🐙 Octo: An Open-Source Generalist Robot Policy

Octo Model Team

Dibya Ghosh[*,1] Homer Walke[*,1] Karl Pertsch[*,1,2] Kevin Black[*,1] Oier Mees[*,1]
Sudeep Dasari[3] Joey Hejna[2] Tobias Kreiman[1] Charles Xu[1] Jianlan Luo[1] You Liang Tan[1]
Dorsa Sadigh[2] Chelsea Finn[2] Sergey Levine[1]

[1]UC Berkeley [2]Stanford [3]Carnegie Mellon University

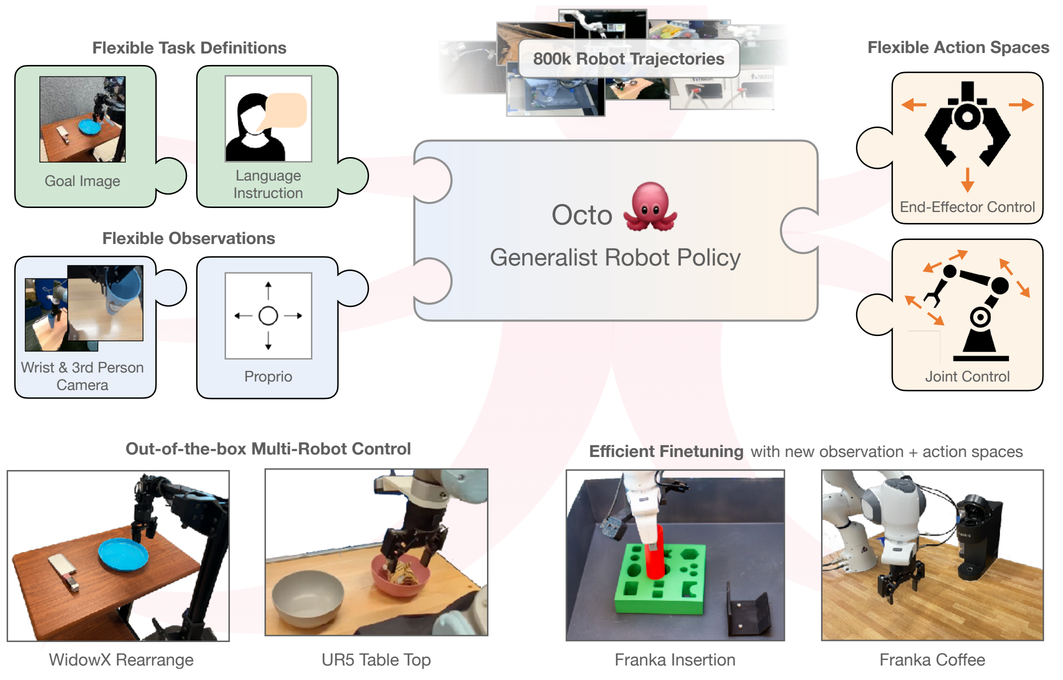

Figure 1: We introduce Octo, our ongoing effort for building open-source, widely applicable generalist policies for robotic manipulation. The Octo model is a transformer-based diffusion policy, pretrained on 800k diverse robot episodes from the Open X-Embodiment dataset. It supports flexible task and observation definitions and can be quickly finetuned to new observation and action spaces.

Abstract

Large policies, pretrained on diverse robot datasets have the potential to transform robot learning: instead of training new policies from scratch, such generalist robot policies may be finetuned with only a little in-domain data, yet generalize broadly. However, existing models restrict downstream users to the exact inputs and action spaces used during pretraining and the largest models are typically not available to the public. In this work, we aim to lay the ground work towards developing open-source, widely applicable, generalist policies for robotic manipulation. As a first step, we introduce Octo 🐙, a transformer-based diffusion policy trained on 800k robot trajectories from the Open X-Embodiment dataset. It can be instructed via language commands or goal images and can be effectively finetuned to robot setups with new sensory inputs and action spaces within a few hours on standard consumer GPUs. In experiments across 6 robotic platforms we demonstrate that Octo serves as a versatile policy initialization that can be effectively finetuned to new observation and action spaces.[2]

[*]Lead authors, ordered alphabetically, see Appendix A for list of contributions.
[2]For models & code, see `https://octo-models.github.io`

1 Introduction

The common approach for robotic learning is to train policies on datasets collected for the specific robot and task at hand. Although a simple and reliable recipe, learning from scratch requires significant data collection efforts for each task, and policies can only generalize narrowly beyond the data collection setup. In principle, collected experience from other robots and tasks offers a possible solution, exposing models to a diverse set of robotic control problems that may improve generalization and performance on downstream tasks.

However, even as general-purpose models become ubiquitous in natural language [OpenAI, 2023, Touvron et al., 2023]) and computer vision [Rombach et al., 2022, Kirillov et al., 2023], there has been little progress on the analogous "general-purpose robot model" that can control many robots for many tasks. A key reason is that training a unified control policy for robotics presents unique challenges, requiring handling different robot embodiments, sensor setups, action spaces, task specifications, environments, and compute budgets.

Recently, several works have proposed models that directly map robot observations to actions and provide zero-shot or few-shot generalization to new domains and robots. We can broadly refer to these models as "generalist robot policies" (GRPs), emphasizing their ability to predict low-level visuomotor control across tasks, environments, and robotic systems [Reed et al., 2022, Bousmalis et al., 2023, Driess et al., 2023, Zitkovich et al., 2023, Brohan et al., 2022, Shah et al., 2023a, AI, 2023, Wayve, 2023, Hu et al., 2023, Yang et al., 2023, Kumar et al., 2023]. For example, the GNM model [Shah et al., 2023b] generalizes across different robotic navigation scenarios, the RoboCat model [Bousmalis et al., 2023] handles different robot embodiments for goal-conditioned tasks, and the RT-X model [Open X-Embodiment Collaboration et al., 2023] performs language-conditioned manipulation across five robot embodiments. We believe that these generalist robot policies have the potential to transform how robot learning research is done: in the same way that current models in NLP are almost universally derived from pretrained large language models, future robot policies might be initialized from GRPs and finetuned with modest amounts of data. However, previously proposed models have been limited in multiple important aspects: they typically constrain downstream users to a pre-defined and often restrictive set of input observations, e.g., a single camera stream, they lack support for effective finetuning to new domains, and importantly, the largest models are not available to the general public.

