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

Team 5
Kim Jaemin
Jung Geumyoung

KAIST

Motion Planning Diffusion: Learning and Planning of Robot Motions with Diffusion Models

Learning Trajectory Priors with Diffusion, Guided Sampling for Optimal Paths

- **Recap Problem:**
  - Optimization-based planners need good initialization
  - Sampling-based planners can be inefficient

- **MPD's Solution:**
  - **Learn Prior :** Diffusion models learn a generative model of expert trajectories
  - **Guided Sampling :** Sample from the posterior by guiding the diffusion reverse process with gradients from motion planning costs

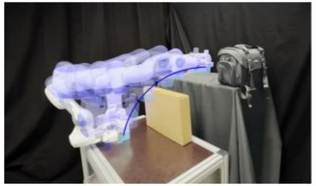
Diffusion Model

Generates diverse trajectories

→

Cost Gradients

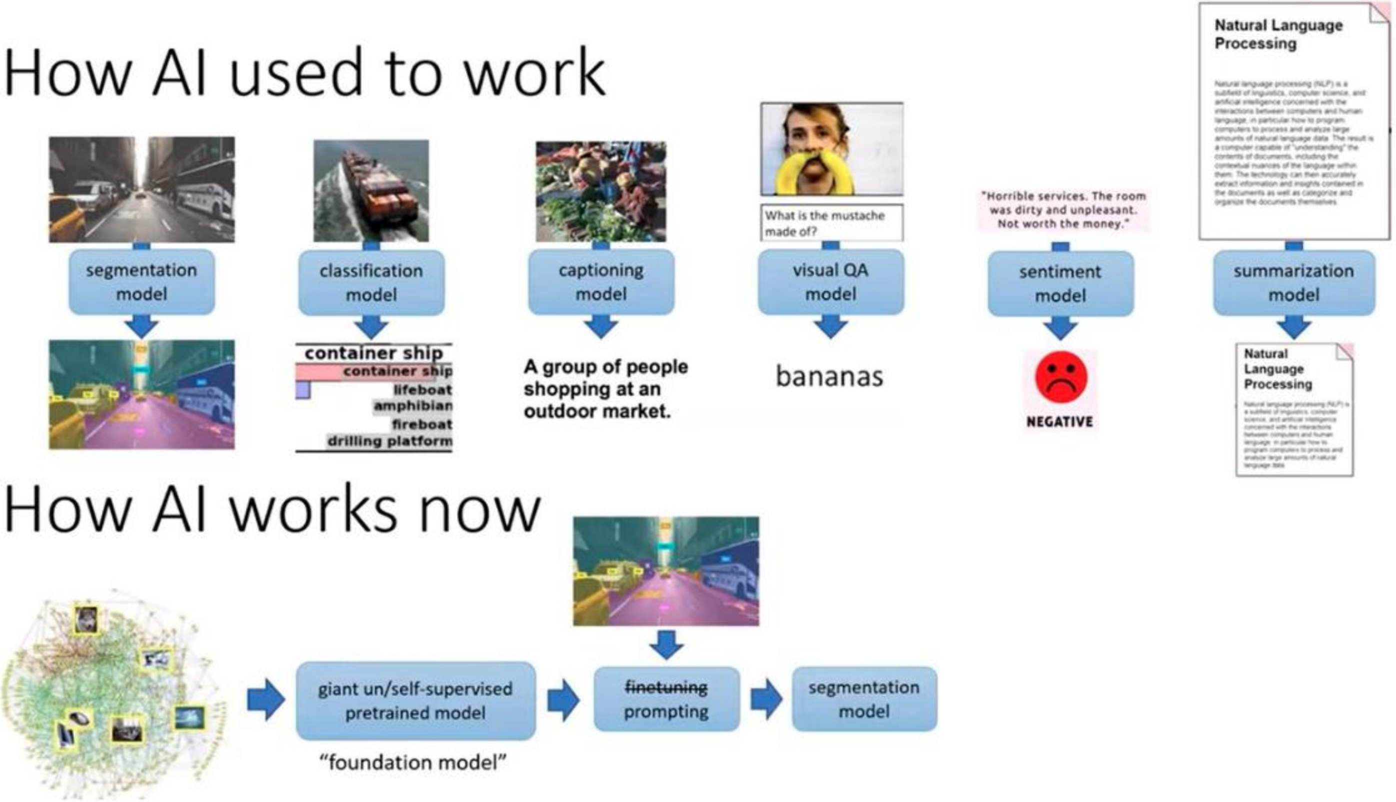
Guiding sampling towards optimal regions



