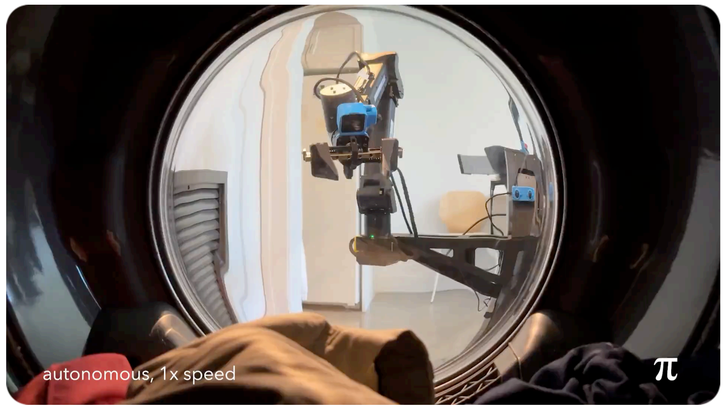
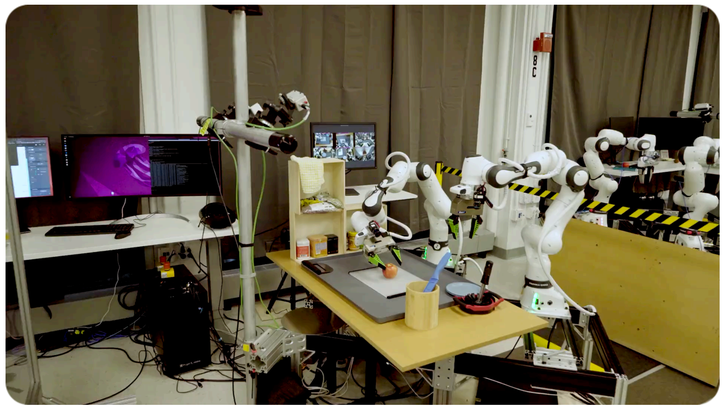
Perspectives on Designing Vision Language Action Models

Ankit Goyal

The Evolving State of Robotics

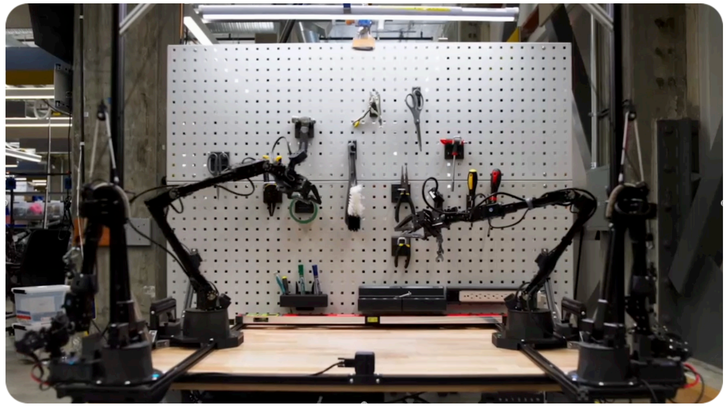


Physical Intelligence



Toyota Research Institute

Dyna Robotics



Google Robotics

Figure Robotics

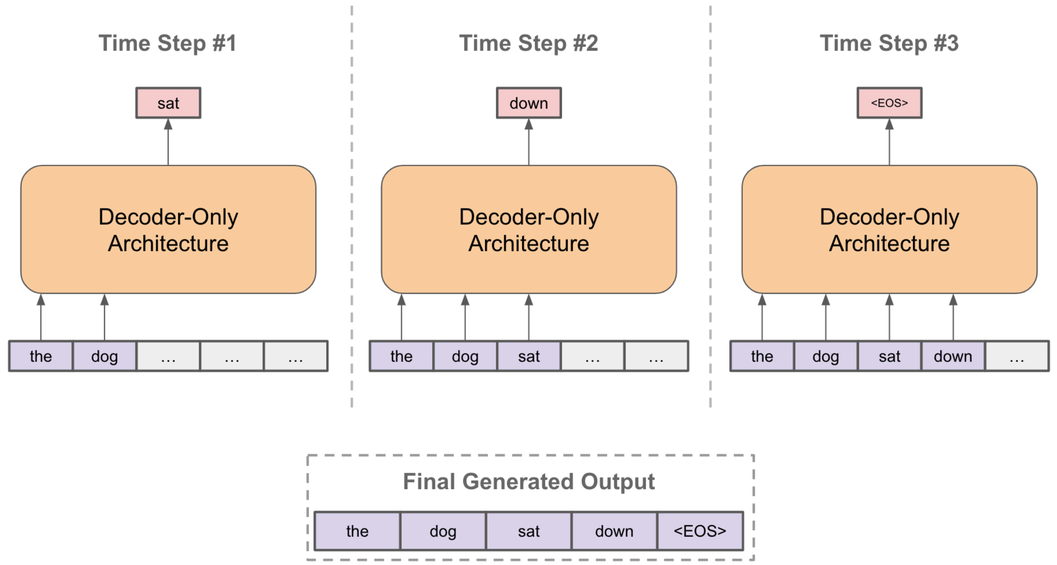
Nvidia GR00T

[Courtesy: Physical Intelligence, TRI, Dyna, Google Robotics, Figure Robotics, NVIDIA GR00T]

Background

LLMs

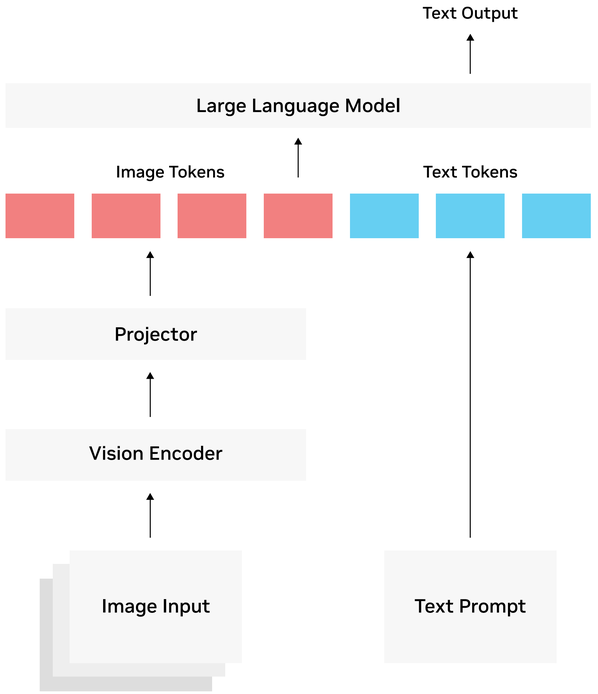
- LLMs (Large Language Models) — They predict one token (similar to sub-word) at a time
- Trained with large scale data and many other tricks!



Background

VLMs

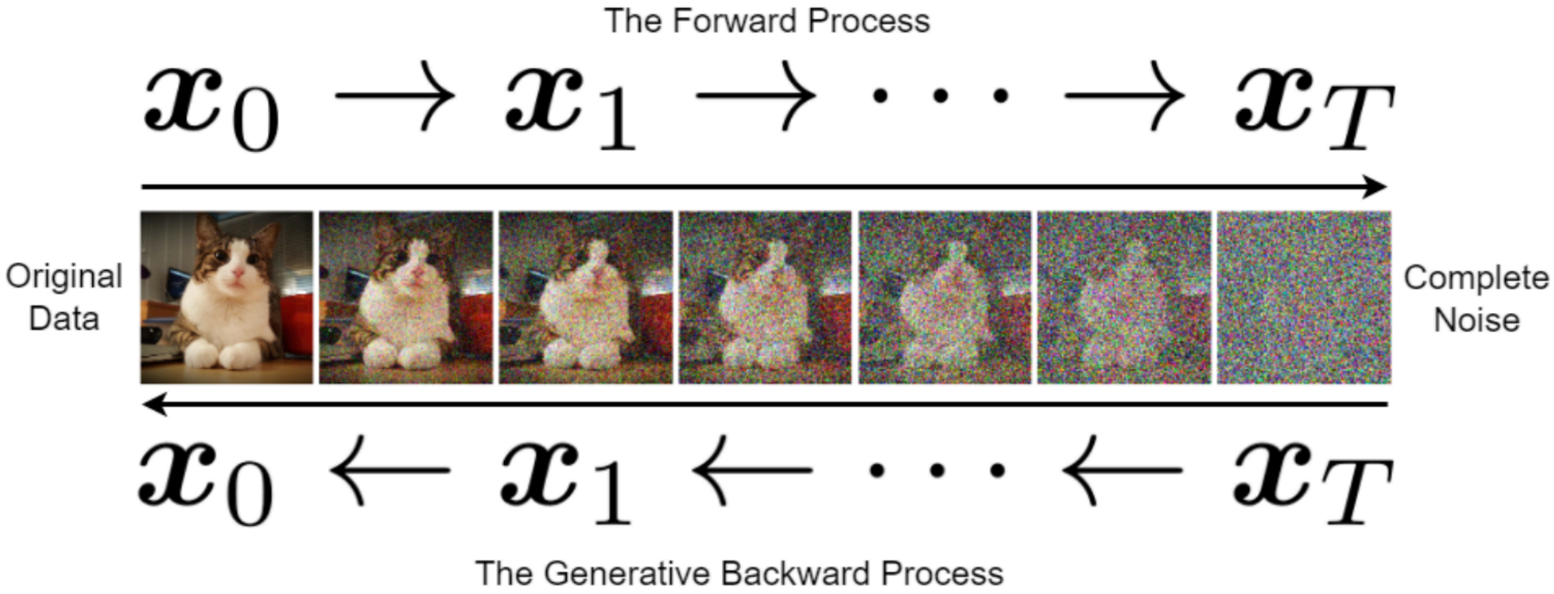
- We start with an LLM and then introduce image tokens to it
- Image tokens come from a pre-trained Vision Encoder
- Fine-tune the LLM on this joint task to create a VLM



Background

Diffusion Model

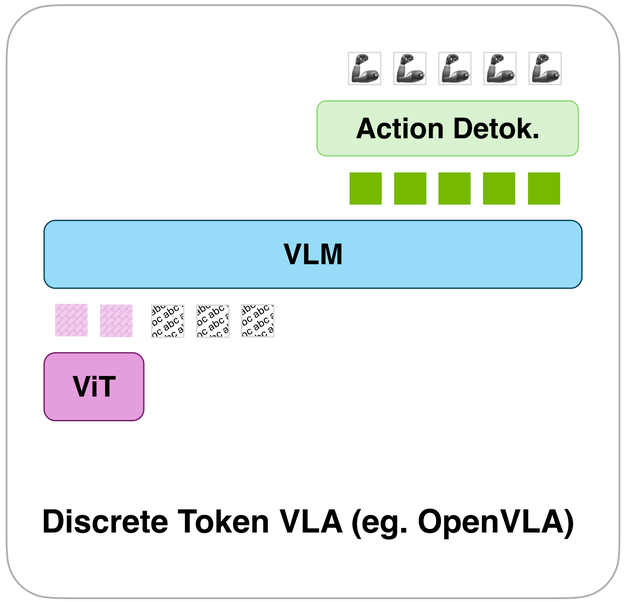
- A class of generative model — i.e. they can be used to generate
- They are trained via adding progressive adding noise to a sample and then asking a network to denoise it
- During generation, we start with noise and progressively denoise it



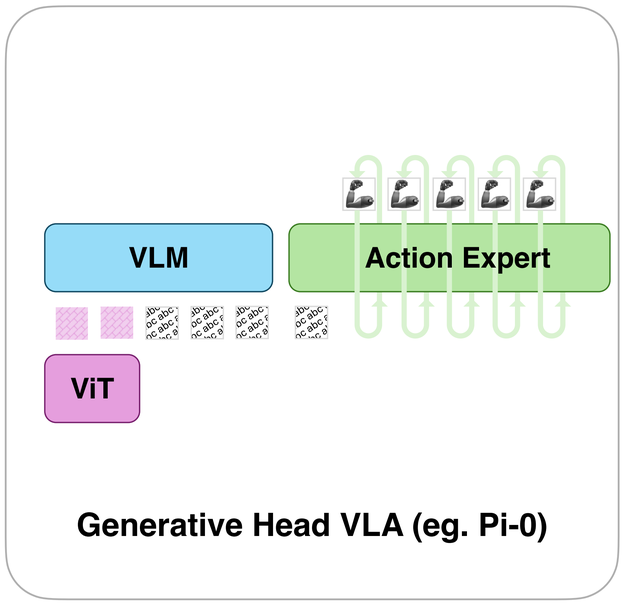
What are Vision Language Action Models ?