Therefore, our aim in this work is to lay the ground work for developing open-source, widely applicable, generalist policies for robotic manipulation. As a first step, we are releasing Octo 🐙 (see Fig. 1), a transformer-based diffusion policy, pretrained on 800k robot trajectories from the Open X-Embodiment dataset [Open X-Embodiment Collaboration et al., 2023]. Octo provides high flexibility: out of the box, it supports multiple RGB camera inputs, can control various robot arms, and can be instructed via language commands or goal images. Importantly, the modular attention structure in Octo's transformer backbone allows it to be effectively finetuned to robot setups with new sensory inputs, action spaces, and morphologies, using only a small target domain dataset and accessible compute budgets.

We are releasing all resources required to train, use, reproduce, and finetune an Octo model. Concretely, we provide (1) pretrained model checkpoints with 27M and 93M parameters respectively, (2) scripts for finetuning these models on new target domains, and (3) our complete pretraining pipeline, including high-quality data loaders, transformer implementations for multimodal inputs, and tools for monitoring training progress.

The rest of this technical report summarizes our on-going efforts for building Octo, a generalist robot policy. We detail Octo's architecture, pretraining data distribution and important design decisions. We also summarize our preliminary experiments on using Octo as both, a powerful zero-shot control policy, as well as a flexible initialization for finetuning on 6 diverse robotic systems. These include new robot observation and action spaces not seen in the pretraining data. We are releasing the initial set of Octo models with this tech report, and we plan to further improve model capabilities in future releases.

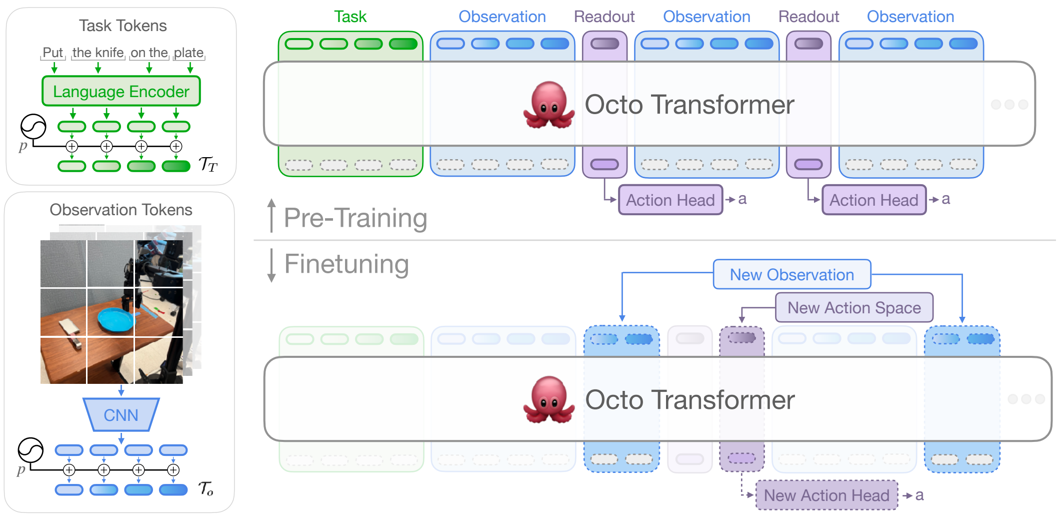

Figure 2: **Model architecture. Left**: Octo tokenizes task descriptions (green) and input observations (blue) using pretrained language models and CNNs respectively. **Top**: The transformer backbone processes the sequence of task and observation tokens and produces readout tokens (purple) that get passed to output heads to produce actions. **Bottom**: The block-wise attention structure of the transformer backbone allows to flexibly add and remove inputs and outputs during finetuning and e.g., add new observations (blue, dashed) or action spaces (purple, dashed) during finetuning.

2 The Octo Model

The design of the Octo model emphasizes flexibility and scale: the model is designed to support a variety of commonly used robots, sensor configurations, and actions, while providing a generic and scalable recipe that can be trained on large amounts of data. Octo supports both natural language instructions and goal images, observation histories, and multi-modal action distributions via diffusion decoding [Chi et al., 2023]. Furthermore, we designed Octo specifically to support efficient finetuning to new robot setups, including robots with different actions and different combinations of cameras and proprioceptive information. This design was selected to make Octo a flexible and broadly applicable generalist robot policy that can be utilized for a variety of downstream robotics applications and research projects.

2.1 Architecture

At its core, Octo is a transformer-based diffusion policy π. It consists of three key parts: **input tokenizers** that transform language instructions ℓ, goals g and observation sequences $o_1, \ldots, o_H$ into tokens $[\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o]$ (Fig. 2, left), a **transformer backbone** that processes the tokens and produces embeddings $e_l, e_g, e_o = T(\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o)$ (Fig. 2, top), and **readout heads** $R(e)$ that produce the desired outputs, i.e., actions a.

Task and observation tokenizers. We convert task definitions, e.g., language instructions ℓ and goal images g, and observations o, e.g., wrist and third-person camera streams, into a common "tokenized" format using modality-specific tokenizers (see Fig. 2, left):

- **Language inputs** are tokenized, then passed through a pretrained transformer that produces a sequence of language embedding tokens. We use the `t5-base` (111M) model [Raffel et al., 2020].
- **Image observations and goals** are passed through a shallow convolution stack, then split into a sequence of flattened patches [Dosovitskiy et al., 2020].

We assemble the input sequence of the transformer by adding learnable position embeddings p to task and observation tokens and then arranging them sequentially $[\mathcal{T}_T, \mathcal{T}_{o,1}, \mathcal{T}_{0,2}, \ldots]$.

