Large AI models for robotics

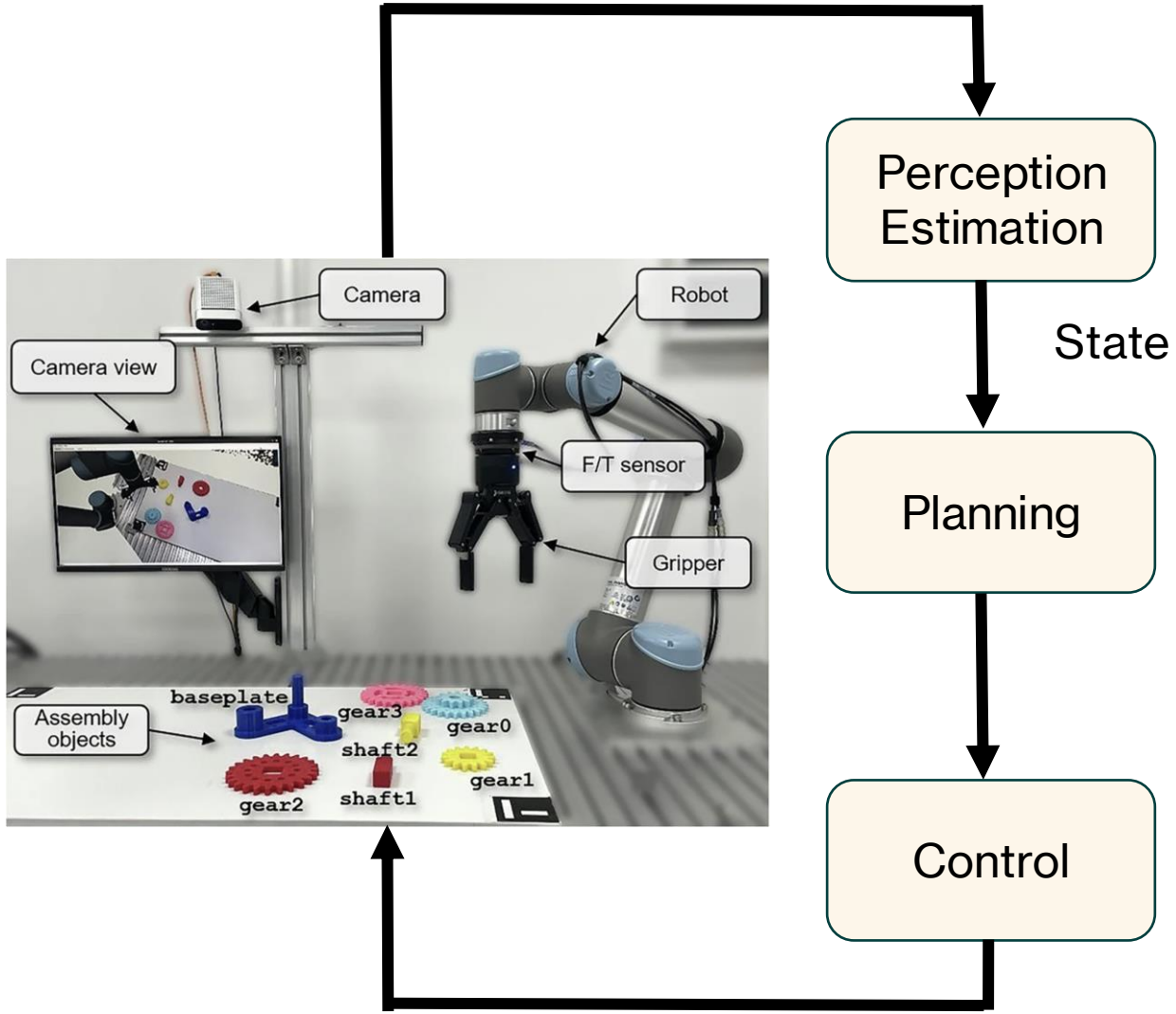
David Filliat

AMIAD

April 1st 2026

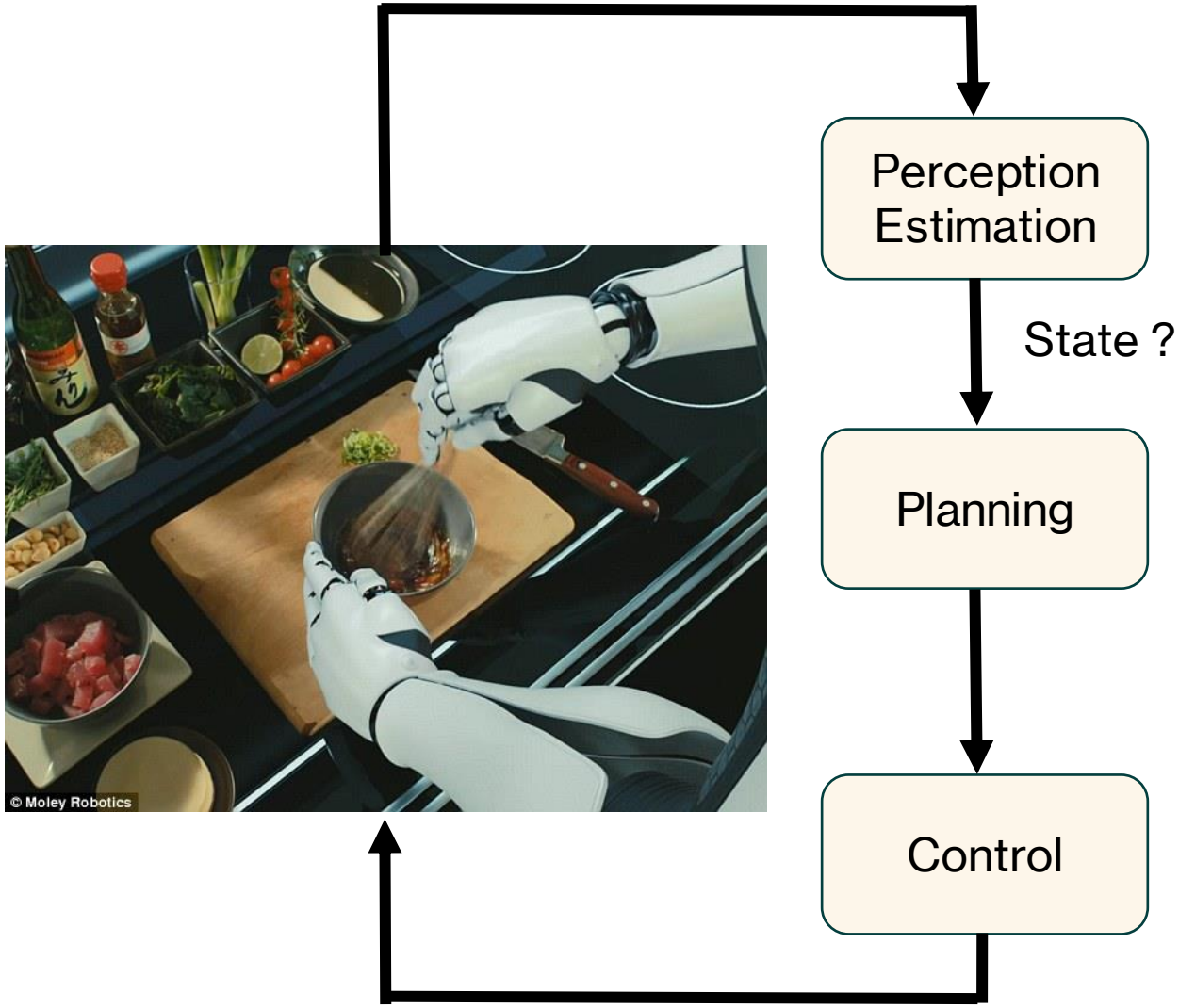
« Classical » robotics pipeline

- Classical systems built from modular pipelines
- Perception, planning, and control
- Mostly based on **states an models of the state evolution / observation** *defined/estimated using human knowledge*
- Need to adapt the environment difficulty to the module's capabilities
- Works well in controlled industrial settings for example



Limitations of « Classical » robotics pipeline

- Heavy engineering effort
- Faces difficulties when environment become more complex
- In particular information required for decision/control may be hard to integrate in the state representation
- Lack adaptability when non-specialist need to adapt robot behavior



