

OpenVLA:

An Open-Source Vision-Language-Action Model

Moo Jin Kim^{*,1} Karl Pertsch^{*,1,2} Siddharth Karamcheti^{*,1,3}
 Ted Xiao⁴ Ashwin Balakrishna³ Suraj Nair³ Rafael Rafailov¹ Ethan Foster¹ Grace Lam
 Pannag Sanketi⁴ Quan Vuong^{5,†} Thomas Kollar³ Benjamin Burchfiel³ Russ Tedrake^{3,6} Dorsa Sadigh¹
 Sergey Levine² Percy Liang¹ Chelsea Finn¹

<https://openvla.github.io>

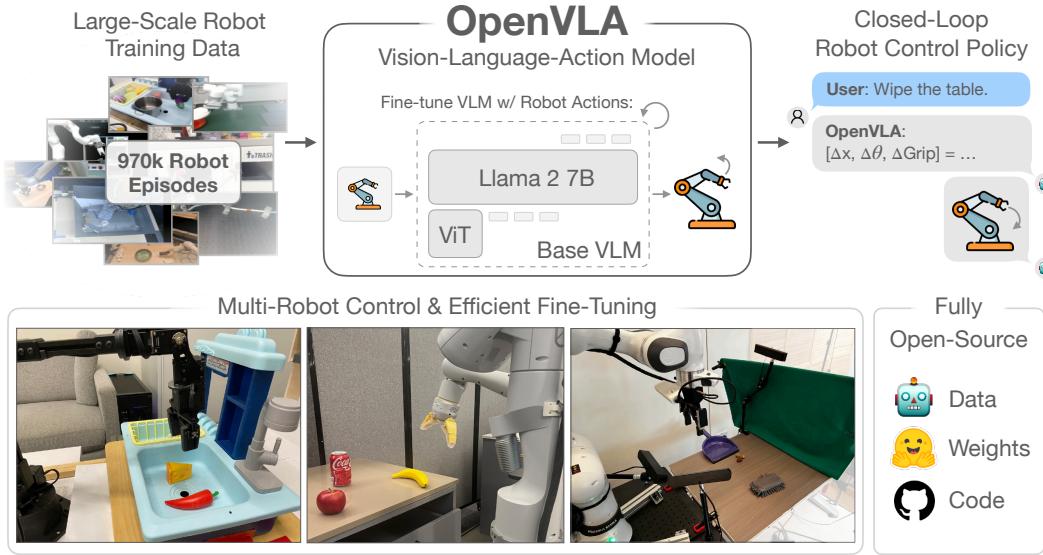


Figure 1: We present OpenVLA, a 7B-parameter open-source vision-language-action model (VLA), trained on 970k robot episodes from the Open X-Embodiment dataset [1]. OpenVLA sets a new state of the art for generalist robot manipulation policies. It supports controlling multiple robots out of the box and can be quickly adapted to new robot domains via parameter-efficient fine-tuning. The OpenVLA checkpoints and PyTorch training pipeline are fully open-source and models can be downloaded and fine-tuned from HuggingFace.

Abstract: Large policies pretrained on a combination of Internet-scale vision-language data and diverse robot demonstrations have the potential to change how we teach robots new skills: rather than training new behaviors from scratch, we can fine-tune such vision-language-action (VLA) models to obtain robust, generalizable policies for visuomotor control. Yet, widespread adoption of VLAs for robotics has been challenging as 1) existing VLAs are largely closed and inaccessible to the public, and 2) prior work fails to explore methods for efficiently fine-tuning VLAs for new tasks, a key component for adoption. Addressing these challenges, we introduce OpenVLA, a 7B-parameter open-source VLA trained on a diverse collection of 970k real-world robot demonstrations. OpenVLA builds on a Llama 2 language model combined with a visual encoder that fuses pretrained features from DINOv2 and SigLIP. As a product of the added data diversity and new model components, OpenVLA demonstrates strong results for generalist manipulation, outperforming closed models such as RT-2-X (55B) by 16.5% in absolute task success rate across 29 tasks and multiple robot embodiments, with 7x fewer parameters. We further show that we can effectively fine-tune OpenVLA for new settings, with especially

^{*}: denotes equal contribution

Correspondence to: moojink@stanford.edu, pertsch@berkeley.edu, skaramcheti@stanford.edu

¹Stanford University, ²UC Berkeley, ³Toyota Research Institute, ⁴Google Deepmind, ⁵Physical Intelligence,

⁶MIT, [†]Work done in part while at Google Deepmind

strong generalization results in multi-task environments involving multiple objects and strong language grounding abilities, and outperform expressive from-scratch imitation learning methods such as Diffusion Policy by 20.4% We also explore compute efficiency; as a separate contribution, we show that OpenVLA can be fine-tuned on consumer GPUs via modern low-rank adaptation methods and served efficiently via quantization without a hit to downstream success rate. Finally, we release model checkpoints, fine-tuning notebooks, and our PyTorch codebase with built-in support for training VLAs at scale on Open X-Embodiment datasets.

1 Introduction

A key weakness of learned policies for robotic manipulation is their inability to generalize beyond their training data: while existing policies trained for individual skills or language instructions have the capacity to extrapolate behaviors to new initial conditions such as object positions or lighting [2, 3], they lack robustness to scene distractors or novel objects [4, 5] and struggle to execute unseen task instructions [6, 7]. Yet beyond robotics, existing foundation models for vision and language such as CLIP [8], SigLIP [9], and Llama 2 [10] are capable of these types of generalization and more, stemming from the priors captured by their Internet-scale pretraining datasets. While reproducing this scale of pretraining for robotics is still an open challenge — even the largest robot manipulation datasets [1, 11] only have 100K to 1M examples — this imbalance suggests an opportunity: using existing foundation models for vision and language *as a core building block* for training robotic policies that can generalize to objects, scenes, and tasks beyond their training data.

Towards this goal, existing work has explored integrating pretrained language and vision-language models for robotic representation learning [12–14] and as a component in modular systems for task planning and execution [15, 16]. More recently, they have been used for directly learning vision-language-action models [VLAs; 1, 7, 17, 18] for control. VLAs provide a direct instantiation of using pretrained vision-and-language foundation models for robotics, directly fine-tuning visually-conditioned language models (VLMs) such as PaLI [19, 20] to generate robot control actions. By building off of strong foundation models trained on Internet-scale data, VLAs such as RT-2 [7] demonstrate impressive robustness results, as well as an ability to generalize to novel objects and tasks, setting a new standard for generalist robot policies. Yet, there are two key reasons preventing the widespread use of existing VLAs: 1) current models [1, 7, 17, 18] are *closed*, with limited visibility into model architecture, training procedures, and data mixture, and 2) existing works do not provide best practices for *deploying and adapting VLAs* to new robots, environments, and tasks — especially on commodity hardware (e.g., consumer-grade GPUs). We argue that to develop a rich foundation for future research and development, robotics needs open-source, generalist VLAs that support effective fine-tuning and adaptation, akin to the existing ecosystem around open-source language models [21–24].

To this end, we introduce OpenVLA, a 7B-parameter open-source VLA that establishes a new state of the art for generalist robot manipulation policies.¹ OpenVLA consists of a pretrained visually-conditioned language model backbone that captures visual features at multiple granularities, fine-tuned on a large, diverse dataset of 970k robot manipulation trajectories from the Open-X Embodiment [1] dataset — a dataset that spans a wide range of robot embodiments, tasks, and scenes. As a product of increased data diversity and new model components, OpenVLA outperforms the 55B-parameter RT-2-X model [1, 7], the prior state-of-the-art VLA, by 16.5% absolute success rate across 29 evaluation tasks on the WidowX and Google Robot embodiments. We additionally investigate *efficient fine-tuning strategies for VLAs*, a new contribution not explored in prior work, across 7 diverse manipulation tasks spanning behaviors from object pick-and-place to cleaning a table. We find that fine-tuned OpenVLA policies clearly outperform fine-tuned pretrained policies such as Octo [5]. Compared to from-scratch imitation learning with diffusion policies [3], fine-tuned OpenVLA shows substantial improvement on tasks involving grounding language to behavior in

¹OpenVLA uses multiple pretrained model components: SigLIP [9] and DinoV2 [25] vision encoders and a Llama 2 [10] language model backbone. For all three models, weights are open, but not their training data or code. We release training data, code and model weights for reproducing OpenVLA on top of these components.

multi-task settings with multiple objects. Following these results, we are the first to demonstrate the effectiveness of compute-efficient fine-tuning methods leveraging low-rank adaptation [LoRA; 26] and model quantization [27] to facilitate adapting OpenVLA models on consumer-grade GPUs instead of large server nodes without compromising performance. As a final contribution, we open-source all models, deployment and fine-tuning notebooks, and the OpenVLA codebase for training VLAs at scale, with the hope that these resources enable future work exploring and adapting VLAs for robotics.

2 Related Work

Visually-Conditioned Language Models Visually-conditioned language models (VLMs), which are trained on Internet-scale data to generate natural language from input image(s) and language prompts, have been adopted for myriad applications from visual question answering [28–31] to object localization [32, 33]. One of the key advances fueling recent VLMs are model architectures that bridge features from pretrained vision encoders [8, 9, 25] with pretrained language models [10, 23, 34–36], directly building on advances in both computer vision and natural language modelling to create powerful multimodal models. While early work explored various architectures for cross-attending between vision and language features [37–41], new open-source VLMs [20, 42–44] have converged on a simpler “patch-as-token” approach, in which patch features from pretrained visual transformers are treated as tokens, and are then projected into the input space of a language model. This simplicity makes it easy to repurpose existing tools for training language models at scale for VLM training. We employ these tools in our work to scale VLA training, and specifically use VLMs from Karamcheti et al. [44] as our pretrained backbone, as they are trained from *multi-resolution visual features*, fusing low-level spatial information from DINOv2 [25] with higher-level semantics from SigLIP [9] to aid in visual generalization.

Generalist Robot Policies A recent trend in robotics works towards training multi-task “generalist” robot policies [2, 6, 45–49] on large diverse robot datasets [1, 2, 6, 11, 45, 49–56], spanning many different robot embodiments [1, 5, 53, 57–66]. Notably, Octo [5] trains a generalist policy that can control multiple robots out-of-the-box and allows for flexible fine-tuning to new robot setups. A key difference between these approaches and OpenVLA is the model architecture. Prior works like Octo typically compose pretrained components such as language embeddings or visual encoders with additional model components initialized from scratch [2, 5, 6], learning to “stitch” them together during the course of policy training. Unlike these works, OpenVLA adopts a more end-to-end approach, directly fine-tuning VLMs to generate robot actions by treating them as tokens in the language model vocabulary. Our experimental evaluation shows that this simple yet scalable pipeline substantially boosts performance and generalization ability over prior generalist policies.

Vision-Language-Action Models A number of works have explored the use of VLMs for robotics, e.g., for visual state representations [12, 13], object detection [67], high-level planning [16], and for providing a feedback signal [68–71]. Others integrate VLMs directly into end-to-end visuomotor manipulation policies [14, 15], but incorporate significant structure into the policy architecture or require calibrated cameras, which limits their applicability. A number of recent works have explored similar recipes to ours and directly fine-tuned large pretrained VLMs for predicting robot actions [1, 7, 17, 18, 72–74]. Such models are often referred to as vision-language-action models (VLAs), since they fuse robot control actions directly into VLM backbones. This has three key benefits: (1) it performs alignment of pretrained vision and language components on a large, Internet-scale vision-language dataset, (2) the use of a generic architecture, not custom-made for robot control, allows us to leverage the scalable infrastructure underlying modern VLM training [75–77] and scale to training billion-parameter policies with minimal code modifications, and (3) it provides a direct pathway for robotics to benefit from the rapid improvements in VLMs. Existing works on VLAs either focus on training and evaluating in single robot or simulated setups [72–74, 78] and thus lack generality, or are closed and do not support efficient fine-tuning to new robot setups [1, 7, 17, 18]. Most closely related, RT-2-X [1] trains a 55B-parameter VLA policy on the Open X-Embodiment dataset and demonstrates state-of-the-art generalist manipulation policy performance. However, our work differs from RT-2-X in multiple important aspects: (1) by combining a strong open VLM

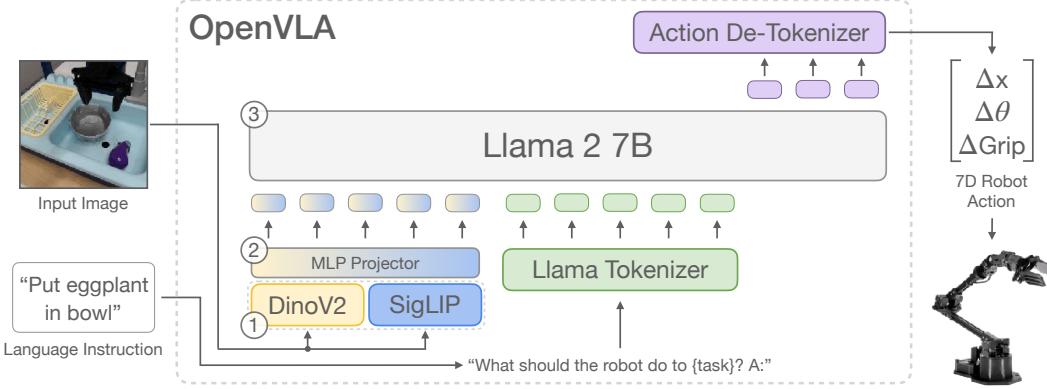


Figure 2: **OpenVLA model architecture.** Given an image observation and a language instruction, the model predicts 7-dimensional robot control actions. The architecture consists of three key components: (1) a **vision encoder** that concatenates Dino V2 [25] and SigLIP [79] features, (2) a **projector** that maps visual features to the language embedding space, and (3) the **LLM backbone**, a Llama 2 7B-parameter large language model [10].

backbone with a richer robot pretraining dataset, OpenVLA outperforms RT-2-X in our experiments while being an order of magnitude smaller; (2) we thoroughly investigate fine-tuning of OpenVLA models to new target setups, while RT-2-X does not investigate the fine-tuning setting; (3) we are the first to demonstrate the effectiveness of modern parameter-efficient fine-tuning and quantization approaches for VLAs; and (4) OpenVLA is the first generalist VLA that is open-source and thus supports future research on VLA training, data mixtures, objectives, and inference.

3 The OpenVLA Model

We introduce the OpenVLA model, a 7B-parameter vision-language-action model (VLA) trained on 970k robot demonstrations from the Open X-Embodiment dataset [1]. There are many, largely unexplored, questions around best practices for developing VLA models, e.g., what are the best model backbones, datasets, and hyperparameters to use for training. Below, we detail our approach for developing OpenVLA and summarize our key learnings. Concretely, we first provide a brief overview of modern VLMs, which form the backbone of OpenVLA (Section 3.1); then describe our basic training recipe and dataset (Section 3.2 and Section 3.3); discuss key design decisions (Section 3.4); and provide details of the used infrastructure for training and inference (Section 3.5).

