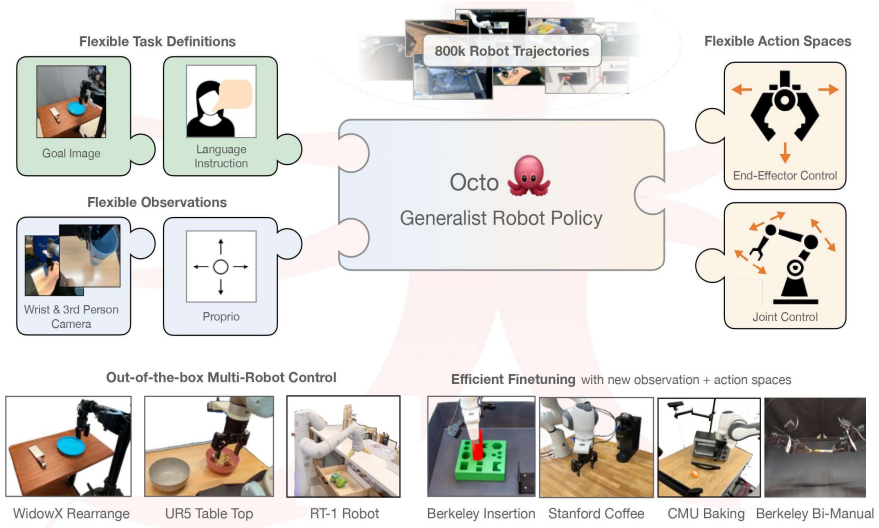
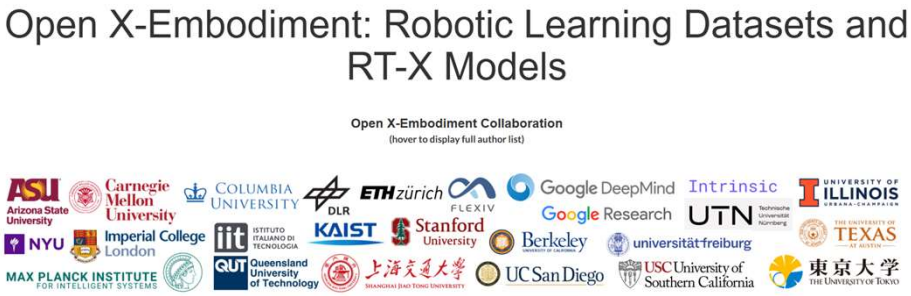
Octo： 开源、通用的机器人操作策略模型





Octo： 开源、通用的机器人操作策略模型

目标：开发通用机器人操作策略

方法：在各种机器人数据集上预训练大型策略，+少量领域内数据微调，得到广泛

挑战：处理不同传感器，动作空间，适应各种机器人平台等

Octo介绍：
1、基于大型transformer的策略模型（VLA）
2、在Open X-Embodiment数据集上训练（800k trajectories）
3、可通过语言命令或目标图像指示操作
4、可在标准消费级GPU上快速微调

Octo： 开源、通用的机器人操作策略模型

研究问题：如何设计一个开源的、通用的、可适配不同机器人和任务的机器人操作策略模型

挑战：
1、现有方法大多针对特定机器人和任务，泛化能力有限，训练代价高。
2、不同机器人在硬件构型，传感器类型，动作空间等方面存在很大差异，难以设计一个统一的模型架构，灵活适配不同机器人的观察和动作接口。
3、大规模机器人操作数据的缺乏
4、模型的实用性和可访问性有待提高。先进模型——私有 or 计算资源要求高

Octo原理简介：
将任务定义（如语言指令、目标图像）和机器人观察（如相机图像、传感器读数）统一编码为一个token序列，然后用一个通用的Transformer骨干网络提取特征，最后用轻量级的输出头解码成机器人动作。