Family of VLAs

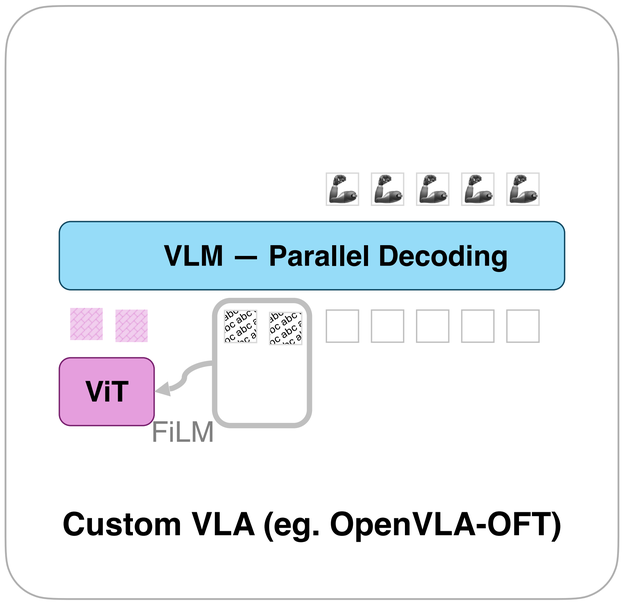
- LLMs power Language Understanding, VLMs bridge Vision and Language — VLAs extend this to Actions
- VLAs build on pretrained VLMs — Adapting them to reason about Actions
- VLAs in the existing literature can be classified into three categories



Discrete Token VLA (eg. OpenVLA)



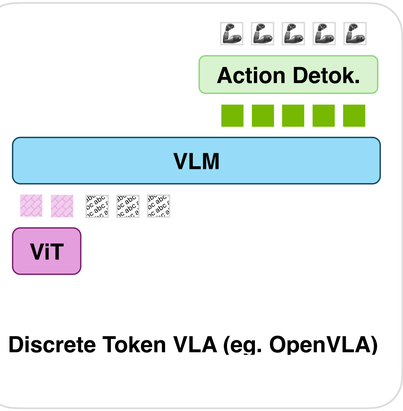
Generative Head VLA (eg. Pi-0)



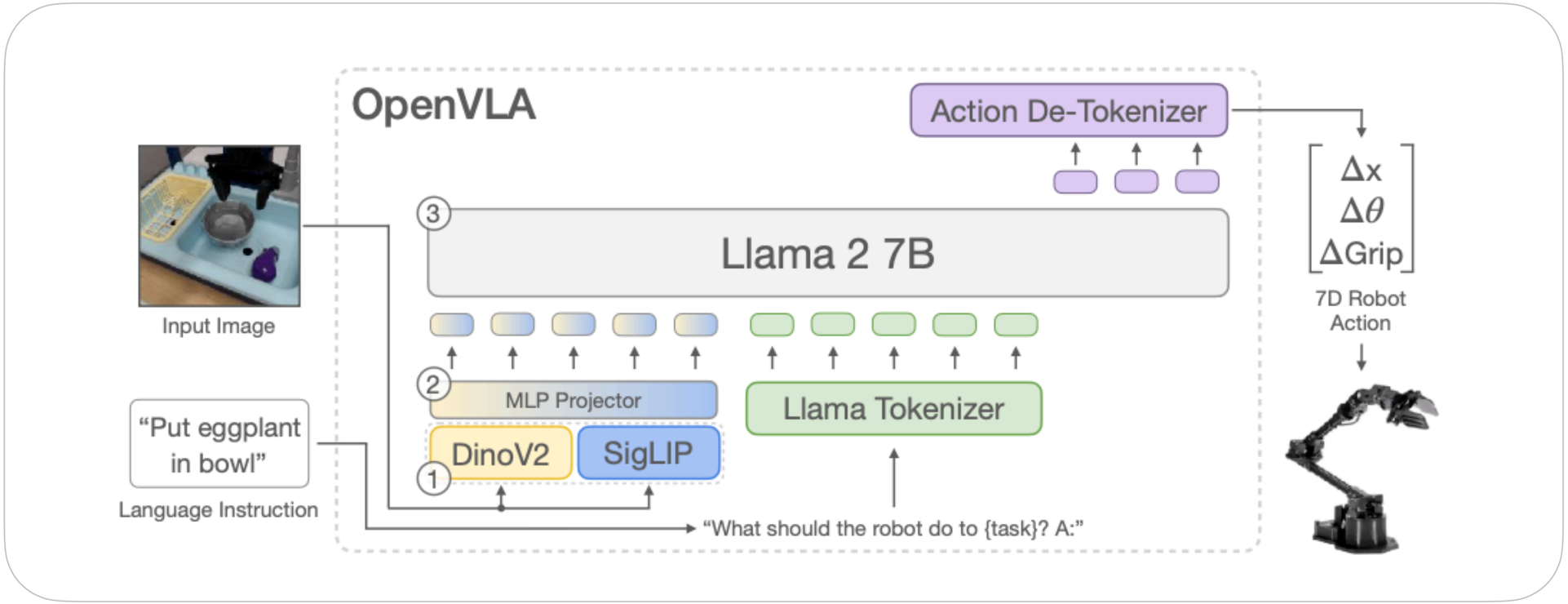
Custom VLA (eg. OpenVLA-OFT)

I. Discrete Token VLAs

E.g. OpenVLA



- Replace or modify the vocabulary to introduce Discrete Action Tokens
- Limited Action Resolution
- Compromised pretrained language representations

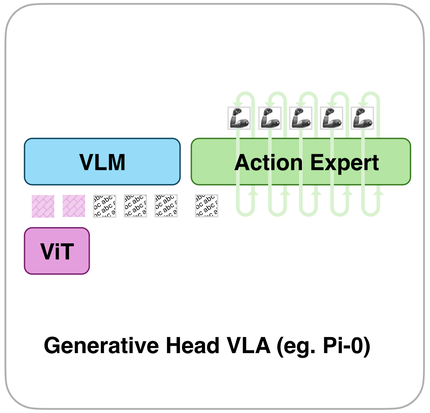


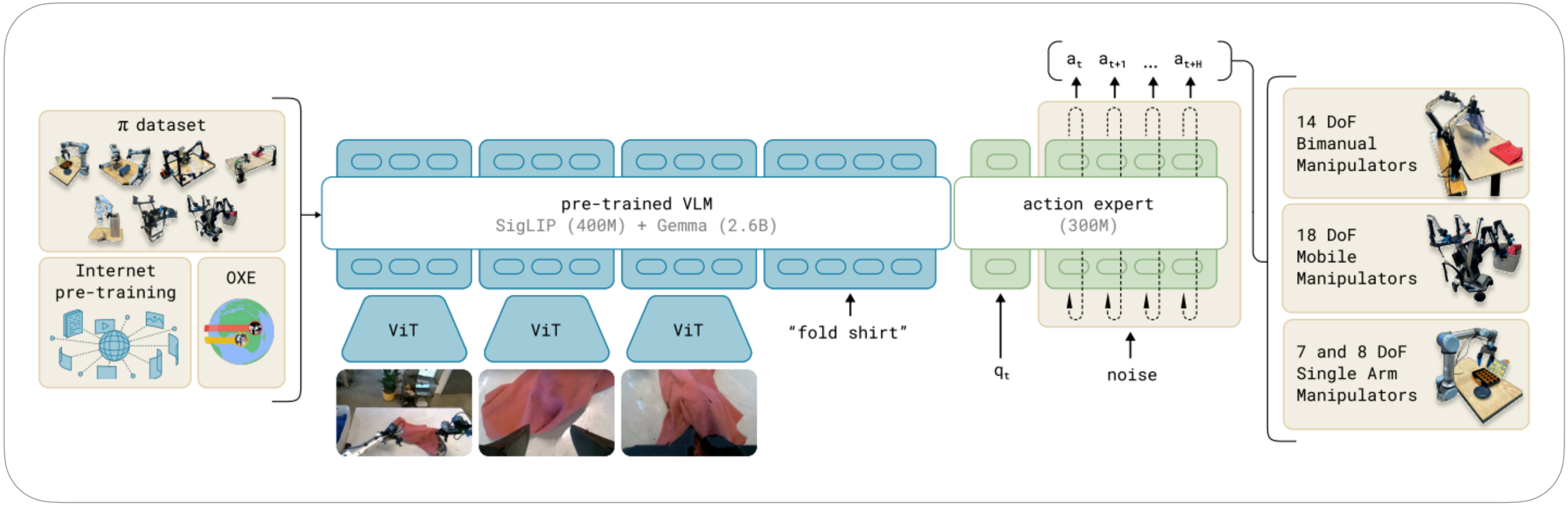
[Kim et al: CoRL 2024]

II. Generative Head VLAs

Eg. Pi-0

- VLM predicts latent vector, fed into a Generative Head (Diffusion/Flow Match.) to predict actions
- Non-pretrained components reduce model generalization
- Compromised pretrained language representations



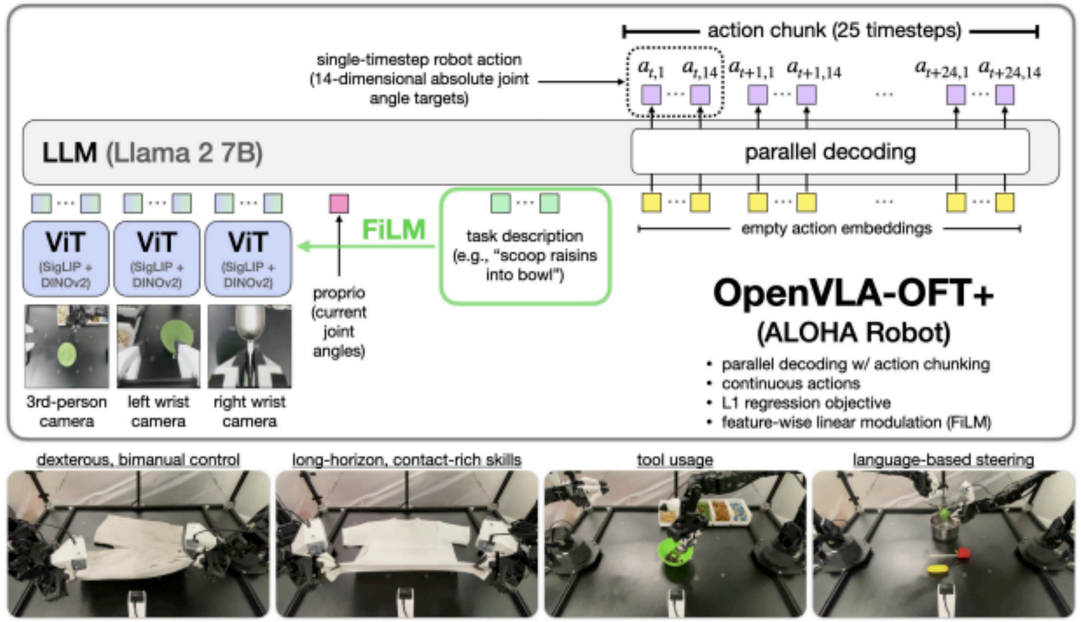


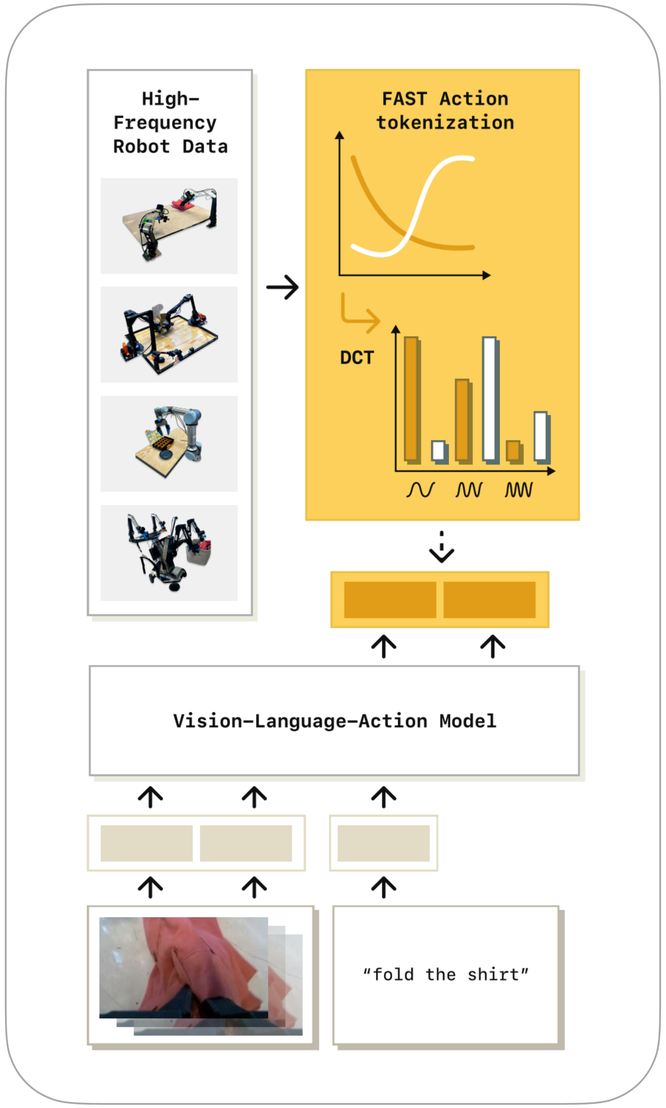
[Black et al.]

