Octo: An Open-Source Generalist Robot Policy

Octo: An Open-Source Generalist Robot Policy

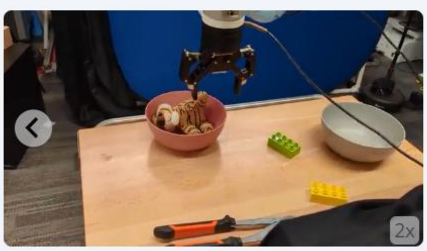
Octo Model Team

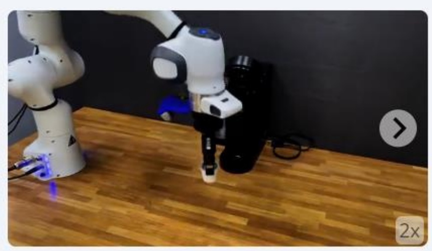
Dibya Ghosh*[1] Homer Walke*[1] Karl Pertsch*[1,2] Kevin Black*[1] Oier Mees*[1]
Sudeep Dasari[3] Joey Hejna[2] Tobias Kreiman[1] Charles Xu[1] Jianlan Luo[1] You Liang Tan[1]
Lawrence Yunliang Chen[1] Pannag Sanketi[4] Quan Vuong[4] Ted Xiao[4] Dorsa Sadigh[2]
Chelsea Finn[2] Sergey Levine[1]

*denotes equal contribution, listed in alphabetical order

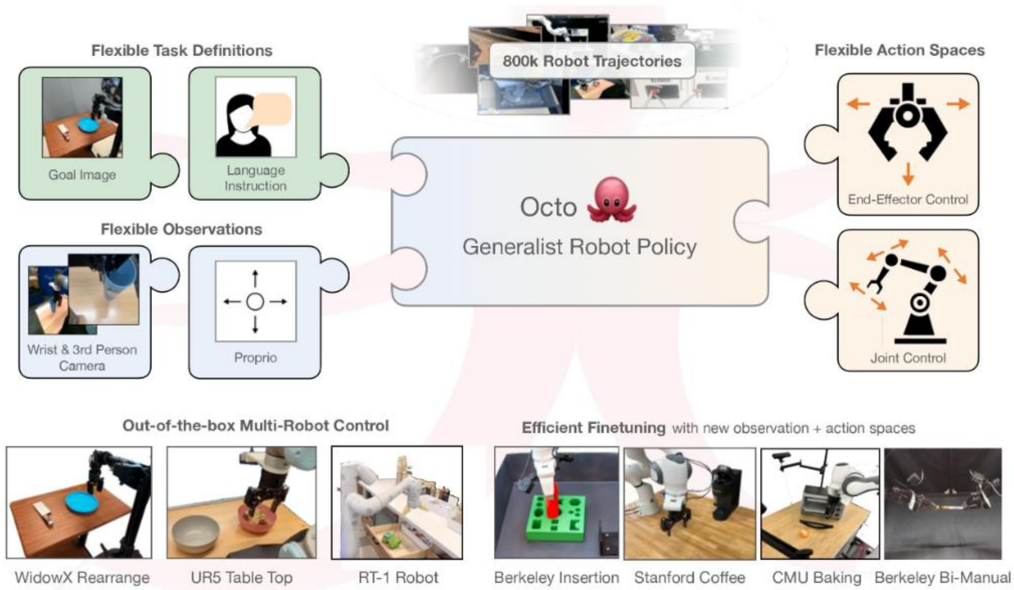
1. UC Berkeley 2. Stanford University 3. Carnegie Mellon University
4. Google DeepMind