Using AI/machine learning in robotics

With the modular approach

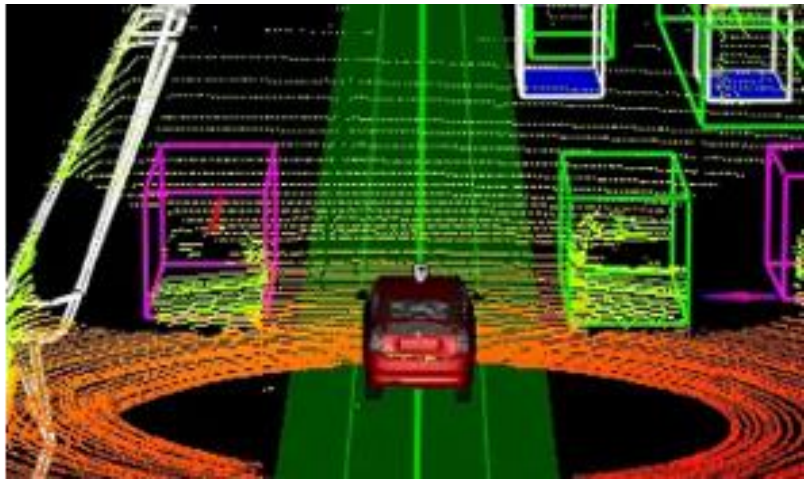
- Used a lot in perception (image, LiDAR…)
- Can contribute to planning, control (prediction, heuristics, RL …)
- Modularity may prevent global optimisation
- Still need to define interfaces between modules
- Still very task specific

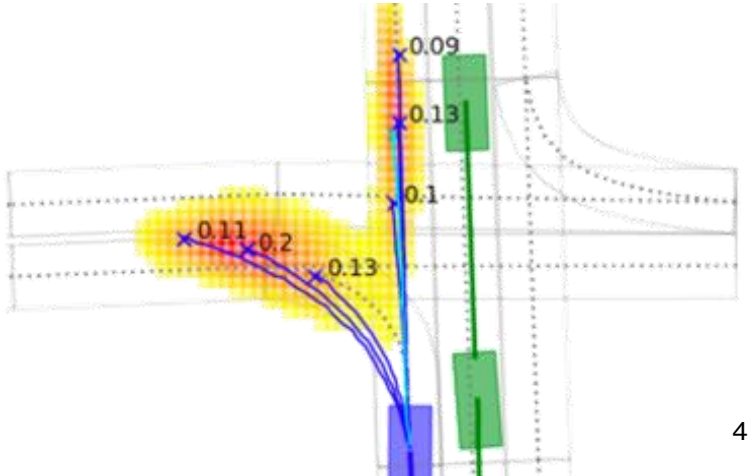
Using large foundation models ?

Training end-to-end models ?

Training large end-to-end models ?

More general approach : **Vision-Language-Action** models





Why Large AI Models?

- Offer general capabilities to robotics
  - Language understanding learned from massive datasets
  - Robustness to objects/environment not seen during (robotic) training
- Hope to drastically improve generalization between
  - Tasks
  - Embodiments (robots, sensor setup)
  - Environment conditions
- Foundation model paradigm
  - Model trained on large datasets and compute
  - Transformer architectures

Overview

- Large AI model basics
- Using Large Models in Robotics Systems
- Vision-Language-Action models
- Large models for navigation
- Reinforcement learning
- State representation learning
- Transfering simulation-based RL to reality