3.1 Preliminaries: Vision-Language Models

The architecture of most recent VLMs [20, 42–44] consists of three main parts (see Fig. 2): (1) a **visual encoder** that maps image inputs to a number of “image patch embeddings”, (2) a **projector** that takes the output embeddings of the visual encoder and maps them into the input space of a language model, and (3) a **large language model (LLM) backbone**. During VLM training, the model is trained end-to-end with a next text token prediction objective on paired or interleaved vision and language data curated from various Internet sources.

In this work, we build on the Prismatic-7B VLM [44]. Prismatic follows the same standard architecture described above, with a 600M-parameter **visual encoder**, a small 2-layer **MLP projector**, and a 7B-parameter Llama 2 **language model backbone** [10]. Notably, Prismatic uses a *two-part* visual encoder, consisting of pretrained SigLIP [79] and DinoV2 [25] models. Input image patches are passed separately through both encoders and the resulting feature vectors are concatenated channel-wise. In contrast to the more commonly used vision encoders such as CLIP- [80] or SigLIP-only encoders, the addition of DinoV2 features has been shown to be helpful for improved spatial reasoning [44], which can be particularly helpful for robot control.

SigLIP, DinoV2, and Llama 2 do not release details about their training data, which likely consists of trillions of tokens of Internet-sourced image-text, image-only, and text-only data respectively. The Prismatic VLM is fine-tuned on top of these components using the LLaVA 1.5 data mixture [43], which contains a total of approximately 1M image-text and text-only data samples from open-source datasets [29, 42, 81–83].

3.2 OpenVLA Training Procedure

To train OpenVLA, we fine-tune a pretrained Prismatic-7B VLM backbone for robot action prediction (see Fig. 2). We formulate the action prediction problem as a “vision-language” task, where an input observation image and a natural language task instruction are mapped to a string of predicted robot actions [7]. To enable the VLM’s language model backbone to predict robot actions, we represent the actions in the output space of the LLM by mapping continuous robot actions to discrete tokens used by the language model’s tokenizer. Following Brohan et al. [7], we discretize each dimension of the robot actions separately into one of 256 bins. For each action dimension, we set the bin width to uniformly divide the interval between the 1st and 99th quantile of the actions in the training data. Using quantiles instead of the min-max bounds Brohan et al. [7] used allows us to ignore outlier actions in the data that could otherwise drastically expand the discretization interval and reduce the effective granularity of our action discretization.

Using this discretization, we obtain N discrete integers $\in [0 \dots 255]$ for an N -dimensional robot action. Unfortunately, the tokenizer used by OpenVLA’s language backbone, the Llama tokenizer [10], only reserves 100 “special tokens” for tokens newly introduced during fine-tuning, which is too few for the 256 tokens of our action discretization. Instead, we again opt for simplicity and follow Brohan et al. [7]’s approach by simply overwriting the 256 *least used* tokens in the Llama tokenizer’s vocabulary (which corresponds to the last 256 tokens) with our action tokens. Once the actions are processed into a sequence of tokens, OpenVLA is trained with a standard next-token prediction objective, evaluating the cross-entropy loss on the predicted action tokens only. We discuss key design decisions for implementing this training procedure in Section 3.4. Next, we describe the robot dataset we use for OpenVLA training.

3.3 Training Data

The goal in constructing the OpenVLA training dataset is to capture a large diversity of robot embodiments, scenes, and tasks. This enables the final model to control various robots out of the box *and* admits efficient fine-tuning to new robot setups. We leverage the Open X-Embodiment dataset [1] (OpenX) as a base to curate our training dataset. The full OpenX dataset, at the time of writing, consists of more than 70 individual robot datasets, with more than 2M robot trajectories, that were pooled into a coherent and easy-to-use data format in a large community effort. To make training on this data practical, we apply multiple steps of data curation to the raw dataset.

The goals of this curation are to ensure (1) a coherent input and output space across all training datasets, and (2) a balanced mix of embodiments, tasks, and scenes in the final training mixture.² To address (1), we follow [1, 5] and restrict our training dataset to contain only manipulation datasets with at least one 3rd person camera and use single-arm end-effector control. For (2), we leverage the data mixture weights of Octo [5] for all datasets that pass the first round of filtering. Octo heuristically down-weights or removes less diverse datasets and up-weights datasets with larger task and scene diversity; see Octo Model Team et al. [5] for details.

We also experimented with incorporating a few additional datasets into our training mixture that were added to the OpenX dataset since the release of Octo, including the DROID dataset [11], although at a conservative mixture weight of 10%. In practice, we found that the action token accuracy on DROID remained low throughout training, suggesting a larger mixture weight or model may be required to fit its diversity in the future. To not jeopardize the quality of the final model, we removed DROID from the data mixture for the final third of training. We provide a complete overview of the used datasets and mixture weights in Appendix A.

3.4 OpenVLA Design Decisions

When developing the OpenVLA model, we explored various design decisions in smaller-scale experiments before starting the final model training run. Concretely, we trained and evaluated OpenVLA models on BridgeData V2 [6] for our initial experiments, instead of training on the full

²Octo [5] demonstrated training across datasets with heterogeneous sensory inputs. While very promising, we leave an investigation of VLA training across heterogeneous sensor modalities and action spaces to future work.

OpenX mixture, to increase iteration speed and reduce computational cost. We summarize key learnings from these explorations below.

VLM Backbone. Initially, we experimented with multiple VLM backbones. Apart from Prismatic [44], we tested fine-tuning IDEFICS-1 [84] and LLaVA [85] for robot action prediction. We found that LLaVA and IDEFICS-1 performed comparably on tasks with only one object in the scene, but LLaVA demonstrated stronger language grounding in tasks that involved multiple objects in the scene and required the policy to manipulate the *correct* object, i.e., the object specified in the language instruction. Concretely, LLaVA improved upon IDEFICS-1 by 35% in absolute success rate, averaged across five language grounding tasks in a BridgeData V2 sink environment. The fine-tuned Prismatic VLM policy achieved further improvements, outperforming the LLaVA policy by roughly 10% in absolute success rate across both simple single-object tasks and multi-object, language grounding tasks. We attribute this performance delta to improved spatial reasoning capabilities afforded by the fused SigLIP-DinoV2 backbones (see [Section 3.1](#)). In addition to the performance enhancements, Prismatic also provides a modular and easy-to-use codebase, so we ultimately chose it to be the backbone for the OpenVLA model.

Image Resolution. The resolution of input images has significant impact on the computational requirements of VLA training, since higher-resolution images result in more image patch tokens and thus longer context lengths that quadratically increase training compute. We compared VLAs with 224×224 px and 384×384 px inputs, but found no performance difference in our evaluations, while the latter takes 3x longer to train. We thus opt for a resolution of 224×224 px for the final OpenVLA model. Note that on many VLM benchmarks, increased resolution does improve performance [44, 86, 87], but we did not see this trend (yet) for VLAs.

Fine-Tuning Vision Encoder. Prior work on VLMs found that freezing vision encoders during VLM training typically leads to higher performance [44]. Intuitively, a frozen vision encoder may better preserve the robust features learned from its Internet-scale pretraining. However, we found fine-tuning the vision encoder during VLA training to be crucial for good VLA performance. We hypothesize that the pretrained vision backbone may not capture sufficient fine-grained spatial details about important parts of the scene to enable precise robotic control.

Training Epochs. Typical LLM or VLM training runs complete at most one or two epochs through their training dataset. In contrast, we found it important for VLA training to iterate through the training dataset significantly more times, with real robot performance continually increasing until training action token accuracy surpasses 95%. Our final training run completes 27 epochs through its training dataset.

Learning Rate. We swept the learning rate across multiple orders of magnitude for VLA training, and achieved the best results using a fixed learning rate of 2e-5 (the same learning rate used during VLM pretraining [44]). We did not find learning rate warmup to provide benefits.

3.5 Infrastructure for Training and Inference

The final OpenVLA model is trained on a cluster of 64 A100 GPUs for 14 days, or a total of 21,500 A100-hours, using a batch size of 2048. During inference, OpenVLA requires 15GB of GPU memory when loaded in bf16 precision (i.e., without quantization) and runs at approximately 6Hz on one NVIDIA RTX 4090 GPU (without compilation, speculative decoding, or other inference speed-up tricks). We can further reduce the memory footprint of OpenVLA during inference via quantization, without compromising performance in real-world robotics tasks, as shown in [Section 5.4](#). We report inference speed on various consumer- and server-grade GPUs in [Fig. 6](#). For convenience, we implement a remote VLA inference server to allow real-time remote streaming of action predictions to the robot – removing the requirement of having access to a powerful local compute device to control the robot. We release this remote inference solution as part of our open-source code release ([Section 4](#)).

4 The OpenVLA Codebase

Along with our model, we release the OpenVLA codebase, a modular PyTorch codebase for training VLA models (see <https://openvla.github.io>). It scales from fine-tuning VLAs on individual GPUs to training billion-parameter VLAs on multi-node GPU clusters, and supports modern

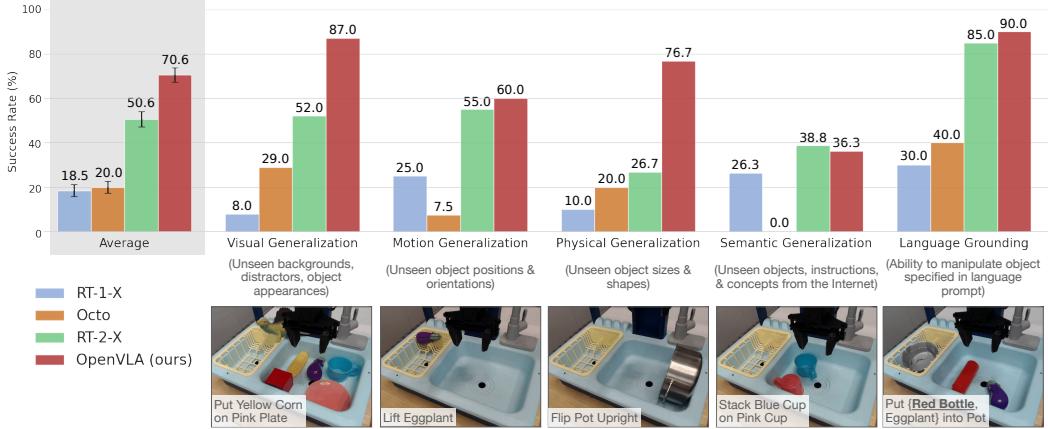


Figure 3: **BridgeData V2 WidowX robot evaluation tasks and results.** We evaluate OpenVLA and prior state-of-the-art generalist robot policies on a comprehensive suite of tasks covering several axes of generalization, as well as tasks that specifically assess language conditioning ability. OpenVLA achieves highest overall performance and even outperforms closed-source model RT-2-X in all categories except for semantic generalization. Average success rates \pm StdErr are computed across 170 total rollouts per approach. See Table 4 for detailed results.

techniques for large transformer model training such as automatic mixed precision (AMP, PyTorch [75]), FlashAttention [76], and fully sharded data parallelism (FSDP, Zhao et al. [77]). Out of the box, the OpenVLA codebase has full support for training on the Open X dataset, integrates with HuggingFace’s [21] AutoModel class, and supports LoRA fine-tuning [26] and quantized model inference [27, 88].

5 Experiments

The goal of our experimental evaluations is to test OpenVLA’s ability to serve as a powerful multi-robot control policy out of the box, as well as be a good initialization for fine-tuning to new robot tasks. Concretely, we aim to answer the following questions:

1. How does OpenVLA compare to prior generalist robot policies, when evaluating on multiple robots and various types of generalization?
2. Can OpenVLA be effectively fine-tuned on a new robot setup and task, and how does it compare to state-of-the-art data-efficient imitation learning approaches?
3. Can we use parameter-efficient fine-tuning and quantization to reduce the computational requirements for training and inference of OpenVLA models and make them more accessible? What are the performance-compute trade-offs?

5.1 Direct Evaluations on Multiple Robot Platforms

Robot Setups and Tasks. We evaluate OpenVLA’s performance “out-of-the-box” on two robot embodiments: the WidowX robot from the BridgeData V2 evaluations [6] (see Fig. 1, left) and the mobile manipulation robot from the RT-1 and RT-2 evaluations [2, 7] (“Google robot”; see Fig. 1, middle). Both platforms have been extensively used in prior works for evaluating generalist robot policies [1, 2, 5, 7]. We define a comprehensive set of evaluation tasks in each environment that covers various axes of generalization, such as **visual** (unseen backgrounds, distractor objects, colors/appearances of objects); **motion** (unseen object positions/orientations); **physical** (unseen object sizes/shapes); and **semantic** (unseen target objects, instructions, and concepts from the Internet) generalization. We also assess language conditioning ability in scenes with multiple objects, testing whether the policy can manipulate the correct target object, as specified in the user’s prompt. See bottom row of Fig. 3 and Fig. 4 for example task images in the BridgeData V2 and Google robot evaluations, respectively. Overall, we evaluated each method in 170 rollouts (17 tasks with 10 trials each) for BridgeData V2 experiments and 60 rollouts (12 tasks with 5 trials each) for Google robot experiments. A detailed breakdown of all tasks and how they differ from the training data is in

Appendix B. All evaluations in this and the following sections are conducted as A/B evaluations, using the same tasks with the same sets of initial robot and object states, to ensure fair comparison.

Comparisons. We compare OpenVLA’s performance to three prior generalist manipulation policies: **RT-1-X** [1], **RT-2-X** [1], and **Octo** [5]. **RT-1-X** (35M parameters) and **Octo** (93M parameters) are transformer policies trained from scratch on subsets of the OpenX dataset; Octo is the state-of-the-art model among open-source manipulation policies. **RT-2-X** (55B parameters) is a state-of-the-art, closed-source VLA that leverages Internet-pretrained vision and language backbones.

The results are summarized in Fig. 3 for BridgeData V2 evaluations and Fig. 4 for Google robot evaluations (per-task breakdown in Appendix, Table 4 and Table 6). We find that both RT-1-X and Octo struggle on the tested tasks, often failing to manipulate the correct object, especially when distractors are present, and in some cases causing the robot to wave its arm around aimlessly. Note that our evaluations test even larger degrees of generalization than the evaluations performed in those prior works to challenge the Internet-pretrained VLA models. Thus, lower performance of models without Internet pretraining is expected. RT-2-X clearly outperforms both RT-1-X and Octo, demonstrating the benefits of large, pretrained VLMs for robotics.

Notably, OpenVLA performs comparably to RT-2-X on Google robot evaluations and significantly outperforms RT-2-X on BridgeData V2 evaluations despite being an order of magnitude smaller (7B vs. 55B parameters). Qualitatively, we find that both RT-2-X and OpenVLA exhibit markedly more robust behaviors than the other tested models, such as approaching the correct object when distractor objects are present, properly orienting the robot’s end-effector to align with the orientation of the target object, and even recovering from mistakes such as insecurely grasping objects (see <https://openvla.github.io> for qualitative rollout examples). RT-2-X achieves higher performance in semantic generalization tasks, as shown in Fig. 3, which is expected given that it uses larger-scale Internet pretraining data and is co-fine-tuned with both robot action data and Internet pretraining data to better preserve the pretraining knowledge, rather than being fine-tuned solely on robot data, like OpenVLA. However, OpenVLA performs comparably or better in all other task categories in both BridgeData V2 and Google robot evaluations. The performance difference can be attributed to a combination of factors: we curated a much larger training dataset for OpenVLA with 970k trajectories (vs. 350k for RT-2-X); we performed more careful cleaning of the training dataset and, e.g., filtered out all-zero actions in the Bridge dataset (see Appendix C for a detailed discussion); and OpenVLA uses a fused vision encoder that combines pretrained semantic and spatial features. See Appendix D for ablation analyses of these components.