Transformer backbone and readout heads. Once the inputs have been cast to a unified token sequence, they are processed by a transformer (see Fig. 2, top). This is similar to prior works that train transformer-based policies on sequences of observations and actions [Wu et al., 2023, Radosavovic et al., 2023]. The attention pattern of the Octo transformer is block-wise masked: observation tokens can only attend causally to tokens from the same or earlier time steps $\mathcal{T}_{o,0:t}$ and task tokens $\mathcal{T}_T$ (green), and tokens corresponding to a non-existing observations are fully masked out (e.g. a dataset without language instructions). This modular design enables us to add and remove observations or tasks during finetuning (see below). In addition to these input token blocks, we insert learned *readout tokens* $\mathcal{T}_{R,t}$ (purple). A readout token at $\mathcal{T}_{R,t}$ attends to observation and task tokens before it in the sequence, but is not attended to by *any* observation or task token – hence, they can only passively read and process internal embeddings without influencing them. A lightweight "action head" is applied on the embeddings for the readout tokens, and used for the diffusion loss.

Our design allows us to flexibly add new task and observation inputs or action output heads to the model during downstream finetuning. When adding new tasks, observations, or loss functions downstream, we can wholly retain the pretrained weights for the transformer, only adding new positional embeddings, a new lightweight encoder, or the parameters of the new head as necessitated by the change in specification (see Fig. 2, bottom).

This flexibility is crucial to make Octo a truly "generalist" robotic model: since we cannot cover all possible robot sensor and action configurations during pretraining, being able to adapt Octo's inputs and outputs during finetuning makes it a versatile tool for the robotics community. Prior model designs that use standard transformer backbones or fuse visual encoders with MLP output heads, lock in the type and order of inputs expected by the model. In contrast, switching the observation or task for Octo *does not* require re-initializing large parts of the model during finetuning.

2.2 Design Decisions

So far, we described Octo's key architectural features that enable us to scale model training to large and diverse datasets while retaining the flexibility to adapt to new task, observation and action spaces. However, there is a number of additional design decisions in which Octo deviates from common policy architectures. We next summarize our findings that motivated these choices.

Shallow vs. deep image encodings. Prior transformer-based policy designs typically encode input images with large ResNet-style [He et al., 2016] encoders and fuse multiple inputs with a comparatively small transformer after [Brohan et al., 2022, Open X-Embodiment Collaboration et al., 2023, Shah et al., 2023a, Chi et al., 2023, Zhao et al., 2023, Mees et al., 2022, Shridhar et al., 2023]. Instead, we opt for a "transformer-first" architecture that uses very shallow CNN patch encoders and concentrates most of the parameters and FLOPS in the transformer backbone for jointly processing all inputs. Empirically, we found this to lead to better-performing policies at scale, potentially because the model can perform most of the processing for all tasks and observations *jointly* using the scalable transformer backbone.

Early vs. late input fusion. While we strive to have an input encoding that is as simple as possible to allow most of the processing to happen in the transformer backbone, there is an inherent trade-off with training speed: since transformer compute requirements scale quadratic in the context length [Vaswani et al., 2017], encoding each input separately results in a large number of tokens and slows training and inference. For incorporating goal images, we make a pragmatic choice and *channel-stack* goal images, if provided, with the observation images before patch tokenization. This "early fusion" matches design decisions with other prior work that train robotic goal-conditioned policies [Shah et al., 2023a, Walke et al., 2023]. Empirically, we found this to work better than fully channel-stacking all input observations, while fully "flattening" all inputs was computationally prohibitive.

Pretrained encoders. It is common to initialize image encoders with weights pretrained on large, non-robotic datasets [Dasari et al., 2023, Brohan et al., 2022, Open X-Embodiment

Collaboration et al., 2023] such as ImageNet [Deng et al., 2009]. In our experiments so far, we found ImageNet-pretrained ResNet encoders to provide no performance improvement over encoders trained from scratch and thus opt for the latter for simplicity, though we believe that more investigation into alternative pretrained representations is needed.

We include a list of other findings of "what worked" and "what did not work" in Appendix E.

3 Training Details

Training data. We train Octo on a mixture of 25 datasets from the Open X-Embodiment Dataset [Open X-Embodiment Collaboration et al., 2023], a diverse collection of robot learning datasets. Our training mixture includes data from a variety of robot embodiments, scenes, and tasks. These datasets are heterogeneous not just in terms of the robot type, but also in the sensors (e.g., including or not including wrist cameras) and labels (e.g., including or not including language instructions). See Fig. 3, Appendix C for the detailed mixture. To create our training mixture D, we first removed all Open-X datasets that contain no image streams and those that do not use delta end-effector control. We then rank the remaining datasets in terms of their diversity and task relevance and remove datasets that are too repetitive, have a low image resolution, or are excessively niche tasks. For the remaining datasets, we roughly categorize them into "more diverse" and "less diverse" datasets based on the tasks and environments, and then double the weight of the more diverse datasets during training. We also down-weight a few large datasets with many data points to balance the mixture. Finally, we zero-pad any missing camera channels and align the gripper action spaces between the datasets such that a gripper command of +1 means "the gripper is open" and 0 means "the gripper is closed." While we found the resulting training mixture to work well, future work should perform more thorough analysis on what constitutes a good data mixture for pretrianing such general-purpose models.

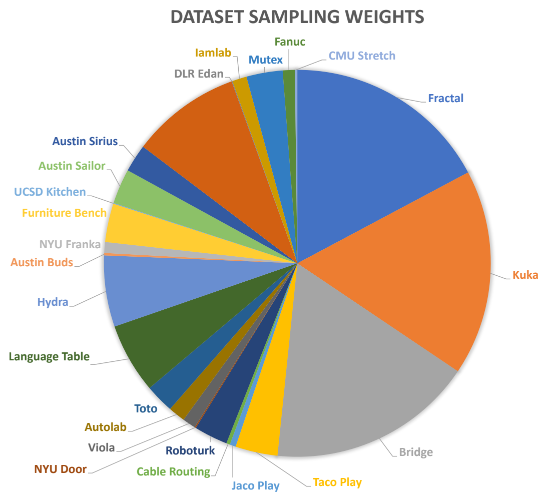

Figure 3: **Training dataset composition.** We curate a subset of 25 datasets from the Open X-Embodiment dataset that have image observations, end-effector actions and show diverse behaviors. The pie chart visualizes the fractions that each dataset contributes to every training batch on average. The dataset weights are determined by the number of samples in each dataset with small modulations to balance dataset size and diversity (see main text for details).