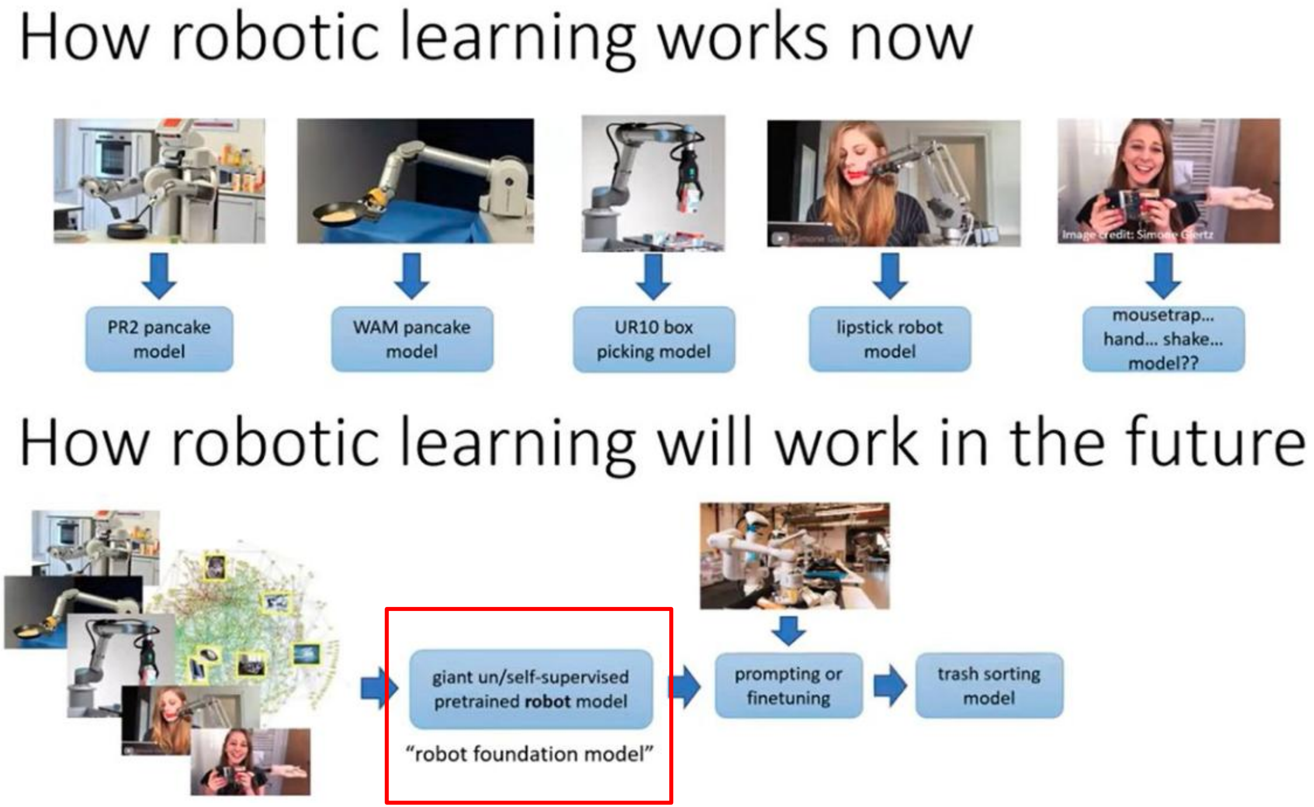
Contents

KAIST

Introduction | Motivation



Introduction | Motivation

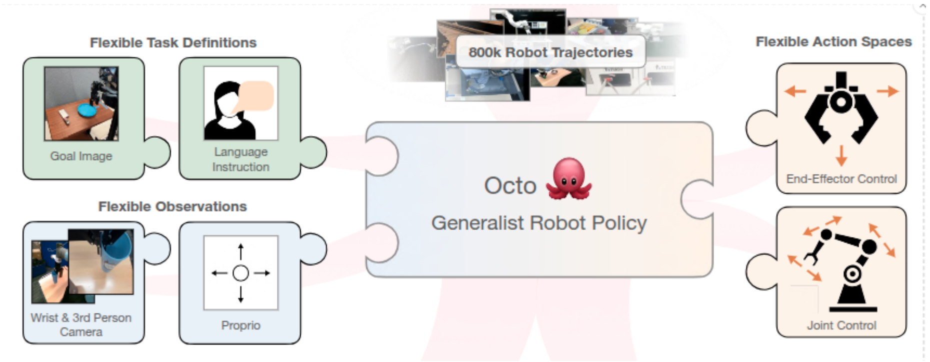


Introduction | Challenges

Existing works are :

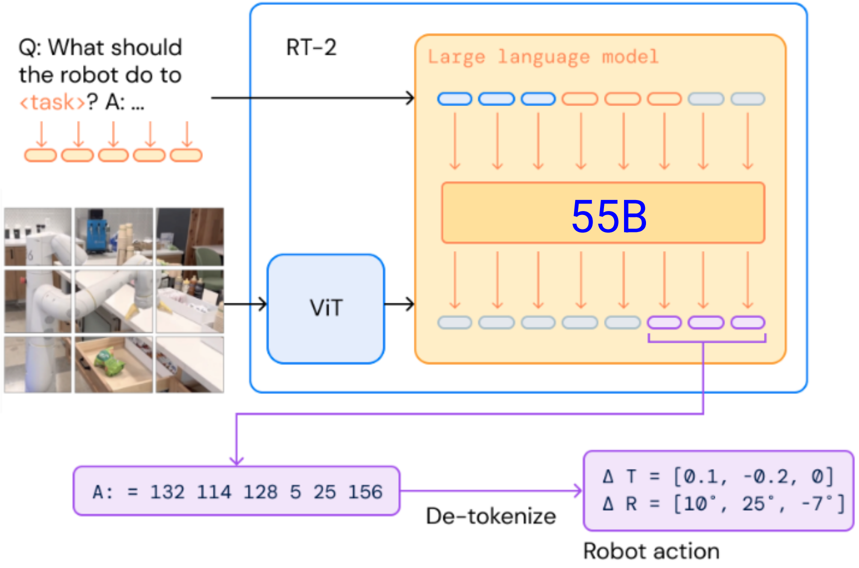
1. Parameter-wise Heavy (~55B parameters)
2. Closed-Source
3. Lacking fine-tuning exploration

Octo: An Open-Source Generalist Robot Policy



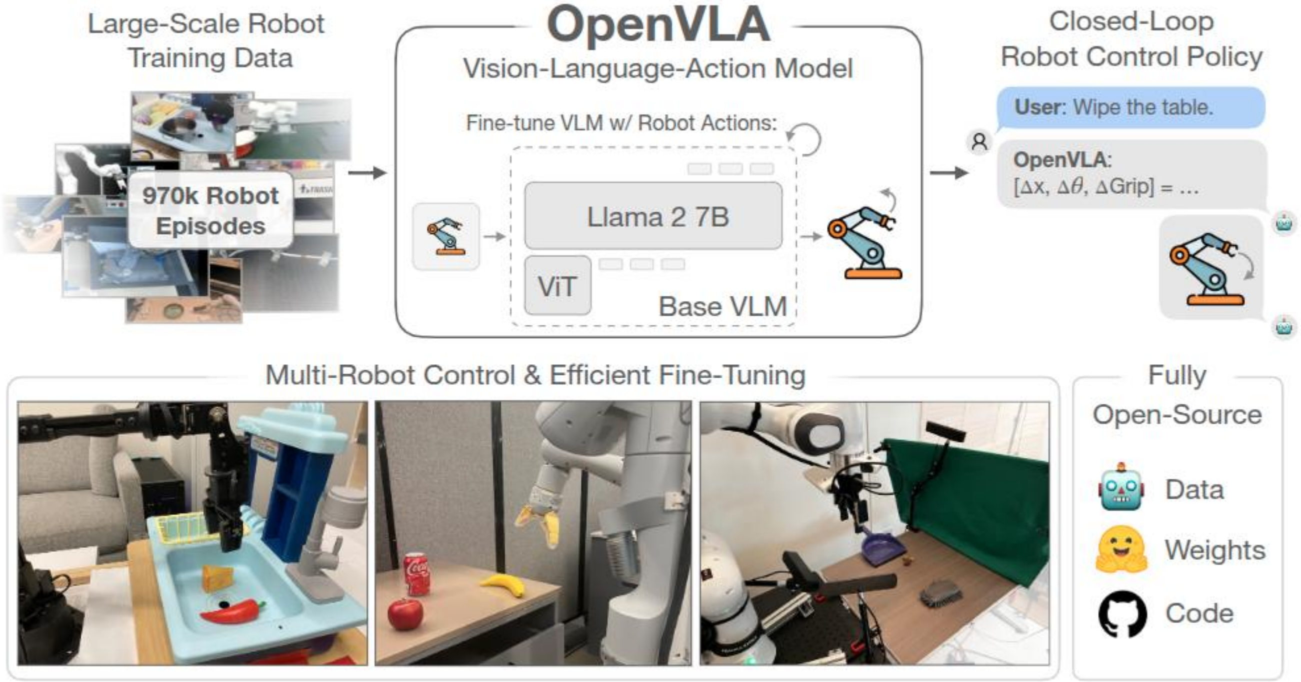
RT-2: Vision-Language-Action Models

Transfer Web Knowledge to Robotic Control



OpenVLA I Robotic **V**ision - **L**anguage - **A**ction model

consists of 7B parameter, fully open-source, support efficient fine-tuning

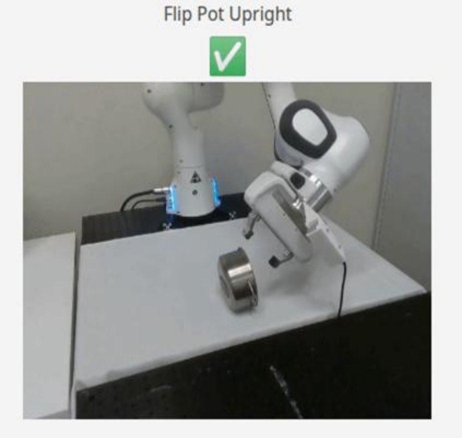


KAIST

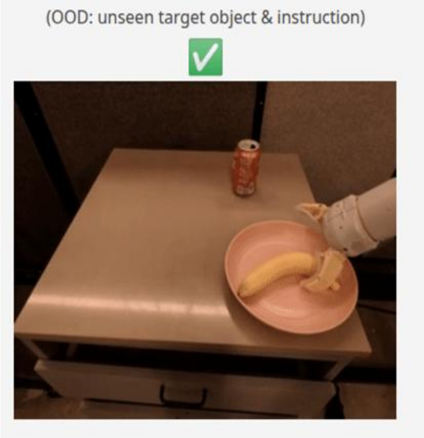
OpenVLA | Contribution

1. **Outperform SOTA** RT-2-X (55B) by 16.5% in absolute task success rate
   - work across 29 tasks, multiple robot embodiments, with fewer parameters(7B)
2. Demonstrate effectiveness of modern **parameter-efficient fine-tuning** and **quantization**
3. First **open-source** generalist VLA thus supports future research