Report Code Colab Weights



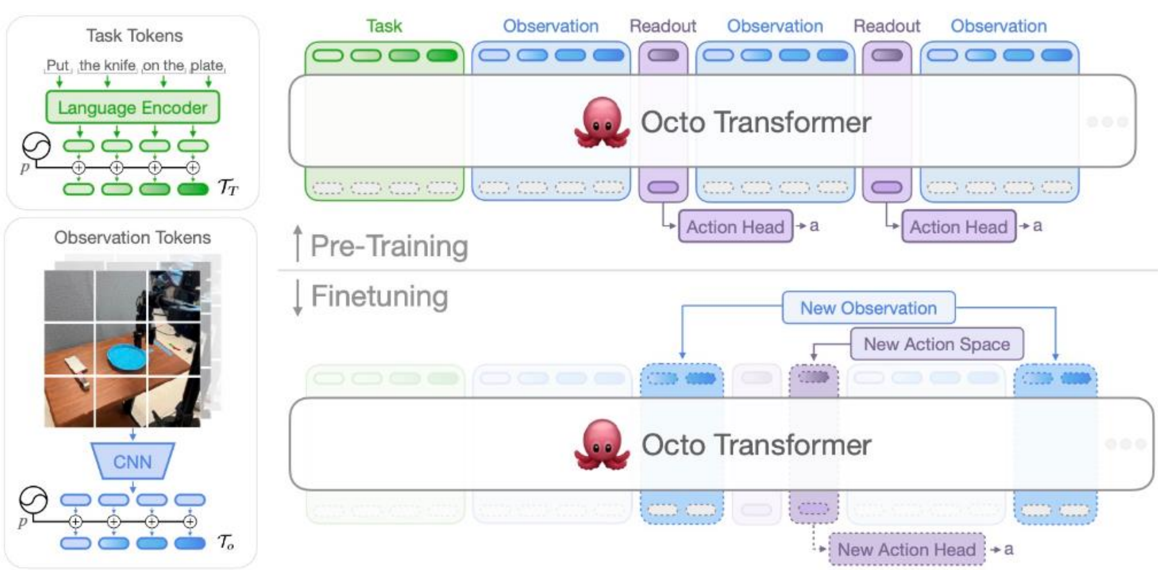


We introduce Octo, our ongoing effort for building open-source, widely applicable generalist policies for robotic manipulation. The Octo model is a transformer-based diffusion policy, pretrained on 800k robot episodes from the Open X-Embodiment dataset. It supports flexible task and observation definitions and can be quickly finetuned to new observation and action spaces. We are introducing two initial versions of Octo, Octo-Small (27M parameters) and Octo-Base (93M parameters).



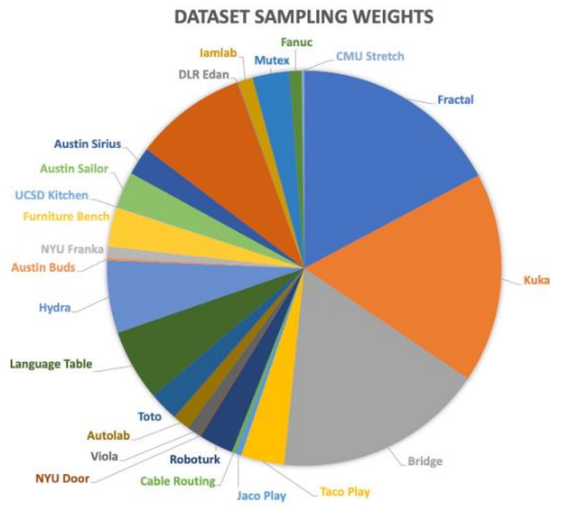
The Model

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. 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 specifically to make Octo a flexible and broadly applicable generalist robotic policy that can be utilized for a variety of downstream robotics applications and research projects.



The Data

We train Octo on a mixture of 25 datasets from the Open X-Embodiment Dataset, 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).



The Results

We evaluate Octo on 9 real robot setups across 4 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").

Zero-shot

	WidowX	UR5	RT-1 Robot
RT-1-X	0.20	0.35	0.60
RT-2-X	**0.50**	—	**0.85**
Octo 🐙	**0.50**	**0.70**	0.80