Training objective. We use a conditional diffusion decoding head to predict continuous, multi-modal action distributions [Ho et al., 2020, Chi et al., 2023]. Importantly, only one forward pass of the transformer backbone is performed per action prediction, after which the multi-step denoising process is carried out entirely within the small diffusion head. We found this policy parametrization to outperform policies trained with MSE action heads or discretized action distributions [Brohan et al., 2022] in both zero-shot and finetuning evaluations. To generate an action, we sample a Gaussian noise vector $x^K \sim \mathcal{N}(0, I)$ and apply K steps of denoising with a learned denoising network $\epsilon_\theta(x^k, e, k)$ that is conditioned on the output x^k of the previous denoising step, the step index k, and the output embedding e of the transformer action readout:

$$x^{k-1} = \alpha(x^k - \gamma\epsilon_\theta(x^k, e, k) + \mathcal{N}(0, \sigma^2 I)). \tag{1}$$

The hyperparameters α, γ and σ correspond to the noise schedule: we use the standard cosine schedule from Nichol and Dhariwal [2021]. We train the diffusion head using the standard DDPM objective first proposed in Ho et al. [2020], where we add Gaussian noise to the dataset actions and train the denoising network $\epsilon_\theta(x^k, e, k)$ to reconstruct the original action. For a detailed explanation of diffusion policy training, see Chi et al. [2023]. We list all used hyperparameters in Appendix D.

Finetuning. We use the same diffusion training objective during finetuning and update the full model, which we found to work better than training only the action head. In all finetuning experiments, we employ the same recipe: given a small target domain dataset with around 100 trajectories, we finetune for 50k steps using a cosine learning rate decay schedule with linear warmup.

Model sizes, infrastructure & data augmentation. We trained two variants of our model: Octo-Small with a transformer backbone that mirrors the size of a ViT-S, and Octo-Base with a transformer backbone that mirrors the size of a ViT-B [Dosovitskiy et al., 2020]. We use the AdamW optimizer [Loshchilov and Hutter, 2017] with an inverse square root decay learning rate schedule [Zhai et al., 2022], with weight decay of 0.1 and gradient clipping of 1.0. The ViT-B was trained for 300k steps with a batch size of 2048 using a TPU v4-128 pod, which took 14 hours. A finetuning run of the same model on a single NVIDIA A5000 GPU with 24GB of VRAM takes approximately 5 hours and can be sped up with multi-GPU training. We apply common image data augmentations during training and use hindsight goal relabeling [Andrychowicz et al., 2017] with randomly sampled future observations. We further randomly zero out the language instruction or goal image per training example to enable Octo to be conditioned on *either* language instructions *or* goal images. For datasets without language annotations, we always use goal image conditioning. This enables our model to learn control mostly from self-supervised visual observations and reduces the burden on language annotation, similar to prior work on multi-context imitation learning [Lynch and Sermanet, 2021]. For more details on the choice of hyperparameters, see Appendix D.

4 Model Checkpoints & Code

We open-source all resources required to train, finetune and run our model (see `https://octo-models.github.io`):

- **Pretrained Octo checkpoints** for Octo-Small (27M params) and Octo-Base (93M params)
- **Finetuning scripts** for Octo models, in JAX
- **Model pretraining pipeline** for Octo pretraining on the Open X-Embodiment dataset, in JAX
- **Standalone data loaders** for Open X-Embodiment data, compatible with JAX and PyTorch pipelines

We provide a simple example for loading and inferencing a pretrained Octo model in Appendix B.

5 Experiments

The goal of our experiments is to answer the following questions:

1. Can Octo control multiple robot embodiments and solve language- and goal-conditioned tasks out of the box?
2. Does Octo serve as a strong initialization for data-efficient finetuning to new tasks and robots, and does it improve over training from scratch and commonly used pretrained representations?
3. Does Octo's compositional design allow finetuning to new observation and action spaces?

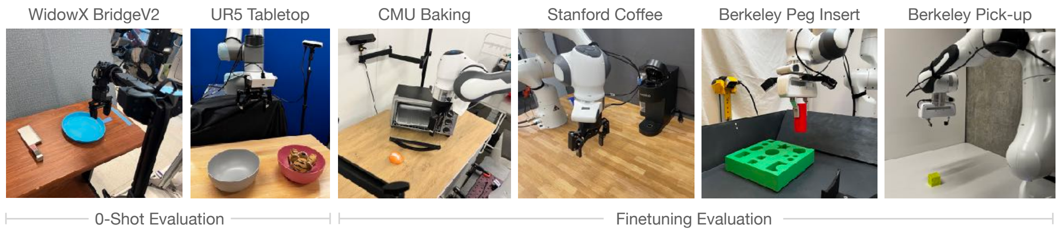

Figure 4: **Evaluation Tasks.** We evaluate Octo on 6 real robot setups across 3 institutions. Our evaluations capture diverse object interactions (e.g., “WidowX BridgeV2”), long task horizons (e.g., “Stanford Coffee”) and precise manipulation (e.g., “Berkeley Peg Insert”). We evaluate Octo’s capabilities to control robots in environments from the pretraining data out-of-the-box and to efficiently finetune to new tasks and environments with small target domain datasets. We also test finetuning with new observations (force-torque inputs for “Berkeley Peg Insert”) and action spaces (joint position control in “Berkeley Pick-up”).

