OpenVLA: LeRobot Research Presentation #5 by Moo Jin Kim

HuggingFace
106K subscribers
Subscribe

431 Share Ask Save

19,343 views Aug 13, 2024
LeRobot Research Presentation #5
Presented by Moo Jin Kim in July 2024
https://moojink.com

This week: OpenVLA: An Open-Source Vision-Language-Action Model
Paper: https://huggingface.co/papers/2406.09246
Project page: https://openvla.github.io

Organized by LeRobot's team at HuggingFace.
Github: https://github.com/huggingface/lerobot

OpenVLA:
An Open-Source Vision-Language-Action Model

Hugging Face LeRobot Paper Discussion
Moo Jin Kim

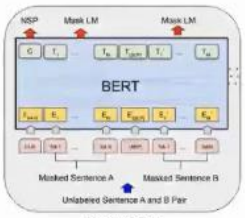
2024-07-24
openvla.github.io

Stanford Cal MIT

1

TOYOTA RESEARCH INSTITUTE Google DeepMind Physical Intelligence (π)

Motivation



Pre-training

- Successful formula in NLP & computer vision:
  (1) Pretrain large "foundation model" on tons of data
  (2) Fine-tune (or deploy zero-shot) on downstream task
- Historically, this approach hasn't been as popular in robotics…
  - Data has been the bottleneck: difficult to collect at large scale in the real world
- However, recent trends show promise for a similar formula in robotics:
  - Open X-Embodiment Dataset: Aggregation of >2M episodes from >20 robots and >20 institutions
  - RT-2 / RT-2-X: Pretrained vision-language models (VLMs) fine-tuned to do robotic control
    - (Coined the term "VLA": Vision-Language-Action model)
    - Broad semantic knowledge from Internet-scale pretraining
      - Enhances robot policy's generalization & reasoning ability → Generalizes beyond robot interaction data
    - High-capacity model with billions of parameters (55B)
      - Capable of fitting large, diverse datasets
- Since RT-2, more VLA models have been developed
  - E.g., RFM-1, Interactive Agent Foundation Model, RoboFlamingo, LEO, 3D-VLA, etc…
  - All these models are either private or trained on simulated robot data only!

OpenVLA Project Goals

- We want to:
  - (1) Develop a strong open-source VLA trained on lots of real-world robot data
    - Generalist policy: Can perform diverse tasks on multiple robots out of the box
  - (2) Develop an effective framework for adapting to downstream tasks
    - Parameter-efficient fine-tuning: Can fine-tune with low compute budget
  - (3) Release all VLA pretraining & fine-tuning code, model weights, data mixtures
    - Enable others to advance research on large pretrained robot models

place the red vegetable in the silver pot

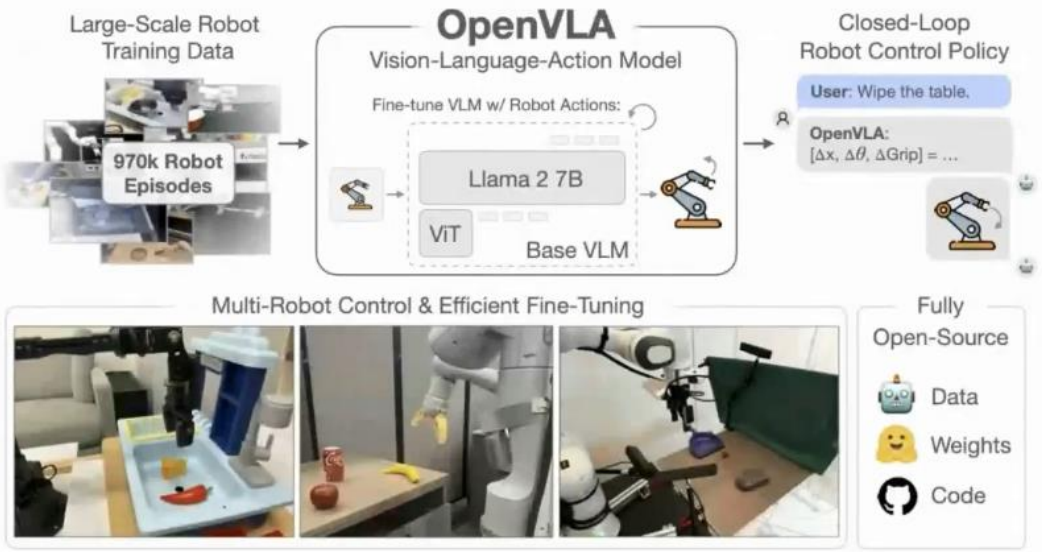
route cable

pick up the blue cup and put it into the brown cup

place the black bowl in the oven

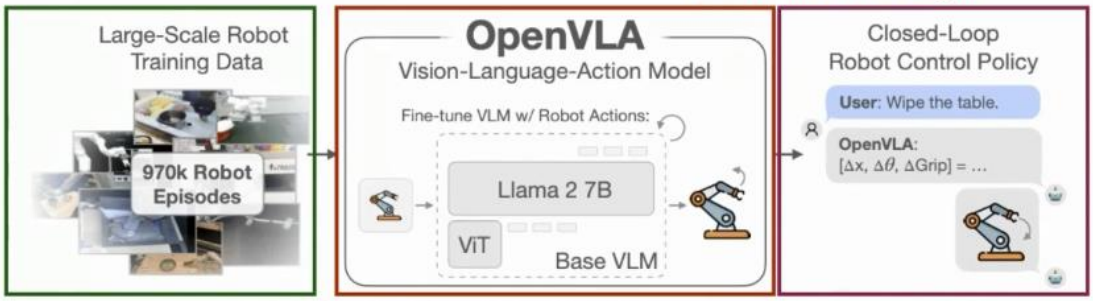
place green can upright

(Videos from the Open X-Embodiment dataset)



What is OpenVLA?

- 7B VLA built on pretrained "Prismatic VLM" backbone
  - LLM: Llama 2
  - Vision: DINOv2 + SigLIP

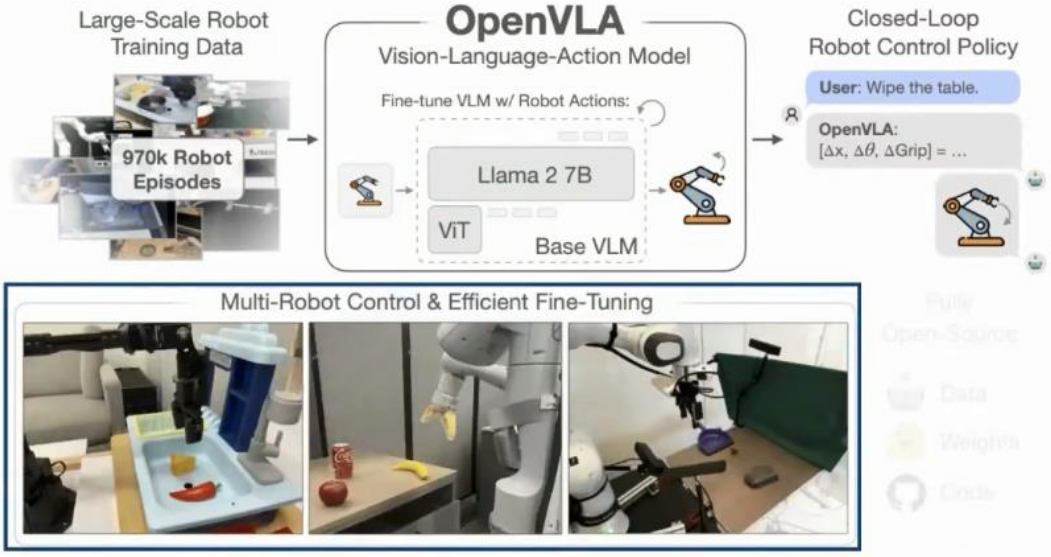


What is OpenVLA trained on?