5.2 Data-Efficient Adaptation to New Robot Setups

While prior works mainly focused on directly evaluating VLAs “out-of-the-box” [1, 7, 16], effective *fine-tuning* of VLA models to new tasks and robot setups is largely unexplored, yet is key for their widespread adoption. In this section, we investigate OpenVLA’s ability to be quickly adapted to a new *real-world* robot setup. (See Appendix E for fine-tuning experiments in simulation.)

Robot setups and tasks. We test a simple fine-tuning recipe for the OpenVLA model: full fine-tuning of all model parameters, using small datasets with 10–150 demonstrations of a target task (see Fig. 5; we explore parameter-efficient fine-tuning approaches in Section 5.3). We test OpenVLA in two setups: **Franka-Tabletop**, a stationary, table-mounted Franka Emika Panda 7-DoF robot arm; and **Franka-DROID**, the Franka robot arm setup from the recently released DROID dataset [11],

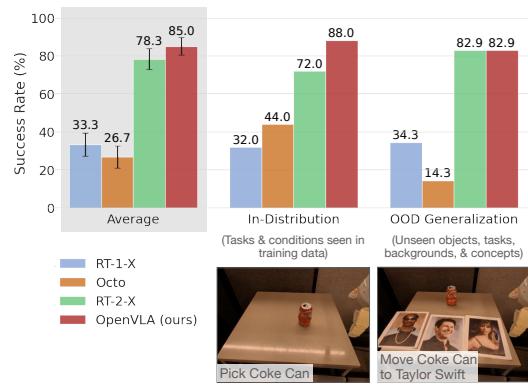


Figure 4: Google robot evaluation results. We evaluate generalist robot policies on in-distribution and out-of-distribution (OOD) tasks on the mobile manipulator used in RT-1 and RT-2 evaluations [2, 7]. We find that OpenVLA and RT-2-X attain comparable performance and significantly outperform RT-1-X and Octo overall. Average success rates \pm StdErr are computed across 60 total rollouts per approach. See Table 6 for detailed results.

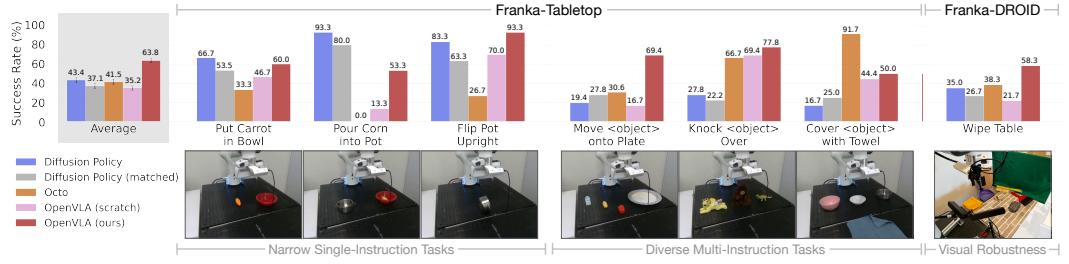


Figure 5: Adapting to new robot setups. We evaluate the state-of-the-art Diffusion Policy trained from scratch on seven Franka Emika Panda tasks (10–150 demonstrations each), as well as generalist robot policies Octo and OpenVLA fine-tuned on the same data. Diffusion Policy exhibits strong performance on narrow single-instruction tasks, while Octo and OpenVLA perform better on diverse fine-tuning tasks involving multiple instructions and distractor objects. Overall, OpenVLA achieves highest aggregate performance across both setups, suggesting that it is an effective default for learning a policy on a downstream task. Average success rates \pm StdErr are computed across 129 rollouts per approach (99 for Franka-Tabletop tasks and 30 for Franka-DROID tasks). See Table 7 for detailed results.

mounted on a movable standing desk. The setups use 5Hz and 15 Hz non-blocking controllers, respectively. We choose Franka robot arms as the target embodiment for our fine-tuning experiments since they are widely used in the robot learning community and thus a likely “target” of OpenVLA fine-tuning. We test on setups with different control frequencies to test OpenVLA’s applicability to a range of use cases.

Comparisons. We compare to **Diffusion Policy** [3], a state-of-the-art data-efficient imitation learning approach, trained from scratch. We also compare to **Diffusion Policy (matched)**, a version of Diffusion Policy that matches the input and output specifications of OpenVLA.³ Additionally, we evaluate **Octo** [5] fine-tuned on the target dataset, since it is currently the best generalist policy that supports fine-tuning (fine-tuning of RT-2-X is not supported through its inference API). We also fine-tune OpenVLA on the same target dataset, and the resulting policy is denoted by **OpenVLA**. Finally, as an ablation experiment, we compare to **OpenVLA (scratch)**, where we directly fine-tune the underlying base Prismatic VLM on the target robot setup – rather than fine-tuning the OpenX-pretrained OpenVLA model – to assess the benefit of large-scale robot pretraining.

We present the results in Fig. 5 (per-task breakdown in Appendix, Table 7). We find that both versions of Diffusion Policy are competitive with or outperform the generalist policies Octo and OpenVLA on narrower single-instruction tasks like “Put Carrot in Bowl” and “Pour Corn into Pot”, but the pretrained generalist policies perform better in more diverse fine-tuning tasks that involve multiple objects in the scene and require language conditioning. OpenX pretraining for Octo and OpenVLA enables the models to better adapt to these more diverse tasks where language grounding is important; we see evidence for this in the lower performance of OpenVLA (scratch).

Overall, we find that OpenVLA achieves the highest average performance. Notably, most prior works achieve strong performance only in *either* narrow single-instruction *or* diverse multi-instruction tasks, resulting in widely varying success rates. OpenVLA is the only approach that achieves at least 50% success rate across all tested tasks, suggesting that it can be a strong default option for imitation learning tasks, particularly if they involve a diverse set of language instructions. For narrower but highly dexterous tasks, Diffusion Policy still shows smoother and more precise trajectories; incorporating action chunking and temporal smoothing, as implemented in Diffusion Policy, may help OpenVLA attain the same level of dexterity and may be a promising direction for future work (see Section 6 for a detailed discussion of current limitations).

³The full Diffusion Policy uses a two-step observation history with both images and proprioceptive state, and performs receding horizon control by predicting a chunk of T future actions and executing the first X actions in open-loop fashion before predicting the next chunk (for 15Hz control, we set $T = 16$, $X = 8$ like in the DROID prior work [11]; for 5Hz control, we reduce the chunk sizes to $T = 8$, $X = 3$). It is also the only method in Section 5.2 that predicts *absolute* Cartesian coordinates to control the robot; all other methods use *relative* position control. Diffusion Policy (matched) uses a single image as input, has no proprioceptive information and no observation history, and predicts a single relative position control action without action chunking.

5.3 Parameter-Efficient Fine-Tuning

The full fine-tuning runs of OpenVLA in the previous section used 8 A100 GPUs for 5-15 hours per task (depending on the dataset size) to achieve high performance. While this is substantially less compute than what is required for VLA pretraining, in this section we explore even more compute- and parameter-efficient fine-tuning approaches and investigate their effectiveness.

Concretely, we compare the following fine-tuning approaches: **full fine-tuning** updates all weights during fine-tuning, as described in [Section 5.2](#); **last layer only** fine-tunes only the last layer of OpenVLA’s transformer backbone and the token embedding matrix; **frozen vision** freezes the vision encoder but fine-tunes all other weights; **sandwich fine-tuning** unfreezes the vision encoder, token embedding matrix, and last layer; and **LoRA** uses the popular low-rank adaptation technique of Hu et al. [26] with multiple rank values r , applied to all linear layers of the model.

Table 1: Parameter-efficient fine-tuning evaluation. LoRA fine-tuning achieves the best performance-compute trade-off, matching full fine-tuning performance while training only 1.4% of the model parameters. Mean success \pm StdErr computed across 33 rollouts per approach on select Franka-Tabletop tasks (see [Table 8](#) for details). *: Sharded across 2 GPUs with FSDP [77].

Strategy	Success Rate	Train Params ($\times 10^6$)	VRAM (batch 16)
Full FT	69.7 \pm 7.2 %	7,188.1	163.3 GB*
Last layer only	30.3 \pm 6.1 %	465.1	51.4 GB
Frozen vision	47.0 \pm 6.9 %	6,760.4	156.2 GB*
Sandwich	62.1 \pm 7.9 %	914.2	64.0 GB
LoRA, rank=32	68.2 \pm 7.5%	97.6	59.7 GB
rank=64	68.2 \pm 7.8%	195.2	60.5 GB

We report fine-tuning success rates across multiple Franka-Tabletop tasks, as well as training parameter count and GPU memory requirements, in [Table 1](#).⁴ We find that only fine-tuning the network’s last layer or freezing the vision encoder leads to poor performance, suggesting that further adaptation of the visual features to the target scene is crucial. In contrast, “sandwich fine-tuning” achieves better performance since it fine-tunes the vision encoder, and it consumes less GPU memory since it does not fine-tune the full LLM backbone. Lastly, LoRA achieves the best trade-off between performance and training memory consumption, outperforming “sandwich fine-tuning” and matching full fine-tuning performance while fine-tuning only 1.4% of the parameters. We find that the LoRA rank has negligible effect on policy performance and thus recommend using a default rank of $r = 32$. With LoRA, we can fine-tune OpenVLA on a new task within 10-15 hours on a *single* A100 GPU – an 8x reduction in compute compared to full fine-tuning.

5.4 Memory-Efficient Inference via Quantization

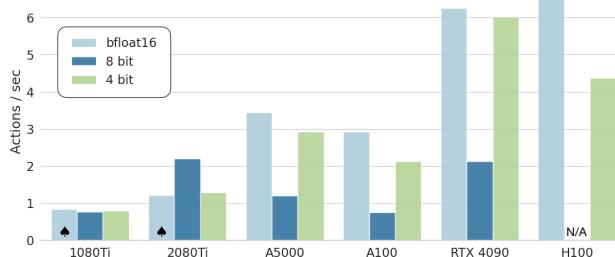


Figure 6: OpenVLA inference speed for various GPUs. Both bfloat16 and int4 quantization achieve high throughput, especially on GPUs with Ada Lovelace architecture (RTX 4090, H100). Further speed-ups are possible with modern LLM inference frameworks like TensorRT-LLM [89]. ♠: Model sharded across two GPUs to fit.

OpenVLA, a 7B-parameter model, consumes more memory at inference time than prior open-source generalist policies such as Octo, which has <100M parameters. We follow best-practices from LLM serving by saving and loading OpenVLA in bfloat16 precision for inference (our default approach), which cuts the memory footprint in half, allowing us to serve OpenVLA on GPUs with only 16GB

Precision	Bridge Success	VRAM
bfloat16	71.3 \pm 4.8%	16.8 GB
int8	58.1 \pm 5.1%	10.2 GB
int4	71.9 \pm 4.7%	7.0 GB

Table 2: Performance with quantized inference. 4-bit quantization matches the performance of bfloat16 inference (our default approach) while reducing the GPU memory footprint by more than half. Mean success \pm StdErr computed across 8 representative BridgeData V2 tasks [6] and 80 rollouts per approach (see [Table 5](#) for details).

⁴In [Section 5.3](#) and [Section 5.4](#), we experiment with a version of the OpenVLA model that is pretrained with a smaller robot data mixture (the same OpenX dataset mixture as Octo) and has a slightly smaller architecture which only uses a SigLIP [79] vision backbone instead of the fused DinoSigLIP encoder. We find that this simpler architecture still achieves strong performance in both fine-tuning tasks and “out-of-the-box” tasks.

of GPU memory. In this section, we test whether we can further reduce the required memory for policy inference and broaden accessibility of VLA policies, by using modern quantization techniques developed for serving LLMs [27, 88]. These approaches load the weights of the network at lower precision, thereby trading off reduced memory requirements for potentially reduced inference speed and accuracy.

Concretely, we investigate serving the OpenVLA model with 8-bit and 4-bit precision on 8 representative BridgeData V2 tasks. We report memory footprint and rollout performance in [Table 2](#). We also report achievable control frequencies on various consumer- and server-grade GPUs in [Fig. 6](#). We observe that 8-bit quantization slows down inference across most GPUs, due to the overhead of the added quantization operations. 4-bit inference achieves higher throughput, since reduced GPU memory transfer compensates for the quantization overhead.

As a result of the reduced inference speed, we observe a substantial performance decrease with 8-bit quantization: on the A5000 GPU we use for our evaluations, we can only run the model at 1.2Hz, which significantly changes the system dynamics compared to the training dataset for the 5Hz non-blocking controller used in the BridgeData V2 tasks.⁵ Notably, 4-bit quantization results in similar performance as bfloat16 half-precision inference despite requiring less than half the amount of GPU memory. 4-bit quantized models can run at 3Hz on the A5000, thus more closely matching the system dynamics during data collection.

6 Discussion and Limitations

In this work, we presented OpenVLA, a state-of-the-art, open-source vision-language-action model that obtains strong performance for cross-embodiment robot control out-of-the-box. We also demonstrated that OpenVLA can be easily adapted to new robot setups via parameter-efficient fine-tuning techniques.

The current OpenVLA model has several limitations. First, it currently only supports single-image observations. In reality, real-world robot setups are heterogeneous, with a wide range of possible sensory inputs [5]. Expanding OpenVLA to support multiple image and proprioceptive inputs as well as observation history is an important avenue for future work. Exploring the use of VLMs pretrained on *interleaved* image and text data may facilitate such flexible-input VLA fine-tuning.

Secondly, improving the inference throughput of OpenVLA is critical to enable VLA control for high-frequency control setups such as ALOHA [90], which runs at 50Hz. This will also enable testing VLAs on more dexterous, bi-manual manipulation tasks than what we investigated in this work. Exploring the use of action chunking or alternative inference-time optimization techniques such as speculative decoding [91] offer potential remedies.

Additionally, there is room for further performance improvements. While OpenVLA outperforms prior generalist policies, it does not yet offer very high reliability on the tested tasks, typically achieving <90% success rate.

Finally, due to compute limitations, many VLA design questions remain underexplored: What effect does the size of the base VLM have on VLA performance? Does co-training on robot action prediction data and Internet-scale vision-language data substantially improve VLA performance? What visual features are best-suited for VLA models? We hope that the release of the OpenVLA model and codebase will enable the community to jointly investigate these questions.