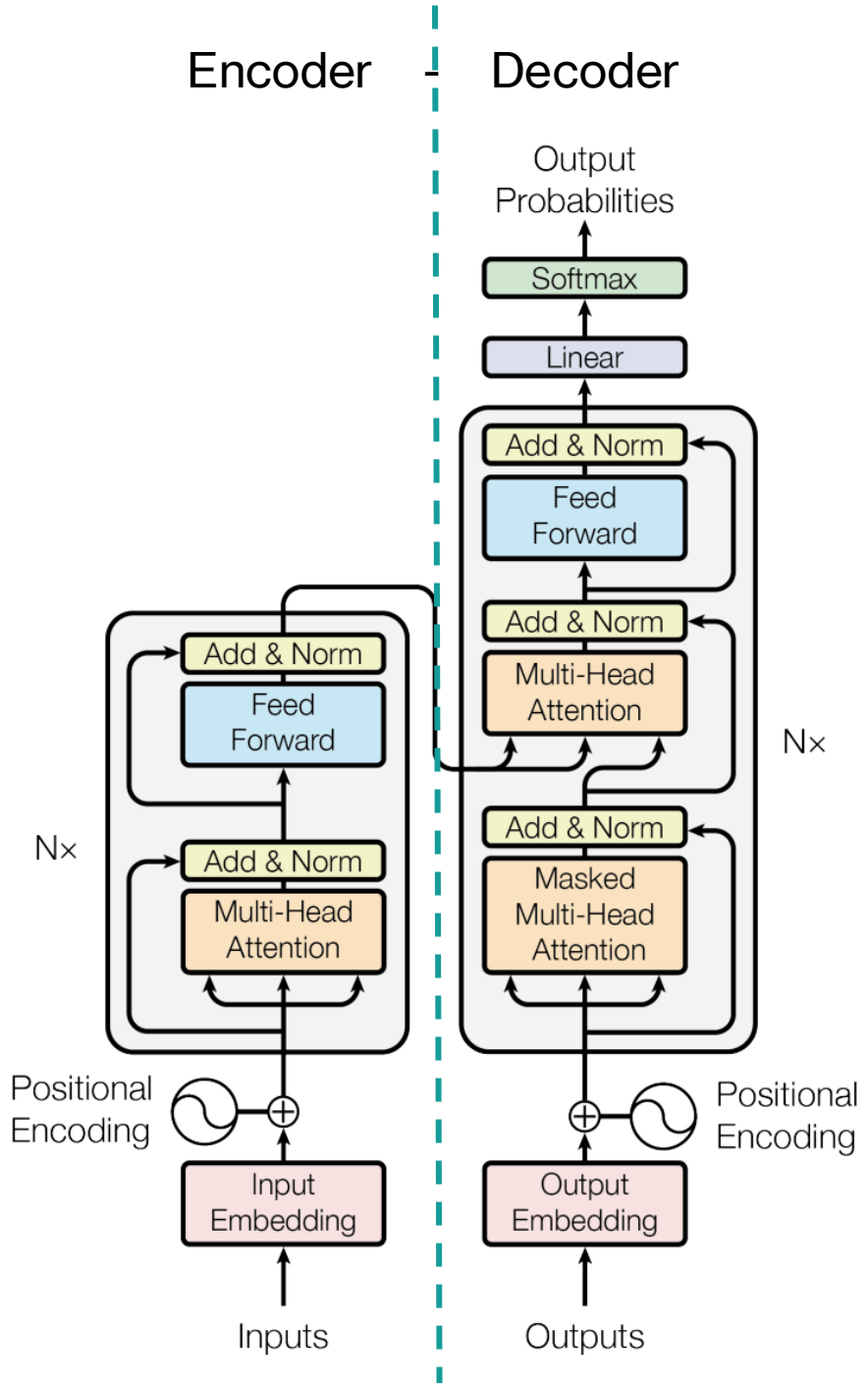
LARGE AI MODELS BASICS

Foundation models

- Train large models on massive datasets
- Mostly train without supervision (self-supervised, unsupervised)
- Learn general representations, "common sense"
- Adapt models to many downstream tasks with limited data
- Mostly using variants of the transformer architecture
  - Can be applied to many data after "tokenizing"
  - Can be trained efficiently with large datasets