Evaluation setups. We evaluate Octo’s capabilities across a representative spectrum of 6 robot learning setups at 3 institutions (see Fig. 4). We test Octo’s ability to control different robots out-of-the-box (“zero-shot”) for language and goal image tasks using robot setups from the pretraining data. We further evaluate Octo for data-efficient finetuning to new environments and tasks, including using new observations (force-torque inputs in “Berkeley Peg Insert”) and new action spaces (joint position control in “Berkeley Pick-up”). Each of the finetuning setups uses ~ 100 in-domain demonstrations and finetunes in < 5 hours on a NVIDIA A5000 GPU, using the same hyperparameters across all setups (see Appendix D). Our evaluation tasks test Octo’s capability to interact with diverse objects (e.g., “WidowX BridgeV2”), solve long-horizon tasks (e.g., “Stanford Coffee”) and perform precise manipulations (e.g., “Berkeley Peg Insert”). For more details on each evaluation setup, see Appendix F.

Comparisons. We compare Octo’s ability to control multiple robots out-of-the-box to the best openly available generalist robot policy, **RT-1-X** [Open X-Embodiment Collaboration et al., 2023], using the released checkpoint. Similar to Octo, RT-1-X is pretrained on a diverse robot dataset and aims to control multiple robots zero-shot, thus forming a natural point of comparison. We further compare Octo’s performance as a policy initialization for data efficient finetuning to the two most common approaches: (1) training on the target domain demonstrations *from scratch* and (2) using pretrained representations. For training from scratch, we found our large transformer architecture to overfit quickly on the small datasets. Instead, we obtained better from-scratch results using a canonical policy architecture employed by many prior works: a ResNet visual encoder with FiLM [Perez et al., 2018] language conditioning, combined with a smaller transformer action decoder (**“ResNet+Transformer Scratch”**), similar to e.g., [Brohan et al., 2022, Zhao et al., 2023, Chi et al., 2023, Lynch et al., 2023]. The default action head was replaced with the same diffusion decoder head used by our method. For the second comparison, we use **VC-1** pretrained representations [Majumdar et al., 2023], combined with an MLP action decoder. VC-1 is a state-of-the-art visual representation pretrained on 4,000 hours of ego-centric videos and ImageNet. We initialized a ViT visual encoder with the VC-1 weights and finetuned the full network to predict expert actions, using an MSE loss.

5.1 Octo Controls Multiple Robots Out-of-the-box

We show the comparison of zero-shot manipulation capability of Octo and RT-1-X in Fig. 5. While both methods are able to solve a diverse range of tasks in the pretraining environments, we find that Octo on average has 33% higher success rate than RT-1-X, the current state-of-the-art, openly available generalist robot policy (35M parameters). For the WidowX evaluations we also compare to existing numbers for RT-2-X [Zitkovich et al., 2023], a 55 billion parameter vision-language model finetuned on the Open X Embodiment dataset to

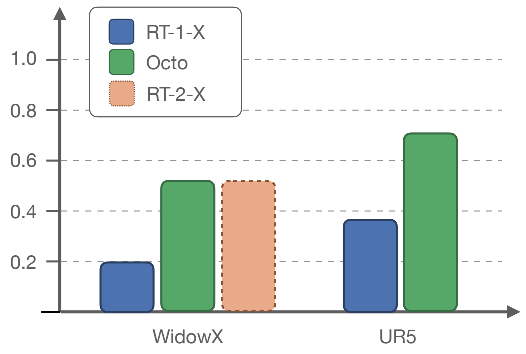

Figure 5: **Zero-Shot Evaluation.** Out-of-the-box, Octo can control multiple robots in environments from the pretraining data. When using natural language to specify tasks, it outperforms RT-1-X [Open X-Embodiment Collaboration et al., 2023], the current best, openly available generalist robot policy across two different robot embodiments and setups and performs similar to RT-2-X [Zitkovich et al., 2023] on the tested WidowX tasks.

produce robot actions[3]. Additionally, while RT-1-X and RT-2-X only support conditioning on language instructions, Octo additionally supports conditioning on goal images. We evaluated our model on the WidowX tasks using goal image conditioning and found that it achieved a 25% higher success rate than when evaluated with language conditioning. This is likely because goal images provide more information about how to achieve the task.

5.2 Octo Enables Data-Efficient Learning in New Domains

	CMU Baking	Stanford Coffee	Berkeley Peg Insert*	Berkeley Pick-up†	Average
ResNet+Transformer Scratch	25%	45%	10%	0%	20%
VC-1 [Majumdar et al., 2023]	30%	0%	5%	0%	9%
Octo (Ours)	**50%**	**75%**	**70%**	**60%**	**64%**

Table 1: **Finetuning Evaluation.** Octo enables data-efficient finetuning to new domains and out-performs training from scratch as well as state-of-the-art pretrained visual representations. Each domain uses $\sim$ 100 target demonstrations and the same hyperparameters across all domains. $*$: New observation input (force-torque proprioception). †: New action space (joint position control).

We report data-efficient finetuning results to new domains in Table 1. We find that finetuning Octo leads to better policies than starting from scratch or with the pretrained VC-1 weights, with an average success rate improvement of 55% across the four evaluation tasks. Importantly, we use the same finetuning recipe for all evaluation tasks (see Section 3), making this a good default configuration for Octo finetuning.

The results also underline Octo's ability to accommodate new observations (force-torque inputs for "Berkeley Peg Insert") and action spaces (joint position control for "Berkeley Pick-up"). This makes Octo applicable to a wide range of robot control problems that go beyond a single camera input and end-effector position control.

6 Discussion

While we demonstrated Octo's strong performance in both zero-shot and finetuning evaluations, we find that the current model still has several short-comings, which we attribute in large parts to characteristics of the training data.

On the one hand, we find that the current Octo model struggles with adequately processing wrist camera information, and often finetuning results were stronger when using only a third person camera instead of combining third person and wrist camera. A likely reason is the lack

[3]We report RT-2-X numbers from Black et al. [2023] and test our model on the same tasks for Bridge, as the RT-2-X model is not openly available.

of wrist camera inputs in the pretraining data: only 27% of the data contains wrist camera information, making it likely that the wrist camera encoders are under-trained. Adding more data with wrist cameras or weight sharing between wrist and third person camera encoders may be able to improve performance.