优势：模块化设计带来了极大的灵活性，通过简单增删输入输出token和头，它可以适配不同机器人和任务，而无需修改预训练的骨干网络参数

Octo： 开源、通用的机器人操作策略模型

模型架构：Octo = Input tokenizers + Transformer backbone + Readout heads

输入编码器：
不同模态输入分别编码为token

language instructions ℓ
goals g
observation sequences $o_1, \ldots, o_H$
→ tokens $[\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o]$

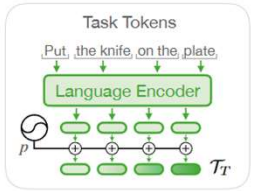
语言：tokenized->预训练的Transformer->language embedding tokens
图像：浅层CNN提取特征，然后划分为多个patches展平->图像token（ViT）

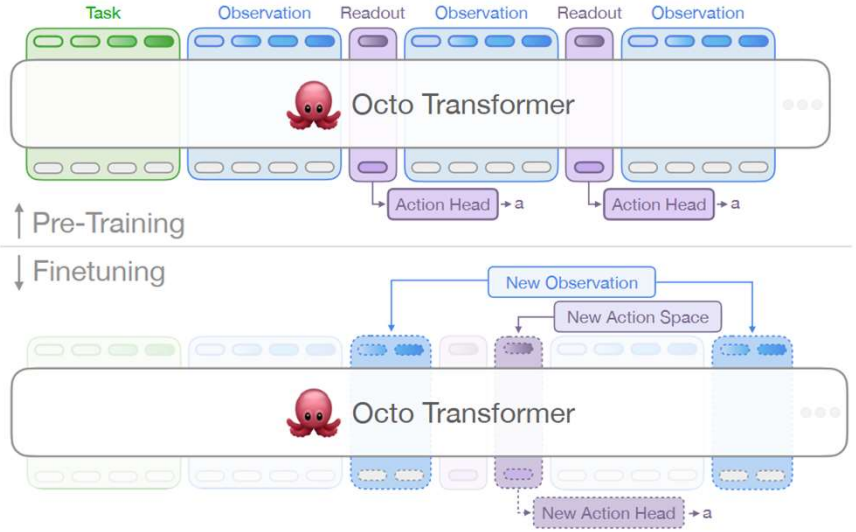
组装输出：添加位置编码，排列组合 $[\mathcal{T}_T, \mathcal{T}_{o,1}, \mathcal{T}_{o,2}, \ldots]$

Transformer： $e_l, e_g, e_o = T(\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o)$
块状注意力掩码 Block-wise masked

$\mathcal{T}_o$ attents to $\mathcal{T}_{o,0:t}$ + $\mathcal{T}_T$
$\mathcal{T}_{R,t}$ attents to $\mathcal{T}_{o,0:t}$ + $\mathcal{T}_T$

动作输出头： actions $a = R(e)$
多种轻量化动作输出头，输出action chunk
Diffusion action head：从初始高斯噪声向量通过逐步去噪解码为连续的机器人动作，每一步去噪过程由浅层MLP实现，输入为当前步的输出、Transformer输出的embedding和步骤编号





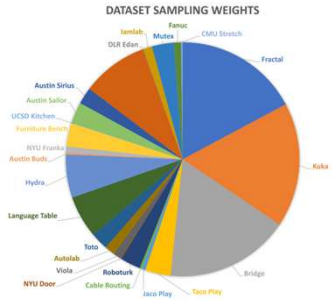
灵活性：
新的观察信息->引入新的对应输入token
新的动作空间->引入新的输出token并搭配新输出头

Octo： 开源、通用的机器人操作策略模型

训练数据:

目前最大规模的机器人操控数据集Open X-Embodiment （包含1500k Trajectories）
精选25个子数据集（800k Trajectories）涵盖多个机器人平台（11 Enbodiment）和任务环境

预处理:
统一动作空间
每条轨迹随机下采样至多100timesteps
数据集采样比例策略（人工选择+动态学习）



训练目标与细节:

采用常见数据增强，正则化等技巧

测试时采用滚动时域预测（TemporalEnsambleWrapper，RHCWrapper）

开源代码与模型:

Model	Inference on 1x NVIDIA 4090	Size
Octo-Base	13 it/sec	93M Params
Octo-Small	17 it/sec	27M Params

Open X-Embodiment: Robotic Learning Datasets and RT-X Models

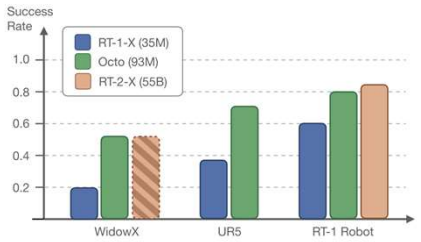
Open X-Embodiment Collaboration
(hover to display full author list)

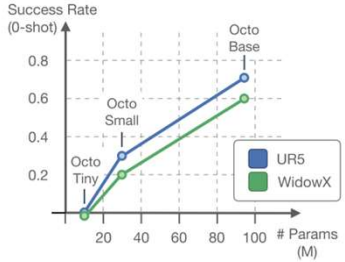
实验结果:

Zero-Shot Evaluation

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 Evaluation

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)

Octo的仿真和实机复现全流程代码

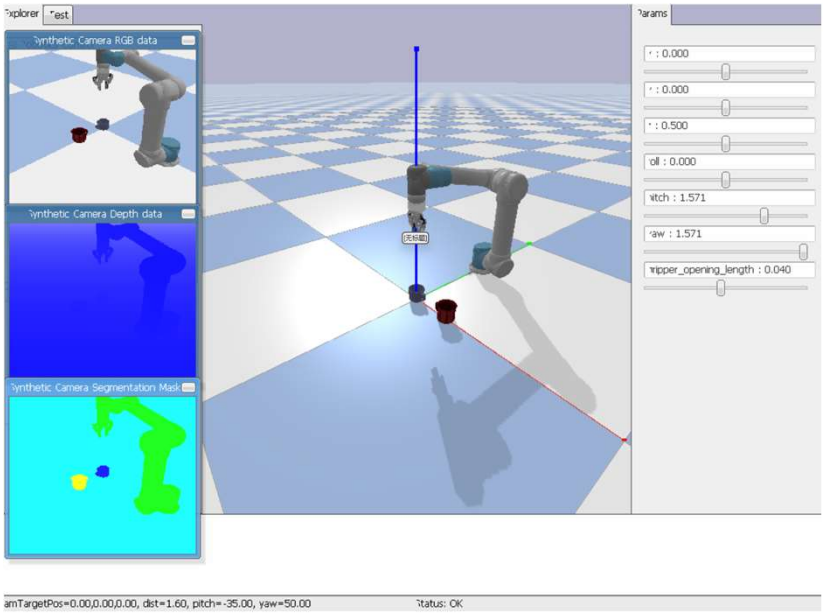
仿真环境：

仿真引擎：Pybullet

机械臂：**UR5**、Panda
夹爪：**robotiq85**、robotiq140

场景物体：pybullet-URDF-models、YCBModels等（URDF文件）

环境格式：gym，dmenv（gym2dmWrapper）



主要组成：

env.py　环境主体代码，内部定义reward，task（language instruction）等
robot.py　机械臂与夹爪类
main.py　用于测试，运行整个环境
utilities.py　工具，如相机类，YCBModel
agent.py　实现简单的控制逻辑，如随机采样
gym2dmenv.py　制作了一个包装器，用于将gym格式的环境包装成dmenv格式
make_dataset.py　使用EnvLogger，在仿真环境中手动采集数据并制作成RLDS格式数据集

Octo的仿真和实机复现全流程代码

数据集:

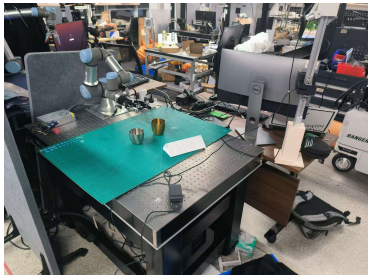
仿真环境:
pybullet_ur5_pick_reset_cup_mug

实机环境（UR3 version1）：
pick_reset_1.00

实机环境（UR3 version2）：
ur3_pick_cup_single (20)
ur3_pick_golden_cup_single (10)
ur3_pick_silver_cup_single (10)

实机环境（UR5）：
ur5_put_cube_on_plate (25)
Ur5_put_cube_on_plate_slow (10)

Example（UR3 version1）：



人眼视角（实际环境）

摄像头视角（观测空间）

ur3_pick_cup_single_slow (20)
ur3_pick_golden_cup_single_slow (10)
ur3_pick_silver_cup_single_slow (10)

数据集保存格式

- **Dataset-name**
  - **dataset-name-1**
    - dataset_info.json
    - features.json
    - data.tfrecord
    - .log *(if exists)*
  - **dataset-name-2**
    - dataset_info.json
    - features.json
    - data.tfrecord
    - .log *(if exists)*
  - **dataset-name-3**
    - dataset_info.json
    - features.json
    - data.tfrecord
    - .log *(if exists)*
  - **...**

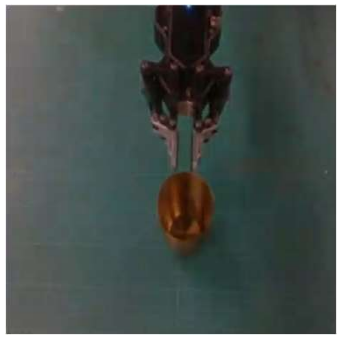
Octo的仿真和实机复现全流程代码

实机环境：

UR3：Pick up & Reset

version1

version2



UR5：put cube on plate

代码结构：

Octo_demo_collection_UR5:

- **real_ur5_env** - 实机环境代码
  - **env.py** - 主体环境　UR5　put cube on plate
  - **gripper_controller.py** - 夹爪控制类
  - **main.py** – 运行环境（手柄控制），用于检查和调整
  - **make_dataset_test.py** - 收集实机数据，制作数据集
  - **reading_stored_trajectories.py** - 读取实机数据集
  - **robot.py** - 机械臂控制类
  - **utilities.py** - 相机、手柄等工具类
  - **client_TCP.py** – 基于TCP的客户端：验证模型时使用
  - **gym_wrappers_easy.py** - gym格式环境包装器
  - **env_pick_up.py** - UR3 pick up环境
- **RTDE_Python_Client_Library** - RTDE库

测试函数：

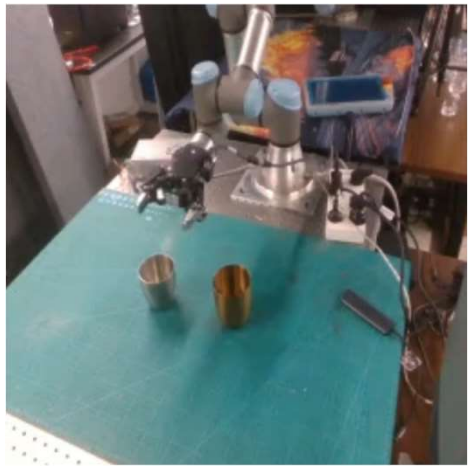
- **test_camera.py** - 测试相机
- **test_control.py** - 测试机械臂控制
- **test_gripper.py** - 测试夹爪控制
- **test_joystick.py** - 测试游戏手柄
- **test_ur_rtde.py** - 测试RTDE连接