III. Custom VLA Designs

Eg. OpenVLA-OFT

- Not in the above two general categories
- Significant architectural changes
- Specialized Training Pipelines

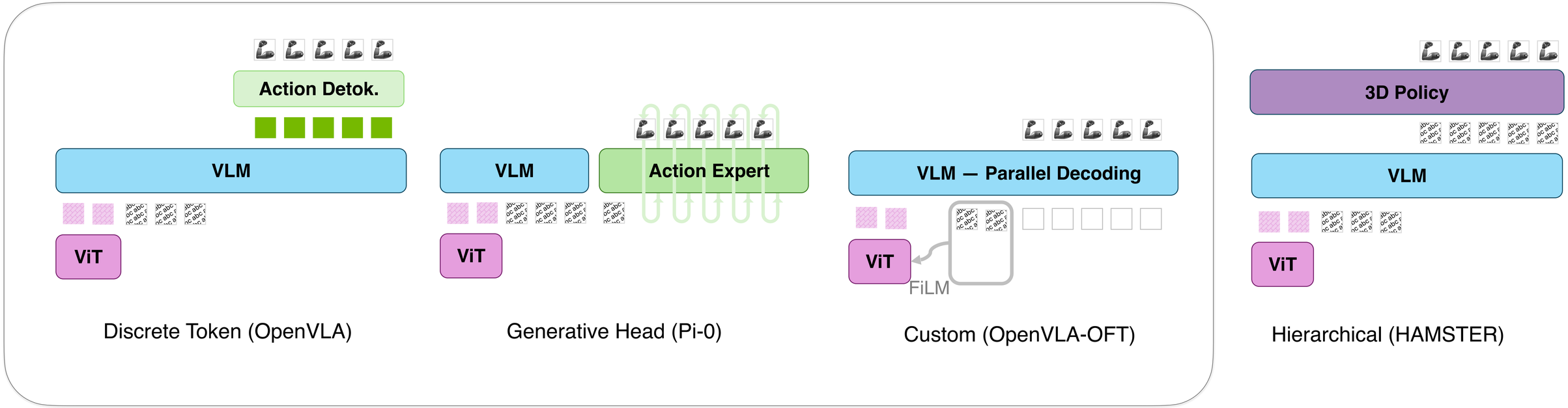




[Kim et al:RSS 2024.; Pertsch et al.]

Family of VLAs

- Hierarchical design instead of monolithic design
- Leverage VLMs for generalization and specialized policies (3D) for action prediction



HAMSTER: Hierarchical Action Models For Open-World Robot Manipulation

ICLR 2025

Put M on Mona Lisa

VLM

The HAMSTER VLM predicts points to denote the 2D paths

Hamster VLA Training Data

High-Level model is trained on a variety of data sources (Sim, Real, Point-QA)

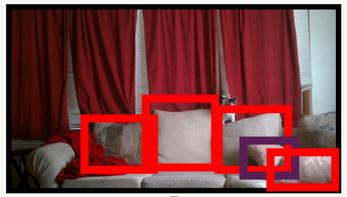
Instr. *z*

Image *img*

Paths *ans*

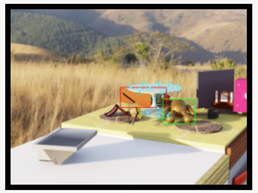
Pixel Point Prediction

Find all instances of cushions



[(0.49, 0.38, 0.08, 0.06), (0.53, 0.42, 0.07, 0.05),...]

Locate object between the marked items

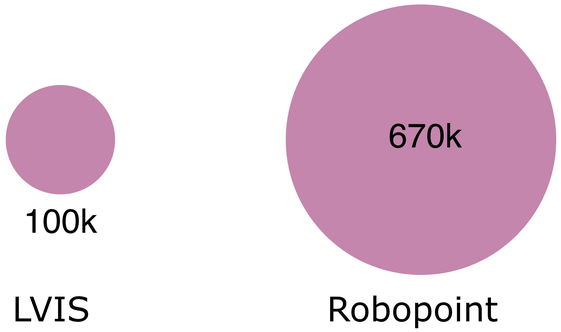


[(0.57, 0.48), (0.58, 0.49),(0.56, 0.45), (0.55, 0.47), ...]

Skills

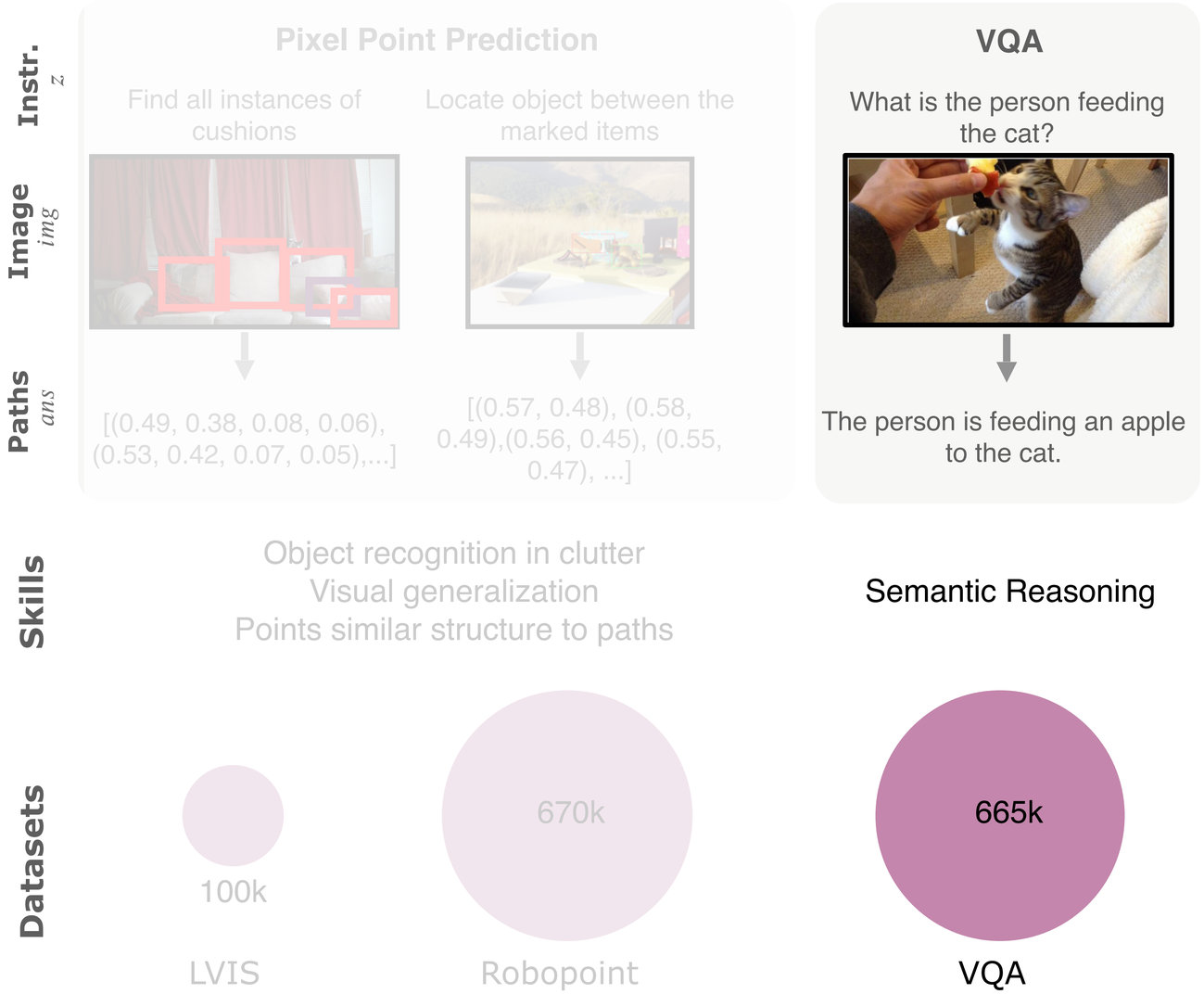
Object recognition in clutter
Visual generalization
Points similar structure to paths

Datasets



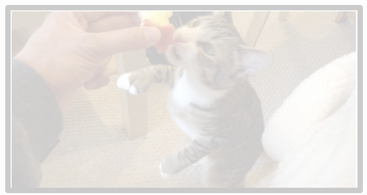
Hamster VLA Training Data

High-Level model is trained on a variety of data sources (Sim, Real, Point-QA)



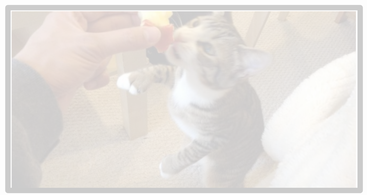
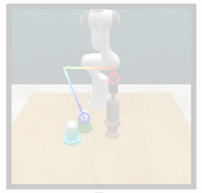
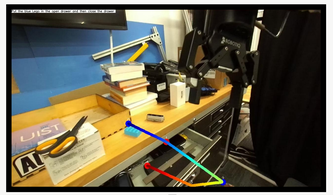
Hamster VLA Training Data

High-Level model is trained on a variety of data sources (Sim, Real, Point-QA)

	Pixel Point Prediction		VQA	Simulated Robot
Instr. z	Find all instances of cushions	Locate object between the marked items	What is the person feeding the cat?	Screw in the green light bulb
Image img				
Paths ans	[(0.49, 0.38, 0.08, 0.06), (0.53, 0.42, 0.07, 0.05),...]	[(0.57, 0.48), (0.58, 0.49),(0.56, 0.45), (0.55, 0.47), ...]	The person is feeding an apple to the cat.	[(0.4, 0.6, close), (0.4, 0.6, close), (0.8, 0.7, open)]
Skills	Object recognition in clutter Visual generalization Points similar structure to paths		Semantic Reasoning	Visual diversity Data-scarce tasks
Datasets	100k LVIS	670k Robopoint	665k VQA	320k RLBench

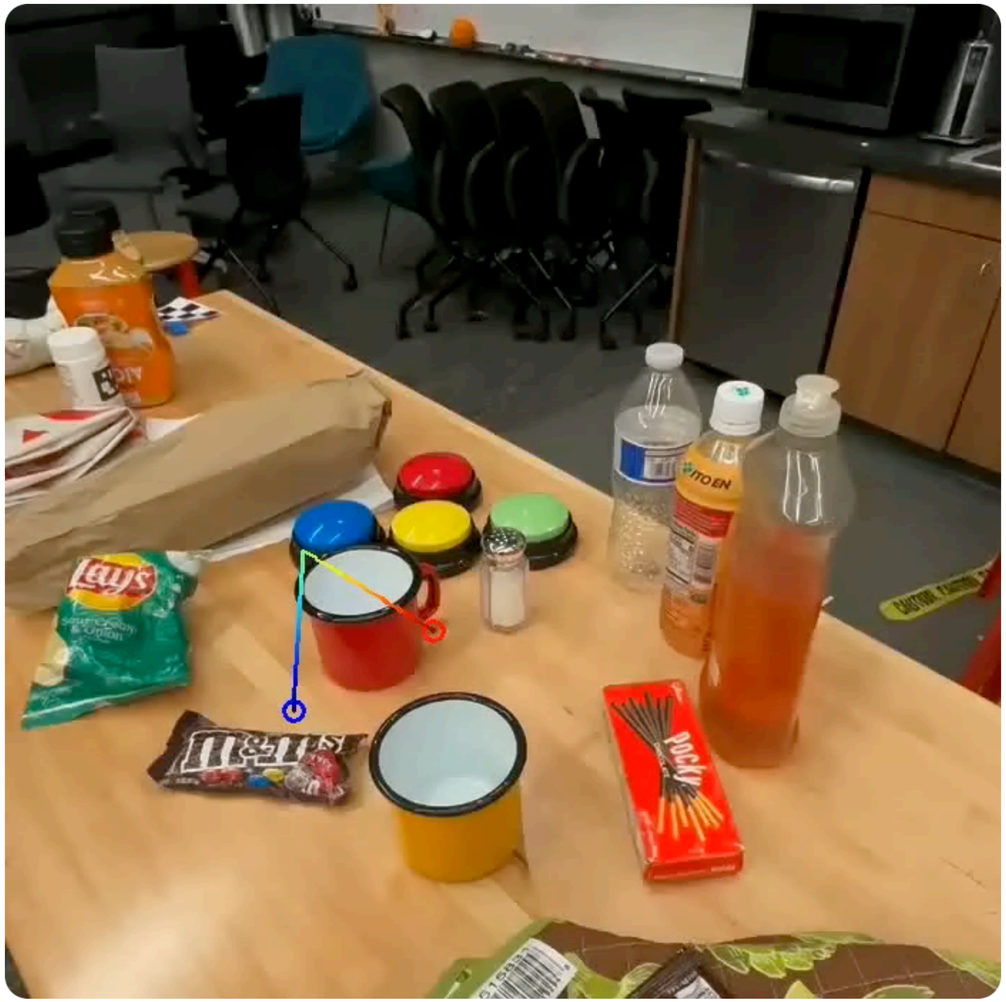
Hamster VLA Training Data

High-Level model is trained on a variety of data sources (Sim, Real, Point-QA)

	Pixel Point Prediction		VQA	Simulated Robot	Off Domain Robot	
Instr. *z*	Find all instances of cushions	Locate object between the marked items	What is the person feeding the cat?	Screw in the green light bulb	Cover the bowl with the towel	Put the marker inside the silver pot
Image *img*						
Paths *ans*	[(0.49, 0.38, 0.08, 0.06), (0.53, 0.42, 0.07, 0.05),...]	[(0.57, 0.48), (0.58, 0.49),(0.56, 0.45), (0.55, 0.47), ...]	The person is feeding an apple to the cat.	[(0.4, 0.6, close), (0.4, 0.6, close), (0.8, 0.7, open)]	[(0.2, 0.2, close), (0.3, 0.2, close), (0.1, 0.2, close), (0.1, 0.3, open)]	[(0.7, 0.5, close), (0.5, 0.6, close), (0.6, 0.7, close), (0.7, 0.6, open)]
Skills	Object recognition in clutter Visual generalization Points similar structure to paths		Semantic Reasoning	Visual diversity Data-scarce tasks	Grounding to real robot prediction	
Datasets	100k LVIS	670k Robopoint	665k VQA	320k RLBench	10k BridgeDatav2	45k DROID