Transformer architecture

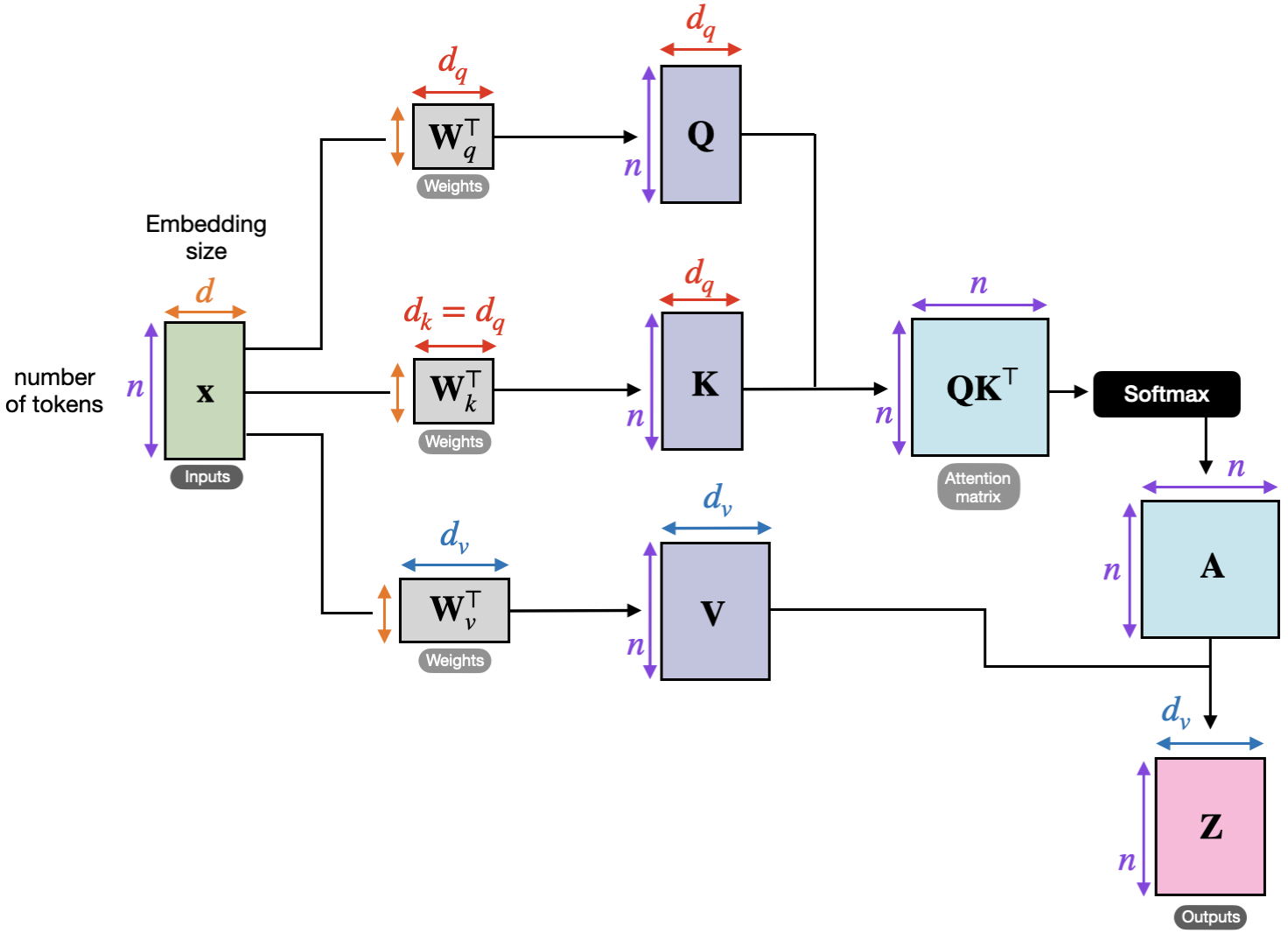
- A very general architecture to transform input sequences using information from all the sequence
- Not restricted to neighborhood (like CNN) or previous time step (like RNN) -> long range dependencies
- Inputs are encoded as **tokens** (discrete elements, represented by real valued vectors)
- Variants
  - Encoder : dependence on the full sequence
  - Decoder : dependence on the past only



Multi-head self-attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{in}}}\right)V$$

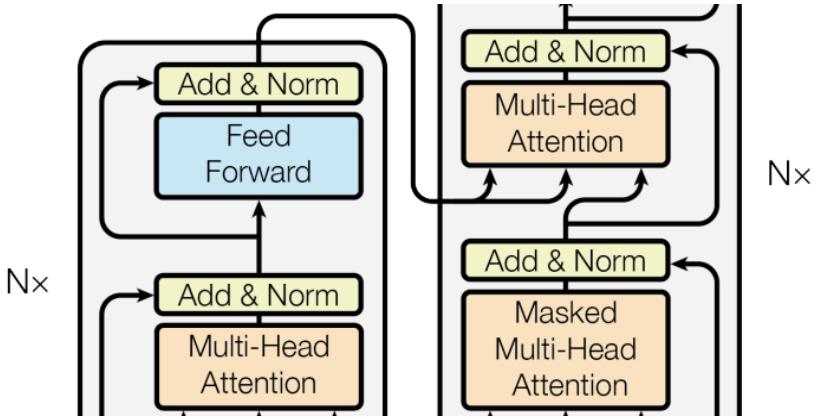
- Tokens are converted to Query / Keys / Values
- Values are summed, weighted by similarities between Keys and Queries
- This is done multiple times (head) and merged in the same size as input in the end
- Implement dependencies between any tokens
- Main computational bottleneck in transformer scaling