Acknowledgments

We are grateful to the Toyota Research Institute for providing significant funding and compute resources required to carry out this research. We also thank the Stanford Center for Research on Foundation Models for providing additional compute resources and Google DeepMind for alpha access to the RT-2-X API for our evaluations. We acknowledge additional support from Volkswagen, Physical Intelligence, ONR grants N00014-22-1-2621 and N00014-22-1-2293, the National Science Foundation through IIS-2246811, and DARPA ANSR.

⁵We attribute the performance loss to low inference speed, since both 8-bit and 4-bit quantization achieve comparable token accuracy to bfloat16 inference when evaluated offline on training data. See [Appendix D.4](#) for supporting details.

References

- [1] Open X-Embodiment Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srivama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. M. Shafullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. <https://arxiv.org/abs/2310.08864>, 2023.
- [2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.
- [3] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.
- [4] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *arXiv preprint arXiv:2307.03659*, 2023.
- [5] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. <https://octo-models.github.io>, 2023.
- [6] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale, 2023.
- [7] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

- [8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, volume 139, pages 8748–8763, 2021.
- [9] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *International Conference on Computer Vision (ICCV)*, 2023.
- [10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [11] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O’Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.
- [12] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *CoRL*, 2022.
- [13] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. *ArXiv*, abs/2302.12766, 2023. URL <https://api.semanticscholar.org/CorpusID:257205716>.
- [14] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.
- [15] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, et al. Open-world object manipulation using pre-trained vision-language models. *arXiv preprint arXiv:2303.00905*, 2023.
- [16] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [17] A. S. et al. Introducing rfm-1: Giving robots human-like reasoning capabilities, 2024. URL <https://covariant.ai/insights/introducing-rfm-1-giving-robots-human-like-reasoning-capabilities/>.
- [18] Wayve. Lingo-2: Driving with natural language. 2024. URL <https://wayve.ai/thinking/lingo-2-driving-with-language/>.
- [19] X. Chen, X. Wang, S. Changpinyo, A. J. Piergiovanni, P. Padlewski, D. M. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer, A. Kolesnikov, J. Puigcerver, N. Ding, K. Rong, H. Akbari, G. Mishra, L. Xue, A. V. Thapliyal, J. Bradbury, W. Kuo, M. Seyedhosseini, C. Jia, B. K. Ayan, C. Riquelme, A. Steiner, A. Angelova, X. Zhai, N. Houlsby, and R. Soricut. Pali: A jointly-scaled multilingual language-image model. *ArXiv*, abs/2209.06794, 2022. URL <https://api.semanticscholar.org/CorpusID:252222320>.

- [20] X. Chen, X. Wang, L. Beyer, A. Kolesnikov, J. Wu, P. Voigtlaender, B. Mustafa, S. Goodman, I. M. Alabdulmohsin, P. Padlewski, D. M. Salz, X. Xiong, D. Vlasic, F. Pavetic, K. Rong, T. Yu, D. Keysers, X.-Q. Zhai, and R. Soricut. PaLI-3 vision language models: Smaller, faster, stronger. *arXiv preprint arXiv:2310.09199*, 2023.
- [21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, and ... Transformers: State-of-the-art natural language processing. In *Proceedings of the 6th International Conference on Learning Representations*, 2020. URL <https://arxiv.org/abs/1910.03771>.
- [22] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [23] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- [24] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- [25] M. Oquab, T. Dariseti, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [26] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [27] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.
- [28] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [29] D. A. Hudson and C. D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [30] A. Singh, V. Natarajan, M. Shah, Y. Jiang, X. Chen, D. Batra, D. Parikh, and M. Rohrbach. Towards VQA models that can read. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [31] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh. VizWiz: nearly real-time answers to visual questions. In *User Interface Software and Technology (UIST)*, pages 333–342, 2010.
- [32] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, 2014.
- [33] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *European Conference on Computer Vision (ECCV)*, 2016.
- [34] T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Huszenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy,

- H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- [35] Y. Li, S. Bubeck, R. Eldan, A. D. Giorno, S. Gunasekar, and Y. T. Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.
 - [36] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
 - [37] J. Li, D. Li, C. Xiong, and S. C. H. Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning (ICML)*, 2022.
 - [38] J. Li, D. Li, S. Savarese, and S. C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning (ICML)*, 2023.
 - [39] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. A. Li, P. Fung, and S. C. H. Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint arXiv:2305.06500*, 2023.
 - [40] H. H. Tan and M. Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
 - [41] H. Laurençon, L. Saulnier, L. Tronchon, S. Bekman, A. Singh, A. Lozhkov, T. Wang, S. Karamcheti, A. M. Rush, D. Kiela, M. Cord, and V. Sanh. OBELICS: An open web-scale filtered dataset of interleaved image-text documents. In *Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2023.
 - [42] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
 - [43] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.
 - [44] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.
 - [45] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.
 - [46] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv*, 2021.
 - [47] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

- [48] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, et al. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv preprint arXiv:2312.02976*, 2023.
- [49] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. *arXiv preprint arXiv:2309.01918*, 2023.
- [50] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.
- [51] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.
- [52] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.
- [53] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *CoRL*, 2019.
- [54] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. de Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *RSS*, 2019.
- [55] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.
- [56] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 3:5, 2023.
- [57] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2017.
- [58] E. S. Hu, K. Huang, O. Rybkin, and D. Jayaraman. Know thyself: Transferable visual control policies through robot-awareness. In *International Conference on Learning Representations*, 2022.
- [59] J. H. Yang, D. Sadigh, and C. Finn. Polybot: Training one policy across robots while embracing variability. In *7th Annual Conference on Robot Learning*, 2023. URL <https://openreview.net/forum?id=HEIRj511cS>.
- [60] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.
- [61] G. Salhotra, I.-C. A. Liu, and G. Sukhatme. Bridging action space mismatch in learning from demonstrations. *arXiv preprint arXiv:2304.03833*, 2023.
- [62] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik. Robot learning with sensorimotor pre-training. In *Conference on Robot Learning*, 2023.

- [63] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.
- [64] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.
- [65] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning*, 2023. URL <https://arxiv.org/abs/2306.14846>.
- [66] J. Yang, C. Glossop, A. Bhorkar, D. Shah, Q. Vuong, C. Finn, D. Sadigh, and S. Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.
- [67] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.
- [68] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.
- [69] Y. J. Ma, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR, 2023.
- [70] X. Zhang, Y. Ding, S. Amiri, H. Yang, A. Kaminski, C. Esselink, and S. Zhang. Grounding classical task planners via vision-language models. *arXiv preprint arXiv:2304.08587*, 2023.
- [71] S. Sontakke, J. Zhang, S. Arnold, K. Pertsch, E. Bıyık, D. Sadigh, C. Finn, and L. Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36, 2024.
- [72] J. Huang, S. Yong, X. Ma, X. Linghu, P. Li, Y. Wang, Q. Li, S.-C. Zhu, B. Jia, and S. Huang. An embodied generalist agent in 3d world. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- [73] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.
- [74] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.
- [75] PyTorch. Automatic mixed precision. URL <https://pytorch.org/docs/stable/amp.html>.
- [76] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- [77] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.
- [78] N. Dorka, C. Huang, T. Welschehold, and W. Burgard. What matters in employing vision language models for tokenizing actions in robot control? In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

- [79] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.
- [80] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [81] P. Sharma, N. Ding, S. Goodman, and R. Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.
- [82] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.
- [83] O. Sidorov, R. Hu, M. Rohrbach, and A. Singh. Textcaps: a dataset for image captioning with reading comprehension. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 742–758. Springer, 2020.
- [84] H. Face. Introducing idefics: An open reproduction of state-of-the-art visual langage model. *Hugging Face Blog*, 2024.
- [85] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.
- [86] B. McKinzie, Z. Gan, J.-P. Fauconnier, S. Dodge, B. Zhang, P. Dufter, D. Shah, X. Du, F. Peng, F. Weers, et al. Mm1: Methods, analysis & insights from multimodal lilm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.
- [87] J. Lin, H. Yin, W. Ping, Y. Lu, P. Molchanov, A. Tao, H. Mao, J. Kautz, M. Shoeybi, and S. Han. Vila: On pre-training for visual language models. *arXiv preprint arXiv:2312.07533*, 2023.
- [88] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 2022.
- [89] NVIDIA. Tensorrt-llm. URL <https://github.com/NVIDIA/TensorRT-LLM>.
- [90] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- [91] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.
- [92] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [93] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard. Latent plans for task agnostic offline reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.
- [94] O. Mees, J. Borja-Diaz, and W. Burgard. Grounding language with visual affordances over unstructured data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

- [95] S. Dass, J. Yapeter, J. Zhang, J. Zhang, K. Pertsch, S. Nikolaidis, and J. J. Lim. CLVR jaco play dataset, 2023. URL https://github.com/clvrai/clvr_jaco_play_dataset.
- [96] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine. Multi-stage cable routing through hierarchical imitation learning. *arXiv preprint arXiv:2307.08927*, 2023.
- [97] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. RoboTurk: A crowdsourcing platform for robotic skill learning through imitation. *CoRR*, abs/1811.02790, 2018. URL <http://arxiv.org/abs/1811.02790>.
- [98] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors, 2023.
- [99] L. Y. Chen, S. Adebola, and K. Goldberg. Berkeley UR5 demonstration dataset. <https://sites.google.com/view/berkeley-ur5/home>.
- [100] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu, P. Abbeel, L. Pinto, C. Finn, and A. Gupta. Train offline, test online: A real robot learning benchmark, 2023.
- [101] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.
- [102] S. Belkhale, Y. Cui, and D. Sadigh. Hydra: Hybrid robot actions for imitation learning. *arxiv*, 2023.
- [103] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.
- [104] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.
- [105] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.
- [106] G. Yan, K. Wu, and X. Wang. ucsd kitchens Dataset. August 2023.
- [107] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on Robot Learning (CoRL)*, 2022.
- [108] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.
- [109] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel. Shared Control Templates for Assistive Robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, page 7, Paris, France, 2020.
- [110] S. Saxena, M. Sharma, and O. Kroemer. Multi-resolution sensing for real-time control with vision-language models. In *7th Annual Conference on Robot Learning*, 2023. URL <https://openreview.net/forum?id=WuBv9-IGDUA>.
- [111] R. Shah, R. Martín-Martín, and Y. Zhu. MUTEX: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023. URL <https://openreview.net/forum?id=PwqiqaEzJ>.

- [112] X. Zhu, R. Tian, C. Xu, M. Ding, W. Zhan, and M. Tomizuka. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot. 2023.
- [113] R. Mendonca, S. Bahl, and D. Pathak. Structured world models from human videos. *CoRL*, 2023.
- [114] J. Luo, C. Xu, F. Liu, L. Tan, Z. Lin, J. Wu, P. Abbeel, and S. Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *arXiv preprint arXiv:2401.08553*, 2024.
- [115] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home, 2023.
- [116] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [117] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

A Data Mixture Details

We list our used data mixture in [Table 3](#). The mixture mostly follows [5], with a few additional datasets.

OpenVLA Training Dataset Mixture	
Fractal [92]	12.7%
Kuka [45]	12.7%
Bridge[6, 47]	13.3%
Taco Play [93, 94]	3.0%
Jaco Play [95]	0.4%
Berkeley Cable Routing [96]	0.2%
Roboturk [97]	2.3%
Viola [98]	0.9%
Berkeley Autolab UR5 [99]	1.2%
Toto [100]	2.0%
Language Table [101]	4.4%
Stanford Hydra Dataset [102]	4.4%
Austin Buds Dataset [103]	0.2%
NYU Franka Play Dataset [104]	0.8%
Furniture Bench Dataset [105]	2.4%
UCSD Kitchen Dataset [106]	<0.1%
Austin Sailor Dataset [107]	2.2%
Austin Sirius Dataset [108]	1.7%
DLR EDAN Shared Control [109]	<0.1%
IAMLab CMU Pickup Insert [110]	0.9%
UTAustin Mutex [111]	2.2%
Berkeley Fanuc Manipulation [112]	0.7%
CMU Stretch [113]	0.2%
BC-Z [55]	7.5%
FMB Dataset [114]	7.1%
DobbE [115]	1.4%
DROID [11]	10.0% ⁶

Table 3: OpenVLA training data mixture using datasets from the Open X-Embodiment dataset [1], following [5] with a few additions.

B Evaluation Tasks and Detailed Results

In this section, we provide more details on the BridgeData V2 WidowX and Google robot evaluations discussed in [Section 5.1](#), as well as the Franka-Tabletop and Franka-DROID fine-tuning evaluations discussed in [Section 5.2](#).

B.1 BridgeData V2 WidowX Evaluation Details

Here we focus specifically on BridgeData V2 evaluations discussed in [Section 5.1](#).

B.1.1 BridgeData V2 Evaluation Tasks

As described in [Section 5.1](#), we evaluate each generalist robot manipulation policy on 17 tasks with 10 trials each. In this section, we provide details on the task categories and individual tasks.

In total, we evaluate on 5 visual generalization tasks, 2 motion generalization tasks, 3 physical generalization tasks, 4 semantic generalization tasks, and 3 language grounding tasks. Note that all tasks we evaluate on introduce some form of distribution shift since we are unable to procure the exact objects used in the original dataset (other distribution shifts naturally arise as we reproduce a real-world test environment originally constructed at a different location; see [Appendix B.1.2](#) for a detailed discussion on such distribution shifts). All 17 tasks are depicted in [Fig. 7](#). Each rollout is

⁶We remove DROID for the last third of training due to slow learning progress (see [Section 3.3](#)) and redistribute its mixture weights across all other datasets.