Hamster VLA Results

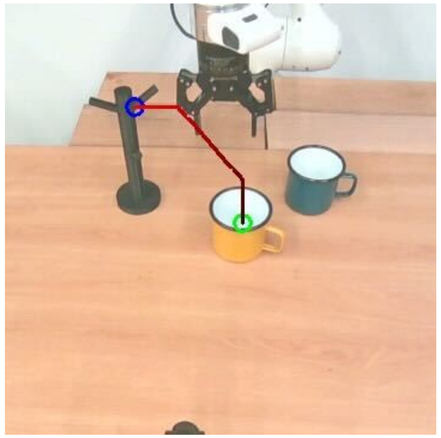
Robust To Novel Camera Position

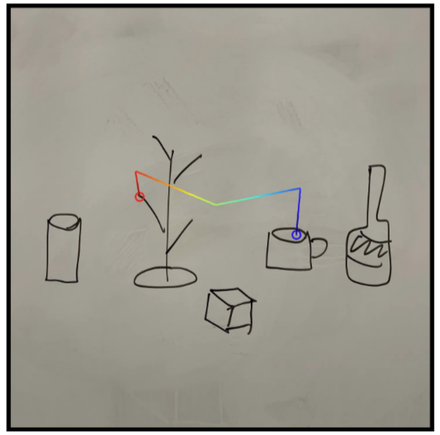


Instruction: Pick up the M&M chocolate and put it in the yellow mug

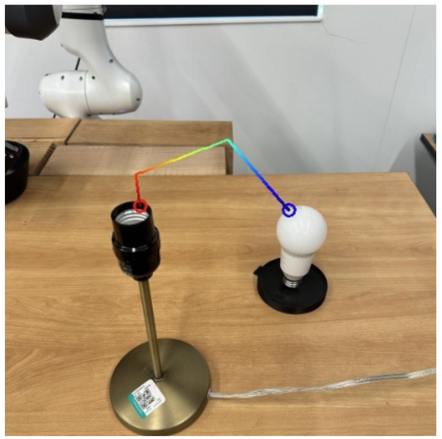
Hamster VLA Results

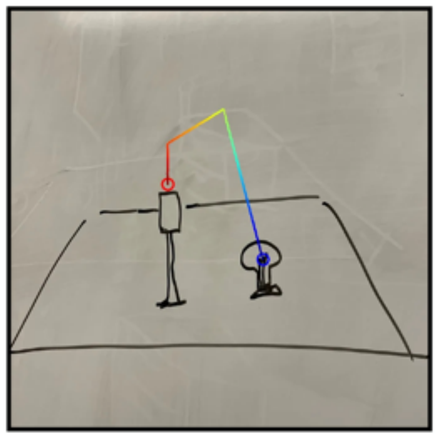
Sim to Real to Sketch





Instruction: Put the cup in the cup holder

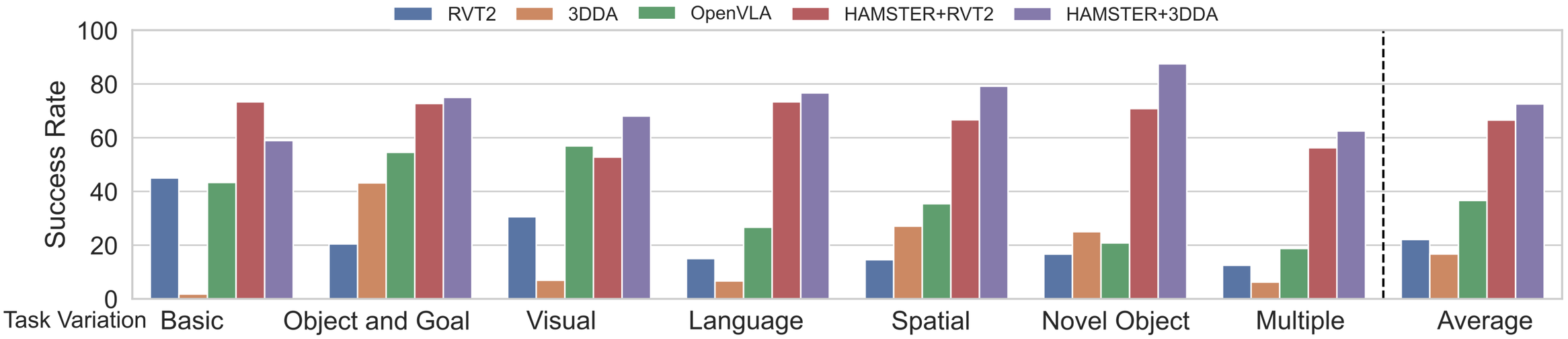




Instruction: Screw in the light bulb

Hamster Results

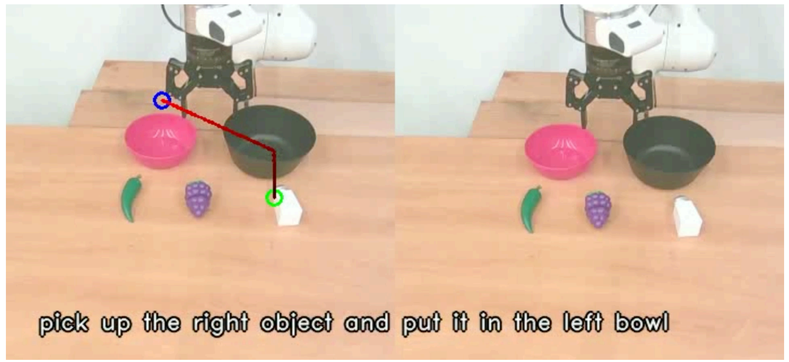
Outperforms OpenVLA as well as 3D Policies



Failure Analysis

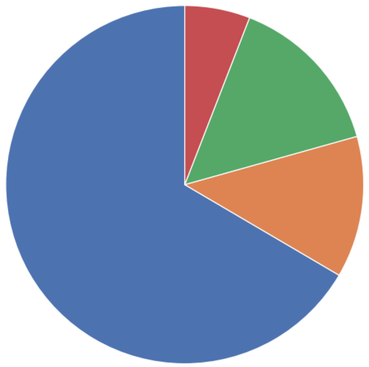
Low-Level Policy Struggles

VLM Failure

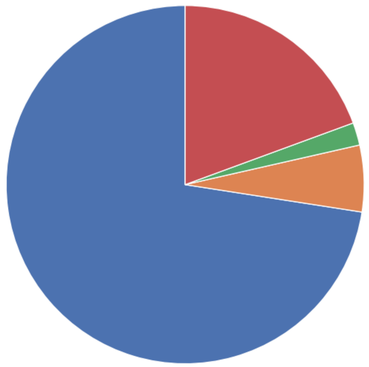


Fail to Follow Trajectory

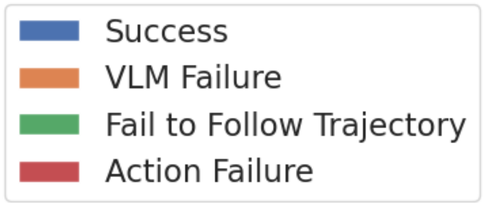
Action Failure



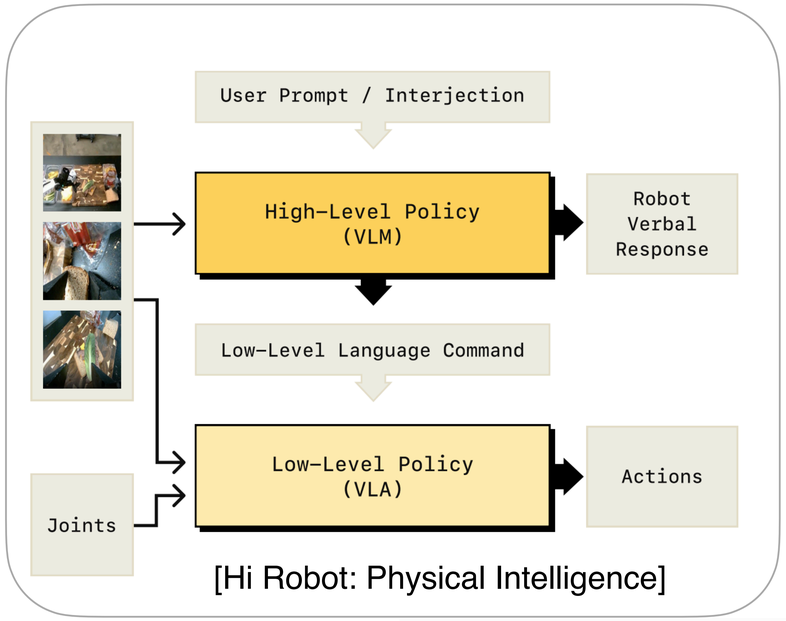
HAMSTER with RVT-2



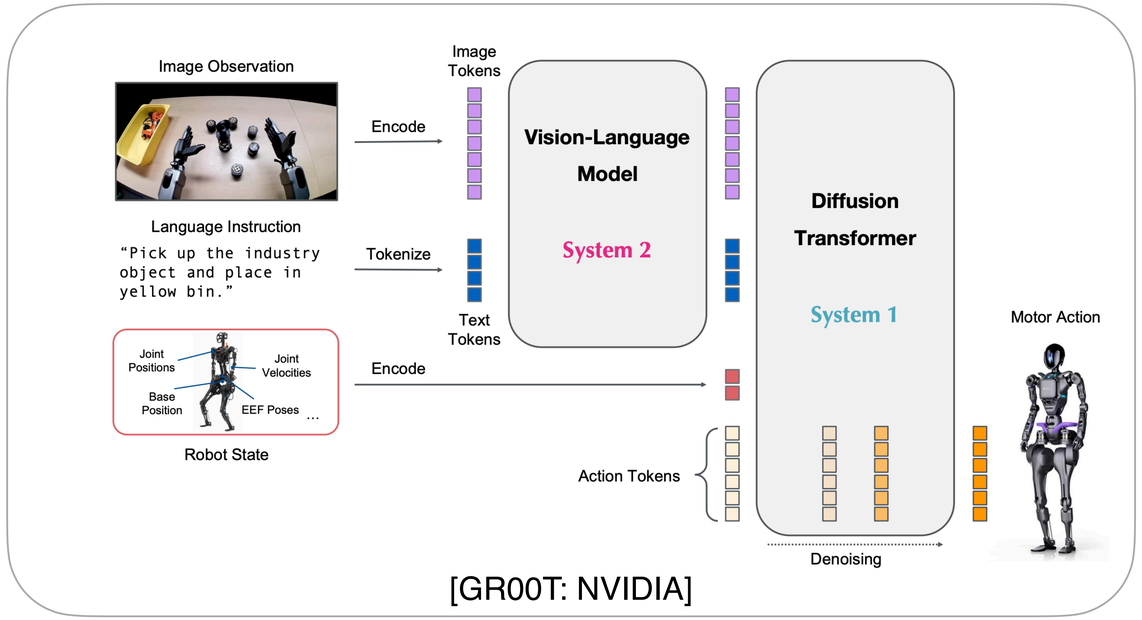
HAMSTER with 3D-DA



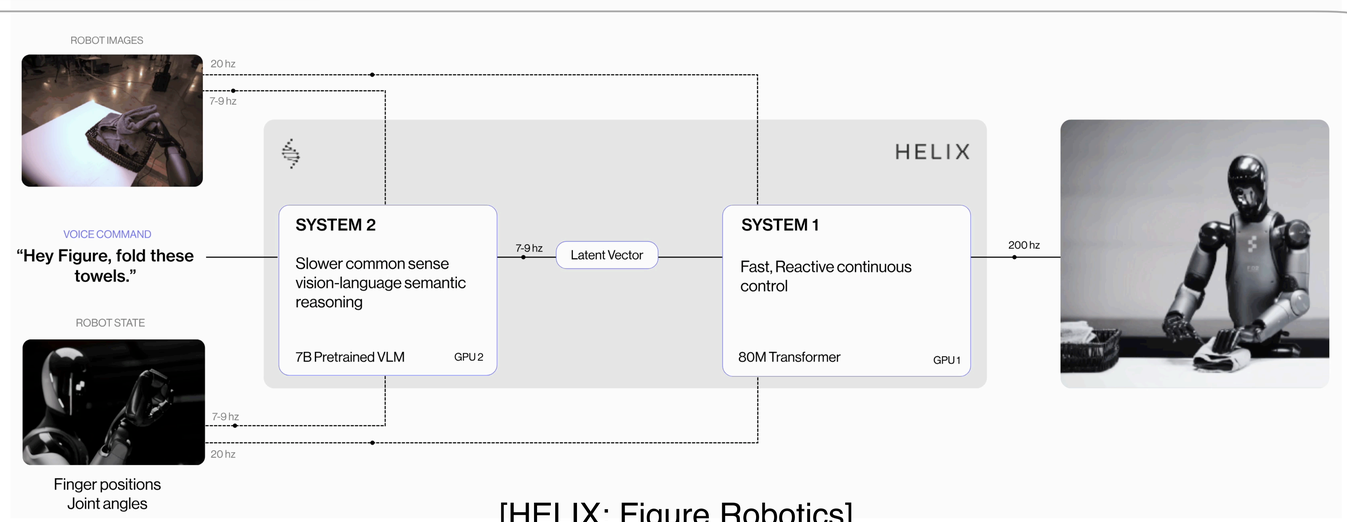
Hierarchical VLAs: Concurrent Works



[Hi Robot: Physical Intelligence]