Additionally, we notice a large difference between language-conditioned policy performance and goal-conditioned policy performance. Again, only 56% of the pretraining data contains language annotations, which may contribute to the lower performance of the language conditioned policy. Beyond adding more language-annotated data to the pretraining mix, there is room to explore alternative approaches for fusing language instruction information into the policy, e.g., cross-attention between observation and language instruction features.

7 Conclusion and Future Plans

We introduced Octo, our ongoing effort towards building generalist robotics models. As a first step, we have released the Octo model, a large transformer-based diffusion policy, pretrained on 800k robot trajectories. We demonstrated that Octo can solve a variety of tasks out-of-the-box and showed how Octo's compositional design enables finetuning to new inputs and action spaces, making Octo a versatile initialization for a wide range of robotic control problems. Apart from the model itself, we have released our full training and finetuning code, as well as a number of tools that make it easier to train on large robot datasets.

While this release marks an important milestone for us, there remains work to improve the Octo model — towards better language conditioning, support for wrist cameras, and data beyond optimal demonstration data — which we hope to incorporate into updated models in the near future. We hope that these models offer a simple launchpad for researchers and practitioners to access larger robotic datasets, and to use pretrained robotics models in a way that allows for efficient learning of new tasks and broad generalization.

Acknowledgements

We are grateful to the Google Research Cloud for providing the compute used in this project. We thank Lawrence Chen and Ilija Radosavovic for their help in setting up the UR5 and Franka Picking robot environments. We are grateful to Kyle Stachowitz for helping to brainstorm the model design, Vivek Myers for help with the initial design of Fig. 1, Andre He for help with the initial integration of language encoders into the Octo codebase and Lucy Shi for help with running robot evaluations at Stanford.

References

OpenAI. GPT-4 Technical Report, March 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models, February 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, April 2022.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment Anything, April 2023.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning*, 2023a. URL https://arxiv.org/abs/2306.14846.

Scale AI. Introducing scale's automotive foundation model, 2023. URL https://scale.com/blog/afm1.

Wayve. Lingo: Natural language for autonomous driving, 2023. URL https://wayve.ai/thinking/lingo-natural-language-autonomous-driving/.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving, 2023.

Jonathan Heewon Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=HEIRj51lcS.

Vikash Kumar, Rutav Shah, Gaoyue Zhou, Vincent Moens, Vittorio Caggiano, Abhishek Gupta, and Aravind Rajeswaran. Robohive: A unified framework for robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=0H5fRQcpQ7.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A General Navigation Model to Drive Any Robot. In *International Conference on Robotics and Automation (ICRA)*. arXiv, May 2023b. doi: 10.48550/arXiv.2210.03370.

Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *International Conference on Machine Learning*, 2023.

Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. *Conference on Robot Learning*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

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

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale, 2023.

Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021.

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

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *arXiv preprint arXiv:2303.18240*, 2023.

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. *arxiv*, 2023.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task agnostic offline reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. Grounding language with visual affordances over unstructured data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. MUTEX: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023c. URL https://openreview.net/forum?id=PwqiqaaEzJ.

Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on Robot Learning (CoRL)*, 2022.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. RoboTurk: A crowdsourcing platform for robotic skill learning through imitation. *CoRR*, abs/1811.02790, 2018. URL http://arxiv.org/abs/1811.02790.

Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, Chelsea Finn, and Abhinav Gupta. Train offline, test online: A real robot learning benchmark, 2023.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. https://sites.google.com/view/berkeley-ur5/home.

Saumya Saxena, Mohit Sharma, and Oliver 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.

Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors, 2023a.

Xinghao Zhu, Ran Tian, Chenfeng Xu, Mingyu Ding, Wei Zhan, and Masayoshi Tomizuka. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot. 2023b.

Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

Kris Wu Ge Yan and Xiaolong Wang. ucsd kitchens Dataset. August 2023.

Shivin Dass, Jullian Yapeter, Jesse Zhang, Jiahui Zhang, Karl Pertsch, Stefanos Nikolaidis, and Joseph J. Lim. CLVR jaco play dataset, 2023. URL https://github.com/clvrai/clvr_jaco_play_dataset.

Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and Sergey Levine. Multi-stage cable routing through hierarchical imitation learning. *arXiv preprint arXiv:2307.08927*, 2023a.

Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *CoRL*, 2023.

Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation, 2021.

Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Joern Vogel. Shared Control Templates for Assistive Robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, page 7, Paris, France, 2020.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *NeurIPS*, 2019.

Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. FMB: A functional manipulation benchmark for generalizable robotic learning. https://functional-manipulation-benchmark.github.io, 2023b.

Yixin Lin, Austin S. Wang, Giovanni Sutanto, Akshara Rai, and Franziska Meier. Polymetis. https://facebookresearch.github.io/fairo/polymetis/, 2021.

A Contributions

Dibya Ghosh: led model development, proposed and implemented large parts of the final model design, babysitted the training runs and touched all parts of the codebase, helped with model evaluations and tech report writing.

Homer Walke: led model evaluations, designed the main Bridge evaluation benchmark, contributed to the initial model implementation and ran many of the evals for this tech report.

Karl Pertsch: managed the overall project, led Open-X data integration and curation, led writing of this tech report, contributed to model development and implementation and ran model evaluations for the tech report.

Kevin Black: led data loading and training infrastructure, managed TPU pod training, created the project website, contributed to model development and implementation, and helped with robot evaluations and tech report writing.

Oier Mees: ran countless ablations for model development, contributed to model implementation, helped with evals and writing of the tech report.

Sudeep Dasari: contributed the model evaluations at CMU, experimented with pretrained encoders.

Joey Hejna: contributed the model evaluations at Stanford.

Tobias Kreiman: contributed model evaluations in simulated environments.

Charles Xu: contributed the model evaluations for Berkeley Peg Insert.

Jianlan Luo: contributed the model evaluations for Berkeley Peg Insert.

You Liang Tan: helped diagnose and resolve bottlenecks in data loading.

Dorsa Sadigh, Chelsea Finn, Sergey Levine: provided guidance throughout the project and feedback on the writing of this tech report.