Franka Emika Panda Robot

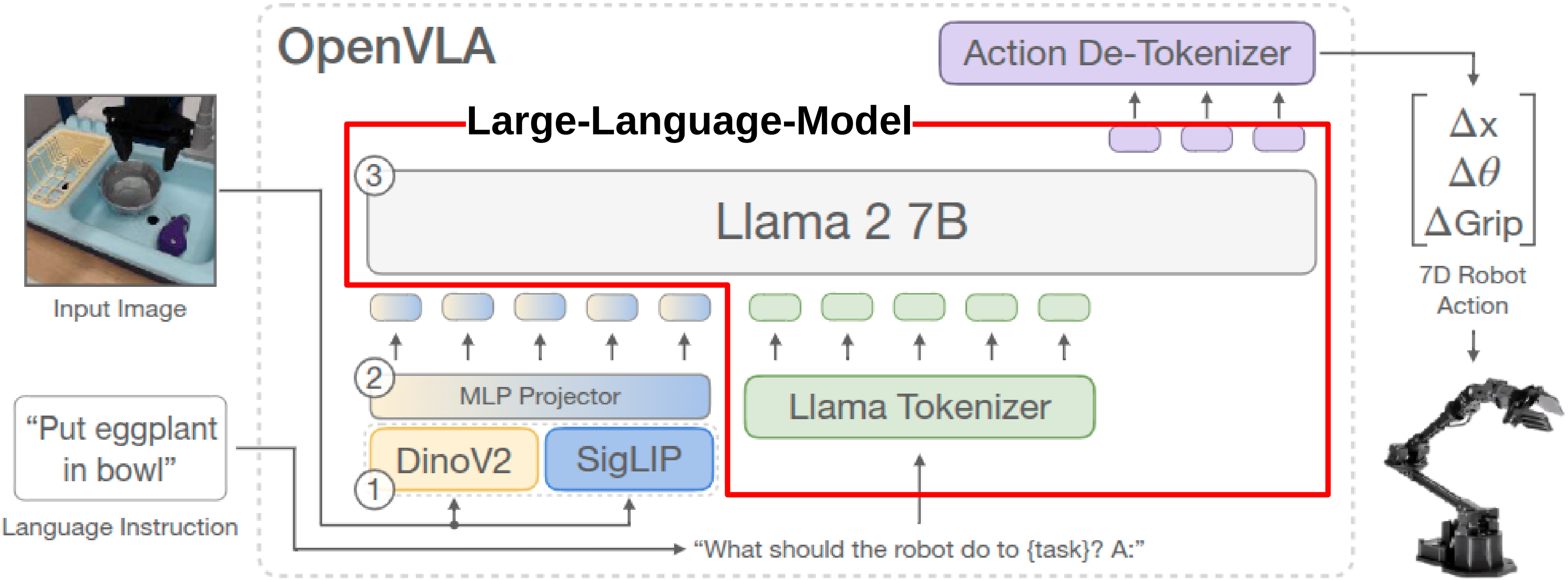


Google Robot



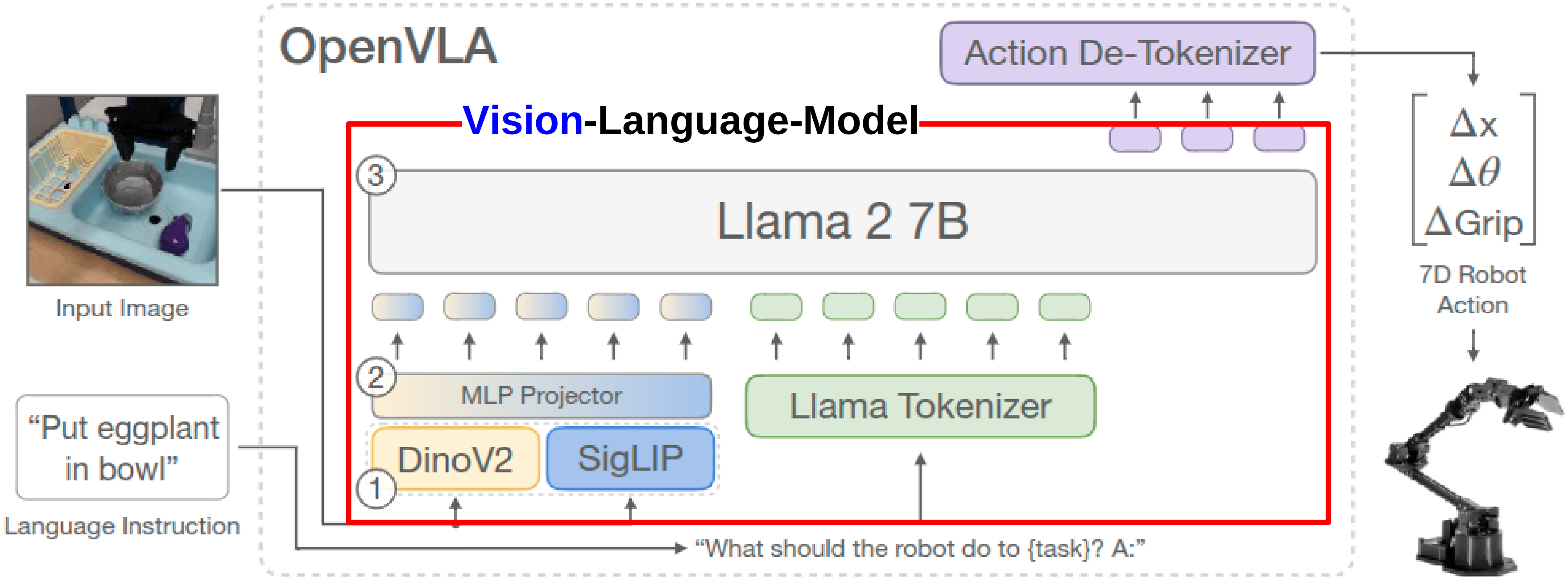
Bridge V2 WidowX Robot

OpenVLA I Architecture



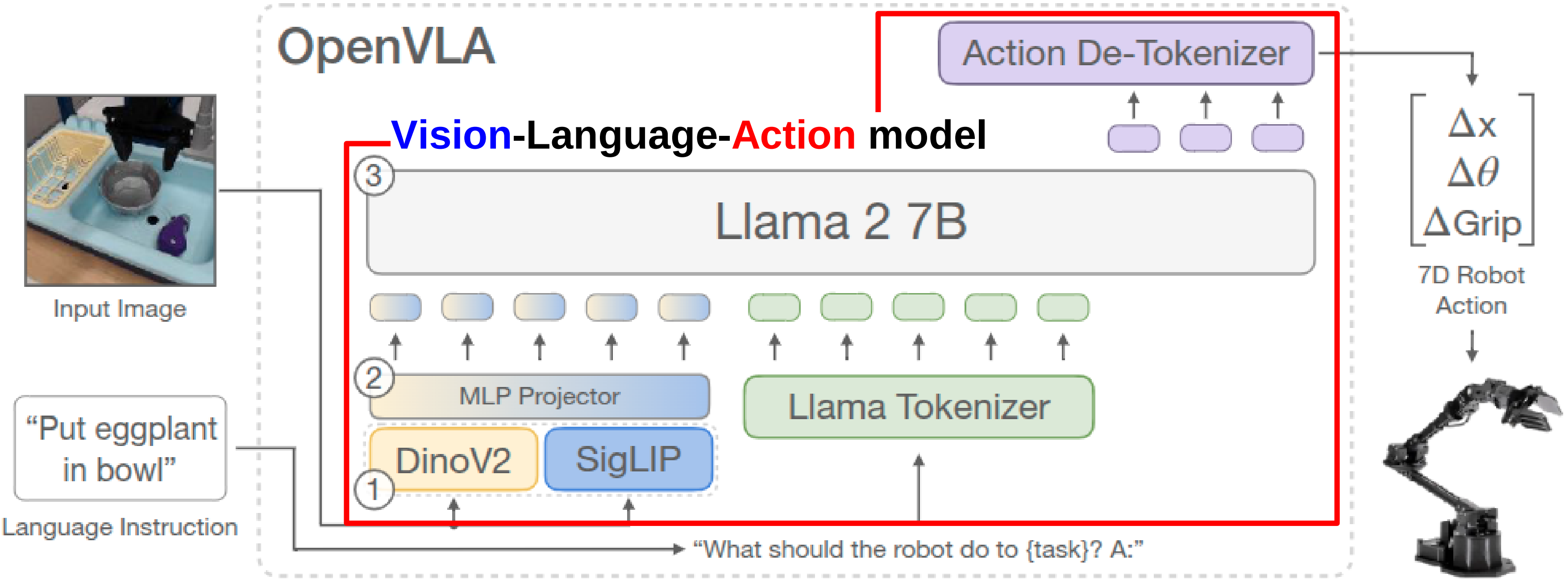
KAIST

OpenVLA | Architecture



KAIST

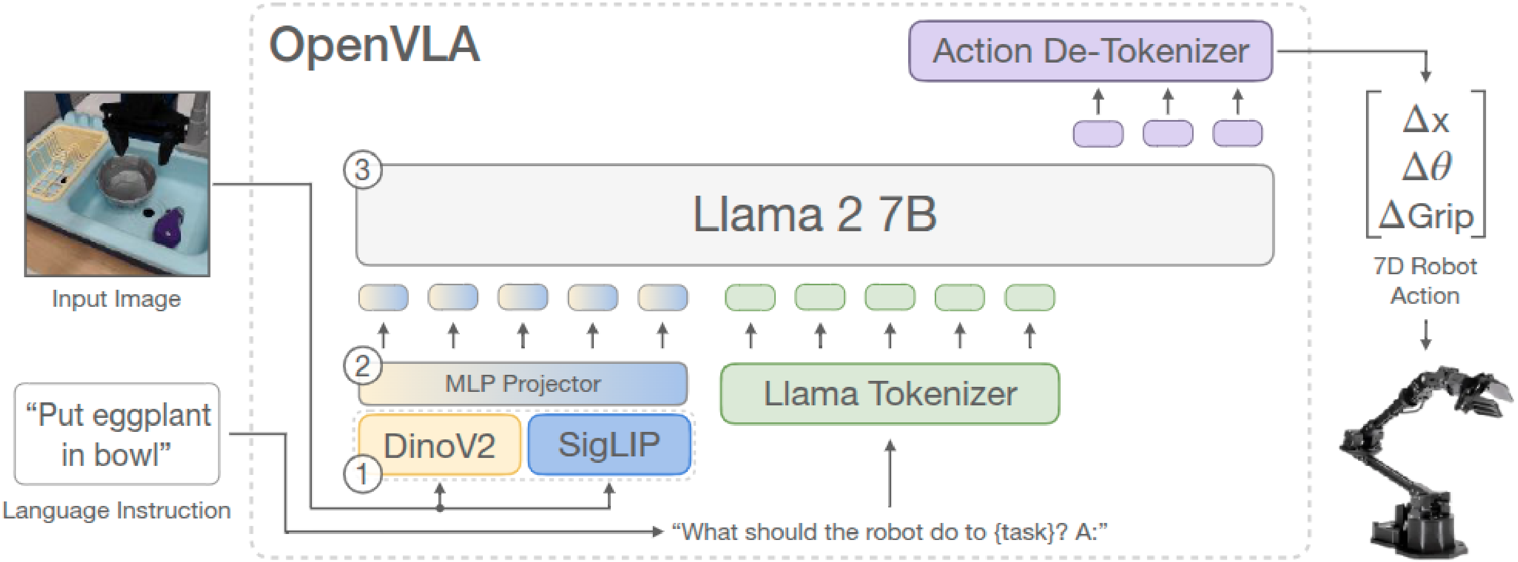
OpenVLA | Architecture



OpenVLA | Architecture

OpenVLA architecture consists of **three key components** :

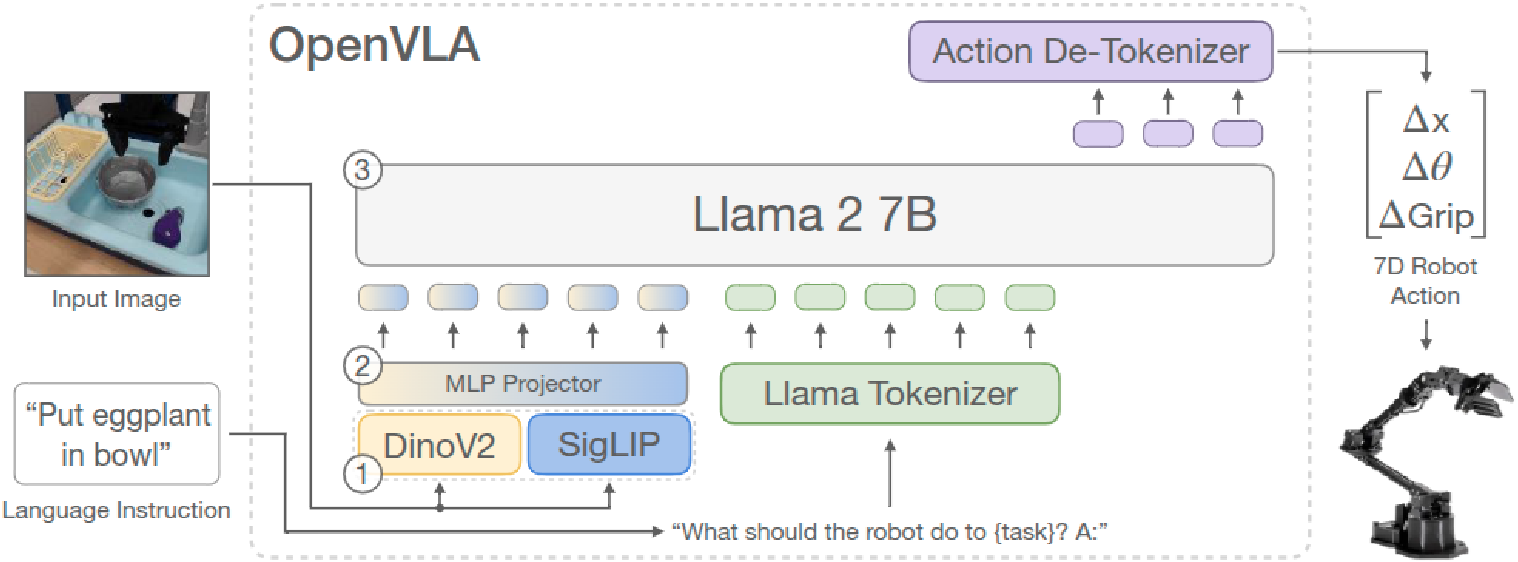
1. Vision encoder that concatenates Dino V2 and SigLIP features
2. Projector that maps visual features to the language embedding space
3. Llama 2 7B-parameter large language model



OpenVLA I Architecture

OpenVLA architecture consists of **three key components** :

1. **Vision encoder that concatenates Dino V2 and SigLIP features**
2. Projector that maps visual features to the language embedding space
3. Llama 2 7B-parameter large language model