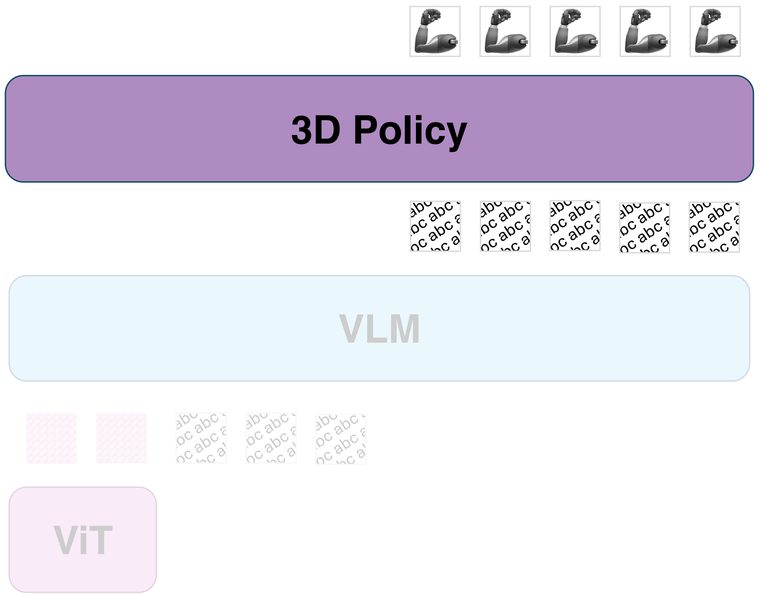
[GR00T: NVIDIA]



[HELIX: Figure Robotics]

3D Policies

Specialized and Efficient



- Take in Scene depth and / or camera calibration
- Specialized policies — Require very few demonstrations

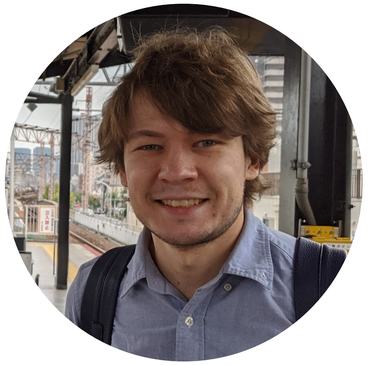
RVT: Robotic View Transformer for 3D Object Manipulation

CoRL 2023

Ankit Goyal

Jie Xu

Yijie Guo



Valts Blukis

Yu-Wei Chao

Dieter Fox

RVT: Robotic View Transformer

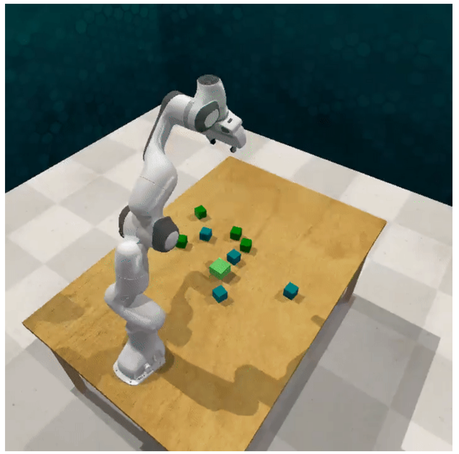
A single model achieves 90% success across tasks with just ~10 demos each.

RVT: Robotic View Transformer for 3D Manipulation

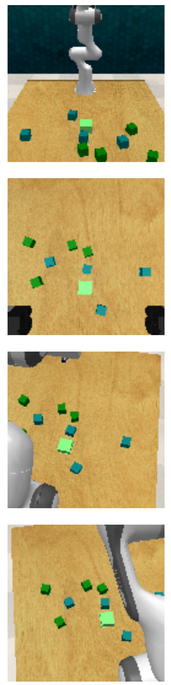
Pipeline Illustration

Instructions

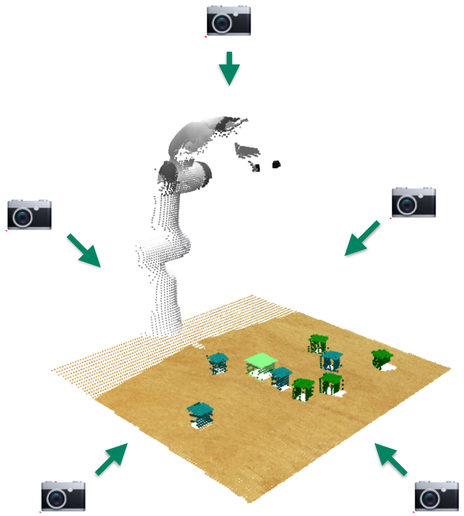
"*Stack the teal blocks*"



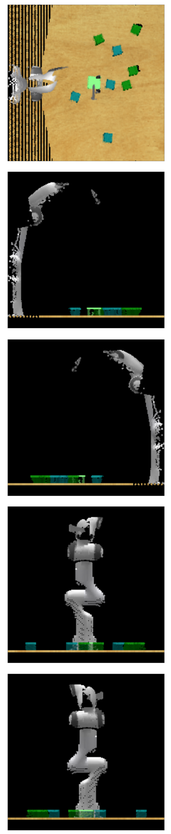
Scene



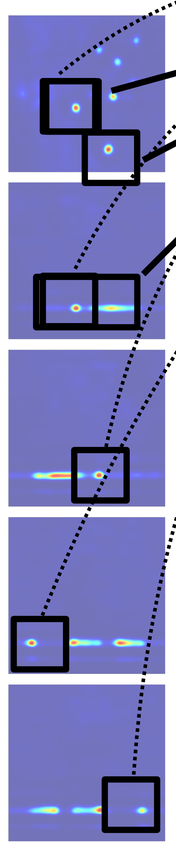
Input RGB-D Images



Reconstructed Point Cloud



Virtual Images



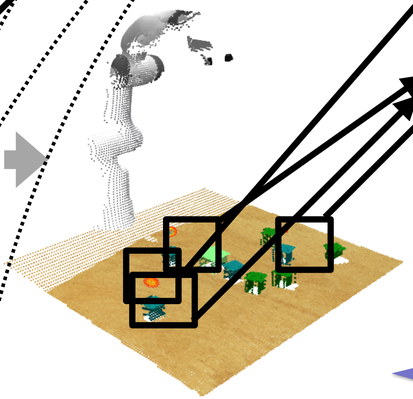
Predicted 2D Heatmap

Multiple modes

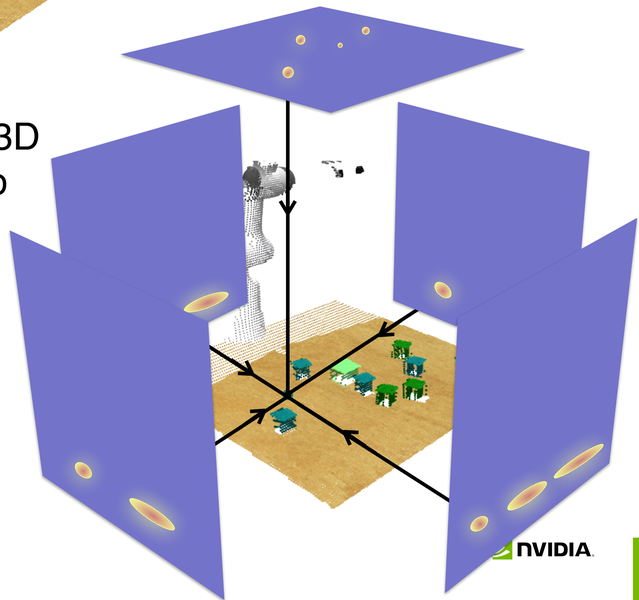
Gripper Rotation and State

Gripper Location

Multiple modes



Predicted 3D Heatmap

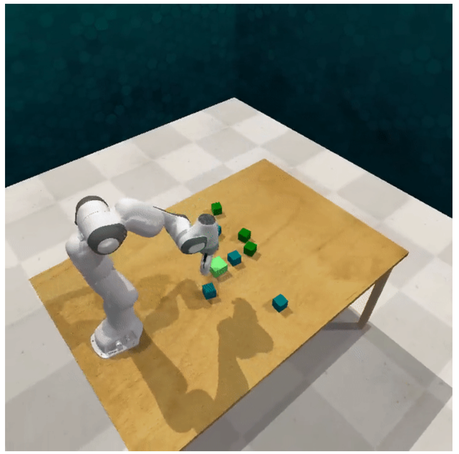


RVT: Robotic View Transformer for 3D Manipulation

Pipeline Illustration

Instructions

"*Stack the teal blocks*"



Scene

Input RGB-D Images

Reconstructed Point Cloud

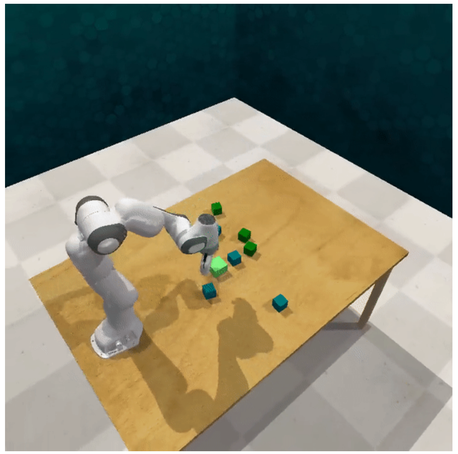
Virtual Images

Predicted 2D Heatmap

Predicted 3D Heatmap

Gripper Rotation and State

Gripper Location

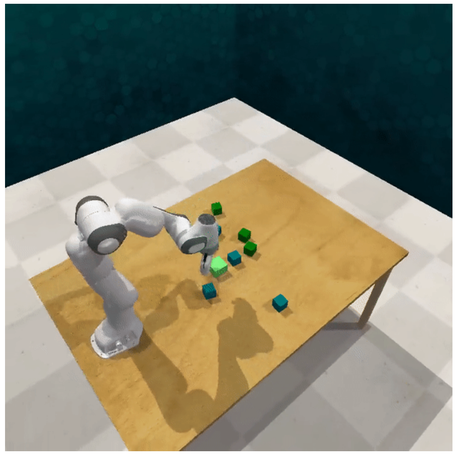


RVT: Robotic View Transformer for 3D Manipulation

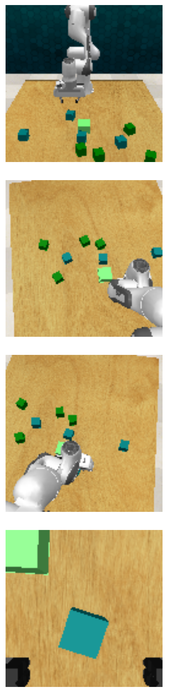
Pipeline Illustration

Instructions

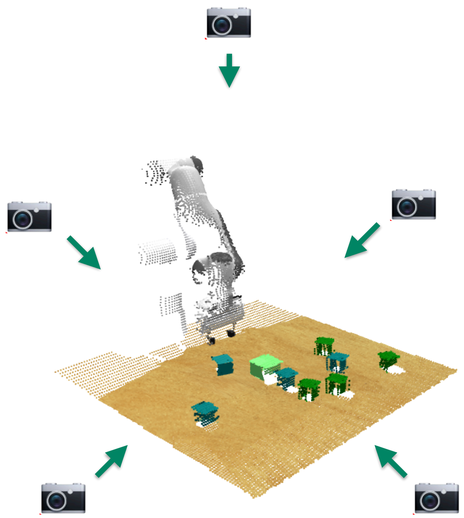
"*Stack the teal blocks*"