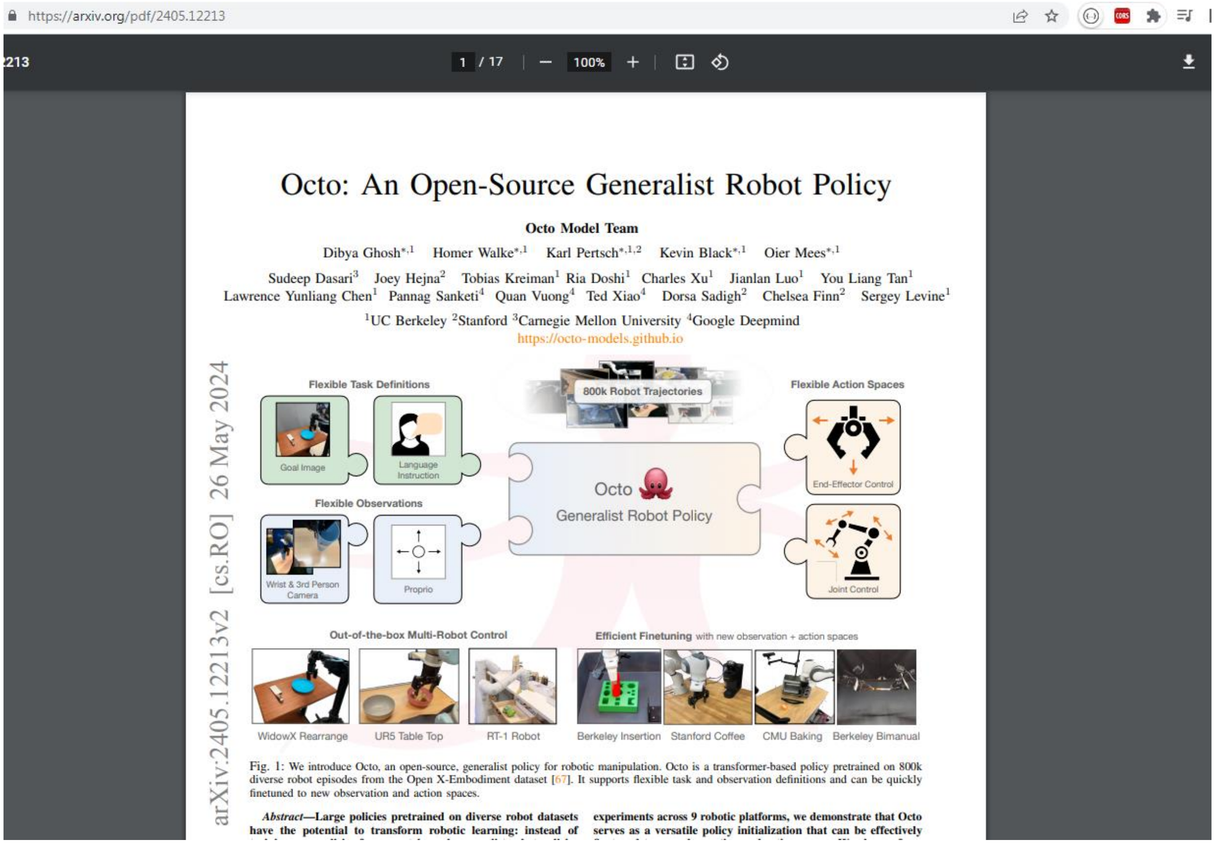
Finetuning

	CMU Baking	Stanford Coffee	Berkeley Peg Insert*	Berkeley Pick-Up†	Berkeley Bimanual†	Berkeley Coke	Average
From Scratch	0.25	0.45	0.10	0.00	0.20	0.20	0.20
VC-1	0.30	0.00	0.05	0.00	0.50	0.10	0.15
Octo 🐙	**0.50**	**0.75**	**0.70**	**0.60**	**0.80**	**1.00**	**0.72**

*New observation input (force-torque proprioception) †New action space (joint position control)

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: the current best, openly available generalist robotic policy. It performs similarly to RT-2-X, a 55-billion parameter model. Additionally, while RT-1-X and RT-2-X only support language conditioning, Octo also supports goal image conditioning. On the WidowX tasks, we found that Octo achieved even better performance with goal image conditioning — 25% higher on average — likely because goal images provide more information about how to achieve the task.

We also find that finetuning Octo leads to better policies than starting from scratch or with the pretrained VC-1 weights. On average across the six evaluation setups, Octo outperforms the next best baseline by 52%. Each task uses ~100 target demonstrations. Importantly, we use the same finetuning recipe for all evaluation tasks, 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 Insertion"), action spaces (joint position control for "Berkeley Pick-Up") and new robot embodiments ("Berkeley Bi-Manual" and "Berkeley Coke"). This makes Octo applicable to a wide range of single and dual arm robotic manipulation problems that go beyond a single camera input and end-effector position control.



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] Ria Doshi[1] Charles Xu[1] Jianlan Luo[1] You Liang Tan[1] Lawrence Yunliang Chen[1] Pannag Sanketi[4] Quan Vuong[4] Ted Xiao[4] Dorsa Sadigh[2] Chelsea Finn[2] Sergey Levine[1]

