, which we test by switching to continuous (normally distributed) actions, as well as by ablating the auto-regressive action representation; (iii) the ImageNet pre-trained initialization of the components, which we test by initializing the model's weights randomly; and (iv) access to the short history, which we test by excluding observation history. More concretely, we ablate our model by (1) decreasing the model size (from 35M to 21M parameters), (2) removing the Transformer architecture (using a pre-trained EfficientNet instead), (3) using a continuous instead of discrete action space (using an MSE loss and multivariate normal output), (4) auto-regressively conditioning on actions, (5) removing ImageNet pre-training of the FiLM EfficientNet, and (6) removing history (reducing the sequence of six images as input to a single image). For each ablation we compare on the axes of performance on seen tasks, performance on unseen tasks, as well as inference speed and robustness to distractors and backgrounds (with a more detailed description of each category in Section IV-A and Appendix I).

Table XIII shows the results of each ablation and the delta performance compared to the full RT-1. RT-1 achieves impressive performance on tasks and new environments, and particularly outperforms baselines on the most challenging robustness problems. We also find that each design decision is important, though at varying levels. We first evaluate a model that replaces the per-dimension discretized action representation in our model with a more standard continuous Gaussian distribution. We observe a significant decline in performance from this modification. The per-dimension discretization allows our model to represent complex multi-modal distributions, while the Gaussian distribution captures only a single mode. These results suggest that this standard and popular choice is highly suboptimal with the more complex and diverse demonstration data used by our system. ImageNet pre-training is particularly important for model generalization and robustness, decreasing the unseen task performance rate by 33%, as a result of the large and diverse visuals of the ImageNet dataset. Adding history has an impact primarily on generalization to distractors, while removing the Transformer component has a uniform but small negative impact across the seen tasks, unseen tasks and distractors. In order to keep the ImageNet pre-training while reducing the model size, we reduce the number of parameters only by 40% (from 31M to 25M). Resulting performance drops across training and generalization tasks but not as much as in other ablations. Finally, autoregressively conditioning on actions, as used in [51, 4, 34], did not benefit performance and slowed inference by more than 2x.

As described in Sec. III-B, in order to run large Transformer models on real robots, we require a model that supports fast inference for real-time operation. Note that in order to achieve our target control rate of 3Hz (described in Sec. III-B), we also need to consider other sources of latency in the pipeline, such as the camera latency and communication overhead. However, these factors will be constant for all the models, and therefore we focus our evaluation on just the network inference time. The last column of Table XIII shows the inference speed of all the models. RT-1 is almost an order of magnitude faster than Gato with a similar number of parameters, but it is also considerably slower than a ResNet-based BC-Z. In terms of the different ablations of our model, we observe that the biggest slow-down is caused by including auto-regressive actions (∼2x slow-down), and since this does not significantly influence the performance, the final version of RT-1 does not generate actions auto-regressively.

			Distractors				Backgrounds	
Model	Seen Tasks	Unseen Tasks	All	Easy	Medium	Hard	All	Inference Time (ms)
Gato [51]	65 (-32)	52 (-24)	43 (-40)	71	44	29	35 (-24)	129
BC-Z [23]	72 (-25)	19 (-57)	47 (-36)	100	67	7	41 (-18)	5.3
BC-Z XL	56 (-41)	43 (-33)	23 (-60)	57	33	0	35 (-24)	5.9
RT-1 (ours)	**97**	**76**	**83**	100	100	64	**59**	15
RT-1 w/o big model	89 (-8)	62 (-14)	77 (-6)	100	100	50	53 (-6)	13.5
RT-1 w/o pre-training	84 (-13)	43 (-33)	60 (-23)	100	67	36	41 (-18)	15
RT-1 w/ continuous actions	68 (-29)	43 (-33)	37 (-46)	71	67	0	35 (-24)	16
RT-1 w/ auto-regressive actions	85 (-12)	71 (-5)	67 (-16)	100	78	43	**65 (+6)**	36
RT-1 w/o history	82 (-15)	62 (-14)	50 (-33)	71	89	14	**59 (+0)**	15
RT-1 w/o Transformer	86 (-13)	62 (-14)	67 (-16)	100	100	29	**59 (+0)**	26