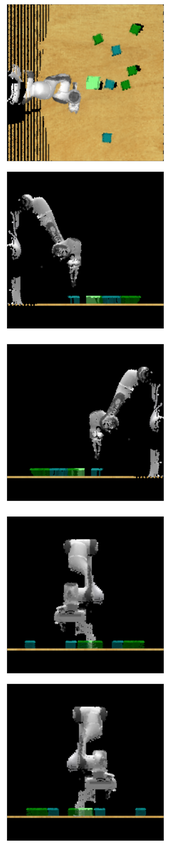


Scene



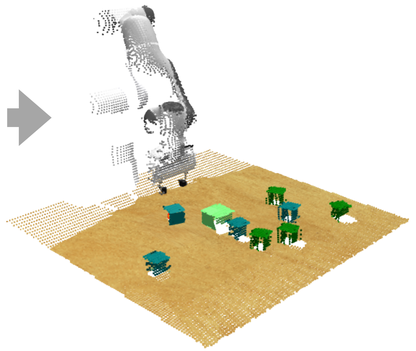
Input RGB-D Images





Virtual Images

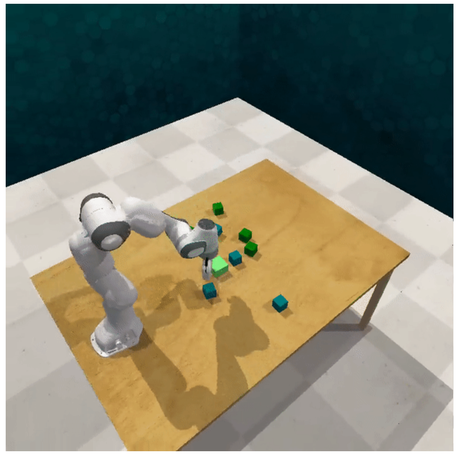
Predicted 2D Heatmap

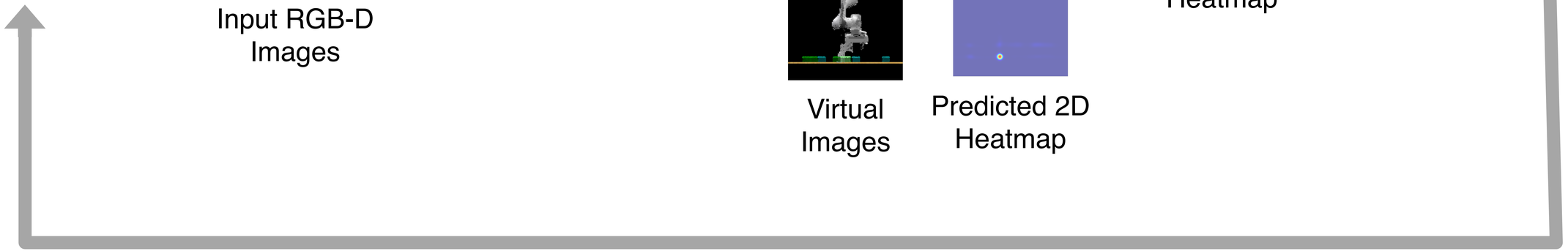


Predicted 3D Heatmap

Gripper Rotation and State

Gripper Location



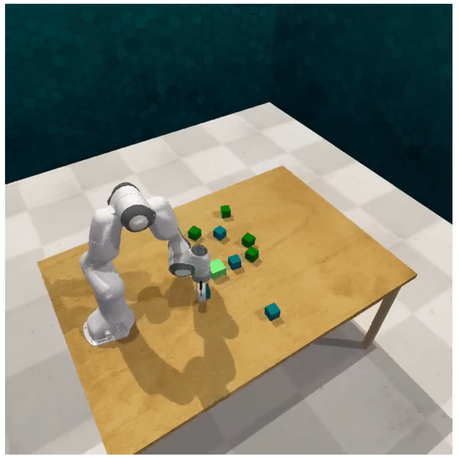


RVT: Robotic View Transformer for 3D Manipulation

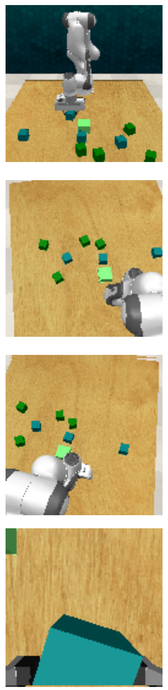
Pipeline Illustration

Instructions

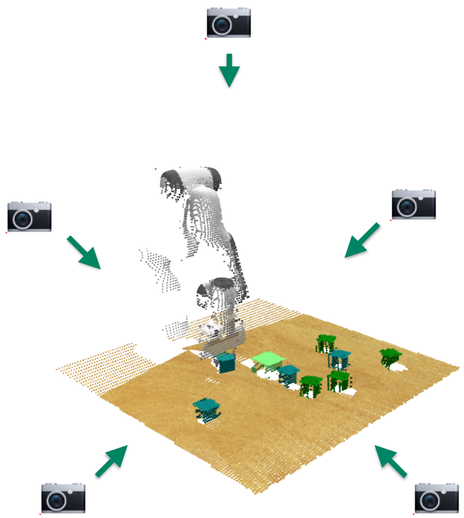
"*Stack the teal blocks*"

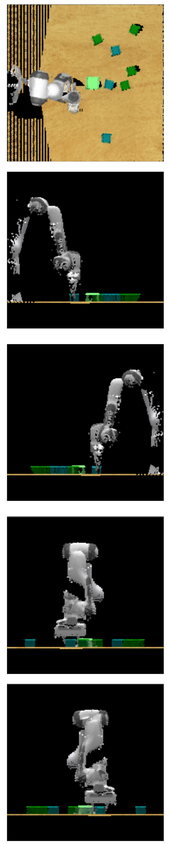


Scene

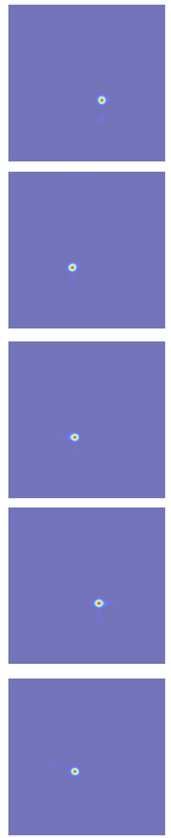


Input RGB-D Images





Virtual Images

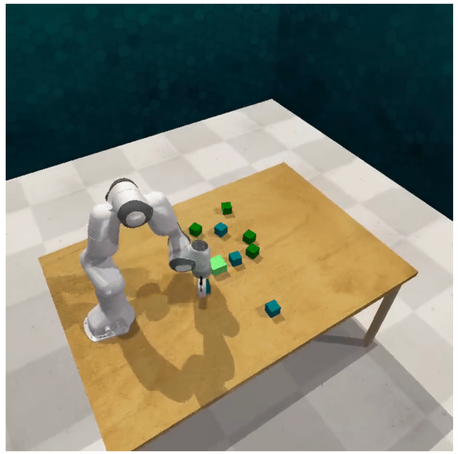


Predicted 2D Heatmap

Predicted 3D Heatmap

Gripper Rotation and State

Gripper Location

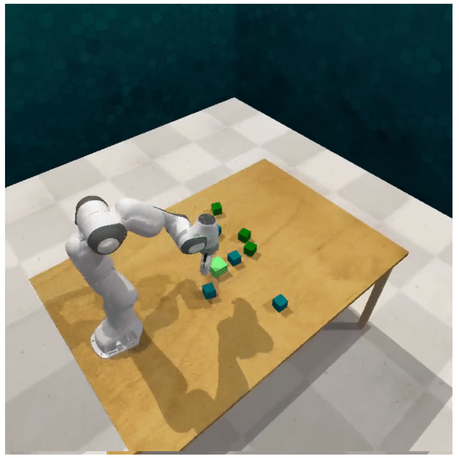


RVT: Robotic View Transformer for 3D Manipulation

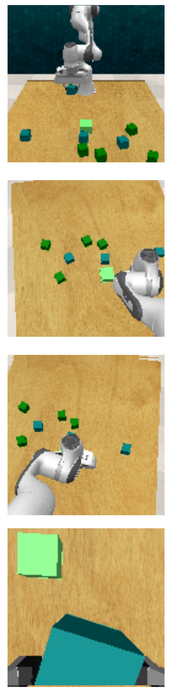
Pipeline Illustration

Instructions

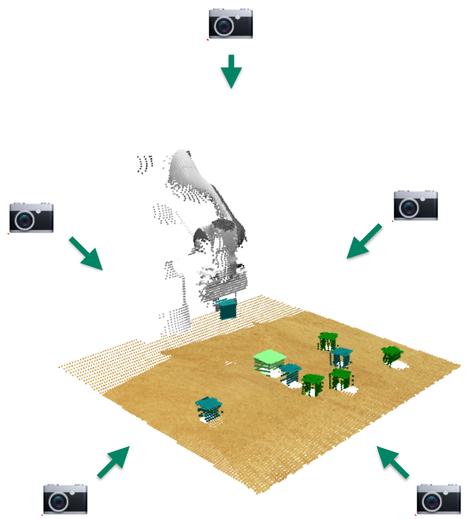
"*Stack the teal blocks*"