- 970k episodes from Open X-Embodiment dataset
  - 27 real robot datasets
- Trained on 15 more datasets than RT-2-X

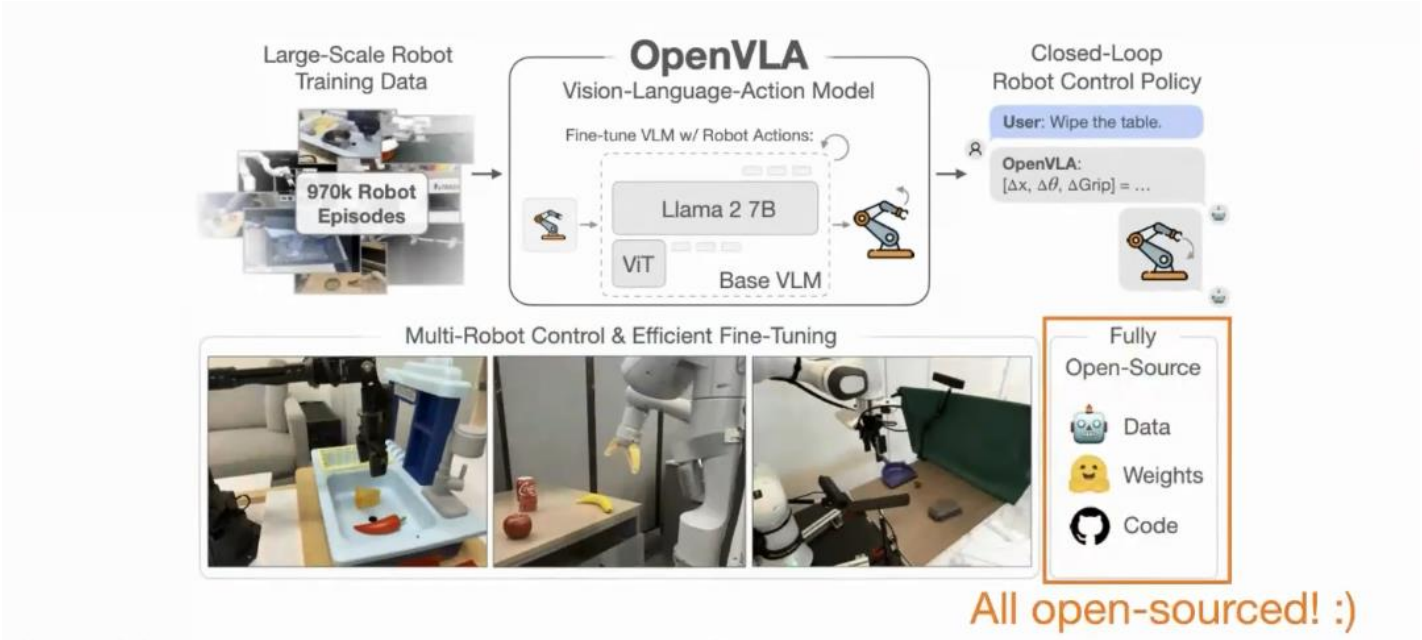
OpenVLA Inputs / Outputs?

- Inputs: Prompt w/ single image & language instruction
  - <Image> In: What action should the robot take to put eggplant into pot? Out:
- Outputs: Tokenized robot action



Which robots can OpenVLA control?

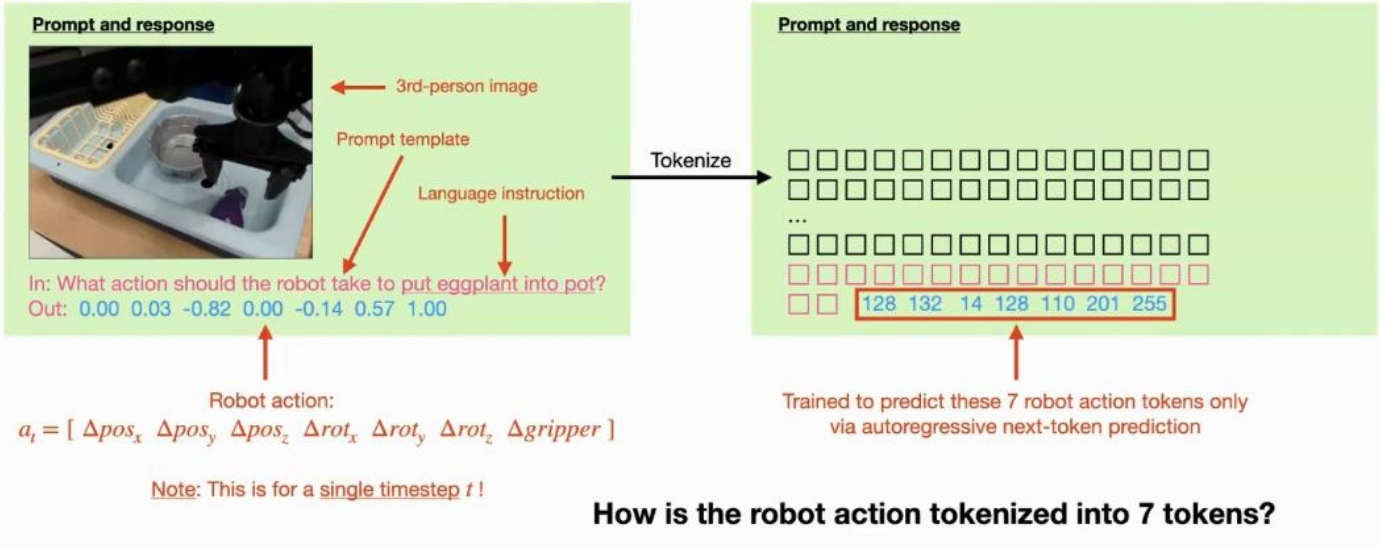
- WidowX, Google robot, Franka, & several more!



How is OpenVLA Trained?

- Imitation learning: Behavioral cloning with expert demonstrations
  - Given: Trajectories of (o_t = observation, a_t = action) pairs, where $t = 1, \ldots, T$
  - Goal: Train model π_θ to predict $\pi_\theta(\cdot \mid o_t) = \hat{a}_t \approx a_t$
- Each action $\hat{a}_t$ is discretized: represented as a string of 7 tokens
  - Allows us to train OpenVLA just like an LLM — via next-token prediction with cross-entropy loss
  - No architectural modifications!