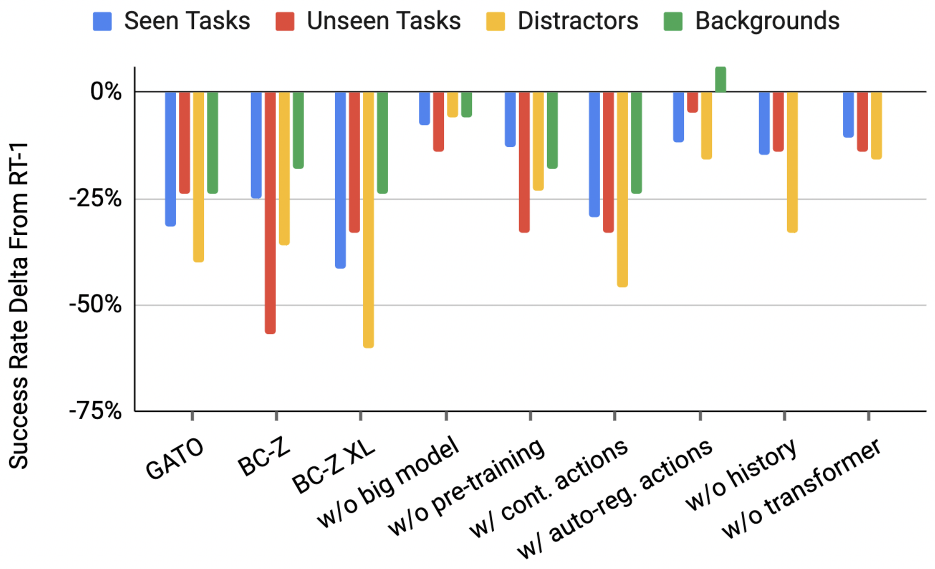


TABLE XIII: Various model ablations of RT-1 across seen tasks, generalization to unseen tasks, and robustness to distractors and backgrounds.

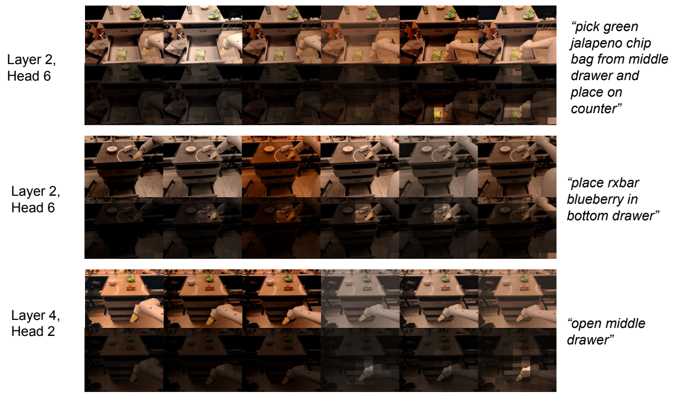

Layer 2, Head 6
"pick green jalapeno chip bag from middle drawer and place on counter"
Layer 2, Head 6
"place rxbar blueberry in bottom drawer"
Layer 4, Head 2
"open middle drawer"

Fig. 13: In this figure we show the attention map of the RT-1 policy. Different layers and heads generally focus on different part of the image. Most commonly, they focus on the parts of the scene with the richest interaction affordances, such as graspable objets. For example, Layer 2 Head 6 focuses on the jalapeno chips and pepsi can in grasping tasks; and Layer 4 Head 2 focuses on the drawer in drawer opening tasks.

N. Summary and Analysis

In this section, we summarize some of our findings and propose intuition for RT-1's high performance, generalization, and robustness. First, ImageNet pretraining (along with Universal Sentence Encoder language embedding) has a large impact particularly on unseen tasks. We observe that RT-1 inherits some of the knowledge that results from the generality and diversity of the datasets these models were trained on. Second, continuous actions have a large impact across all aspects of performance. This has been previously observed and may be due to the ability to represent more complex action distributions – the per-dimension discretization allows our model to represent complex multi-modal distributions, while the Gaussian distribution captures only a single mode. Third, given such expressive multitask models, data diversity has a larger impact than data size. Indeed, even datasets collected in simulated environments or from different robotic embodiments can be leveraged by RT-1, opening avenues for new regimes of data collection.

Finally, RT-1 fuses language into the image pipeline early via FiLM conditioning, compared to e.g., Gato's late fusion. This enables image tokens that focus only on relevant features for the instruction at hand, which may be the cause of poor distractor performance for Gato. Figure 13 visualizes the attention during rollouts of RT-1. We see that the attention is focused on relevant features and particularly on interaction between the gripper and the object of interest. The bottleneck of attention layers such as these results in a compact representation which effectively ignores distractors and varying backgrounds.