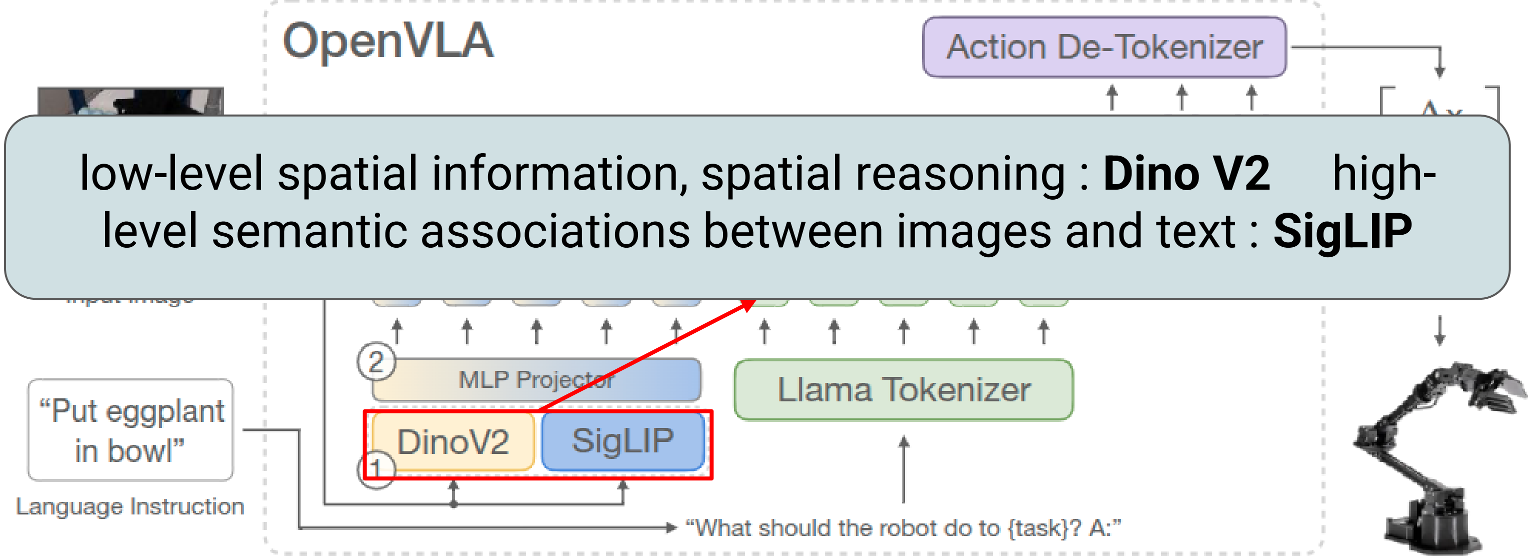


KAIST

OpenVLA I Architecture

OpenVLA architecture consists of **three key components** :

1. **Vision encoder that concatenates Dino V2 and SigLIP features**
2. Projector that maps visual features to the language embedding space
3. Llama 2 7B-parameter large language model



low-level spatial information, spatial reasoning : **Dino V2** high-level semantic associations between images and text : **SigLIP**

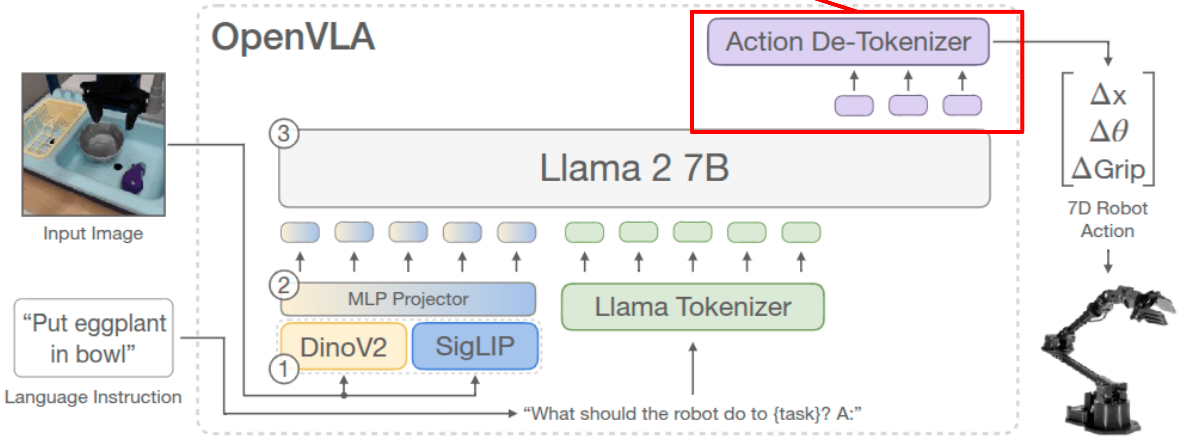
OpenVLA | Architecture

Enabling VLM to predict robot actions : **next-token prediction (CE loss)**

continuous robot action - - → **discrete tokens**

7- dim : [Δx, Δy, Δz, Δroll, Δpitch, Δyaw, gripper status]

Example Output : [0.1, 0.23, 0.42, 0.16, 0.81, 0.39, 0.33]



OpenVLA | Training

Training Data : **Open X-Embodiment dataset** 70 robot dataset with 2M trajectories

Curated under below conditions, selected **970k robot episodes**

1. Single arm manipulations, with 3rd person view camera
2. Ensure a balanced mix of embodiments, tasks, scenes

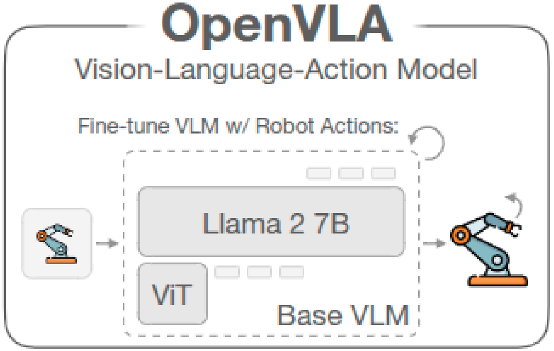
OpenVLA Training Dataset Mixture	
Fractal [92]	12.7%
Kuka [45]	12.7%
Bridge[6, 47]	13.3%
Taco Play [93, 94]	3.0%
Jaco Play [95]	0.4%
Berkeley Cable Routing [96]	0.2%
Roboturk [97]	2.3%
Viola [98]	0.9%
Berkeley Autolab UR5 [99]	1.2%
Toto [100]	2.0%
Language Table [101]	4.4%
Stanford Hydra Dataset [102]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

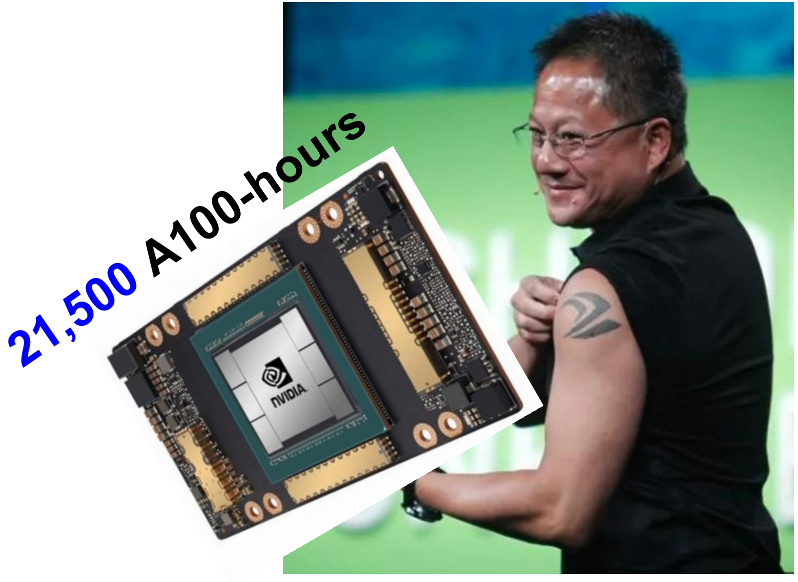
KAIST

OpenVLA | Training

Other considerations during training :