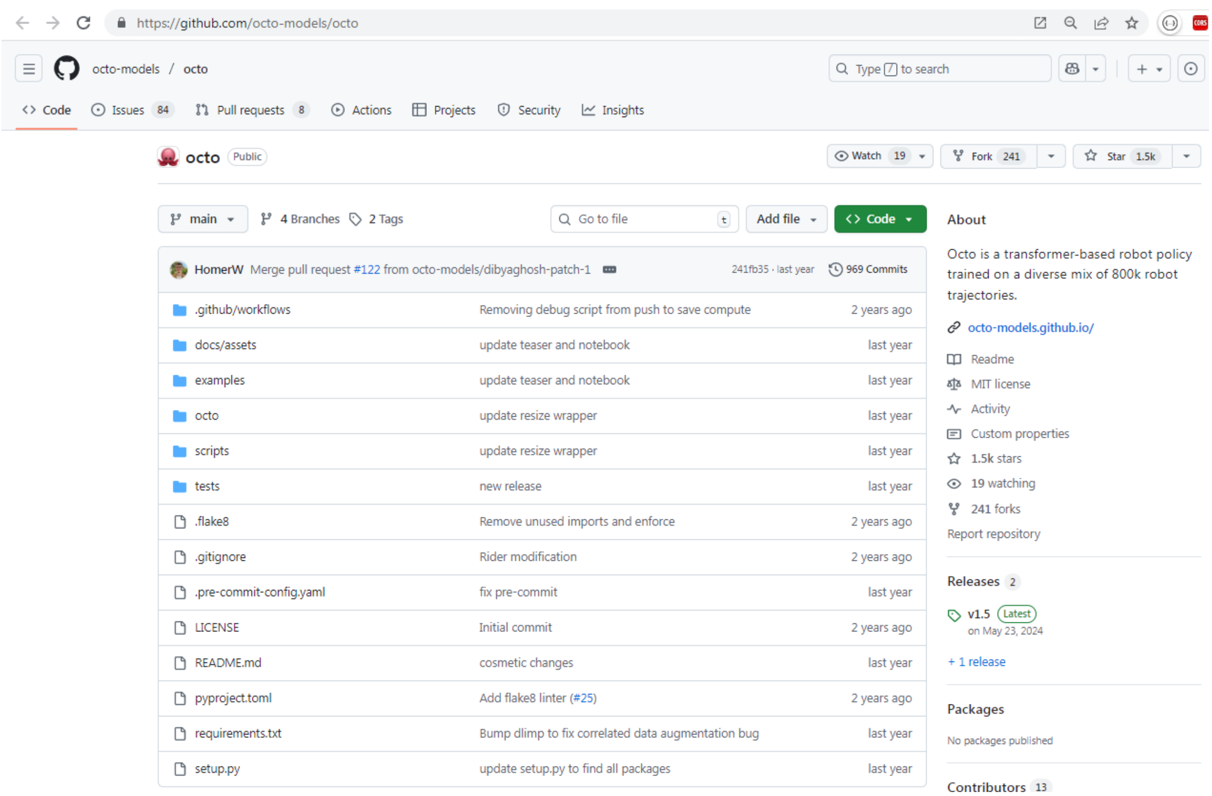
[1]UC Berkeley [2]Stanford [3]Carnegie Mellon University [4]Google Deepmind

https://octo-models.github.io

Fig. 1: We introduce Octo, an open-source, generalist policy for robotic manipulation. Octo is a transformer-based policy pretrained on 800k diverse robot episodes from the Open X-Embodiment dataset [67]. 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 robotic learning: instead of

experiments across 9 robotic platforms, we demonstrate that Octo serves as a versatile policy initialization that can be effectively



README MIT license

Languages

Python 100.0%

Octo

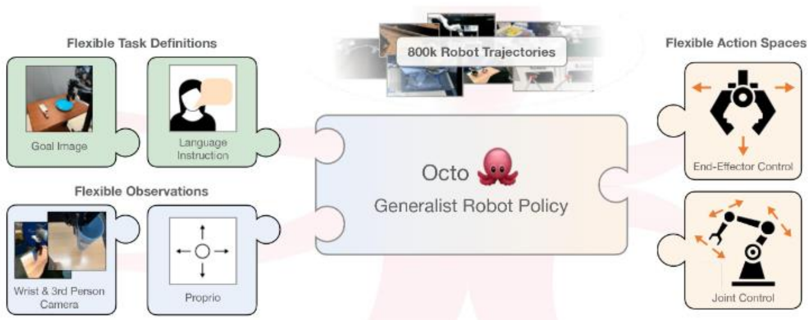
Open in Colab | License MIT | Project Page

This repo contains code for training and finetuning Octo generalist robotic policies (GRPs). Octo models are transformer-based diffusion policies, trained on a diverse mix of 800k robot trajectories.