How is OpenVLA Trained? (2)



We basically mask out all the tokens except for the robot action tokens as above, the OpenVLA model is trained at training time to predict only the 7 robot action tokens for the next step autoregressively

OpenVLA Robot Action Tokenization

- Robot action space: 7 dimensions
  - 6-DoF delta end-effector pose: $\Delta pos_x, \Delta pos_y, \Delta pos_z, \Delta rot_x, \Delta rot_y, \Delta rot_z$
  - 1-DoF gripper control: $\Delta gripper$ (binary: 0 = close, 1 = open)
- Each dimension is scaled to [-1, +1], then discretized into 255 uniform bins
  - Δpos_x → -1 [1 | 2 | . . . | 254 | 255] +1
  - Δpos_y → -1 [1 | 2 | . . . | 254 | 255] +1
  - …
- Therefore, each action $\hat{a}_t$ can be represented by a string of 7 tokens
  - Example:
    - Raw action: [0.00 0.03 -0.82 0.00 -0.14 0.57 1.00]
    - Tokenized: " 128 132 14 128 110 201 255 "
- Only need 255 tokens to represent the entire action space!
- In practice: We override the 255 least frequently used tokens in vocab
- Note: RT-2 (PaLM-E variant) used similar tokenization scheme

Experiments

- Main experiments:
  - **(1) How does OpenVLA compare to prior generalist robot policies when deployed out of the box?**
  - **(2) Can OpenVLA be effectively fine-tuned on a new robot setup/task?**
- Additional experiments:
  - (3) Can we use parameter-efficient fine-tuning methods (e.g., LoRA) to fine-tune OpenVLA efficiently with a smaller compute budget?
  - (4) Can we use inference-time quantization to load & run OpenVLA with limited GPU memory?

Experiment #1 — Out-of-the-Box Evals

- **(1) How does OpenVLA compare to prior generalist robot policies when deployed out of the box?**
- Comparisons
  - **RT-1-X (35M params)**
    - Transformer model with FiLM-conditioned EfficientNet image encoder
    - Trained on 12 datasets from OpenX (mostly from scratch)
  - **Octo (93M params) — Prior SOTA open-source generalist policy**
    - Transformer model with flexible input/output spaces & diffusion action head
    - Trained on 25 datasets from OpenX (mostly from scratch)
  - **RT-2-X (55B params) — Prior SOTA closed-source generalist policy**
    - Pretrained PaLI-X VLM with 22B ViT image encoder & 32B encoder-decoder LLM
    - Fine-tuned on 12 datasets from OpenX (same as RT-1-X)
  - **OpenVLA (7B params)**
    - Pretrained Prismatic VLM with fused ~300M DINOv2 + ~400M SigLIP image encoders & 7.1B Llama 2 decoder-only LLM
    - Fine-tuned on 27 datasets from OpenX

Experiment #1 — Out-of-the-Box Evals

- We evaluate on two setups: BridgeData V2 WidowX robot + Google RT-X robot
- Evaluation setup #1: BridgeData V2 WidowX
  - 17 tasks total, 10 rollouts per task
    - Various axes of OOD generalization: visual, motion, physical, semantic
    - Language grounding tasks
      - Given multiple objects in scene, can policy manipulate correct target object (specified in user's prompt)?

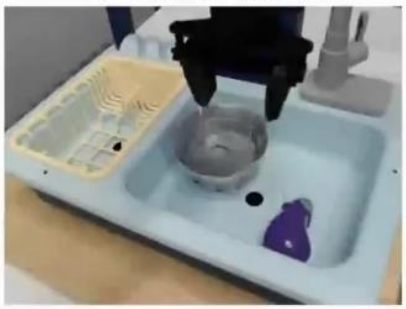
Experiment #1 — Out-of-the-Box Evals

Sample **visual generalization** tasks

Put Cup from Counter into Sink

(unseen cup color)

Put Eggplant into Pot



(unseen pot)

Put Yellow Corn on Pink Plate

(unseen clutter / distractor objects)

Experiment #1 — Out-of-the-Box Evals

Sample **visual generalization** tasks

Put Cup from Counter into Sink

(unseen cup color)

Put Eggplant into Pot

(unseen pot)

Put Yellow Corn on Pink Plate

(unseen clutter / distractor objects)

(These are all real OpenVLA policy rollouts)

Experiment #1 — Out-of-the-Box Evals

Sample **motion generalization** tasks

Lift Eggplant

(unseen object position)

Put Carrot on Plate

(unseen plate height)

Experiment #1 — Out-of-the-Box Evals

Sample **motion generalization** tasks

Lift Eggplant

(unseen object position)

Put Carrot on Plate

(unseen plate height)

Experiment #1 — Out-of-the-Box Evals

Sample **motion generalization** tasks

Lift Eggplant

(unseen object position)

Put Carrot on Plate

(unseen plate height)

Experiment #1 — Out-of-the-Box Evals

Sample **physical generalization** tasks

Flip Pot Upright

(unseen pot size/shape)

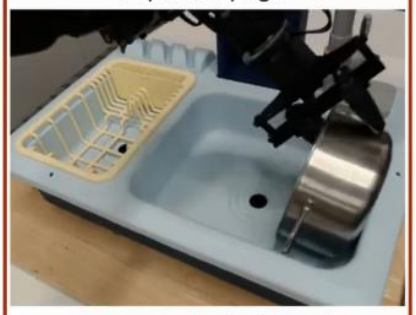
Lift AAA Battery

(unseen target object size)

Experiment #1 — Out-of-the-Box Evals

Sample **physical generalization** tasks

Flip Pot Upright



(unseen pot size/shape)

Lift AAA Battery

(unseen target object size)

Experiment #1 — Out-of-the-Box Evals

Sample **physical generalization** tasks

Flip Pot Upright

(unseen pot size/shape)

Lift AAA Battery

(unseen target object size)

Experiment #1 — Out-of-the-Box Evals

Sample **semantic generalization** tasks

Move Skull into Drying Rack

(unseen target object)

Stack Blue Cup on Pink Cup

(unseen language instruction)

Experiment #1 — Out-of-the-Box Evals

Sample **semantic generalization** tasks

Move Skull into Drying Rack

(unseen target object)

Stack Blue Cup on Pink Cup

(unseen language instruction)