- **OpenVLA model is trained on a cluster of 64 A100 GPUs for 14 days**
  - Inference : 15GB of GPU memory when loaded bfloat16





Experiments

Experiments validate 3 key aspects of OpenVLA:

1. Performance as a Generalist Policy
Zero-shot generalization tests

2. Effectiveness of Fine-tuning
New robot setups & tasks

3. Performance with Limited Hardware
Parameter-efficient FT, quantization

KAIST

Experiments

Experiments validate 3 key aspects of OpenVLA:

1. Performance as a Generalist Policy
Zero-shot generalization tests

2. Effectiveness of Fine-tuning
New robot setups & tasks

3. Performance with Limited Hardware
Parameter-efficient FT, quantization

Experiment | Generalization

1. Performance as a Generalist Policy
Zero-shot generalization tests

Settings : Robot and tasks from **pre-trained data**

WidowX robot from BridgeData V2 evaluation

Mobile manipulation robot from RT-1 and RT-2 evaluation

Mobile manipulation robot

Categorizing the term "Generalization" :

Visual : unseen backgrounds, distractor objects, appearances of objects

Motion : unseen object positions/orientations

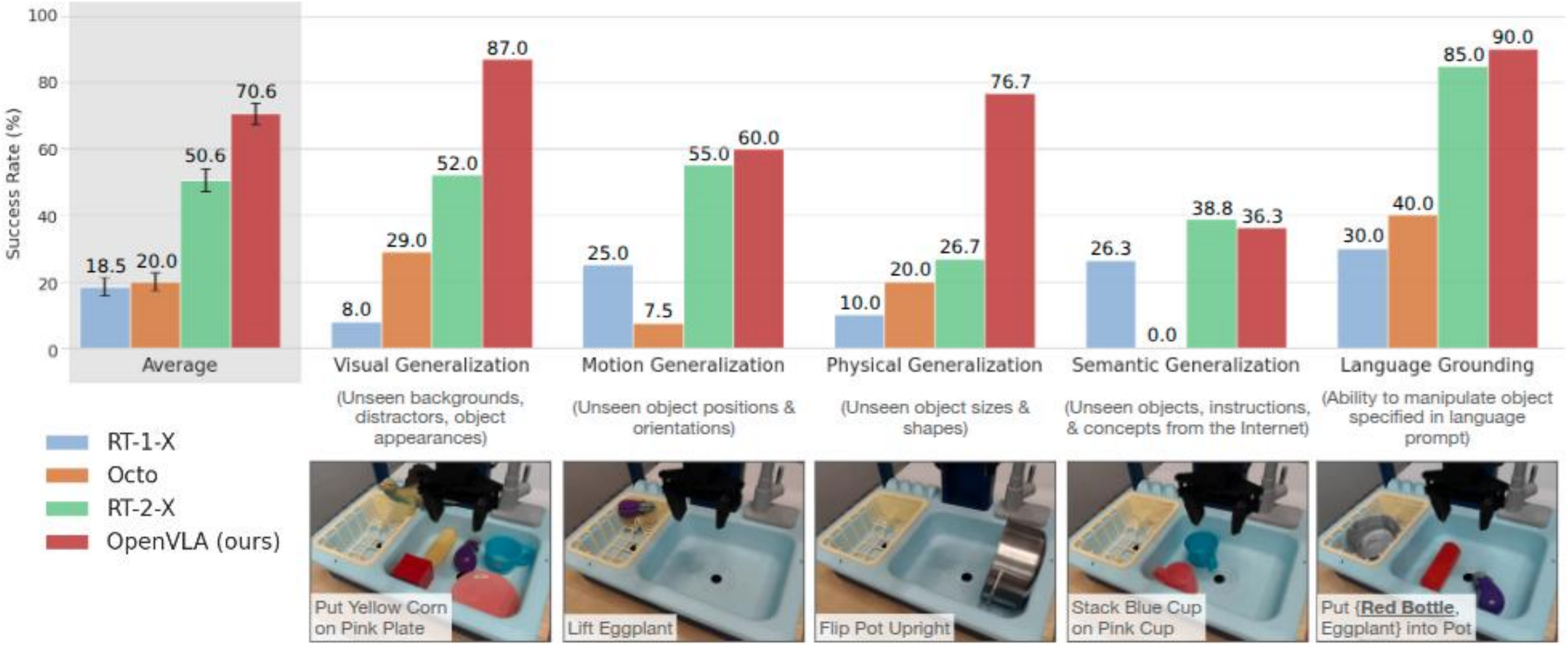
Physical : unseen object sizes/shapes

Semantic : unseen target objects, instructions, concepts from the Internet

WidowX robot

Experiment | Generalization

1. Performance as a Generalist Policy
Zero-shot generalization tests



Experiment

Experiments validate 3 key aspects of OpenVLA:

1. Performance as a Generalist Policy
Zero-shot generalization tests

2. Effectiveness of Fine-tuning
New robot setups & tasks

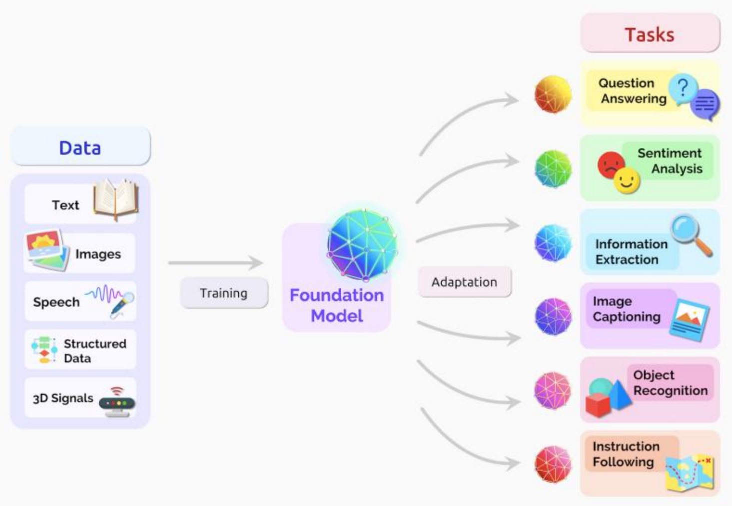
3. Performance with Limited Hardware
Parameter-efficient FT, quantization

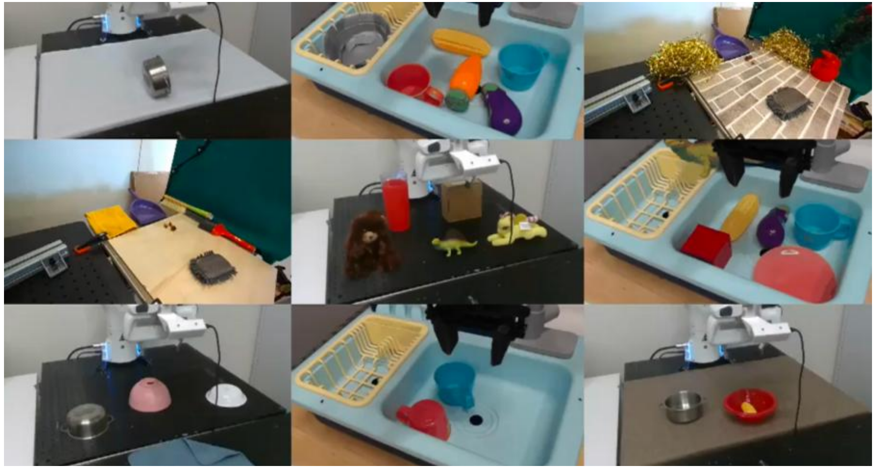
Experiments | Fine-Tuning

2. Effectiveness of Fine-tuning
New robot setups & tasks

"Adapting a pretrained VLA model to a **new robot, new environment, or new task**"

- Quick adaptation to a new setup, with a much smaller dataset
- Implicitly captures per-setup differences
    - Camera pose, embodiment, environment, reference frame, ...