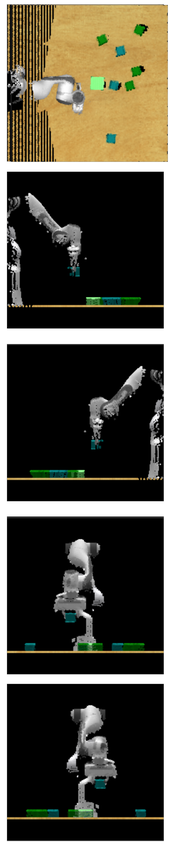


Scene



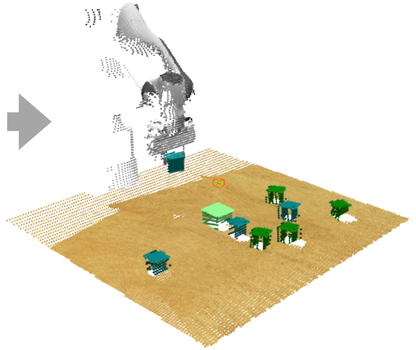
Input RGB-D Images





Virtual Images

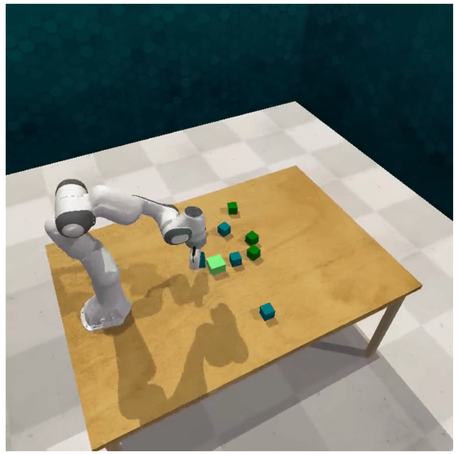
Predicted 2D Heatmap



Predicted 3D Heatmap

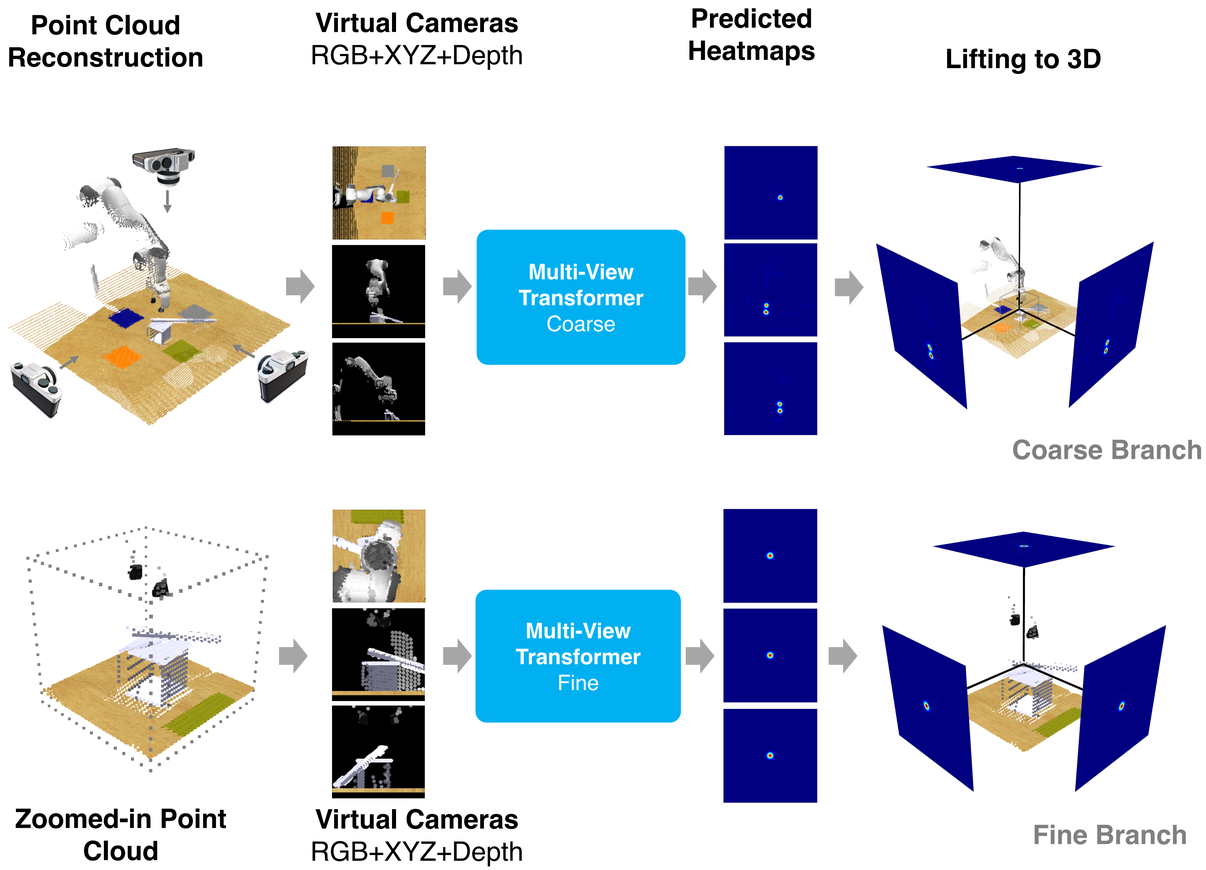
Gripper Rotation and State

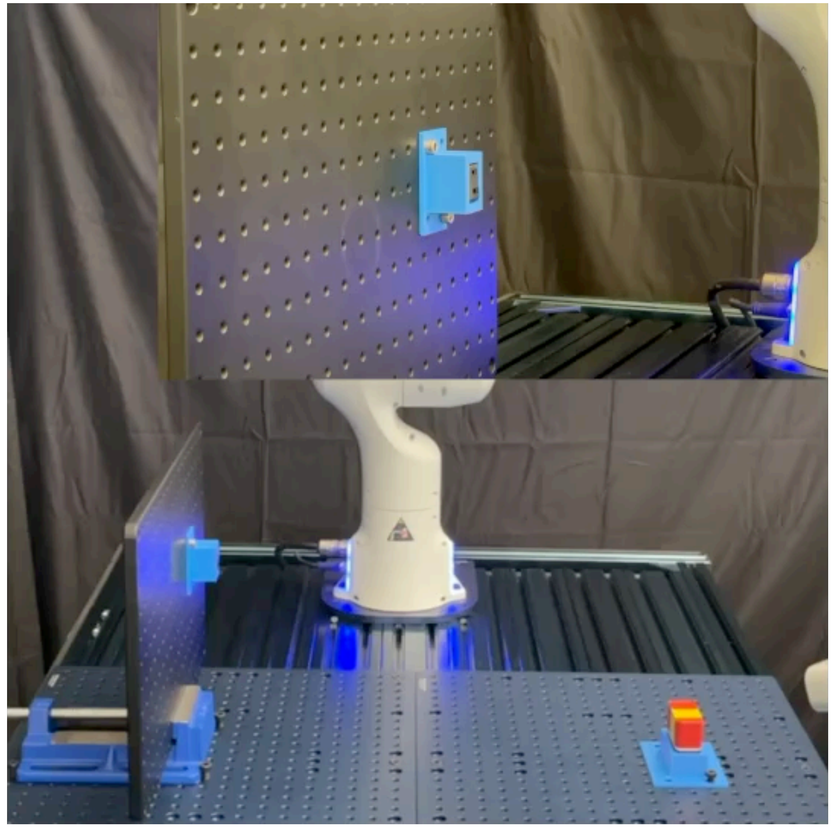
Gripper Location



RVT-2: Learning Precise Manipulation from Few Examples

RSS 2024

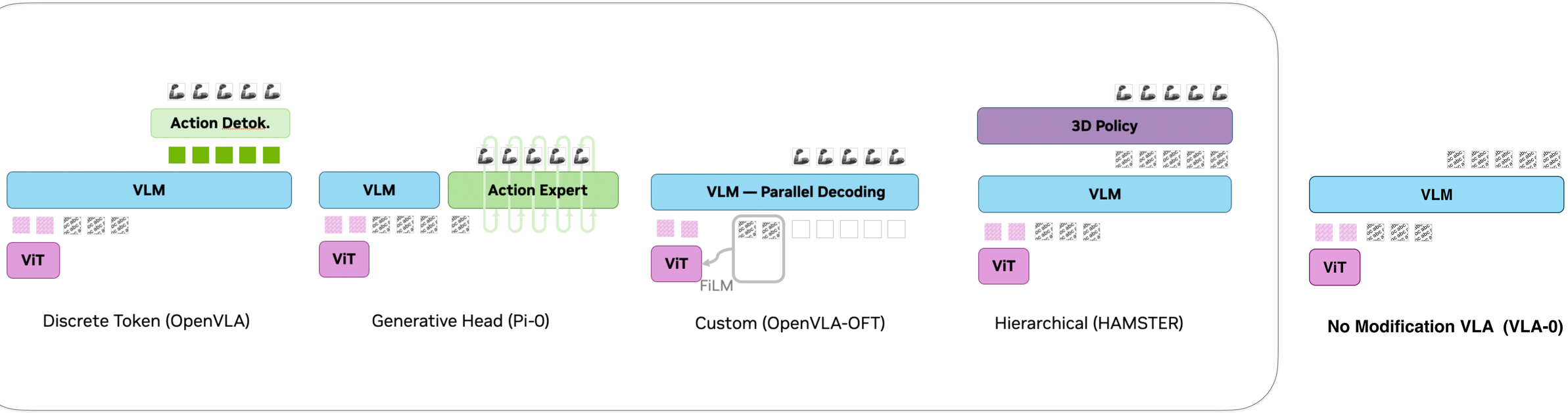




Multiscale design allows for precise manipulation with just 10 demos

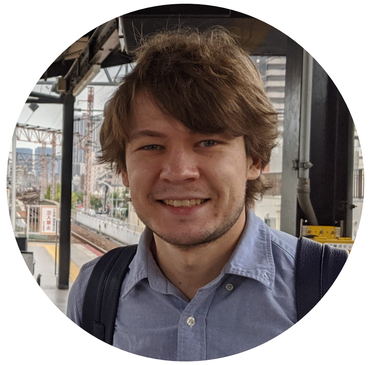
Family of VLAs

- How about the simplest variant?
- Predict Action as Text — No modification



VLA-0: Building State-of-the-Art VLAs with Zero Modification




VLA-0

No Modification to the VLM

- VLA with no modification to the VLM
- Predict Action as Text
- No change to tokenization
- No new architectural component
- Training recipe is important - Check the paper for details

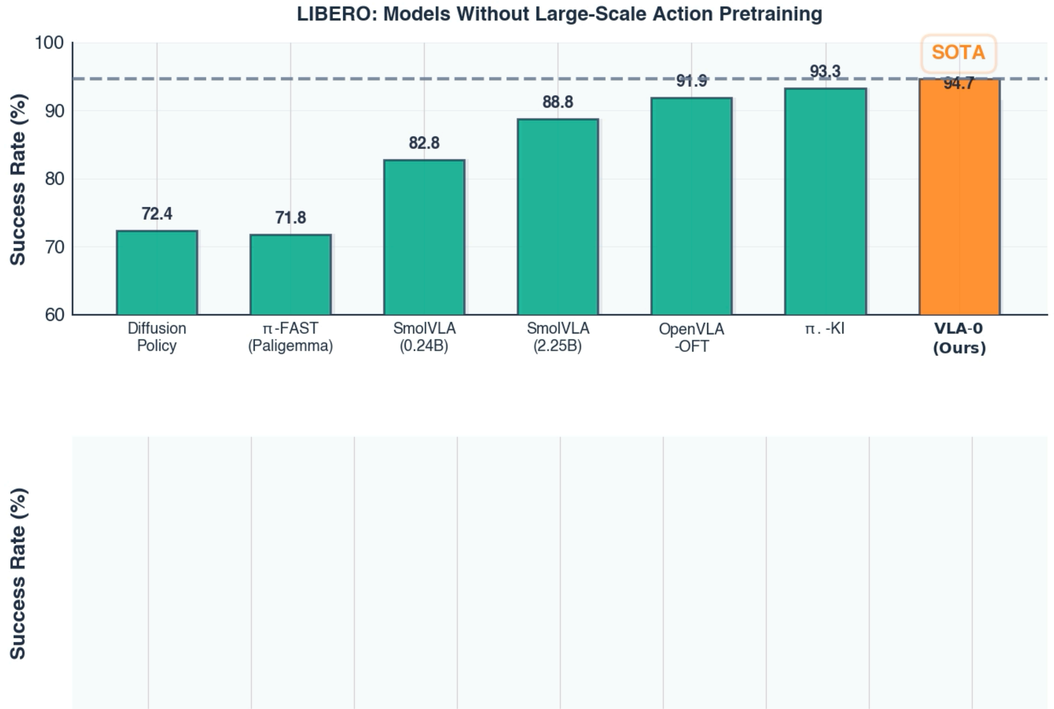
VLA-0

Results on Libero

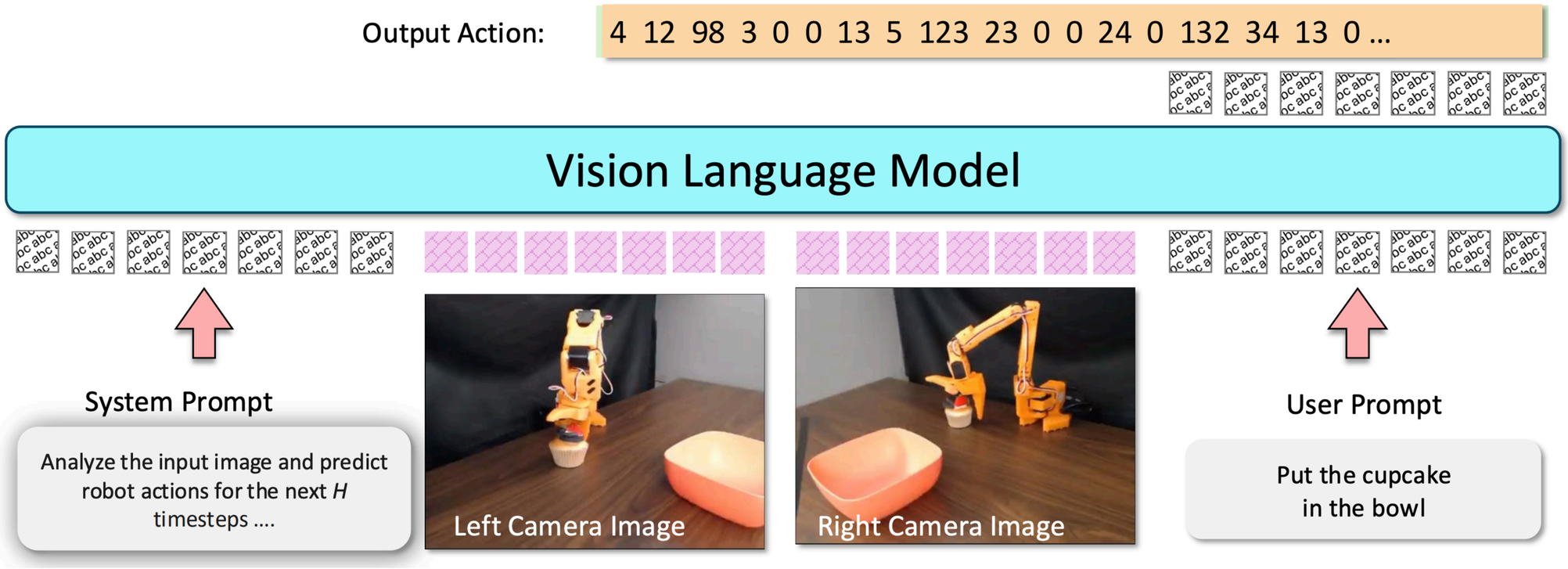
- #1 among all non-pretrained architectures — Outperforming Pi-FAST, Pi-0.5-KI, SmolVLA, OpenVLA-OFT

Even more surprising:

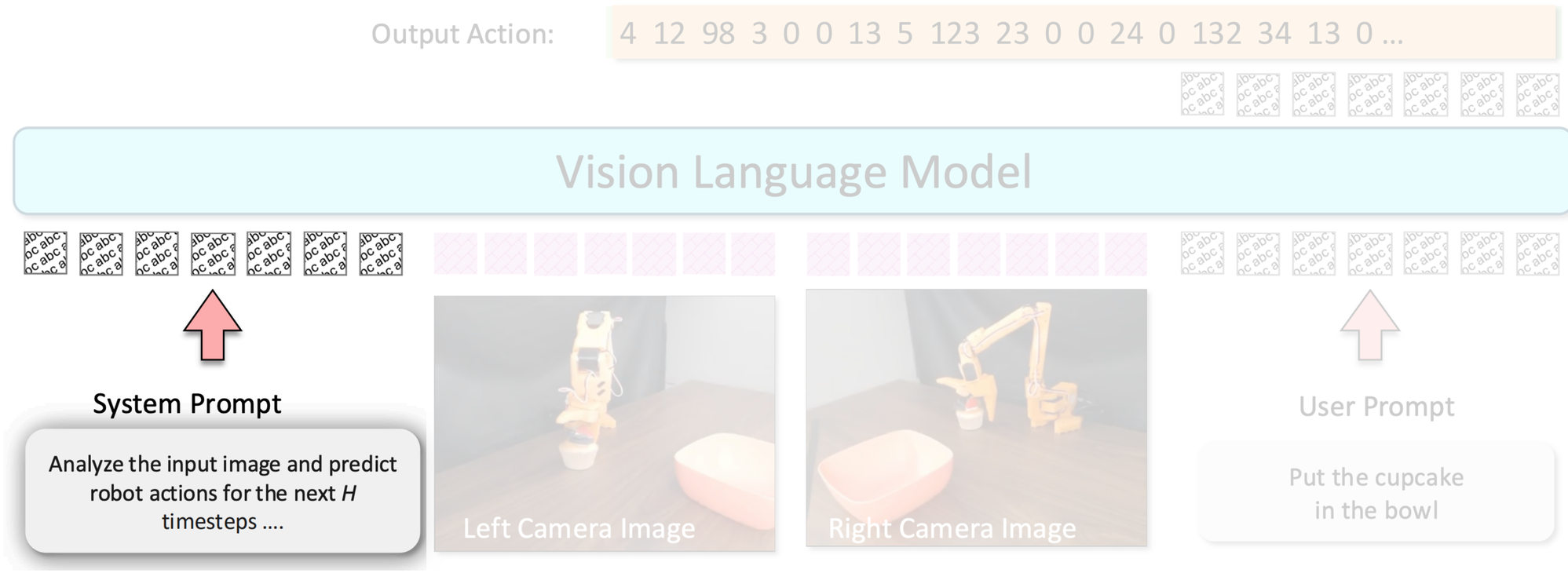
- Without any action pretraining, outperforms leading pretrained models like Pi-0, Pi-0.5-K, GR00T-N1, MomlmoAct