Experiment #1 — Out-of-the-Box Evals

Sample **language grounding** tasks

Put **Eggplant** into Pot

Put **Red Bottle** into Pot

Experiment #1 — Out-of-the-Box Evals

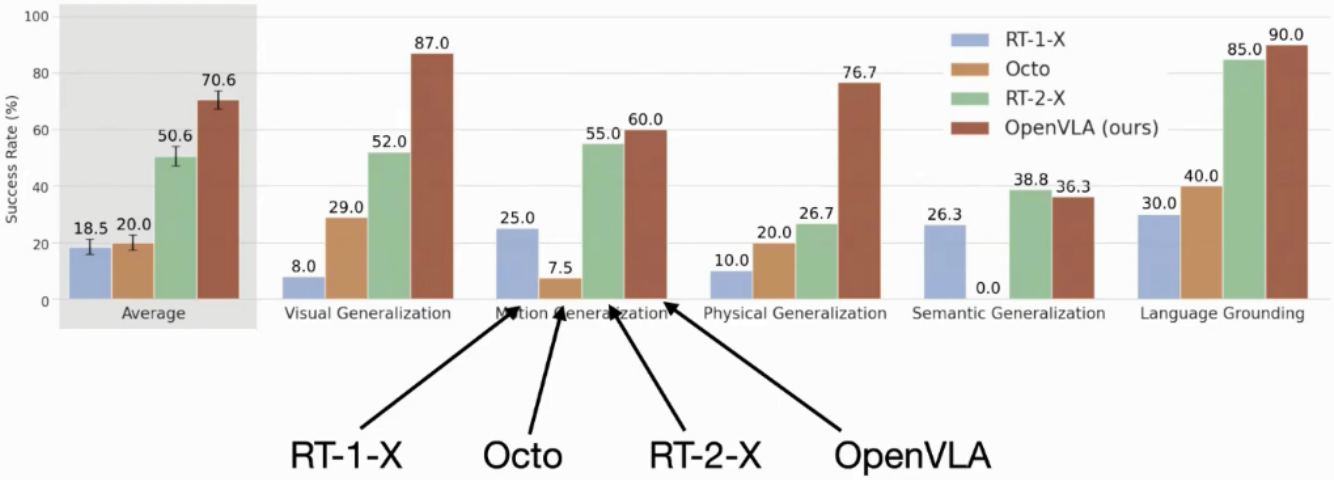
Sample **language grounding** tasks

Put **Eggplant** into Pot

Put **Red Bottle** into Pot

Experiment #1 — Out-of-the-Box Evals

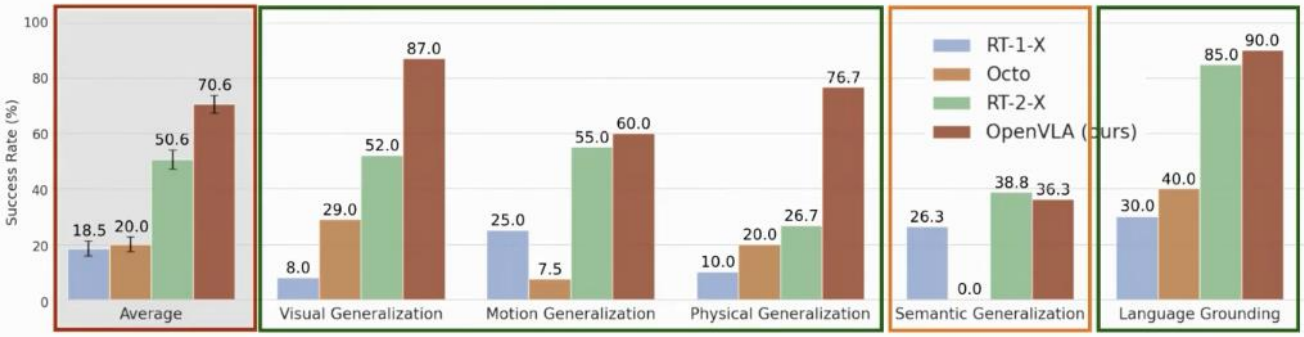
BridgeData V2 Results



RT-1-X Octo RT-2-X OpenVLA

Experiment #1 — Out-of-the-Box Evals

BridgeData V2 Results



OpenVLA performs best overall, outperforming next best model (RT-2-X) by +20%

OpenVLA is strongest in most task categories

Exception: For semantic gen, RT-2-X performs best

Potentially due to:
(1) Larger-scale Internet pretraining
(2) Co-fine-tuning on both robot data + VLM pretaining data

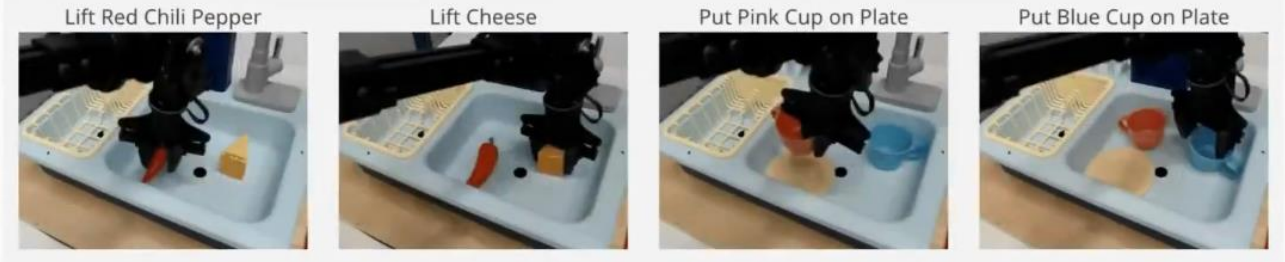
Experiment #1 — Out-of-the-Box Evals

Qualitative Results

Experiment #1 — Out-of-the-Box Evals

Qualitative Results

Similarly, we can prompt OpenVLA to manipulate a different target object given same initial states



Experiment #1 — Out-of-the-Box Evals

Qualitative Results

In some cases, after an initial mistake, OpenVLA can recover and successfully complete the task

Put Blue Cup on Plate

Put Eggplant into Pot

Experiment #1 — Out-of-the-Box Evals

- Evaluation setups:
  - BridgeData V2 WidowX Robot ✅
  - Google RT-X manipulator
    - 12 tasks total, 5 rollouts per task
      - 5 in-distribution tasks
      - 7 OOD tasks
        - Unseen objects, backgrounds, object relations, web data concepts

OOD is Out of Distribution.

Experiment #1 — Out-of-the-Box Evals

Sample **in-distribution** tasks

Pick Coke Can

Move Apple near Green Can

Open Middle Drawer

Experiment #1 — Out-of-the-Box Evals

Sample **OOD** generalization tasks

Move Orange near Brown Chip Bag

(unseen objects & backgrounds)

Place Banana in Pan

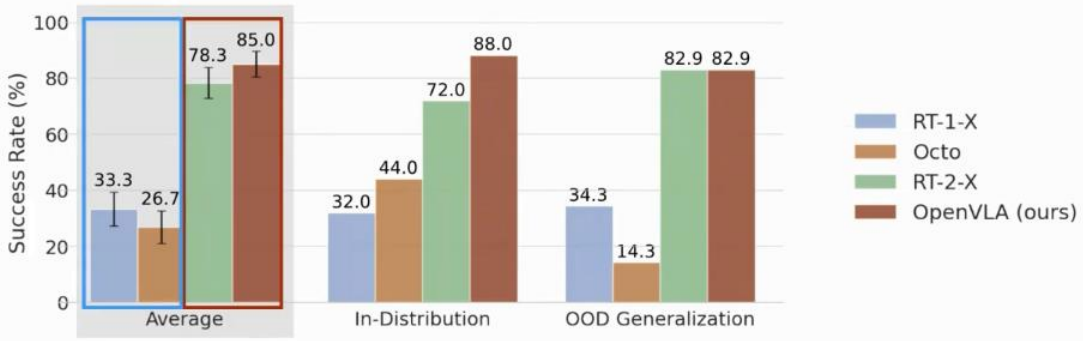
(unseen objects & instruction)