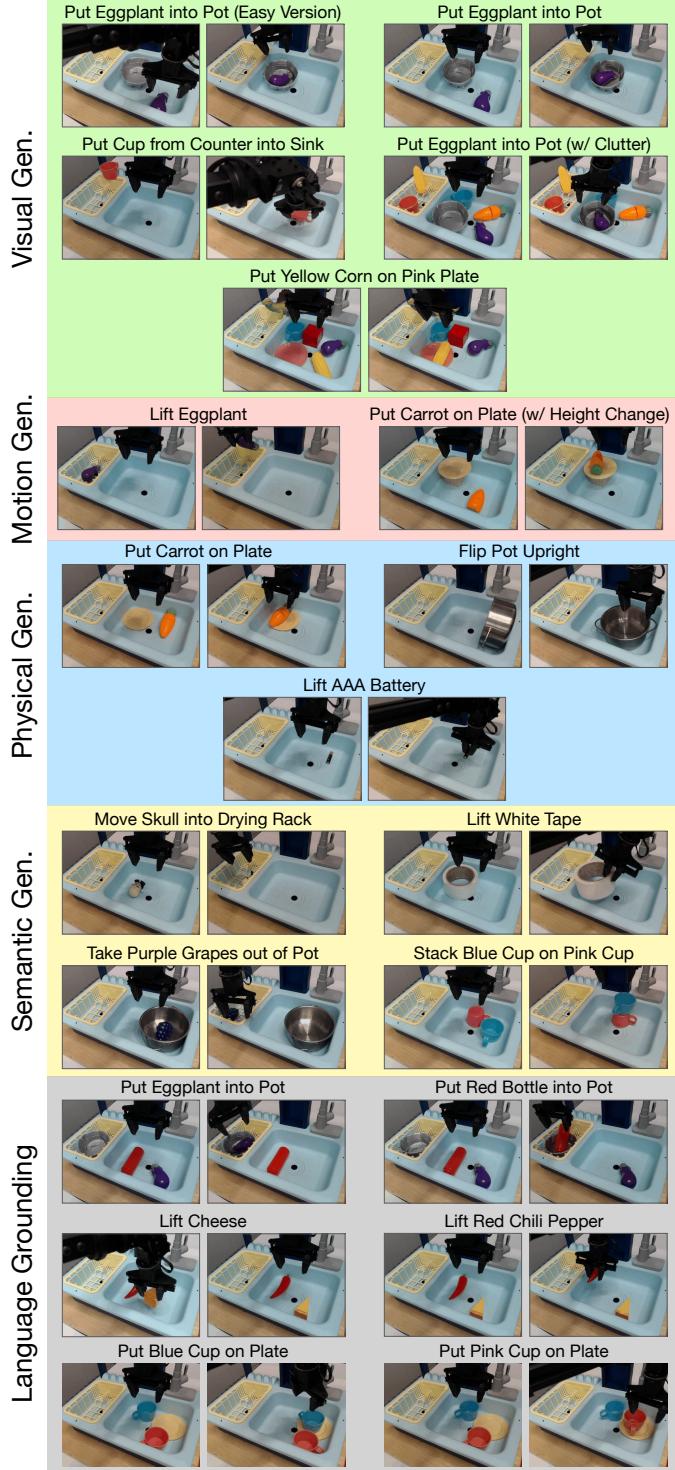


Figure 7: BridgeData V2 WidowX robot evaluation tasks. We evaluate every generalist robot policy on 4 types out-of-distribution (OOD) generalization tasks: **visual**, **motion**, **physical**, and **semantic** (as defined in [Section 5.1](#)). Every pair of images shows the start state and an example end state after the robot completes the task. We also rigorously assess **language grounding** in the 3 tasks shown in the bottom 3 rows, by changing the prompt while fixing the initial state and testing whether the policy can approach the correct target object.

marked as a failure (0) or success (1). In some more difficult tasks, we record partial successes (0.5); we describe the conditions for partial credit in the task descriptions below.

Below we describe each of the 17 tasks, in the order shown in Fig. 7:

1. **Put Eggplant into Pot (Easy Version):** The robot’s goal is to pick up the eggplant and drop it into the pot. This is a **visual generalization** task because we use a handcrafted paper pot that has a different appearance than the pot used in the original BridgeData V2 training dataset (since we are unable to procure the original pot). Unlike all 16 other tasks, for this particular task we initialize the robot’s end-effector directly above the eggplant before rolling out the policy; hence, we call this the “Easy Version” of the “Put Eggplant into Pot” task.
2. **Put Eggplant into Pot:** This is the same task as described above, except that the robot’s end-effector is not initialized directly above the eggplant. Instead, we initialize it in a position that is fixed across all rollouts, which means that the robot must horizontally reach for the eggplant first before manipulating it. (Note: The same applies to all other tasks described below.) This is a **visual generalization** task for the same reason as above.
3. **Put Cup from Counter into Sink:** The robot’s goal is to pick up the pink cup from either the kitchen countertop or drying rack and place it into the sink on the right. This is a **visual generalization** task because we use a pink cup rather than a blue cup (a blue cup is used in the original BridgeData V2 dataset, but we find that none of the methods we evaluate is able to manipulate it reliably – most likely because the color of the cup blends in with the color of the sink).
4. **Put Eggplant into Pot (w/ Clutter):** This is the same task as the “Put Eggplant into Pot” task, except that it is more difficult due to the presence of several distractor objects. It is a **visual generalization** task for the same reason discussed in the normal “Put Eggplant into Pot” task, and even more so given unseen distractors in the scene. *Partial credit (0.5 out of 1) is rewarded when the robot moves towards the correct target object.*
5. **Put Yellow Corn on Pink Plate:** The robot’s goal is to pick up the yellow corn and place it on the pink plate. This is a **visual generalization** task due to the presence of unseen distractor objects in the scene, such as a green dinosaur on the countertop in the back section of the sink. *Partial credit (0.5 out of 1) is rewarded when the robot moves towards the correct target object.*
6. **Lift Eggplant:** The robot’s goal is to grasp and lift the eggplant into the air. This is a **motion generalization** task because the eggplant is initialized in unseen positions and/or orientations, and the robot is forced to move beyond its training distribution of positions and/or orientations and often perform long-range reaching in order to complete the task. (Note: Long-range reaching is not demonstrated in this environment in the original BridgeData V2 demonstrations; see Appendix B.1.2 for details.) We find that this task, though seemingly simple, is deceptively challenging for many policies. *Partial credit (0.5 out of 1) is rewarded when the robot makes contact with the eggplant.*
7. **Put Carrot on Plate (w/ Height Change):** The robot’s goal is to pick up the carrot and place it on the yellow plate. This is a **motion generalization** task because the plate is elevated from its usual position at the bottom of the sink, and the robot must adjust its trajectory to correctly place the carrot on the elevated platform (without knocking down the plate in the process). *Partial credit (0.5 out of 1) is rewarded when the robot grasps the carrot and touches the plate with it.*
8. **Put Carrot on Plate:** This is the same task as above, except that the plate is at its normal position (at the bottom of the sink or drying rack). We consider this a **physical generalization** task because the carrot has a different size and shape than the one used in the original BridgeData V2 dataset, which is shorter and narrower. (Note that the previous version of this task listed above would also technically be a physical generalization task since it involves the same carrot, but we list it under the “motion generalization” category since that is the focus there.)

9. **Flip Pot Upright:** The robot’s goal is to manipulate the pot such that it is oriented upright in the sink at the end of the episode. This is a **physical generalization** task because this pot has a different size and shape than the one used in the original BridgeData V2 training demonstrations (the pot we use is wider and shorter).
10. **Lift AAA Battery:** The robot’s goal is simply to grasp the AAA battery and lift it up into the air. This is considered a **physical generalization** task because the battery is much smaller and thinner than target objects seen in the BridgeData V2 training demonstrations in this environment; see [Appendix B.1.2](#) for details. (Note that this target object does not exist in the original BridgeData V2 demonstrations in this environment, so this is also an instance of “semantic generalization”, but we classify it solely as “physical generalization” since that is the main focus here).
11. **Move Skull into Drying Rack:** The robot’s goal is to grasp the skull windup toy and drop it into the yellow drying rack in the left part of the sink. This is a **semantic generalization** task since the skull is an unseen target object (does not appear in the BridgeData V2 training demonstrations).
12. **Lift White Tape:** The robot’s goal is to grasp and lift the white roll of tape into the air. This is a **semantic generalization** task since the white tape roll is an unseen target object (does not appear in the BridgeData V2 training demonstrations). (Note that this task may also be considered as “physical generalization” because of its shape being different than the objects seen in the training demonstrations in this environment; most policies struggle to grasp objects with this ring structure, and they often move the robot’s end-effector directly into the center region.)
13. **Take Purple Grapes out of Pot:** The robot’s goal is to grasp the purple grapes lying inside the steel pot and remove it from the pot (by lifting it out and/or dropping it anywhere outside the pot). This is a **semantic generalization** task because it is an unseen language instruction; the robot has never seen this task in the original BridgeData V2 training dataset.
14. **Stack Blue Cup on Pink Cup:** The robot’s goal is to grasp the blue cup and place it securely on top of the pink cup. This is a **semantic generalization** task because it is an unseen language instruction; the robot has never seen this task in this environment in the original BridgeData V2 training dataset. *Partial credit (0.5 out of 1) is rewarded when the robot grasps the blue cup and touches the pink cup with the blue cup.*
15. **Put {Eggplant, Red Bottle} into Pot:** This is a **language grounding** task. The robot’s goal is to put the specified target object into the pot. Both the eggplant and red bottle are present in the scene. We conduct paired evaluations: for the same initial state, we prompt the policy to target the eggplant in one episode, and then the red bottle in the next episode. We test each method 5 times with the eggplant and 5 times with the red bottle, using the same set of 5 initial states for both target objects. *Partial credit (0.5 out of 1) is rewarded when the robot moves towards the correct target object.*
16. **Lift {Cheese, Red Chili Pepper}:** This is a **language grounding** task. The robot’s goal is to grasp and lift the specified target object. We conduct paired evaluations as described in the task above. *Partial credit (0.5 out of 1) is rewarded when the robot moves towards the correct target object.*
17. **Put {Blue Cup, Pink Cup} on Plate:** This is a **language grounding** task. The robot’s goal is to grasp the specified target object and place it onto the plate. We conduct paired evaluations as described in other language grounding tasks. *Partial credit (0.5 out of 1) is rewarded when the robot moves towards the correct target object.*

B.1.2 Comparing Evaluation Tasks to Original BridgeData V2 Training Data

We conduct our evaluations in a sink environment used in the original BridgeData V2 dataset [6]. We reproduce the environment to match the original environment in the BridgeData V2 dataset with rough approximations for the robot’s location relative to the sink, as well as the camera’s placement

relative to the scene. Given the lack of precise measurements of these positions in the original dataset, we are unable to reproduce the *exact* environment setup, and natural distribution shifts arise due to slightly different robot, sink, and camera placements. In addition, since we evaluate robot policies in a different location than where the training demonstrations were collected from, other natural distribution shifts arise. For example, the lighting conditions and background (e.g., visible areas behind the sink) are inevitably different than what was seen in the training dataset. Furthermore, we are unable to procure the exact set of objects used in the original BridgeData V2 dataset, so there are distribution shifts between the objects used at train time and those used at test time.

Despite all these challenges, we find that certain generalist policies, such as OpenVLA and RT-2-X, can still generalize and perform various tasks fairly reliably “out-of-the-box”. Other generalist policies, such as RT-1-X and Octo, can also complete some tasks, though they struggle when tested with more difficult generalization tasks in our BridgeData V2 evaluation suite.

The original BridgeData V2 dataset includes demonstrations of the following seven tasks in this specific sink environment: “Flip Pot Upright”, “Put Carrot on Plate”, “Put Cup from Counter (or Drying Rack) into Sink”, “Put Eggplant into Pot”, “Put Knife on Cutting Board”, “Put Spoon in Pot”, and “Turn Lever Vertical to Front”. See Fig. 8 for samples images of all these tasks from the original dataset. Note that all training demonstrations collected in this environment are initialized such that the robot’s end-effector is positioned directly above the target object in the beginning of the episode. (However, this is not the case across all environments in the BridgeData V2 dataset; in some other environments, the robot is initialized farther away from the target object, so it must horizontally reach for the object first before manipulating it.)



Figure 8: Original BridgeData V2 sink environment tasks. Images from sample demonstrations in the sink environment from the original BridgeData V2 dataset reveal that all demonstrations in this environment were initialized such that the robot’s end-effector was positioned immediately above the target object. Note that these initial states are different from the initial states we use in our BridgeData V2 evaluation tasks shown in Fig. 7. In our evaluations, we always initialize the robot’s end-effector to a fixed location above the sink, rather than positioning it directly above the target object (except for one task: “Put Eggplant into Pot (Easy Version)”).

In our BridgeData V2 evaluation suite, only one task – “Put Eggplant into Pot (Easy Version)” – is initialized with the robot’s end-effector hovering directly over the target object; in all 16 other tasks, the end-effector is initialized at a fixed location above the sink such that the robot must horizontally reach towards the object. This initial condition, in combination with the distribution shifts we introduce in the various types of OOD generalization in our evaluation suite, challenges the generalist policies and requires a high degree of robustness in order to complete the tasks successfully. Hence, the success rates for policies like RT-1-X and Octo are lower than what is reported in prior works. However, we find that other policies such as RT-2-X and OpenVLA still achieve relatively strong performance despite all these distribution shifts and challenges.

B.1.3 Detailed BridgeData V2 Evaluation Results

See [Table 4](#) for the full BridgeData V2 WidowX evaluation results. The number of successes for each method, out of 10 trials, is listed for each of 17 tasks. OpenVLA achieves strongest performance in the majority of the tasks and has the highest aggregate success rate among the generalist policies. RT-2-X also shows good performance, outperforming RT-1-X and Octo, though it does not perform as well as OpenVLA. RT-1-X and Octo generally experience difficulty in these generalization tasks.

Table 4: Detailed BridgeData V2 WidowX evaluation results. We report performance on the full evaluation suite of 17 tasks (discussed in [Section 5.1](#)), including visual/motion/physical/semantic generalization tasks and language grounding tasks. Note that *partial success* (score of 0.5) is possible for some tasks; see [Appendix B.1.1](#) for details. We find that OpenVLA performs best in most tasks and achieves highest performance overall, followed by RT-2-X. On the other hand, RT-1-X and Octo struggle in the evaluations, only getting 0–2 successes in several tasks. See [Fig. 7](#) for illustrations of all tasks.

Category	Task	# Trials	RT-1-X # Successes	Octo # Successes	RT-2-X # Successes	OpenVLA (ours) # Successes
Visual gen	Put Eggplant into Pot (Easy Version)	10	1	5	7	10
Visual gen	Put Eggplant into Pot	10	0	1	5	10
Visual gen	Put Cup from Counter into Sink	10	1	1	0	7
Visual gen	Put Eggplant into Pot (w/ Clutter)	10	1	3.5	6	7.5
Visual gen	Put Yellow Corn on Pink Plate	10	1	4	8	9
Motion gen	Lift Eggplant	10	3	0.5	6.5	7.5
Motion gen	Put Carrot on Plate (w/ Height Change)	10	2	1	4.5	4.5
Physical gen	Put Carrot on Plate	10	1	0	1	8
Physical gen	Flip Pot Upright	10	2	6	5	8
Physical gen	Lift AAA Battery	10	0	0	2	7
Semantic gen	Move Skull into Drying Rack	10	1	0	5	5
Semantic gen	Lift White Tape	10	3	0	0	1
Semantic gen	Take Purple Grapes out of Pot	10	6	0	5	4
Semantic gen	Stack Blue Cup on Pink Cup	10	0.5	0	5.5	4.5
Language grounding	Put {Eggplant, Red Bottle} into Pot	10	2.5	4	8.5	7.5
Language grounding	Lift {Cheese, Red Chili Pepper}	10	1.5	2.5	8.5	10
Language grounding	Put {Blue Cup, Pink Cup} on Plate	10	5	5.5	8.5	9.5
Mean Success Rate		18.5±2.7%	20.0±2.6%	50.6±3.5%	70.6±3.2%	

Additionally, in [Table 5](#), we provide the full evaluation results for the quantized inference experiments that were summarized in [Table 2](#). For these evaluations, we test policies on 8 representative BridgeData V2 tasks spanning all task categories in the full evaluation suite.

Table 5: Full quantized inference results. Here we present the detailed version of the results shown in [Table 2](#).