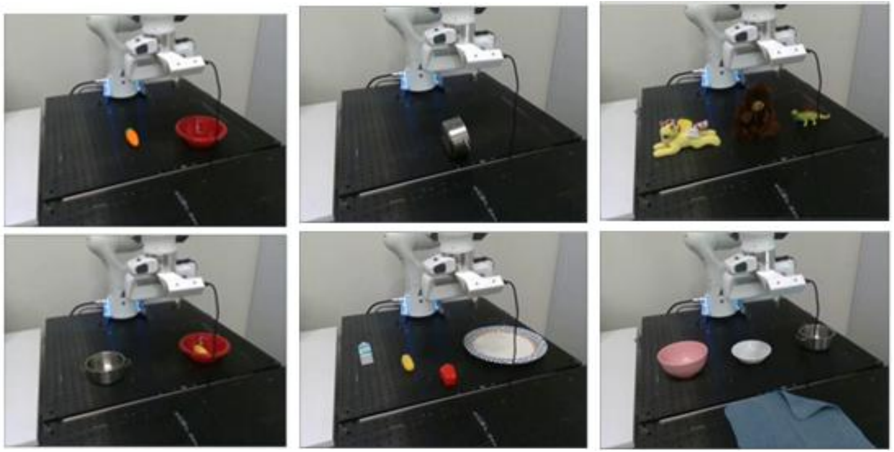




Experiments | Fine-Tuning

<u>"Is OpenVLA adaptable to new robot setups and tasks?"</u>

- Prepared Franka robot arm setups **not in pretraining data**
- Models trained / fine-tuned on 7 tasks, 10-150 demonstrations each
  - Diffusion policies, Octo, OpenVLA (scratch and pretrained)



KAIST

Experiments | Fine-Tuning

2. Effectiveness of Fine-tuning
New robot setups & tasks

1. Diffusion Policy → good for narrow, single-instruction tasks

KAIST

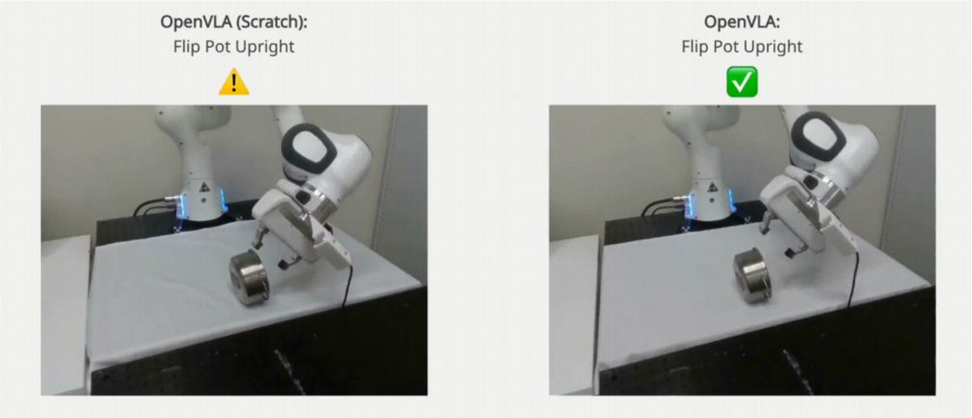
Experiments | Fine-Tuning

1. Diffusion Policy → good for narrow, single-instruction tasks
2. Fine-tuned VLAs → better with multiple objects & language conditioning

Experiments | Fine-Tuning

2. Effectiveness of Fine-tuning
New robot setups & tasks

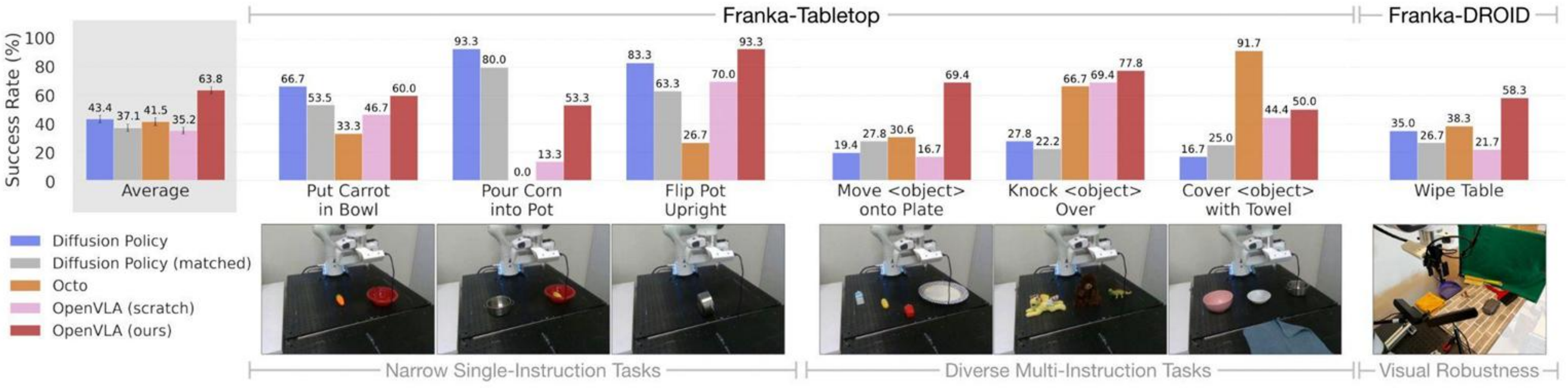
1. Diffusion Policy → good for narrow, single-instruction tasks
2. Fine-tuned VLAs → better with multiple objects & language conditioning
3. Robot data pretraining → significant performance boost



Experiments | Fine-Tuning

2. Effectiveness of Fine-tuning
New robot setups & tasks

1. Diffusion Policy → good for narrow, single-instruction tasks
2. Fine-tuned VLAs → better with multiple objects & language conditioning
3. Robot data pretraining → significant performance boost
4. OpenVLA shows **strong performance across task types**



KAIST

Experiments

Experiments validate 3 key aspects of OpenVLA:

1. Performance as a Generalist Policy
Zero-shot generalization tests

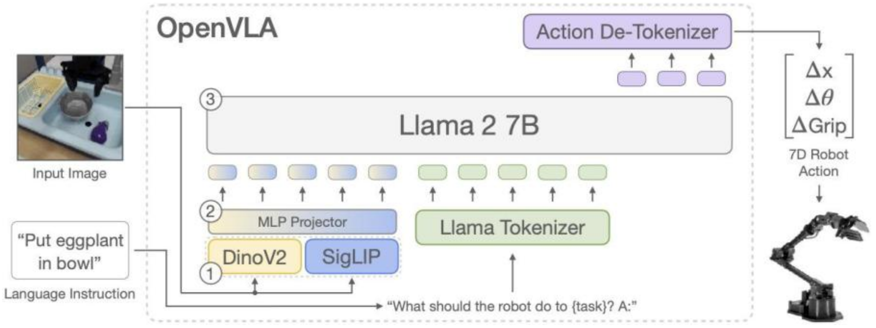
2. Effectiveness of Fine-tuning
New robot setups & tasks

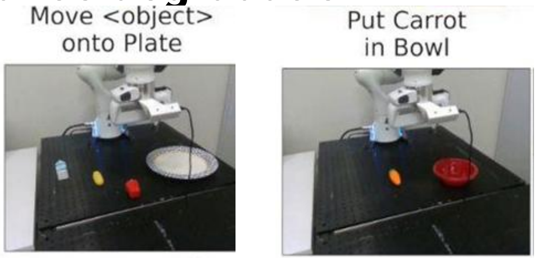
3. Performance with Limited Hardware
Parameter-efficient FT, quantization

Experiments | Parameter-efficient FT

Tested different fine-tuning strategies

- Strategies: Full FT, Last layer only, Frozen vision, Sandwich, LoRA
- Criteria: memory requirement, performance
- Remark: LoRA is **efficient**, with **minimal performance degradation**!