Move Coke Can to Taylor Swift

(unseen photos from Internet)

Experiment #1 — Out-of-the-Box Evals

Google RT-X Robot Results



OpenVLA and RT-2-X perform comparably (overlapping error bars)

RT-1-X and Octo performance is also comparable

Both underperform compared to the VLA models

Experiment #1 — Out-of-the-Box Evals

Qualitative Results

Both RT-2-X & OpenVLA perform reliably on in-distribution + basic OOD tasks

RT-2-X:
Place Banana on Plate
(OOD: unseen target object & instruction)

OpenVLA:
Place Banana on Plate
(OOD: unseen target object & instruction)

Experiment #1 — Out-of-the-Box Evals

Qualitative Results

As in Bridge evals, RT-2-X performs better on difficult semantic generalization tasks

RT-2-X:
Move Coke Can near Taylor Swift
(OOD: unseen concept from Internet)

OpenVLA:
Move Coke Can near Taylor Swift
(OOD: unseen concept from Internet)

Experiments

- Main experiments:
  - **(1) How does OpenVLA compare to prior generalist robot policies when deployed out of the box?**
  - **(2) Can OpenVLA be effectively fine-tuned on a new robot setup/task?**
- Additional experiments
  - (3) Can we use parameter-efficient fine-tuning methods (e.g., LoRA) to fine-tune OpenVLA efficiently with a smaller compute budget?
  - (4) Can we use inference-time quantization to load & run OpenVLA with limited GPU memory?

Experiment #2 — Fine-Tuning Evals

- **(2) Can OpenVLA be effectively fine-tuned on a new robot setup/task?**
- Comparisons
  - **"Diffusion Policy"**: Full Diffusion Policy, trained from scratch
    - Additional bells & whistles compared to OpenVLA — robot proprio state, 2-step observation history, action chunking
  - **"Diffusion Policy (matched)"**: Version of DP that matches OpenVLA inputs/outputs and action space, trained from scratch
  - **"Octo"**: Pretrained Octo-Base model, fine-tuned
  - **"OpenVLA (scratch)"**: (Ablation) OpenVLA model with no OpenX pretaining, fine-tuned
    - I.e., fine-tuning the base pretained VLM directly
  - **"OpenVLA (ours)"**: Full OpenVLA model, fine-tuned

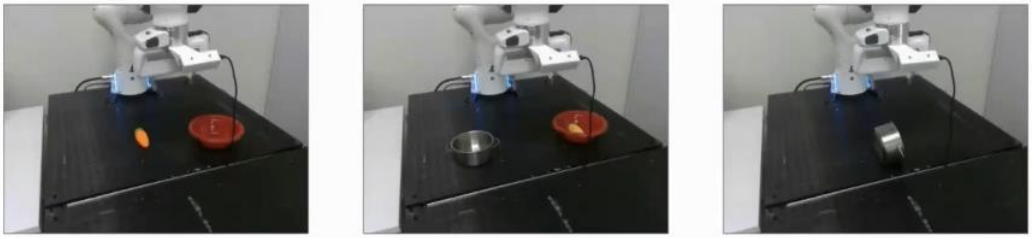
Experiment #2 — Fine-Tuning Evals

- Evaluation setups:
  - **"Franka-Tabletop"**: 7-DoF table-mounted Franka Panda robot, 5 Hz control
    - 3 narrow single-instruction tasks
    - 3 diverse multi-instruction tasks
  - **"Franka-DROID"**: Same hardware setup from DROID dataset paper, 15 Hz control
    - 1 single-instruction task

Experiment #2 — Fine-Tuning Evals

Franka-Tabletop: Narrow Single-Instruction Tasks

Put Carrot in Bowl (50 demos) | Pour Corn into Pot (50 demos) | Flip Pot Upright (10 demos)



OOD Variants

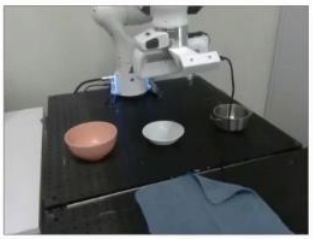
Experiment #2 — Fine-Tuning Evals

Franka-Tabletop: Diverse Multi-Instruction Tasks

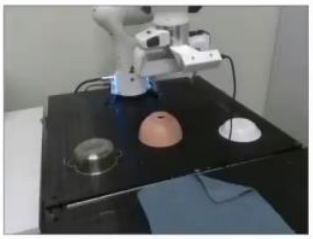
Move <object> onto Plate (150 demos)

Knock <object> Over (70 demos)

Cover <object> with Towel (45 demos)



OOD Variants

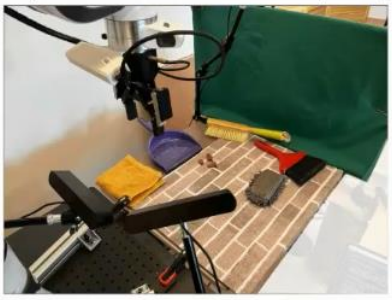


Experiment #2 — Fine-Tuning Evals

Franka-DROID Task

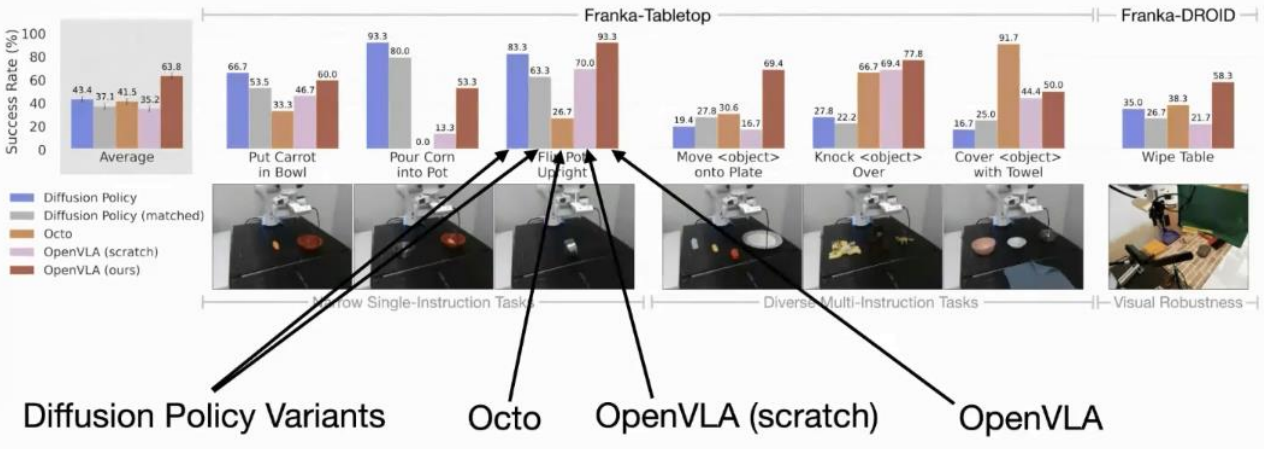
Wipe Table (70 demos)