Category	Task	# Trials	bfloat16 # Successes	int8 # Successes	int4 # Successes
Visual gen	Put Eggplant into Pot (Easy Version)	10	9	7	9
Visual gen	Put Eggplant into Pot	10	7	7	7
Visual gen	Put Cup from Counter into Sink	10	5	3	7
Motion gen	Lift Eggplant	10	6	4	7.5
Physical gen	Put Carrot on Plate	10	6	5	7
Physical gen	Lift AAA Battery	10	7	5	3
Semantic gen	Take Purple Grapes out of Pot	10	8	8	9
Language grounding	Put {Eggplant, Red Bottle} into Pot	10	9	7.5	8
Mean Success Rate		71.3 ± 4.8%	58.1 ± 5.1%	71.9 ± 4.7%	

B.2 Google Robot Evaluation Details

In this section, we provide more details on the Google robot evaluations introduced in [Section 5.1](#).

B.2.1 Google Robot Evaluation Tasks

On the Google robot, we evaluate each generalist robot policy on 12 tasks with 5 rollouts each, for a total of 60 rollouts. The first five tasks test on in-distribution conditions, and the last seven tasks test on more difficult out-of-distribution (OOD) conditions. All tasks are depicted in [Fig. 9](#). Each rollout is marked as a failure (0) or success (1).

We describe the 12 tasks below:

1. **Pick Coke Can** (in-distribution): The robot is positioned in front of a platform with a can of Coke on top of it. The robot’s goal is to grasp and lift the Coke can.

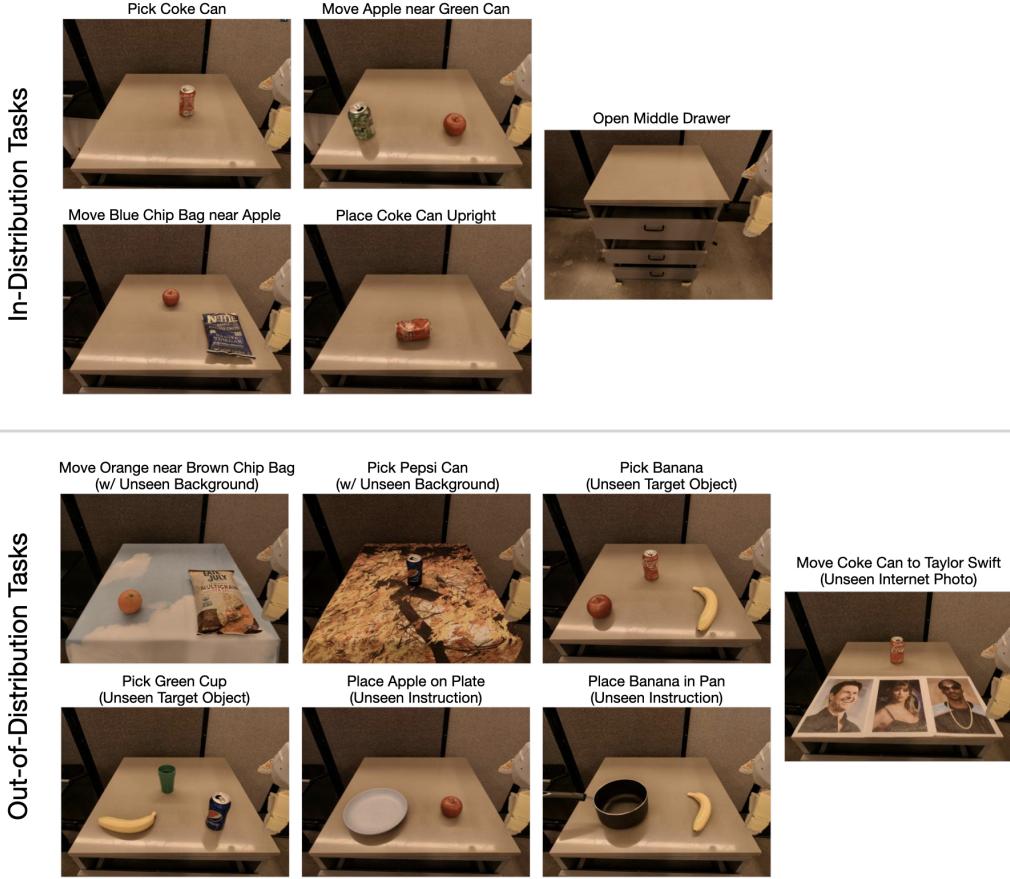


Figure 9: **Google robot evaluation tasks.** We evaluate every generalist robot policy on in-distribution tasks and out-of-distribution (OOD) generalization tasks. OOD tasks involve unseen backgrounds, target objects, instructions/object relations, and semantic concepts (e.g., photos from the Internet that do not appear in robot action data).

2. **Move Apple near Green Can** (in-distribution): The robot is positioned in front of a platform with an apple and a green soda can on top of it. The robot’s goal is to grasp the apple and move it next to the green can.
3. **Move Blue Chip Bag near Apple** (in-distribution): The robot is positioned in front of a platform with a blue bag of chips and an apple on top of it. The robot’s goal is to grasp the blue bag of chips and move it close to the apple.
4. **Place Coke Can Upright** (in-distribution): The robot is positioned in front of a platform with a can of Coke on top of it, and the can is oriented horizontally on its side. The robot’s goal is to grasp the Coke can and orient it to be in a vertical position.
5. **Open Middle Drawer** (in-distribution): The robot is positioned in front of a set of three drawers. The robot’s goal is to grasp the middle drawer handle and pull the drawer open.
6. **Move Orange near Brown Chip Bag** (OOD): The robot is positioned in front of a platform with a brown bag of chips and an orange on top of it. A tablecloth with blue sky and white cloud patterns covers the platform underneath the objects. The robot’s goal is to grasp the orange and bring it next to the bag of chips. This task is OOD because the orange is an unseen object relative to the training dataset, and the tablecloth is an unseen background.⁷

⁷See Appendix of Brohan et al. [7] for a detailed list of OOD conditions in Google robot evaluations.

7. **Pick Pepsi Can** (OOD): The robot is positioned in front of a platform with a can of Pepsi on top of it. A tablecloth with bright yellow/brown patterns covers the platform underneath the can. The robot’s goal is to grasp and lift the can. This task is OOD because the Pepsi can is an unseen object, and the tablecloth is an unseen background.
8. **Pick Banana** (OOD): The robot is positioned in front of a platform with an apple, a can of Coke, and a banana. The robot’s goal is to grasp and lift the banana. This task is OOD because the banana is an unseen target object.
9. **Pick Green Cup** (OOD): The robot is positioned in front of a platform with a banana, a can of Pepsi, and a green cup. The robot’s goal is to grasp and lift the green cup. This task is OOD because all objects in the scene are unseen in the training data.
10. **Place Apple on Plate** (OOD): The robot is positioned in front of a platform with a plate and an apple. The robot’s goal is to grasp the apple and move it onto the plate. This task is OOD because it is a novel instruction describing an unseen object relation: training demonstrations only cover moving the apple *near* the plate, rather than placing it *on top of* the plate.
11. **Place Banana in Pan** (OOD): The robot is positioned in front of a platform with a pan and a banana. The robot’s goal is to grasp the banana and move it into the pan. This task is OOD because the banana is an unseen target object, and it is a novel instruction describing an unseen object relation, as explained in the previous task.
12. **Move Coke Can to Taylor Swift** (OOD): The robot is positioned in front of a platform with a can of Coke and photos of three different celebrities, including Taylor Swift. The robot’s goal is to grasp the can and move it to the photo of Taylor Swift. This task is OOD because the photos of the celebrities are unseen in the robot interaction data.

B.2.2 Detailed Google Robot Evaluation Results

Table 6: **Detailed Google robot evaluation results.** We report full evaluation results for Google robot evaluations discussed in Section 5.1. Each generalist policy is evaluated with 60 rollouts across 12 tasks, covering both in-distribution and out-of-distribution (OOD) testing conditions. In the bottom row, we report mean success rate \pm StdErr for each policy. OpenVLA and RT-2-X both significantly outperform RT-1-X and Octo overall (we bold the mean success rate for both due to overlapping error bars). See Fig. 9 for illustrations of all tasks.

Category	Task	# Trials	RT-1-X		Octo		RT-2-X		OpenVLA (ours)	
			# Successes	RT-1-X	# Successes	Octo	# Successes	RT-2-X	# Successes	OpenVLA (ours)
In-distribution	Pick Coke Can	5	5	1	5	5	5	5	5	5
In-distribution	Move Apple near Green Can	5	3	3	3	3	3	3	3	5
In-distribution	Move Blue Chip Bag near Apple	5	0	3	4	4	4	4	4	5
In-distribution	Place Coke Can Upright	5	0	0	0	4	4	4	4	4
In-distribution	Open Middle Drawer	5	0	4	2	2	2	2	2	3
OOD	Move Orange near Brown Chip Bag	5	1	2	5	5	5	5	5	5
OOD	Pick Pepsi Can	5	3	0	5	5	5	5	5	4
OOD	Pick Banana	5	5	3	5	5	5	5	5	5
OOD	Pick Green Cup	5	1	0	5	5	5	5	5	5
OOD	Place Apple on Plate	5	0	0	4	4	4	4	4	4
OOD	Place Banana in Pan	5	0	0	2	2	2	2	2	4
OOD	Move Coke Can near Taylor Swift	5	2	0	3	3	3	3	3	2
Mean Success Rate			33.3 \pm 6.1%	26.7 \pm 5.8%	78.3\pm5.4%	85.0\pm4.6%				

Full results for the Google robot evaluations are shown in Table 6. Overall, we find that RT-1-X and Octo experience difficulty on the evaluation tasks; they are often unable to achieve a single success out of five trials in several tasks. On the other hand, RT-2-X and OpenVLA demonstrate strong performance, completing every task at least two times out of five trials; these two VLA policies perform comparably with each other on this particular evaluation suite.

B.3 Data-Efficient Adaptation Experiment Details

In this section, we provide more details on the data-efficient adaptation experiments discussed in Section 5.2, where we investigate the effectiveness of fine-tuned OpenVLA policies on new robot setups such as Franka-Tabletop and Franka-DROID.

B.3.1 Franka-Tabletop and Franka-DROID Tasks

We collect 10–150 demonstrations of each of seven tasks. The first six tasks correspond to a robot setup which we denote as “Franka-Tabletop” (Franka Emika Panda robot mounted on top of a table), and the final task corresponds to a robot setup which we call “Franka-DROID”.

In the Franka-Tabletop setup, the first three of six tasks correspond to single-instruction tasks and are narrow, while the last three tasks correspond to multi-instruction tasks in which multiple objects are present in the scene and the robot must manipulate the correct one depending on the language instruction.

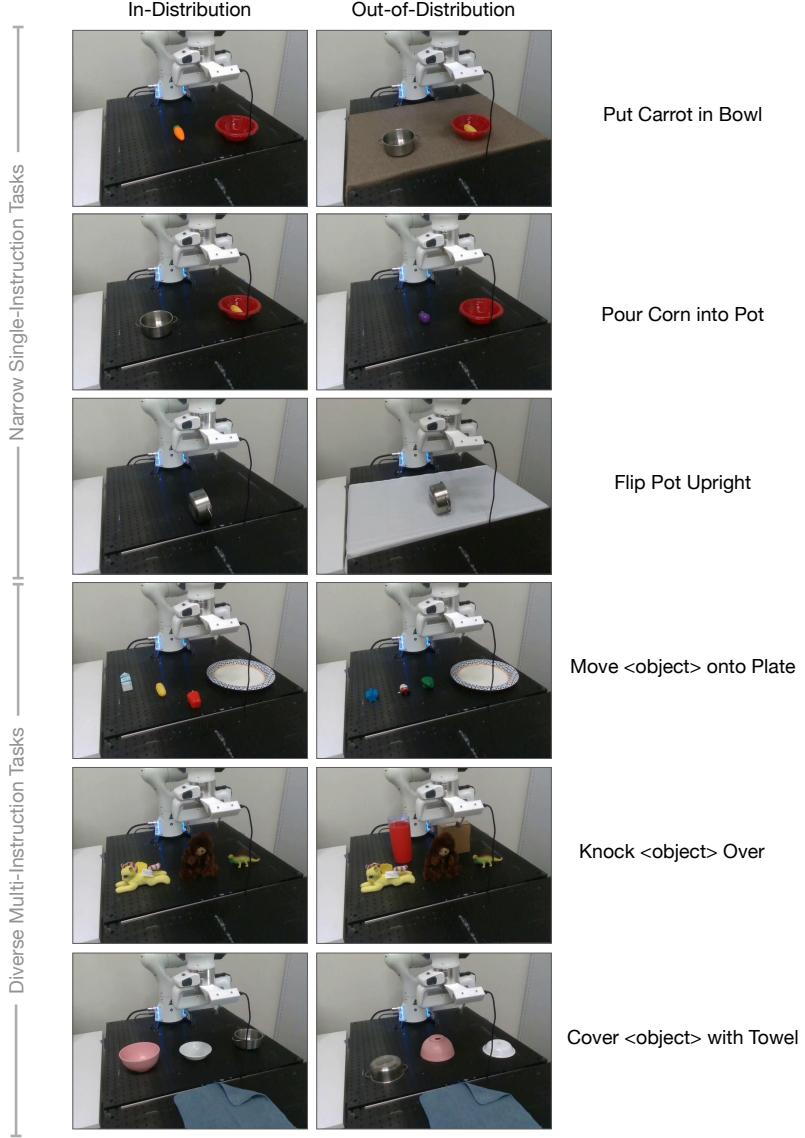


Figure 10: Franka-Tabletop fine-tuning tasks. Franka-Tabletop tasks used in the data-efficient adaptation experiments in Section 5.2 and described in detail in Fig. 10 are depicted above. The first three of six tasks, shown in the top three rows, only involve a single instruction, while the last three tasks in the bottom three rows involve multiple objects and instructions (the instructions specify the target object or target location). The first column shows sample initial states matching the training data distribution, while the second column shows out-of-distribution (OOD) initial states (e.g., unseen backgrounds, target objects, distractors, and object positions/orientations). Every policy in Section 5.2 is evaluated with 10–12 rollouts on in-distribution tasks and 5–6 rollouts on OOD tasks.