Octo的仿真和实机复现全流程代码

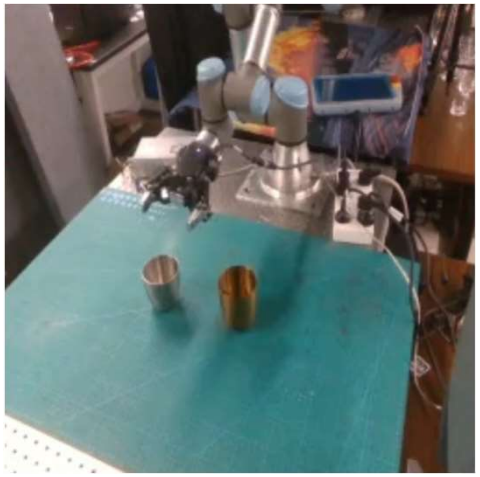
结果展示与分析:

UR3：version1　Task&Language Instruction：Pick up the golden cup and put it down

模型训练时间不够
=>只有夹爪运动

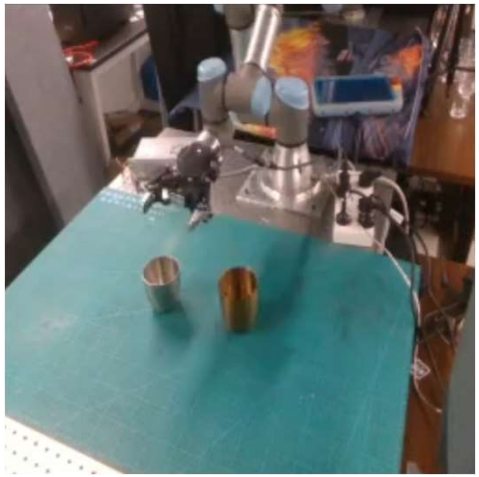


模型训练时间足够
=>夹爪+机械臂运动



Temporal Ensemble

动作较为稳定，策略保守



Receding Horizon Control

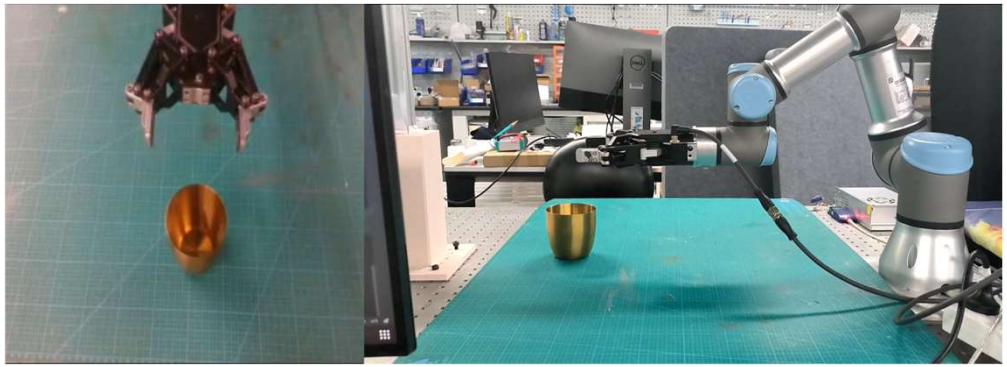
动作较不稳定，策略激进

Octo的仿真和实机复现全流程代码

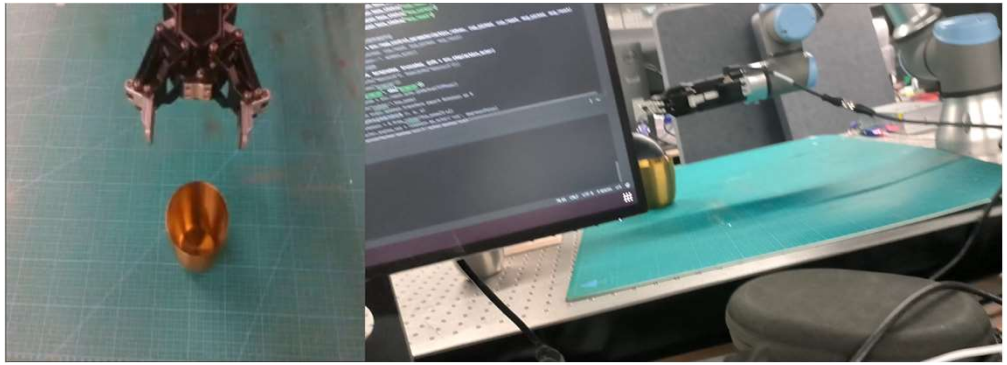
结果展示与分析:

UR3：version2　Task&Language Instruction：Pick up the golden cup and put it down

Temporal Ensemble =>



Receding Horizon Control =>

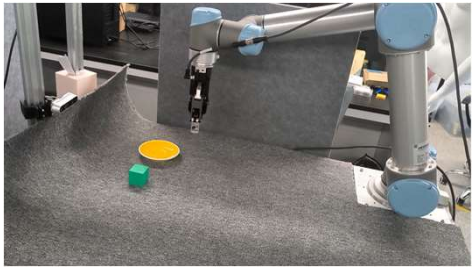


Octo的仿真和实机复现全流程代码

结果展示与分析:

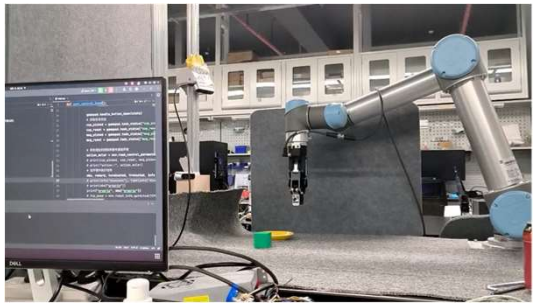
UR5　Task&Language Instruction：put cube on plate

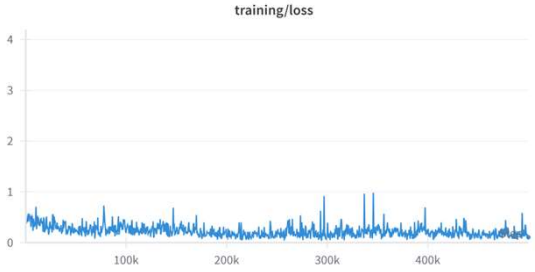
train 780k steps on put_cube_on_plate_25:
一开始移动到了正确的位置（物块上方）



训练轮次不够的时候:
一开始移动到盘子上方

Current best model：9.3 train 500k steps on put_cube_on_plate_slow_10





Octo的仿真和实机复现全流程代码

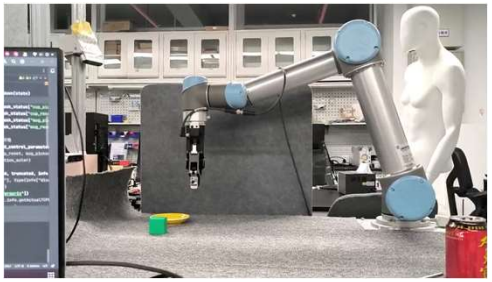
结果展示与分析:

UR5　Task&Language Instruction：put cube on plate

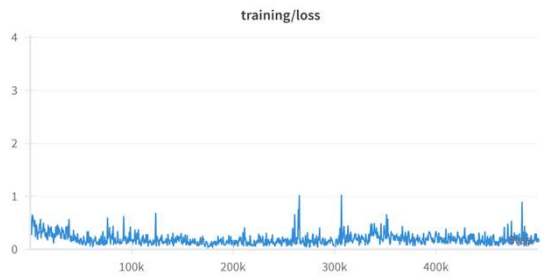
Based on current best model：9.13 train 500k steps on put_cube_on_plate_slow_10

50k steps

300k steps



500k steps



在原有模型基础上再训练，哪怕只是一点（50k），也会模式崩溃
再训练的时间越长，模式崩溃越严重
暂时结论：从头开始的训练效果好，模型继续训练就会模式崩溃

尝试从头训练100k steps ->结论：put cube on plate 任务无法完成