Wipe Table w/ Distractors (OOD)



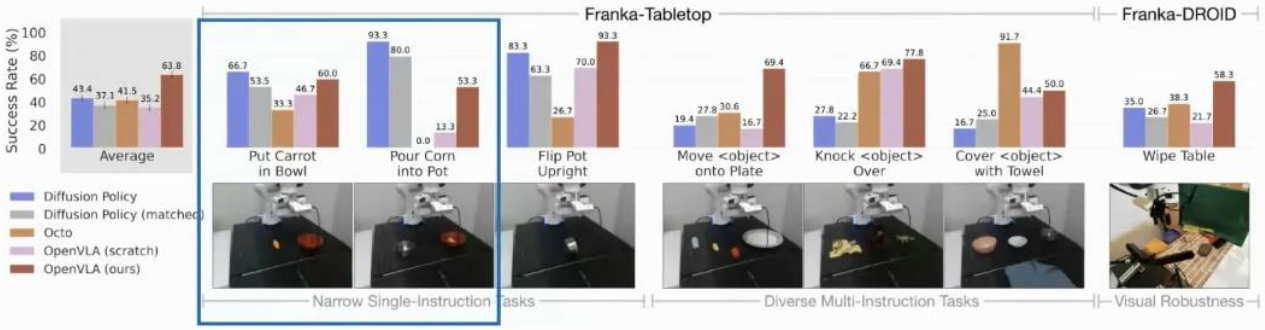
Experiment #2 — Fine-Tuning Evals

Fine-Tuning Results



Experiment #2 — Fine-Tuning Evals

Fine-Tuning Results



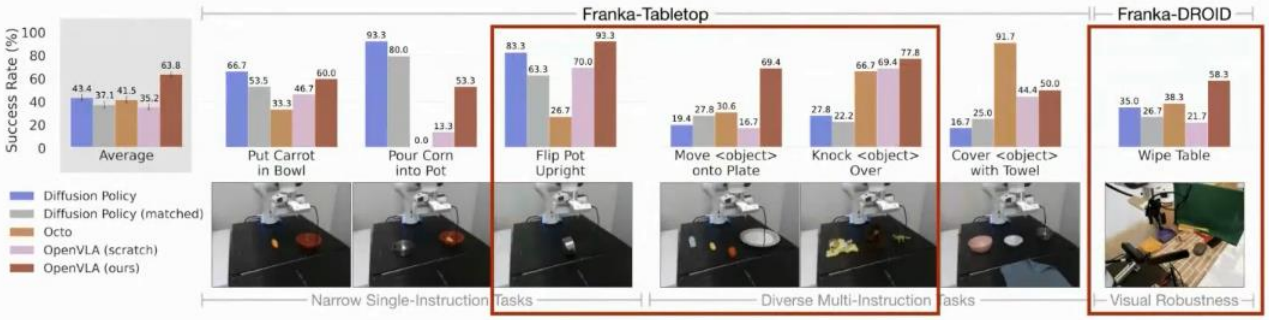
Diffusion Policy variants perform best in 2 narrow single-instruction tasks, where language grounding is not important

Presenter: Moo Jin Kim

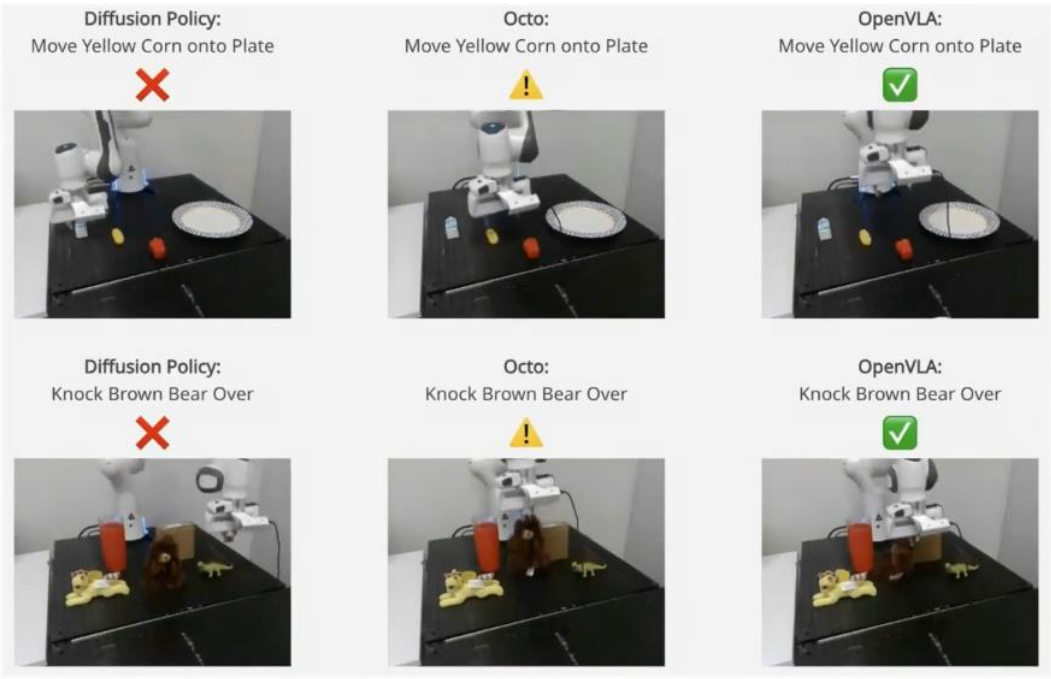
openvla.github.io

Experiment #2 — Fine-Tuning Evals

Fine-Tuning Results

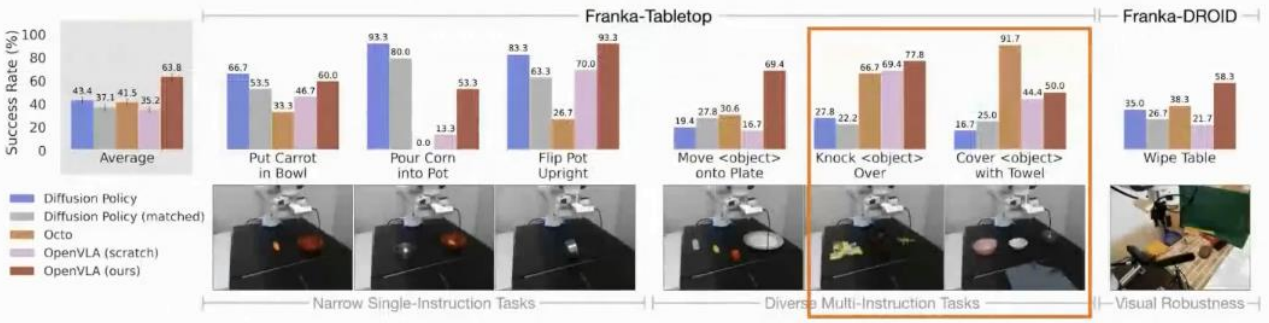


OpenVLA performs best in 4 of the 7 tasks, especially in diverse multi-instruction tasks where language grounding is important