Below we describe each of the six Franka-Tabletop tasks shown in Fig. 10:

1. **Put Carrot in Bowl** (single-instruction): The robot’s goal is to grasp the carrot and place it into the bowl. We collect 50 demonstrations of this task for the training dataset, randomly placing the carrot and the bowl at different locations on the table in every episode. The carrot is always initialized on the left side of the bowl. During evaluation, each trial is recorded as a success (1) or failure (0); there is no partial credit.
2. **Pour Corn into Pot** (single-instruction): The robot’s goal is to grasp the red bowl, move towards the steel pot, and pour the contents (a yellow corn) into the pot. We collect 50 demonstrations of this task for the training dataset, randomly placing the bowl and the pot at different locations on the table in every episode. The bowl is always initialized on the right side of the pot. During evaluation, each trial is recorded as a success (1) or failure (0); there is no partial credit.
3. **Flip Pot Upright** (single-instruction): The robot’s goal is to grasp the steel pot (which is initially oriented vertically), rotate it to be in the upright position, and place it back onto the table. We collect only 10 demonstrations of this task for the training dataset, randomly placing the steel pot at various locations within a small section of the table. During evaluation, each trial is recorded as a success (1), failure (0), or partial success (0.5). Partial successes include grasping the pot but not orienting it upright, or knocking it over to the upright position but not carefully guiding it. The robot must release the pot at the end of the episode for full credit.
4. **Move <object> onto Plate** (multi-instruction): The robot’s goal is to grasp one out of three objects (depending on the target specified in the language instruction) and place it on the plate on the right side of the table. We collect 150 demonstrations of this task for the training dataset, randomly placing different combinations of three objects on the table and selecting one as the target. The plate is always initialized on the right side of the table. During evaluation, each trial is recorded as a success (1), failure (0), or partial success (0.5). Partial success is recorded when the first object that the robot makes contact with is the correct target object (i.e., the object specified in the language instruction), but the robot does not complete the task.
5. **Knock <object> Over** (multi-instruction): The robot’s goal is to approach one out of three objects (depending on the target specified in the language instruction) and push it until it falls over. We collect 70 demonstrations of this task for the training dataset, randomly placing different combinations of three objects on the table and selecting one as the target. During evaluation, each trial is recorded as a success (1), failure (0), or partial success (0.5). Partial success is recorded when the first object that the robot makes contact with is the correct target object (i.e., the object specified in the language instruction), but the robot does not complete the task.
6. **Cover <object> with Towel** (multi-instruction): The robot’s goal is to grasp the blue towel and place it on one out of three objects (depending on the target specified in the language instruction). We collect 45 demonstrations of this task for the training dataset, randomly placing different combinations of three objects on the table. During evaluation, each trial is recorded as a success (1), failure (0), or partial success (0.5). Partial success is recorded when the first object that the robot touches with the towel is the correct target object (i.e., the object specified in the language instruction), but the robot does not complete the task (e.g., it drops the towel onto the table instead of on top of the target object). Full credit is given when any part of the towel is resting over the top surface of the target object, i.e., the object does not need to be fully covered.

For every Franka-Tabletop task, we evaluate each method with 10–12 in-distribution trials and 5–6 OOD generalization trials. The in-distribution and OOD test conditions are depicted in Fig. 10 (second column).

We describe the OOD test conditions for each of the six tasks below:

1. Put Carrot in Bowl (OOD): An eggplant (unseen object) replaces the carrot.
2. Pour Corn into Pot (OOD): An unseen brown tablecloth covers the tabletop.
3. Flip Pot Upright (OOD): An unseen white tablecloth covers the tabletop
4. Move <object> onto Plate (OOD): A set of three unseen objects are placed on the table.
5. Knock <object> Over (OOD): Two unseen distractor objects (red plastic cup and brown box) are positioned behind the set of three seen objects.
6. Cover <object> with Towel (OOD): The three objects on the table are placed upside-down and at unseen positions.

Finally, in the Franka-DROID environment, we experiment with one task and variants of it: **Wipe Table** (see Fig. 11). In this task, the robot’s goal is to grab the brush and sweep all three small brown objects into the dustpan. We collect 70 demonstrations for this task for the training dataset, varying the positions of all the objects.

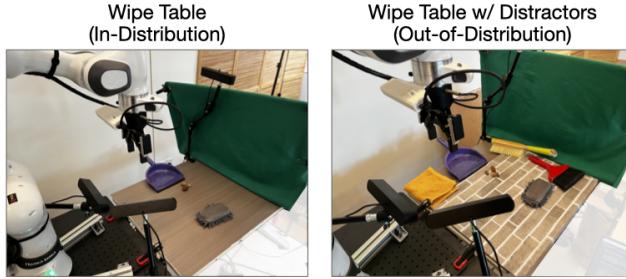


Figure 11: **Franka-DROID fine-tuning task.** The “Wipe Table” task shown here is the final task used in the data-efficient adaptation experiments in Section 5.2. The left image shows the initial conditions for an in-distribution trial. The right image shows an out-of-distribution trial in which unseen distractor objects are present on the table. To fully complete the task, the robot must grab the brush and sweep all three objects into the dustpan.

At test time, we evaluate on in-distribution conditions matching the training data (Fig. 11, left), as well as out-of-distribution (OOD) conditions in which distractor objects are also present in the scene on the table (Fig. 11, right). Since there are various possible outcomes for each trial, we define a scoring rubric as follows: The maximum score for each trial is 2 points. The policy receives the full 2 points if the robot sweeps all three objects into the dustpan. It receives 1 point for successfully sweeping one or two objects into the dustpan. Otherwise, it receives 0 points. We evaluate each policy with 18 in-distribution trials and 12 OOD trials, so each policy receives an aggregate score out of 60 points.

B.3.2 Detailed Franka-Tabletop and Franka-DROID Evaluation Results

Full evaluation results for both Franka-Tabletop and Franka-DROID evaluations are shown in Table 7. We evaluate the methods discussed in Section 5.2. We find that Diffusion Policy demonstrates strong performance on the single-instruction Franka-Tabletop tasks (e.g., “Put Carrot in Bowl” and “Pour Corn in Pot”), outperforming other methods. However, OpenVLA and Octo achieve higher performance in the more diverse multi-instruction tasks (“Move <object> onto Plate”, “Knock <object> Over”, and “Cover <object> with Towel”). In the Franka-DROID environment, OpenVLA obtains best results. Overall, we find that OpenVLA achieves the highest average performance across both tasks.

Additionally, in Table 8, we show the detailed version of the parameter-efficient fine-tuning experiment results summarized in Table 1. In these experiments, we use a representative subset of two Franka-Tabletop tasks, with both in-distribution and OOD variants: one narrow single-instruction task (“Put Carrot in Bowl”) and one diverse multi-instruction task (“Move <object> onto Plate”). We use the same number of training demonstrations used in Section 5.2 (50 and 150, respectively), which is delineated in Appendix B.3.1.

Table 7: Detailed data-efficient adaptation experiment results. Here we present the full breakdown of results summarized in Fig. 5. We report the performance of Diffusion Policy trained from scratch on new robot tasks, as well as generalist policies fine-tuned on the same data. Each policy is tested against both in-distribution and out-of-distribution (OOD) generalization conditions (see Fig. 10 for Franka-Tabletop tasks and Fig. 11 for Franka-DROID tasks). We find that no single policy performs best on all tasks: Diffusion Policy achieves high success rates on single-instruction tasks, while OpenVLA and Octo performs well on diverse multi-instruction tasks. In terms of aggregate performance, however, OpenVLA obtains the highest average success rate across both environments.

		# trials	Diffusion Policy	Diffusion Policy (matched)	Octo	OpenVLA (scratch)	OpenVLA (ours)
Franka-Tabletop (5Hz)	“Put Carrot in Bowl” (in-distribution)	10	90.0%	80.0%	40.0%	70.0%	70.0%
	“Put Carrot in Bowl” (OOD)	5	20.0%	0.0%	20.0%	0.0%	40.0%
	“Pour Corn into Pot” (in-distribution)	10	100.0%	90.0%	0.0%	10.0%	50.0%
	“Pour Corn into Pot” (OOD)	5	80.0%	60.0%	0.0%	20.0%	60.0%
	“Flip Pot Upright” (in-distribution)	10	100.0%	85.0%	40.0%	85.0%	100.0%
	“Flip Pot Upright” (OOD)	5	50.0%	20.0%	0.0%	40.0%	80.0%
	“Move <object> onto Plate” (in-distribution)	12	25.0%	25.0%	41.7%	8.3%	75.0%
	“Move <object> onto Plate” (OOD)	6	8.3%	33.3%	8.3%	33.3%	58.3%
	“Knock <object> Over” (in-distribution)	12	33.3%	25.0%	83.3%	75.0%	75.0%
	“Knock <object> Over” (OOD)	6	16.7%	16.7%	33.3%	58.3%	83.3%
	“Cover <object> with Towel” (in-distribution)	12	16.7%	20.8%	91.7%	41.7%	50.0%
	“Cover <object> with Towel” (OOD)	6	16.7%	33.3%	91.7%	50.0%	50.0%
Average			48.5±4.9%	43.4±4.7%	43.4±4.4%	43.4±4.6%	67.2±4.0%
Franka-DROID (15Hz)	“Wipe Table” (in-distribution)	18	50.0%	27.8%	52.8%	25.0%	55.6%
	“Wipe Table” + Distractors (OOD)	12	12.5%	25.0%	16.7%	16.7%	62.5%
Average			35.0±8.0%	26.7±7.5%	38.3±8.5%	21.7±6.6%	58.3±7.2%

Table 8: Detailed parameter-efficient fine-tuning experiment results. Here we present the detailed task performance results summarized in Table 1.

	# trials	Full FT	Last layer only	Frozen vision	Sandwich	LoRA, r=32	LoRA, r=64
Franka-Tabletop (5Hz)	“Put Carrot in Bowl” (in-distribution)	10	90.0	40.0	40.0	60.0	90.0
	“Put Carrot in Bowl” (OOD)	5	40.0	0.0	40.0	0.0	40.0
	“Move <object> onto Plate” (in-distribution)	12	79.2	33.3	50.0	75.0	62.5
	“Move <object> onto Plate” (OOD)	6	41.7	33.3	58.3	41.7	66.7
Average			69.7±7.2%	30.3±6.1%	47.0±6.9%	62.1±7.9%	68.2±7.5% 68.2±7.8%

C RT-2-X vs. OpenVLA in BridgeData V2 Evaluations

In this section, we provide additional details on RT-2-X vs. OpenVLA comparisons in BridgeData V2 evaluations discussed in Section 5.1. As discussed previously, OpenVLA is pretrained on a larger subset of OpenX data than RT-2-X and uses a fused SigLIP-DinoV2 vision backbone rather than a single visual encoder. However, in addition to these factors, we believe that OpenVLA’s significant improvement upon RT-2-X specifically in BridgeData V2 evaluations (as shown in Fig. 3) also stems from more careful preprocessing of the Bridge dataset.

During the development of the OpenVLA model, we discovered that the original version of the BridgeData V2 dataset contained many transitions with all-zero (no-op) actions. For instance, in every demonstration, an all-zero action was recorded as the ground-truth action in the first timestep. Consequently, training a highly expressive VLA model on the original dataset without any data preprocessing led to a policy that frequently predicted all-zero actions and froze during evaluations. Therefore, we simply filtered out the first transition in every demonstration when training the OpenVLA model, and this was sufficient for mitigating the freezing behavior in most cases.

However, the RT-2-X model was trained without such data preprocessing, so it often suffers the aforementioned freezing behavior if deployed out of the box without modifying the model querying procedure – which severely deteriorates rollout performance. Since this is a proprietary model that is infeasible for us to re-train (e.g., with our preprocessed version of the BridgeData V2 dataset), we mitigated this issue by simply querying the *second-most-likely* action from the model, since the first-most-likely action was often all zeros while the second-most-likely action was not. (Note that this is the same workaround that was applied by the developers of the RT-2-X model for BridgeData V2 evaluations reported in the Open X-Embodiment experiments [1].) This workaround led to much stronger RT-2-X performance on BridgeData V2 evaluations – though we believe that it is still suboptimal compared to re-training the model on the preprocessed version of the dataset.

We also tried to *dynamically* query RT-2-X, i.e., by first sampling the first-most-likely action and then sampling the second-most-likely action if the first one was all zeros. However, we empirically

found that dynamic querying led to worse performance than simply querying the second-most-likely action at all times. We hypothesize that this is due to a change in the robot’s dynamics that arises from dynamic querying: pausing in the middle of a trajectory to re-query the model leads to slight interruptions in the robot’s movement due to non-negligible latency in the querying pipeline, and this leads to subtle performance degradation. Therefore, we report the performance of RT-2-X when always querying the second-most-likely action, as done in the Open X-Embodiment project [1].

D Additional Experiments and Ablations

In this section, we conduct several additional experiments to analyze the effects of individual components of the OpenVLA model architecture and training scheme, as well as provide quantitative evidence for claims made in earlier sections of this work. We aim to answer the following questions:

1. How important is OpenX training and how does it impact OpenVLA’s performance ([Appendix D.1](#))?
2. What effect does using a fused SigLIP-DinoV2 vision encoder have on OpenVLA’s performance, compared to using a SigLIP-only vision encoder ([Appendix D.2](#))?
3. Is it better to fine-tune or freeze the vision encoder in OpenVLA ([Appendix D.3](#))?
4. How do the quantized inference results discussed in [Section 5.3](#) change when policy performance is disentangled from model inference speed ([Appendix D.4](#))?

We discuss the experimental setup and results addressing each of the above questions sequentially in the following sections.

D.1 OpenX Training Data Ablation Experiments

As discussed in [Section 3.3](#), OpenVLA is trained on a large dataset of robot embodiments, scenes, and tasks from the Open X-Embodiment dataset [1] (OpenX). In this section, we ablate the OpenX mixture and train a VLA policy solely on one robot dataset, to assess the impact of OpenX training on policy performance. Note that we have already observed the negative effect of ablating OpenX training in the fine-tuning regime, as discussed in [Section 5.2](#) (see OpenVLA (Scratch)), but we discuss additional experiments on another robot embodiment in this section to provide more supporting evidence.

Experimental setup and tasks. We compare the original OpenVLA model with **OpenVLA-Bridge**, which is produced by taking the same pretrained VLM as OpenVLA (Prismatic VLM [44]) and fine-tuning it solely on BridgeData V2 [6] rather than the entire OpenX training mixture discussed in [Appendix A](#). We evaluate OpenVLA and OpenVLA-Bridge on a subset of 8 representative tasks from the BridgeData V2 WidowX robot evaluation suite discussed in [Appendix B.1.1](#). The tasks are listed in [Table 9](#).

Results. Results for the OpenX training mixture ablation are shown in [Table 9](#). By comparing OpenVLA with OpenVLA-Bridge, we see that performance drops drastically (reduction of 30 percent in absolute success rate), which demonstrates the importance of OpenX pretraining on final policy performance. Although the language grounding performance is not impacted, we observe performance reduction across all generalization categories. This result suggests that the large diversity of scenes, objects, and tasks in the OpenX training mixture is essential for unlocking improved generalization capabilities in the OpenVLA model.

D.2 Dual vs. Single Vision Encoder Experiments

The OpenVLA model architecture consists of a fused vision backbone that combines the SigLIP [9] and DinoV2 [25] encoders. In this section, we ablate the DinoV2 component to assess the importance of using a dual vision encoder.

Experimental setup and tasks. We instantiate a model, **OpenVLA-Bridge-SigLIP**, which is a version of OpenVLA that is trained only on BridgeData V2 and consists of only the SigLIP encoder as the vision backbone. We compare this model with the OpenVLA-Bridge model discussed in the previous section ([Appendix D.1](#)), which shares the same model architecture as the original OpenVLA model and is only trained on Bridge robot data. Therefore, the only difference between OpenVLA-Bridge-SigLIP and OpenVLA-Bridge is that the former omits the DinoV2 encoder in the

Table 9: BridgeData V2 WidowX ablation experiment results. We evaluate various methods on a subset of 8 representative tasks to assess the importance of different components of the OpenVLA model architecture and training scheme. OpenVLA-Bridge is a version of OpenVLA without OpenX training (it is trained only on BridgeData V2), and OpenVLA-Bridge-SigLIP additionally ablates the fused vision backbone by removing the DinoV2 encoder (its vision backbone only consists of the SigLIP encoder). We observe that both OpenX training and the fused vision encoder improve policy performance, though the former has a much greater effect than the latter.