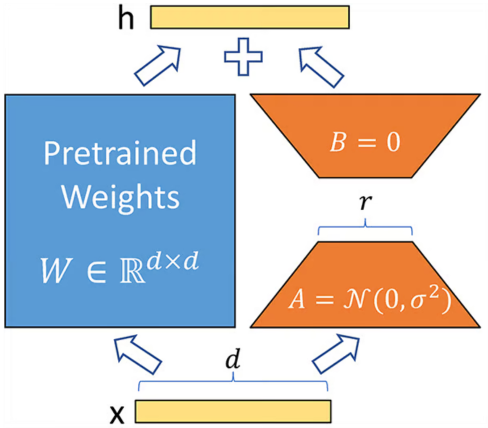


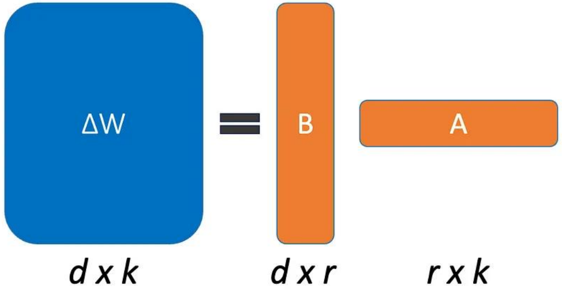
Strategy	Success Rate	Train Params ($\times 10^6$)	VRAM (batch 16)
Full FT	**69.7 $\pm$ 7.2 %**	7,188.1	163.3 GB*
Last layer only	30.3 $\pm$ 6.1 %	465.1	51.4 GB
Frozen vision	47.0 $\pm$ 6.9 %	6,760.4	156.2 GB*
Sandwich	62.1 $\pm$ 7.9 %	914.2	64.0 GB
LoRA, rank=32	**68.2 $\pm$ 7.5%**	**97.6**	**59.7 GB**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

Experiments | Parameter-efficient FT

LoRA: Low-Rank Adaptation of Large Language Models

- Vanilla fine-tuning: update (d x h) weights, expensive
- LoRA: only train B and A, where BA = ΔW (weight updates)
- Insight
    - Pretrained W is near a good solution
    - For fine-tuning, updating all weights is unnecessary!

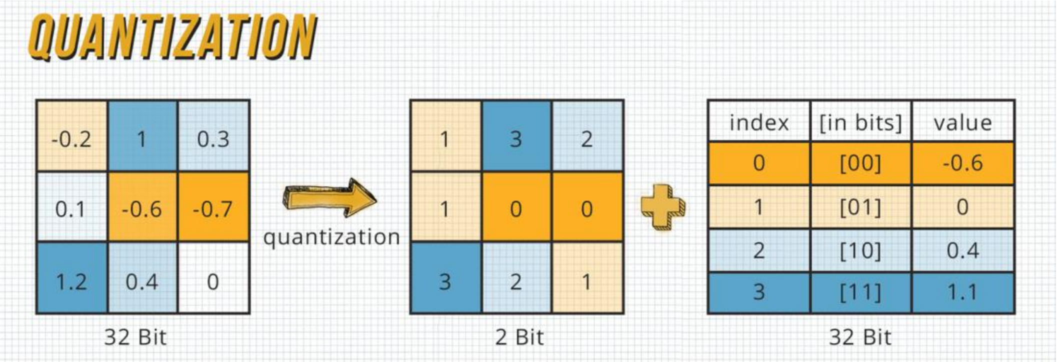




Experiments | Quantization

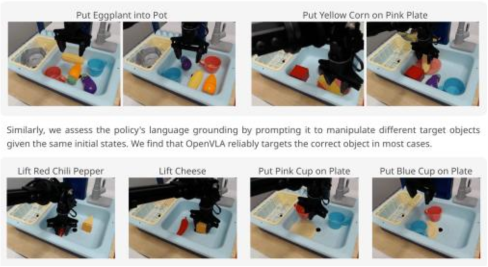
Many foundation models require large GPU memory (even **for inference**)

- Idea: reduce the precision (# of bits used) of model weights
  - Less memory footprint
  - Extra quantization / dequantization overhead
  - Increased arithmetic error
- **Memory ↔ Performance tradeoff**



Experiments | Quantization

- Tested on BridgeData V2 tasks with different quantization levels
- Quantization overhead can reduce throughput
    - 8-bit: throughput too low for a 5Hz control loop
- Quantization can lower GPU memory transfer (**improve throughput**)
    - 4-bit: reduced memory transfer compensates overhead!



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

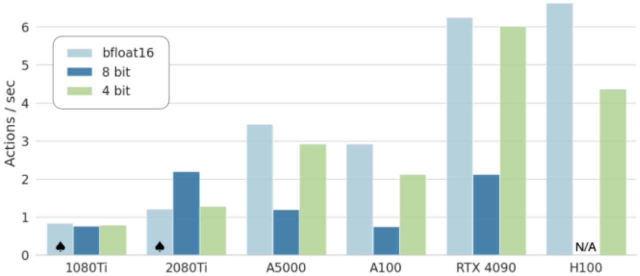


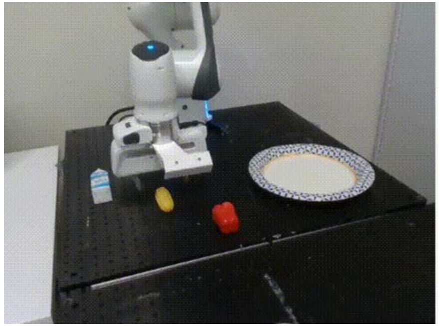
Figure 6: **OpenVLA inference speed for various GPUs.** Both bfloat16 and int4 quantization achieve high throughput, especially on GPUs with Ada Lovelace architecture (RTX 4090, H100). Further speed-ups are possible with modern LLM inference frameworks like TensorRT-LLM [89]. ♠: Model sharded across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

OpenVLA | Limitation

- Only support single-image observation
- Inference throughput (Hz)
- Room for performance improvement (90+ %)

OpenVLA | Summary

- Open-source VLA model for manipulators
    - Image + language instruction → robot action
- CV & NLP advancements on robotic applications
- Towards large, ‘generalist’, widely-deployable robot models
    - Robot data pretraining → general performance
    - Fine-tuning (LoRA) → task-specific adaptation
    - Quantization → memory-efficient inference



Thank you

Quiz

1. According to OpenVLA's fine-tuning experiments, the key benefit of LoRA over full fine-tuning is the drastic improvement in **performance** (task success rate).
   a. True
   b. False

2. Which is **NOT** true about OpenVLA?
   a. OpenVLA can be fine-tuned on new robot tasks with 10-150 demonstrations.
   b. OpenVLA strictly outperforms Diffusion Policy across all tasks.
   c. OpenVLA supports low-memory deployment using quantization.