Experiment #2 — Fine-Tuning Evals

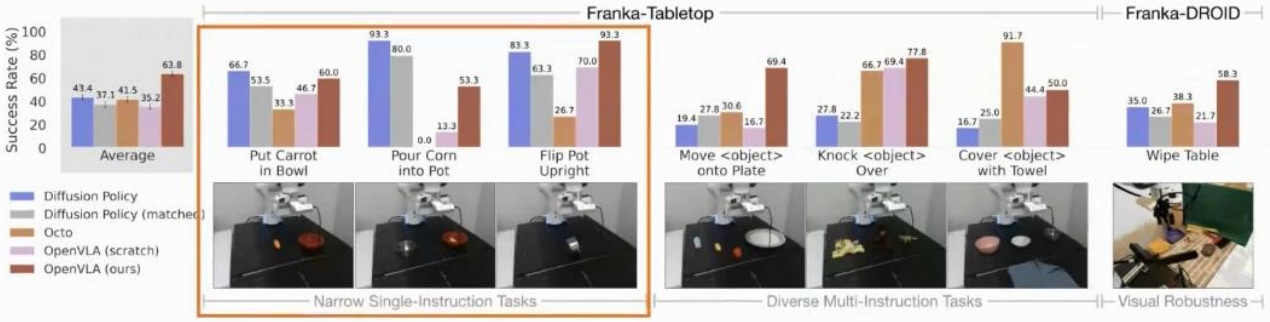
Fine-Tuning Results



We observe that Octo exhibits strong language grounding as well
- Performs well on 2 of 3 “diverse multi-instruction” tasks

Experiment #2 — Fine-Tuning Evals

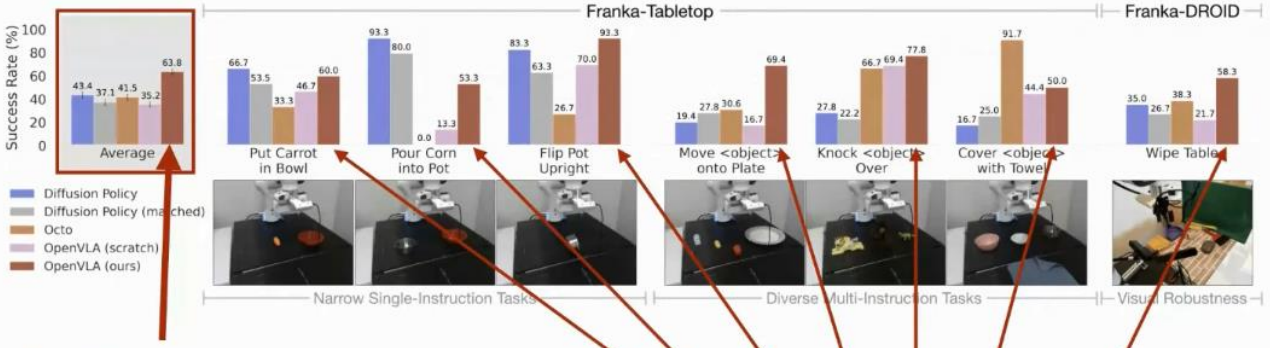
Fine-Tuning Results



However, Octo struggles on the “narrow single-instruction” tasks

Experiment #2 — Fine-Tuning Evals

Fine-Tuning Results

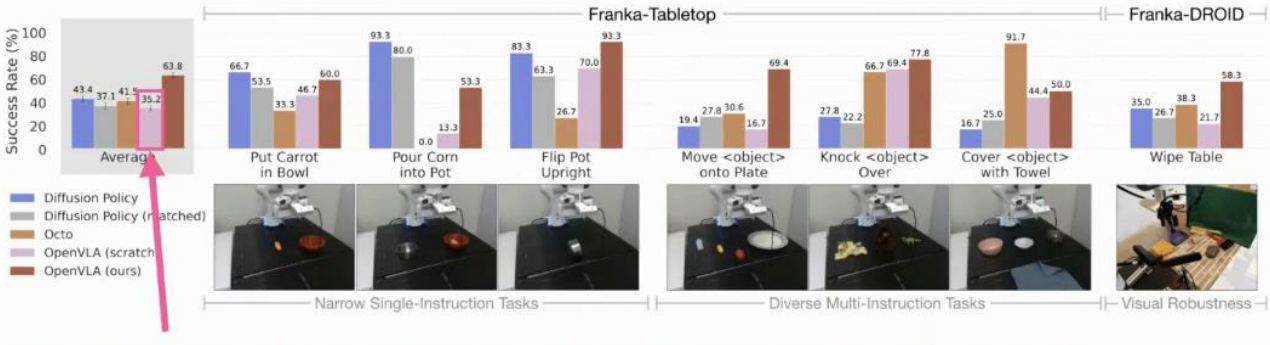


OpenVLA shows strongest performance on average: outperforms next best baseline by +20.4% (absolute)

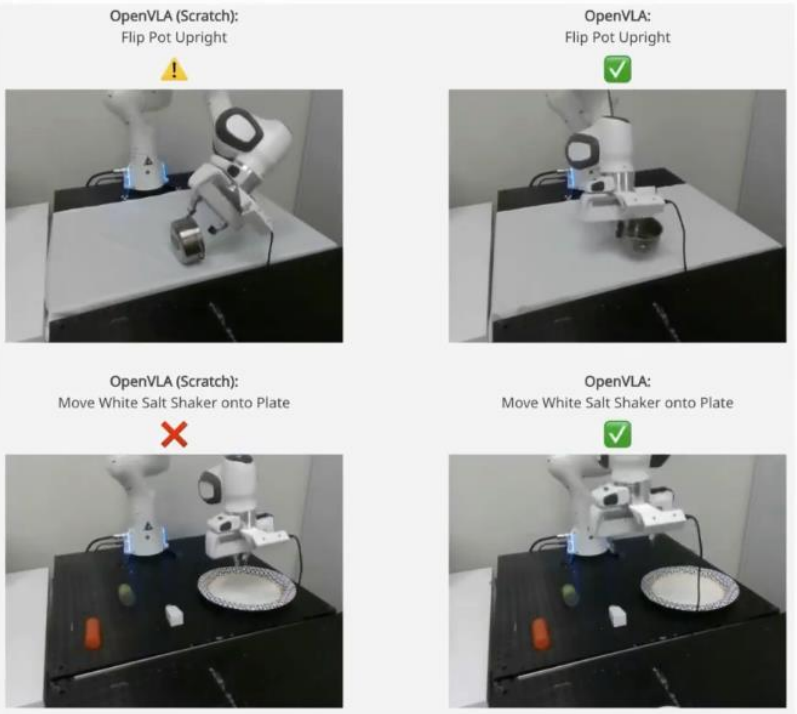
Also: Only method to get at least 50% success in all tasks