B Octo Code Example

Loading a pretrained Octo model and performing inference requires little code:

```python
import jax
from octo.model.octo_model import OctoModel

model = OctoModel.load_pretrained("hf://rail-berkeley/octo-base")
print(model.get_pretty_spec()) # Print out the input-output spec
observation = {"image_primary": img}
task = model.create_tasks(texts=["pick up the fork"])
action = model.sample_actions(
    observation, task, rng=jax.random.PRNGKey(0))
```

Listing 1: Example Python code to perform inference with a pretrained ORCA model.

C Data mixture

We list the detailed training mixture used for training the Octo models in Table 2. The sampling weights are mostly determined by the relative size of the datasets with a few manual adjustments (see Section 3). We rank the datasets of the Open X-Embodiment dataset [Open X-Embodiment Collaboration et al., 2023] in terms of their diversity and task relevance and remove datasets that are too repetitive, have a low image resolution, or are excessively niche tasks. We also down-weight a few large datasets with many data points to balance the mixture.

OctoPretraining Dataset Mixture	
Fractal [Brohan et al., 2022]	17.0%
Kuka [Kalashnikov et al., 2018]	17.0%
Bridge [Walke et al., 2023]	17.0%
BC-Z [Jang et al., 2022]	9.1%
Stanford Hydra Dataset [Belkhale et al., 2023]	6.0%
Language Table [Lynch et al., 2023]	5.9%
Taco Play [Rosete-Beas et al., 2022, Mees et al., 2023]	3.6%
Furniture Bench Dataset [Heo et al., 2023]	3.3%
UTAustin Mutex [Shah et al., 2023c]	3.0%
Austin Sailor Dataset [Nasiriany et al., 2022]	2.9%
Roboturk [Mandlekar et al., 2018]	2.8%
Toto [Zhou et al., 2023]	2.4%
Austin Sirius Dataset [Liu et al., 2023]	2.3%
Berkeley Autolab UR5 [Chen et al.]	1.5%
IAMLab CMU Pickup Insert [Saxena et al., 2023]	1.2%
Viola [Zhu et al., 2023a]	1.2%
Berkeley Fanuc Manipulation [Zhu et al., 2023b]	1.0%
NYU Franka Play Dataset [Cui et al., 2022]	0.9%
UCSD Kitchen Dataset [Ge Yan and Wang, 2023]	<0.1%
Jaco Play [Dass et al., 2023]	0.6%
Berkeley Cable Routing [Luo et al., 2023a]	0.3%
Austin Buds Dataset [Zhu et al., 2022]	0.3%
CMU Stretch [Mendonca et al., 2023]	0.2%
NYU Door Opening [Pari et al., 2021]	0.1%
DLR EDAN Shared Control [Quere et al., 2020]	0.1%

Table 2: Octo pretraining data mixture using datasets from the Open X-Embodiment dataset [Open X-Embodiment Collaboration et al., 2023].

D Training Hyperparameters

We mostly follow documented practices for training vision transformers [Zhai et al., 2022]. We use the AdamW optimizer [Loshchilov and Hutter, 2017] with an inverse square root decay learning rate schedule [Zhai et al., 2022] and learning rate warm-up. We list hyperparamaters used during training in Table 3 and the model parameters for the different sizes in Table 4. We apply standard image augmentations during training. Concretely, for the 3rd person camera we apply stochastic crops followed be a resize to 256 $\times$256, followed by color jitter. Finally, we normalize the input image to have pixels with float values between -1.0 and 1.0. For the wrist camera, we apply the same procedure except without the random crop and resizing to 128 $\times$128 instead.

Hyperparameter	Value
Learning Rate	3e-4
Warmup Steps	2000
LR Scheduler	reciprocal square-root
Weight Decay	0.1
Gradient Clip Threshold	1
Batch Size	2048

Table 3: Hyperparameters used during training.

The images are passed through a shallow convolution stack, then split into a sequence of flattened patches [Dosovitskiy et al., 2020] of size 16 $\times$16. This results in 256 tokens for the 3rd person camera images and 64 tokens for the wrist camera images. For datasets containing language annotations, we use a pretrained `t5-base` (111M) transformer model [Raffel et al., 2020] that produces a sequence of 16 language embedding tokens.

Model	Layers	Hidden size D	MLP size	Heads	Params
Octo-Small	12	768	3072	12	27M
Octo-Base	24	1024	4096	16	86M

Table 4: Details of Octo model variants.

The diffusion action head is characterized by a 3-layer MLP with a hidden dimension of 256, residual connections, and layer normalization. During training we use the standard DDPM objective as introduced by [Ho et al., 2020] with a cosine noise schedule [Nichol and Dhariwal, 2021]. During both training and inference, we use 20 diffusion steps.

E Things that Worked and Did Not Work (Yet)

Things we found improved performance:

- **Adding history during pretraining**: Models with one frame of history as context performed better in zero-shot evals than models pretrained without history. We did not observe benefits of increasing the history length further on the few tasks we evaluated on, though other tasks may benefit.
- **Using action chunking**: We found it helpful to use "action chunking" [Zhao et al., 2023], i.e., to predict multiple actions into the future, for getting more coherent policy movements. We did not find temporal ensembling of future actions to provide additional benefits in the finetuning tasks we tested.
- **Decreasing patch size** Tokenizing images into patches of size 16×16 led to improved performance over patches of size 32×32, particularly for grasping and other fine-grained tasks. This does add compute complexity (the number of tokens is $4\times$), so understanding how to balance compute costs and resolution remains a problem of interest.
- **Increasing shuffle buffer size**: Loading data from 25 datasets in parallel is a challenge. Specifically, we found that achieving good shuffling of frames during training was crucial — zero-shot performance with a small shuffle buffer (20k) and trajectory-level interleaving suffered significantly. We solved this issue by shuffling and interleaving frames from different trajectories *before* decoding the images, allowing us to fit a much larger shuffle buffer (up to 500k). We also subsample at most 100 randomly chosen steps from each training trajectory during data loading to avoid "over-crowding" the shuffle buffer with single, very long episodes.