Get Started

Follow the installation instructions, then load a pretrained Octo model! See examples for guides to zero-shot evaluation and finetuning and Open in Colab for an inference example.

```
from octo.model.octo_model import OctoModel
model = OctoModel.load_pretrained("hf://rail-berkeley/octo-base-1.5")
print(model.get_pretty_spec())
```



https://huggingface.co/rail-berkeley/octo-base

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing | Log In

rail-berkeley/**octo-base** | like 21

Robotics | Transformers | License: mit

Model card | Files and versions | Community 1 | Deploy | Use this model

Octo Base

See https://github.com/octo-models/octo for instructions for using this model.

Octo Base is trained with a window size of 2, predicting 7-dimensional actions 4 steps into the future using a diffusion policy. The model is a Transformer with 93M parameters (equivalent to a ViT-B). Images are tokenized by preprocessing with a lightweight convolutional encoder, then grouped into 16x16 patches. Language is tokenized by applying the T5 tokenizer, and then applying the T5-Base language encoder.

Downloads last month
64

Video Preview

Robotics

× Preview not found

https://huggingface.co/rail-berkeley/octo-small

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing | Log In

rail-berkeley/**octo-small** | like 13

Robotics | Transformers | License: mit

Model card | Files and versions | Community 1 | Deploy | Use this model

Octo Small

See https://github.com/octo-models/octo for instructions for using this model.

Octo Small is trained with a window size of 2, predicting 7-dimensional actions 4 steps into the future using a diffusion policy. The model is a Transformer with 27M parameters (equivalent to a ViT-S). Images are tokenized by preprocessing with a lightweight convolutional encoder, then grouped into 16x16 patches. Language is tokenized by applying the T5 tokenizer, and then applying the T5-Base language encoder.

Downloads last month
69

Video Preview

Robotics

× Preview not found

https://huggingface.co/rail-berkeley

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Community | Docs | Enterprise | Pricing | Log In

RAIL Berkeley

rail-berkeley

Follow

88 followers · 0 following

https://rail.eecs.berkeley.edu/

rail-berkeley

Models 6 — Sort: Recently updated

- rail-berkeley/crossformer — Robotics • Updated Aug 26, 2024 • 30 • 8
- rail-berkeley/ResNet-26-ImageNet — Updated Aug 21, 2024
- rail-berkeley/octo-base-1.5 — Robotics • Updated May 20, 2024 • 43 • 15
- rail-berkeley/octo-small-1.5 — Robotics • Updated May 20, 2024 • 220 • 8
- rail-berkeley/octo-small — Robotics • Updated Dec 14, 2023 • 69 • 13
- rail-berkeley/octo-base — Robotics • Updated Dec 14, 2023 • 64 • 21

Datasets 1

- rail-berkeley/OXE_paraphrases — Updated May 26, 2024 • 30

https://colab.research.google.com/drive/1z0vELj_lX9OWeoMG_WvXnQs43aPOEAhz?usp=sharing

Octo Inference Example — Changes will not be saved

File Edit View Insert Runtime Tools Help

Commands | + Code | + Text | Run all | Copy to Drive | Connect T4

Step 1: Minimal Octo Inference Example

This Colab demonstrates how to load a pre-trained / finetuned Octo checkpoint, run inference on some offline images and compare the outputs to the true actions.

First, let's start with a minimal example!

```
# Download repo
!git clone https://github.com/octo-models/octo.git
%cd octo
# Install repo
!pip3 install -e .
!pip3 install -r requirements.txt
!pip3 install --upgrade "jax[cuda11_pip]==0.4.20" -f https://storage.googleapis.com/jax-releases/jax_cuda_releases.html
!pip install numpy==1.21.1 # to fix colab AttributeError: module 'numpy' has no attribute '_no_nep50_warning', if the error still shows reload
```

```
Cloning into 'octo'...
remote: Enumerating objects: 7079, done.
remote: Counting objects: 100% (255/255), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 7079 (delta 235), reused 230 (delta 228), pack-reused 6824
Receiving objects: 100% (7079/7079), 25.10 MiB | 13.14 MiB/s, done.
Resolving deltas: 100% (4565/4565), done.
/content/octo
Obtaining file:///content/octo
  Installing build dependencies ... done
  Checking if build backend supports build_editable ... done
  Getting requirements to build editable ... done
  Preparing editable metadata (pyproject.toml) ... done
Building wheels for collected packages: octo
  Building editable for octo (pyproject.toml) ... done
  Created wheel for octo: filename=octo-0.0.0-0.editable-py3-none-any.whl size=3223 sha256=5e6f2183455ee950f3186346fcf81862a6fde83eda49063206a30c86cba8d816
```