Experiment #2 — Fine-Tuning Evals

Fine-Tuning Results



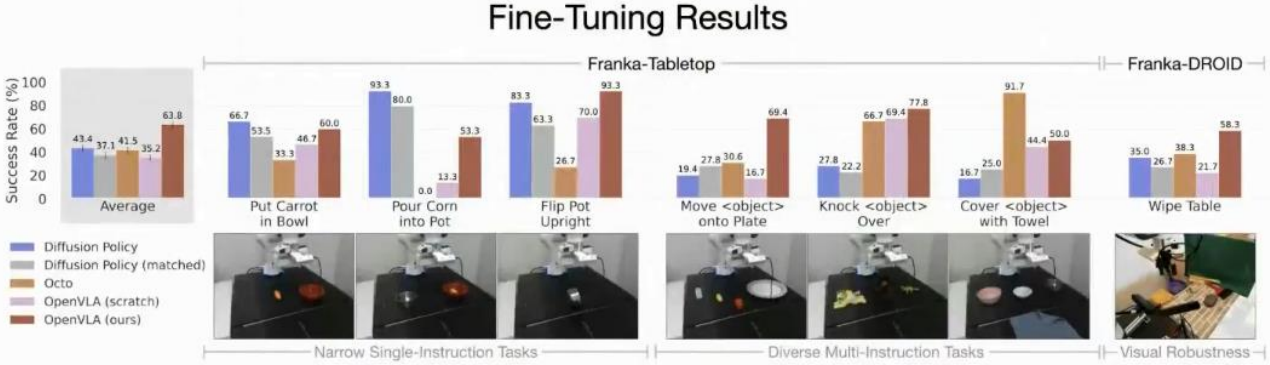
OpenVLA trained from scratch (no OpenX pretaining) struggles overall — showing that robot pretraining is crucial for good fine-tuning performance



Presenter: Moo Jin Kim

openvla.github.io

Experiment #2 — Fine-Tuning Evals



(See openvla.github.io for more qualitative results!)

Experiments

- Main experiments:
  - **(1) How does OpenVLA compare to prior generalist robot policies when deployed out of the box?**
  - **(2) Can OpenVLA be effectively fine-tuned on a new robot setup/task?**
- Additional experiments
  - (3) Can we use parameter-efficient fine-tuning methods (e.g., LoRA) to fine-tune OpenVLA efficiently with a smaller compute budget?
  - (4) Can we use inference-time quantization to load & run OpenVLA with limited GPU memory?

Additional Experiments

- Q: Do we need to fully fine-tune OpenVLA to get good performance on downstream tasks?
- **Parameter-efficient fine-tuning**
  - We compare various partial fine-tuning strategies:
    - **"Last layer only"**: Fine-tune only last layer of LLM backbone
    - **"Frozen vision"**: Freeze vision encoder, unfreeze all else
    - **"Sandwich"**: Fine-tune vision encoder + last LLM layer, freeze all else
    - **"LoRA"**: Low-rank adaptation method, applied to all linear layers of vision/LLM backbones
- We evaluate each method on a subset of Franka-Tabletop tasks

Additional Experiments

Parameter-Efficient Fine-Tuning Results

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

LoRA is most effective: With rank=32, trains only 1.4% of model params but matches full fine-tuning performance!

Additional Experiments

Parameter-Efficient Fine-Tuning Results

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

Freezing the vision encoder leads to poor performance — even if you fine-tune the entire LLM backbone!

Additional Experiments

Parameter-Efficient Fine-Tuning Results

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

"Sandwich fine-tuning" only fine-tunes vision encoder + LLM last layer
Boost in performance demonstrates importance of fine-tuning vision encoder

Additional Experiments

- Q: Do we need a GPU with large memory to run OpenVLA models?
- **4-bit quantization (at inference time)**
  - Normally, we cast OpenVLA to bfloat16 at test time → Requires ~16 GB VRAM
  - What if we use int4 quantization?
  - Results (on subset of BridgeData V2 tasks)

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

  - **TLDR**:
    - Only need 7 GB VRAM to run OpenVLA with 4-bit quantization
    - Did not observe degradation in performance in our evals!
    - (8-bit quantization performs poorly… possibly due to low inference speed)

OpenVLA Design Decisions & Insights

- Which VLM to fine-tune?
  - Tried IDEFICS (v1), LLaVA (v1.5), and Prismatic VLM
  - IDEFICS << LLaVA (+35% *absolute* SR in Bridge) < Prismatic VLM (+10%)
- Which image resolution?
  - Tried 224x224 and 384x384
  - Observed no difference in performance in Bridge evals, but 384x384 takes 3x longer to train → We opted for 224x224
- Freeze or fine-tune vision encoder?
  - Prior work on VLMs found freezing vision to be better
  - We found the opposite: Fine-tuning was crucial for both zero-shot + fine-tuning evals
- How long to train?
  - Consistently found that training for more epochs was better (e.g. 30 epochs!)
- Which learning rate?
  - LR sweep → Same LR used in VLM pretaining (2e-5) led to best results
  - Used constant LR schedule for all experiments

OpenVLA Limitations

- Only supports single-frame input and single-step action output
  - (Because large model → low training/inference speed with more inputs/outputs)
  - Q: How would performance change with additional inputs/outputs (e.g., robot proprio state, observation history, action chunking)?
- Lack of support for high-frequency control / bi-manual manipulation robots
  - E.g., ALOHA
  - (Difficult to run OpenVLA at >7 Hz control without special tricks — e.g. if we are using local GPUs or even remote H100 servers)
  - Q: Can VLM inference speedup tricks enable high-frequency VLA control?
- VLM fine-tuned only on robot action data (not co-fine-tuned on robot data + VQA data, as done in RT-2/RT-2-X)
  - Potential catastrophic forgetting of Internet-pretraining concepts that do not appear in robot interaction data
  - Q: Would OOD generalization performance improve significantly if OpenVLA is also fine-tuned on VQA data?

- We hope researchers build off of OpenVLA to answer some of these questions!

Thankful for my amazing collaborators!

OpenVLA:

An Open-Source Vision-Language-Action Model

Moo Jin Kim[*,1], Karl Pertsch[*,1,2], Siddharth Karamcheti[*,1,3],
Ted Xiao[4], Ashwin Balakrishna[3], Suraj Nair[3], Rafael Rafailov[1], Ethan Foster[1], Grace Lam,
Pannag Sanketi[4], Quan Vuong[5], Thomas Kollar[3], Benjamin Burchfiel[3], Russ Tedrake[3,6], Dorsa Sadigh[1],
Sergey Levine[2], Percy Liang[1], Chelsea Finn[1],

[*]Equal contribution

[1]Stanford University, [2]UC Berkeley, [3]Toyota Research Institute, [4]Google DeepMind, [5]Physical Intelligence,
[6]MIT,