Category	Task	# Trials	OpenVLA # Successes	OpenVLA-Bridge # Successes	OpenVLA-Bridge-SigLIP # Successes
Visual gen	Put Eggplant into Pot (Easy Version)	10	10	8	8
Visual gen	Put Eggplant into Pot	10	10	2	3
Visual gen	Put Cup from Counter into Sink	10	7	4	2
Motion gen	Lift Eggplant	10	7.5	5.5	6.5
Physical gen	Put Carrot on Plate	10	8	4	1
Physical gen	Lift AAA Battery	10	7	2	2
Semantic gen	Take Purple Grapes out of Pot	10	4	3	3
Language grounding	Put {Eggplant, Red Bottle} into Pot	10	7.5	8	7
Mean Success Rate		76.3 ± 4.8%	45.6 ± 5.6%	40.6 ± 5.5%	

vision backbone. We evaluate these models on the same subset of 8 Bridge tasks described in the previous section.

Results. Results for the dual vision encoder ablation are shown in Table 9. The drop in performance from OpenVLA-Bridge to OpenVLA-Bridge-SigLIP implies that additionally including the DinoV2 encoder in the vision backbone improves policy performance. However, the 5 percent reduction in performance here is not as significant as the 30 percent drop in performance observed from ablating OpenX training. The low-level spatial features represented in DinoV2 appear to aid generalization in only some cases.

D.3 Fine-Tuned vs. Frozen Vision Encoder Experiments

As discussed in Section 3.4, prior work on VLMs observed higher performance from freezing the vision encoder than fine-tuning its parameters [44]. However, when training OpenVLA, we fine-tuned all 7B parameters in the model, including the SigLIP-DinoV2 vision backbone, as we discovered early on during development that fine-tuning the vision encoder led to higher-performing VLAs — a finding which held across various pretrained VLMs and model architectures. We discuss details of such findings below.

Experimental setup and tasks. In this section, we report the performance of two VLA policies produced by fine-tuning two different pretrained models from the Prismatic VLMs [44] repository on BridgeData V2. The two pretrained models are named **SigLIP ViT-SO 224px** and **LLaVa v1.5 7B (Reproduction)**; see Karamcheti et al. [44] for details on their architectures and training mixtures. We evaluate both policies on various Bridge tasks shown in Table 10. Note that the evaluation configurations here differ from previously discussed Bridge evaluations, so the results are not directly comparable to results in other similar experiments.

Results. Results for the fine-tuned vs. frozen vision encoder experiments are shown in Table 10. We find that for both VLAs tested, fine-tuning the vision encoder leads to significantly higher success rates across various tasks. Qualitatively, in some cases, deploying the frozen vision encoder policies leads to unstable robot behaviors that are clearly suboptimal. Consequently, we decided early on during development to not conduct further experimentation with frozen vision encoders.

D.4 Additional Quantized Inference Experiments: Disentangling Policy Performance and Model Inference Speed

In Section 5.3, we evaluated OpenVLA with different levels of precision at inference time: half precision (bfloating16), 8-bit quantization, and 4-bit quantization. 8-bit quantization led to lower BridgeData V2 performance relative to the other two approaches, and we hypothesized that the reduction in performance was caused by lower model inference speed from the operations used in 8-bit quantization. In this section, we conduct experiments to assess the veracity of this claim.

Specifically, we evaluate OpenVLA again with the three different levels of precision listed above, but now with *blocking control*. In other words, each action is fully executed on the robot before the next one is predicted by the policy and executed by the controller. This scheme controls system

Table 10: **Fine-tuned vs. frozen vision encoder experiment results.** We evaluate the performance of fine-tuning (“Fine-Tuned”) vs. freezing the vision encoder (“Frozen Vision”) in two VLA policies built on top of two different pretrained VLMs from the Prismatic VLMs [44] repository. BridgeData V2 WidowX tasks shown here are performed in the same sink environment used for other Bridge experiments in this work (however, the initial environment configurations here differ, as these evaluations were conducted at an earlier stage in the project). We find that fine-tuning the vision encoder is crucial to obtain good policy performance. Certain frozen vision encoder evaluations were discontinued due to very poor (near-zero) performance and unstable robot behaviors. Among the evaluations where both frozen vision and fine-tuned approaches are tested, fine-tuning the vision encoder leads to 80.0% average success versus 46.7% average success from leaving it frozen.

Task	# Trials	SigLIP ViT-SO 224px		LLaVa v1.5 7B (Reproduction)	
		Frozen Vision	Fine-Tuned	Frozen Vision	Fine-Tuned
		# Successes	# Successes	# Successes	# Successes
Put Eggplant into Pot	10	7	10	5	9
Put Corn on Plate	10	10	9	0	9
Mean Success Rate		85	95	25	90
Put { Eggplant, Red Bottle } into Pot	4	2	4	–	3
Put { Blue Cup, Pink Cup } on Plate	4	0	0	–	0
Lift { Cheese, Red Chili Pepper }	4	0	3	–	2
Put { Strawberry, Lime } into Pot	4	1	0	–	3
Move { Sushi, Grapes }	4	3	4	–	3
Mean Success Rate		30	55	–	55

dynamics across methods with varying amounts of latency and thus allows us to test the quality of a policy’s action predictions, independent of its prediction speed. Effectively, the precision levels that have higher throughput – bfloat16 and 4-bit quantization – are forced to run slower to match the dynamics observed when deploying OpenVLA with 8-bit precision. Therefore, we expect OpenVLA’s performance with 8-bit precision to match the performance of bfloat16 and 4-bit precision under blocking control.

Experimental setup and tasks. We report the performance of OpenVLA with blocking control and quantized inference on the same subset of 8 BridgeData V2 tasks used in [Appendix D.1](#) and [Appendix D.2](#).

Results. Quantized inference experiment results with blocking control are shown in [Table 11](#). Unlike in [Table 2](#), where 8-bit quantization led to the worst rollout performance due to low inference speed, here we observe that 8-bit quantization performs comparably to bfloat16 precision and 4-bit quantization given that we evaluate with blocking control to remove the influence of varying inference speeds on task performance. This confirms our hypothesis about the effect of inference speed on 8-bit quantization performance in previous experiments (when using non-blocking control). We also see no substantial performance degradation when using the lowest precision, 4-bit, as also observed in [Section 5.3](#).

Table 11: **Quantized inference experiment results with blocking control.** We report the success rate and standard error of OpenVLA on various BridgeData V2 WidowX tasks with bfloat16 precision (the default approach), 8-bit quantization (int8), and 4-bit quantization (int4) at inference time. All average success rates have overlapping error bars, which suggests that all methods perform comparably.

Category	Task	# Trials	bfloat16 # Successes	int8 # Successes	int4 # Successes
Visual gen	Put Eggplant into Pot (Easy Version)	10	10	10	10
Visual gen	Put Eggplant into Pot	10	9	10	10
Visual gen	Put Cup from Counter into Sink	10	5	5	3
Motion gen	Lift Eggplant	10	8	7	7.5
Physical gen	Put Carrot on Plate	10	10	10	10
Physical gen	Lift AAA Battery	10	3	6	4
Semantic gen	Take Purple Grapes out of Pot	10	2	2	2
Language grounding	Put {Eggplant, Red Bottle} into Pot	10	9	9.5	8.5
Mean Success Rate					
$70.0 \pm 5.1\%$					

E LIBERO Simulation Experiments

Our previous discussions in [Section 5.2](#) and [Section 5.3](#) focused on adapting OpenVLA to novel *real-world* robot setups and tasks. This section explores adapting OpenVLA to *simulated* robot setups and tasks, specifically utilizing the **LIBERO** benchmark [116]. Our experimentation in simulation offers two key advantages:

1. Demonstration of versatility: We show that OpenVLA, despite having been pretrained exclusively on real-world robot data, can effectively adapt to simulated domains, overcoming potential disparities between real-world and simulated environments and dynamics.
2. Enhanced accessibility and reproducibility: Integration of OpenVLA into a publicly available simulation platform makes our model more accessible to other researchers, especially those who may not have access to robotic hardware. Additionally, simulated experiments are more easily reproduced than their real-world counterparts.

We discuss the experimental setup in [Appendix E.1](#) and the results in [Appendix E.2](#). We release the materials required to reproduce the experiments along with the OpenVLA codebase.

E.1 LIBERO Simulation Experimental Setup

Simulation setup and tasks. The LIBERO benchmark [116] consists of four task suites designed for studying lifelong learning in robotic manipulation, and the original paper therefore investigates both forward and backward transfer to a variety of tasks. In our experiments, we focus solely on supervised fine-tuning on the target task suite, measuring the performance of various policies trained via behavioral cloning on successful demonstrations of the tasks.

We perform experiments with the following four task suites, which each contain 10 tasks with 50 human-teleoperated demonstrations each:

- **LIBERO-Spatial** consists of the same set of objects but different layouts, and tests the model’s understanding of spatial relationships.
- **LIBERO-Object** consists of the same scene layouts but different objects, and tests the model’s understanding of object types.
- **LIBERO-Goal** consists of the same objects and layouts but different task goals, and tests the model’s knowledge of different task-oriented behaviors.
- **LIBERO-Long** (also called **LIBERO-10**) consists of *long-horizon* tasks with diverse objects, layouts, and tasks.

We make the following modifications to each of the training datasets above:

1. To accommodate methods requiring higher-resolution images (such as $256 \times 256\text{px}$ or $224 \times 224\text{px}$), we regenerate all demonstrations at an increased resolution of $256 \times 256\text{px}$. Originally, the dataset provided by the benchmark consists of $128 \times 128\text{px}$ images. We find that simply upscaling these images to $256 \times 256\text{px}$ results in poor image quality. Therefore, we choose to begin with higher-resolution images, which can be downsampled as necessary, ensuring higher image quality across various resolution requirements. These higher-resolution images were obtained by stepping through the simulation environments with the actions stored in the provided human-collected demonstrations and saving the images rendered by the simulator.
2. We filter out all “no-op” actions from the dataset, i.e., actions that have near-zero magnitude in the translation and rotation components and do not change the state of the robot’s gripper. We find that this simple data cleaning step is crucial for highly expressive single-step policies such as OpenVLA, which otherwise learn to imitate these no-op actions and consequently freeze indefinitely at certain states during evaluation.
3. We rotate all third-person images at both train and test time by 180 degrees because we observe that the LIBERO environments return images that are upside down on our hardware.

4. Since we train policies via imitation learning, which expects demonstrations to be successful, we replay all demonstrations in the corresponding simulation environments and filter out the demonstrations that fail to complete the task (as determined by the environments’ success criteria). As a result, we remove 68 of 500 LIBERO-Spatial demonstrations, 46 of 500 LIBERO-Object demonstrations, 72 of 500 LIBERO-Goal demonstrations, and 121 of 500 LIBERO-Long demonstratinos.
5. For all methods in our comparisons, we only utilize the static third-person camera images; we do not use the wrist camera images that are additionally provided in the original datasets. This is for sake of having fair comparisons, as OpenVLA’s visual inputs only consist of third-person camera images.

Comparisons. The methods that we compare include **Diffusion Policy**⁸ [3] trained from scratch, **Octo** [5] fine-tuned on the target dataset, and **OpenVLA** fine-tuned on the target dataset via LoRA ($r = 32$) as described in [Section 5.3](#). Each policy is trained independently on each of the task suites above (rather than training a single policy on all four suites combined). All policies are trained with the same set of demonstrations, so all methods benefit from the data cleaning steps described above.

Evaluation details. To ensure lower variance in the experimental results, all methods are evaluated across 500 trials for each task suite, and the reported performance is the average success rate over three random seeds (resulting in 1500 total trials per statistic). Although we modify the training datasets, as described earlier, we do not change the test environments but rather use the same initial environment configurations provided by the original LIBERO benchmark.

E.2 LIBERO Simulation Experimental Results

We present the LIBERO experimental results in [Table 12](#). Importantly, we observe that OpenVLA can be effectively adapted to tasks in the LIBERO simulation environments, as it obtains highest average success rate and rank among the tested methods. However, we find that the overall margin between OpenVLA and the other methods are tighter here than in the real-world fine-tuning experiments discussed in [Section 5.2](#). We attribute this to the fact that OpenVLA was pretrained with purely real-world robot data and no simulation data, which suggests that fine-tuning the model on simulated robot tasks may not be as effective as fine-tuning it on real-world tasks due to the domain gap between simulated and real-world environments and dynamics. We see evidence for this notion in the results obtained by Octo – another policy pretrained on large amounts of real-world robot data – which also only achieves a small boost in overall performance relative to a simple, strong baseline such as Diffusion Policy trained from scratch. We expect increased gains in performance for the pretrained and fine-tuned methods if simulation data is added to the pretraining data mixture.

Table 12: LIBERO simulation benchmark results. We report the success rate (SR) and standard error of each method for the four task suites in the LIBERO benchmark, averaged over three random seeds with 500 trials each. In addition, we show the ranking of each method within each task suite, where a rank of 1 indicates the strongest method in the suite and a rank of 3 indicates the weakest method. (The average ranking is important to note since it informs which method may be most suitable to use as a default for a variety of tasks; it is more informative than the average success rate, which is not normalized by individual task suite difficulty.) Overall, we find that fine-tuned OpenVLA achieves highest average success rate and rank, followed by fine-tuned Octo and then Diffusion Policy trained from scratch.

	LIBERO-Spatial		LIBERO-Object		LIBERO-Goal		LIBERO-Long		Average	
	SR (\uparrow)	Rank (\downarrow)	SR (\uparrow)	Rank (\downarrow)	SR (\uparrow)	Rank (\downarrow)	SR (\uparrow)	Rank (\downarrow)	SR (\uparrow)	Rank (\downarrow)
Diffusion Policy from scratch	$78.3 \pm 1.1\%$	3	$92.5 \pm 0.7\%$	1	$68.3 \pm 1.2\%$	3	$50.5 \pm 1.3\%$	3	$72.4 \pm 0.7\%$	2.5
Octo fine-tuned	$78.9 \pm 1.0\%$	2	$85.7 \pm 0.9\%$	3	$84.6 \pm 0.9\%$	1	$51.1 \pm 1.3\%$	2	$75.1 \pm 0.6\%$	2
OpenVLA fine-tuned (ours)	$84.7 \pm 0.9\%$	1	$88.4 \pm 0.8\%$	2	$79.2 \pm 1.0\%$	2	$53.7 \pm 1.3\%$	1	$76.5 \pm 0.6\%$	1.5

⁸We use the implementation of Diffusion Policy that is described in the DROID dataset paper [11], which conditions action generation on DistilBERT [117] language embeddings of the task label.