Things that did not work (yet):

- **MSE Action Heads**: i.e., replacing our diffusion decoding head with a simple L2 loss, lead to "hedging" policies that move very slowly and e.g., fail to rotate the gripper in Bridge evals.
- **Discrete Action Heads**: i.e., discretizing actions into 256 bins per dimension and training with cross-entropy loss like in Brohan et al. [2022]; lead to more "decisive" policies, yet often observe early grasping issues.
- **ResNet Encoders**: train faster as they compress the image into fewer tokens, but attain worse zero-shot performance.
- **Pretrained Encoders**: ImageNet pretrained ResNet encoders did not provide benefit on zero-shot evals, though may be confounded with ResNet architectures underperforming as mentioned above.
- **Relative Gripper Action Representation**: when aligning the gripper action representations of the different datasets, we tried (A) absolute gripper actions, i.e.,

actions are +1 when the gripper is open and -1 if it is closed, and (B) relative gripper actions, i.e., gripper action is +1/-1 only in the timestep when the gripper opens/closes. We found that the latter tends to open/close grippers less often since most of the training data represents "do not change gripper" actions, leading to a slightly higher grasp success rate. At the same time, the relative representation led to less retrying behavior after a grasp failed, which was ultimately worse. Thus, we chose the absolute gripper action representation.

- **Adding Proprioceptive Inputs**: resulting policies seemed generally worse, potentially due to a strong correlation between states and future actions. We hypothesize this might be due to a causal confusion between the proprioceptive information and the target actions [de Haan et al., 2019].
- **Finetuning Language Model**: In order to improve the visuo-lingual grounding of Octo we experimented with: i) varying sizes of the T5 encoder [Raffel et al., 2020] `small` (30M), `base` (111M), `large` (386M) and ii) finetuning the last two layers of the encoder. While using the `base` model resulted in better language-conditioned policies, we did not find improvements when using even larger encoders or finetuning the encoder. We hypothesize this might be due to the lack of rich, diverse, free-form language annotations in most of the datasets.

F Experimental Setups

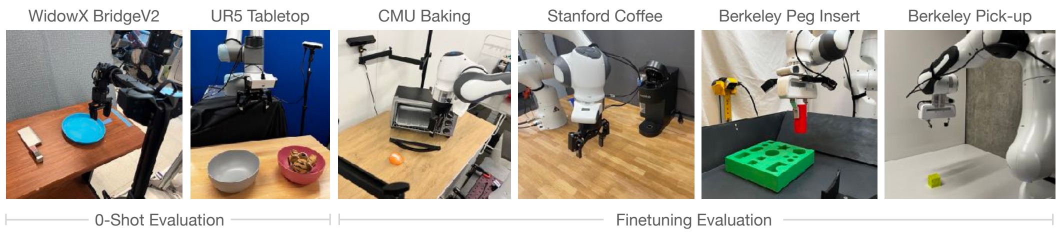

Figure 6: **Evaluation Tasks.** Replicated from the main text for convenience. We evaluate Octo on 6 real robot setups across 3 institutions in zero-shot and finetuning scenarios.

F.1 Zero-Shot Evaluations

WidowX BridgeV2. Uses the setup of Walke et al. [2023], in which a Trossen WidowX robot performs diverse table top manipulation tasks. Concretely, we evaluate on two tasks in which a the robot needs to "place carrot on plate" and "put eggplant in the pot". Both tasks are challenging since they are out of distribution of the Bridge pre-training data and require generalization to new objects. The robot observation consists of a single third person camera stream and the action space are end-effector velocity actions.

UR5. Uses the setup of Chen et al.. A UR5 robot arm performs multiple table top manipulation tasks, namely picking a toy tiger from a bowl and placing it into a different bowl and wiping a table with a cloth. The task requires generalization over initial positions and, since the training data was collected months ago, miscellaneous changes in the environment. Policies are trained with a single third-person camera input and predict end-effector velocities.

F.2 Finetuning Evaluations

CMU Baking. The robot must pick up the toy bread object, place it in the toaster, and shut the toaster. This task requires generalization across initial positions (of both the toaster and object) and the shape of the target toy bread object. We use an eef velocity (Cartesian pos + rotation delta) action space. Observations come from the 3rd person front-facing Zed camera. Actions are predicted at 15 Hz, and executed on the robot using the R2D2 Franka controller. The finetuning dataset consists of 120 demos collected via expert VR tele-operation, and every policy was evaluated using 20 trials (4 novel test objects w/ 5 positions each).

Stanford Coffee. The robot is tasked with picking up one of four different Keurig Coffee Pods and placing it inside of Keurig machine. This task requires both generalization across initial positions and colors of the coffee pod, and precision placement in the Keurig machine. We use a cartesian delta and rotation end-effector space with an open source controller running at 10 Hz based on polymetis found here. We use only a single third-person wrist observation. Our training dataset contained 118 expert demonstrations from varied coffee pods and positions collected via VR tele-operation. We evaluated policies for 20 episodes, five episodes for each of four different color coffee pods.

Berkeley Peg Insertion. The task is for a robot to insert a pre-grasped 3d-printed peg into a matching slot on a 3d-printed board inside the bin, as pictured in Fig. **??**. The matching tolerance between the peg and the hole is 1.5mm; which makes it a contact-rich precise part-mating task. The robot must learn an appropriate policy to "search" for the matching opening through contact, which necessitates the use of relevant input modalities such as external force/torque measurements. The observation space of the policy consists of a single side-view camera image, the end-effector twist, and the end-effector force/torque reading. The policy sends action commands as the robot's end-effector twists at 5 HZ, tracked at 1000 HZ by a low-level impedance controller. Our finetuning dataset is composed of 100 human demonstrations from the FMB dataset [Luo et al., 2023b], we evaluated trained policies for 20 trials with randomized board positions.

Berkeley Pick Up. We use the setup of Radosavovic et al. [2023]: the robot needs to pick up a block from a table top surface after being trained on a dataset of 100 pickups of various objects. The robot uses joint position control with an underlying Polymetis control stack [Lin et al., 2021]. It is conditioned on a wrist camera input image as well as the proprioceptive readings of the robot.