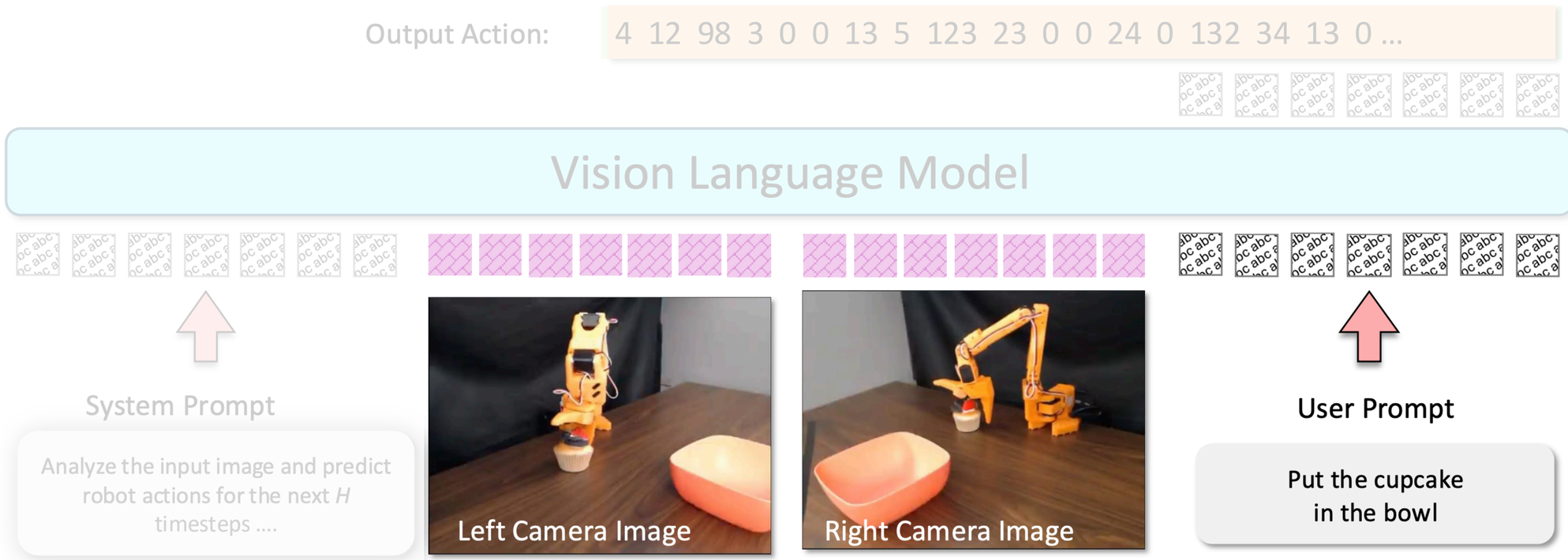
VLA-0


Output Action:
4 12 98 3 0 0 13 5 123 23 0 0 24 0 132 34 13 0 ...
Vision Language Model
System Prompt
Analyze the input image and predict robot actions for the next H timesteps ....
Left Camera Image
Right Camera Image
User Prompt
Put the cupcake in the bowl

VLA-0

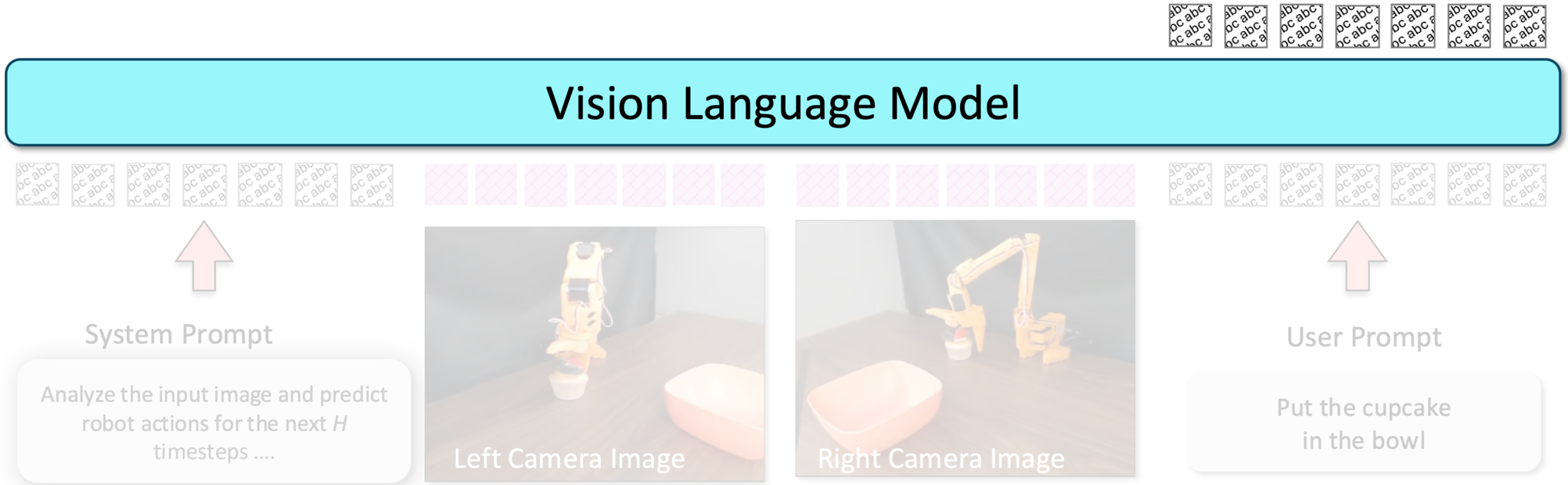

Output Action:
4 12 98 3 0 0 13 5 123 23 0 0 24 0 132 34 13 0 ...
Vision Language Model
System Prompt
Analyze the input image and predict robot actions for the next H timesteps ....
Left Camera Image
Right Camera Image
User Prompt
Put the cupcake in the bowl

VLA-0


Output Action:
4 12 98 3 0 0 13 5 123 23 0 0 24 0 132 34 13 0 ...
Vision Language Model
System Prompt
Analyze the input image and predict robot actions for the next H timesteps ....
Left Camera Image
Right Camera Image
User Prompt
Put the cupcake in the bowl

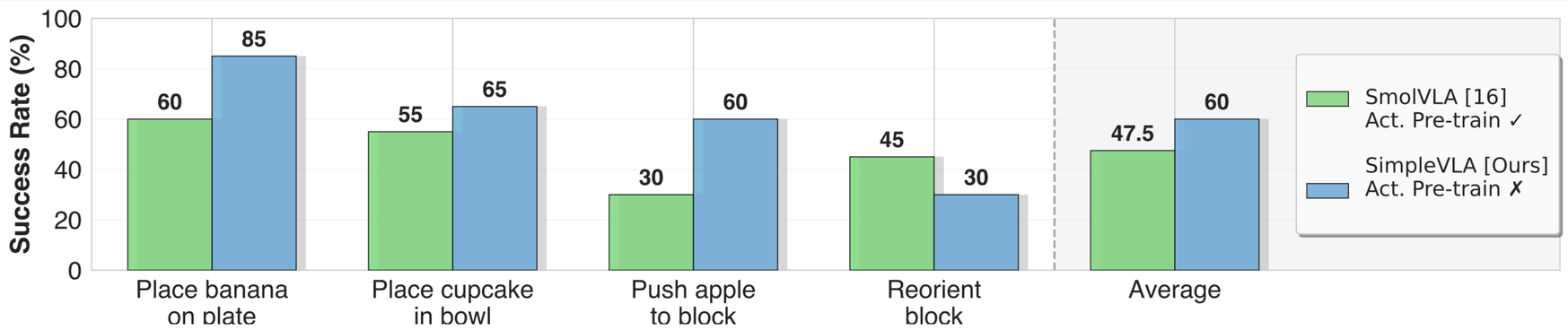
VLA-0

Output Action: 4 12 98 3 0 0 13 5 123 23 0 0 24 0 132 34 13 0 …



VLA-0

Results in Real



- On real world data, outperforms SmolVLA
- SmolVLA — Pretrained on large-scale SO-100 data + Finetuned on 100 demos per task
- VLA-0 trained from scratch with 100 demos per task

VLA-0

vla0.github.io

VLA-0: Building State-of-the-Art VLAs with Zero Modification

Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, Fabio Ramos
NVIDIA

***Abstract*— Vision-Language-Action models (VLAs) hold immense promise for enabling generalist robot manipulation. However, the best way to build them remains an open question. Current approaches often add complexity, such as modifying the existing vocabulary of a Vision-Language Model (VLM) with action tokens or introducing special action heads. Curiously, the simplest strategy of representing actions directly as text has remained largely unexplored. This work introduces VLA-0 to investigate this idea. We find that VLA-0 is not only effective; it is surprisingly powerful. With the right design, VLA-0 outperforms more involved models. On LIBERO, a popular benchmark for evaluating VLAs, VLA-0 outperforms all existing methods trained on the same robotic data, including $\pi_0.5$-KI, OpenVLA-OFT and SmolVLA. Furthermore, without large-scale robotics-specific training, it outperforms methods trained on large-scale robotic data, like $\pi_0.5$-KI, π_0, GR00T-N1 and MolmoAct. These findings also translate to the real world, where VLA-0 outperforms SmolVLA, a VLA model pre-trained on large-scale real data. This paper summarizes our unexpected findings and spells out the specific techniques required to unlock the high performance of this simple yet potent VLA design. Visual results, code, and trained models are provided at: https://vla0.github.io/.**

I. Introduction

Following the success of Large Language Models (LLMs) in text processing and Vision-Language Models (VLMs) in handling both visual and textual inputs, a natural next step is to explore Vision-Language-Action models (VLAs), i.e. systems that not only understand visual and textual information, but also predict actions for robotic agents. VLAs are typically built by modifying a base VLM to predict actions. However, it is still unclear what the 'correct' way to do this is, if there is one at all. Recent research has taken various approaches, which we broadly categorize into three families, as shown in Figure 2: (1) Discrete Token VLAs, (2) Generative Action Head VLAs, and (3) Custom Architecture VLAs.

Discrete Token VLAs. It is one of the initial strategies popularized by models such as RT-2 [24] and OpenVLA [11]. Robot actions, originally continuous, are discretized into bins; each bin is then assigned a token from the VLM vocabulary, using either new or infrequent tokens. The model is then trained to predict these action tokens using the same cross-entropy loss as used to train the base VLM. Although straightforward, this approach has two main limitations: (i) it restricts the resolution of the action space, since fine-grained control can require thousands of bins, which conflicts with sharing the text vocabulary; and (ii) it compromises the pretrained language understanding of the VLM by repurposing its vocabulary for actions. Given these limitations, such VLAs do not perform as well as other alternatives. (see Tab. I)

Generative Action Head VLAs. Another common strategy is to attach an action generation head on top of the VLM, as done by methods like π_0 [2] or SmolVLA [19]. The VLM is fine-tuned to predict a latent vector, which is then decoded into actions using a generative model such as a diffusion process or flow matching. While this method improves action fidelity, it also introduces a new neural network that needs to be finetuned. This often leads to a decline in the language understanding and grounding capabilities of the underlying VLM [9], and introducing a non-pretrained action head may compromise generalization of the overall system.

Custom Architecture VLAs. Beyond the above categories, some methods propose architectural modifications or custom tokenizers tailored to action prediction. For instance, OpenVLA-OFT [10] introduces a specialized ACT head. Another example is π-FAST [16] that create a special tokenization scheme for actions using discrete cosine transform (DCT). π-FAST can also be considered a discrete token VLA, but for the purposes of this work, we classify them as custom VLA as it involves a custom tokenization scheme. While these custom methods are effective, they typically involve significant architectural changes, additional param-

Fig. 1: **Schematic representation of VLA-0.** VLA-0 converts a VLM into a VLA by prompting the VLM to predict action as text. This strategy is surprisingly effective and achieves state-of-the-art results akin to alternatives.

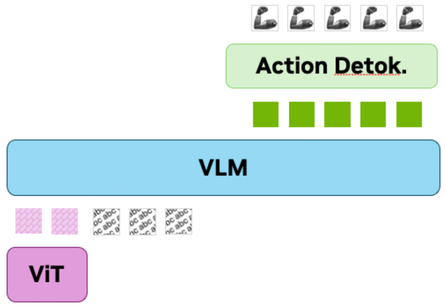
SCAN ME

SCAN ME

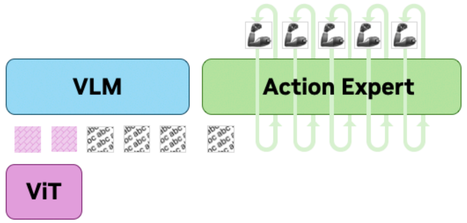
Summary

Thank you! Questions?

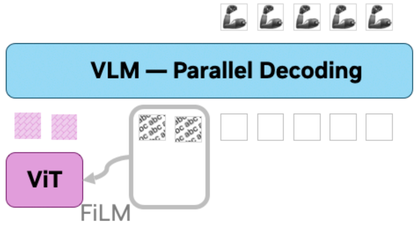
- Hierarchical VLAs — Combine the strength of VLMs and 3D Policies
- 3D Policies — Efficient task specific learners
- Among monolithic VLAs — Simplest design (VLA-0) is surprisingly effective



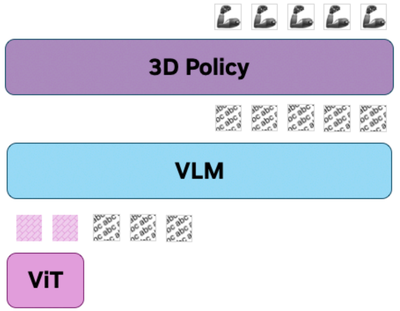
Discrete Token (OpenVLA)



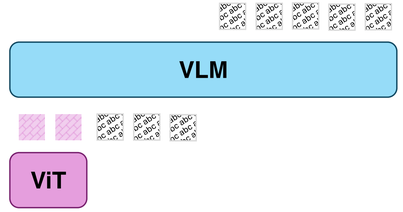
Generative Head (Pi-0)



Custom (OpenVLA-OFT)



Hierarchical (HAMSTER)



No Modification VLA (VLA-0)