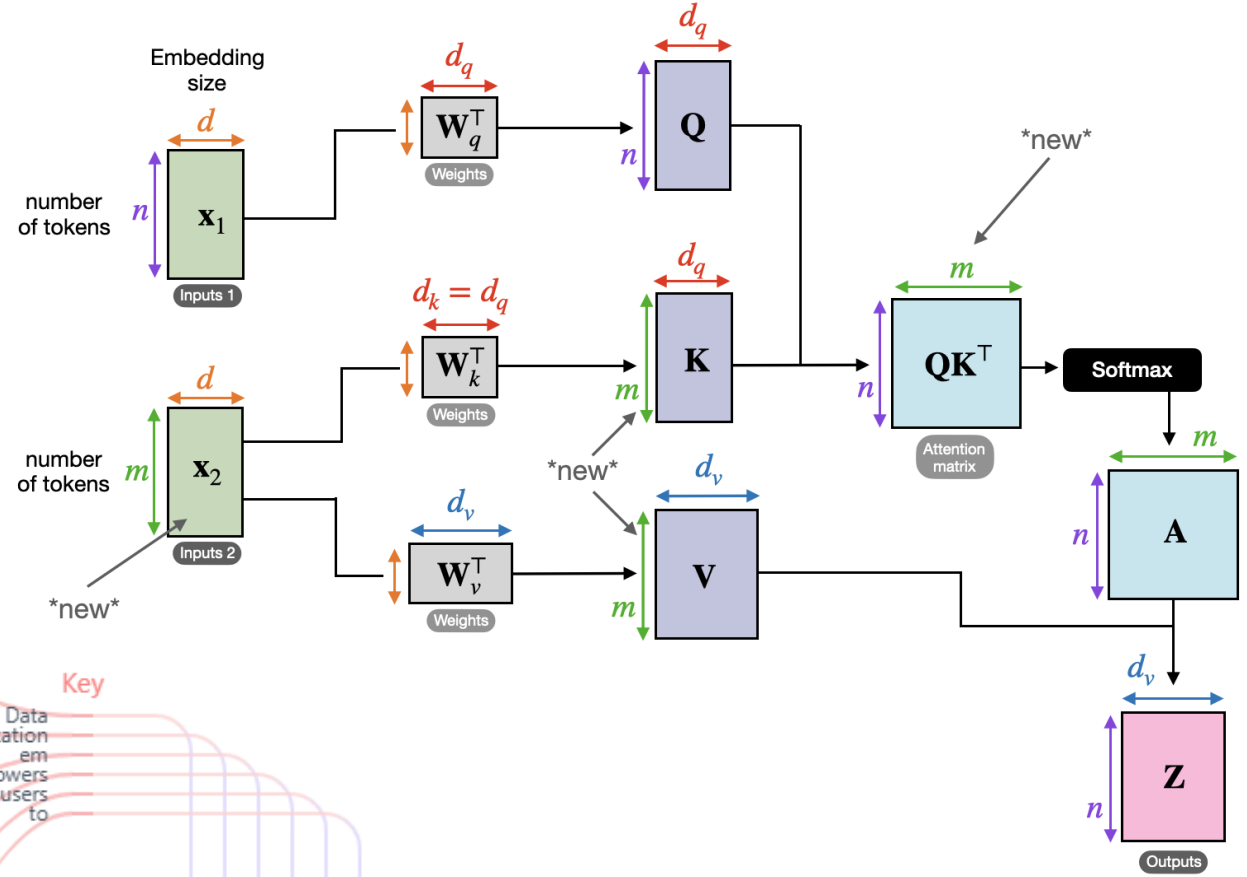
https://sebastianraschka.com/images/blog/2023/self-attention-from-scratch/cross-attention-summary.png

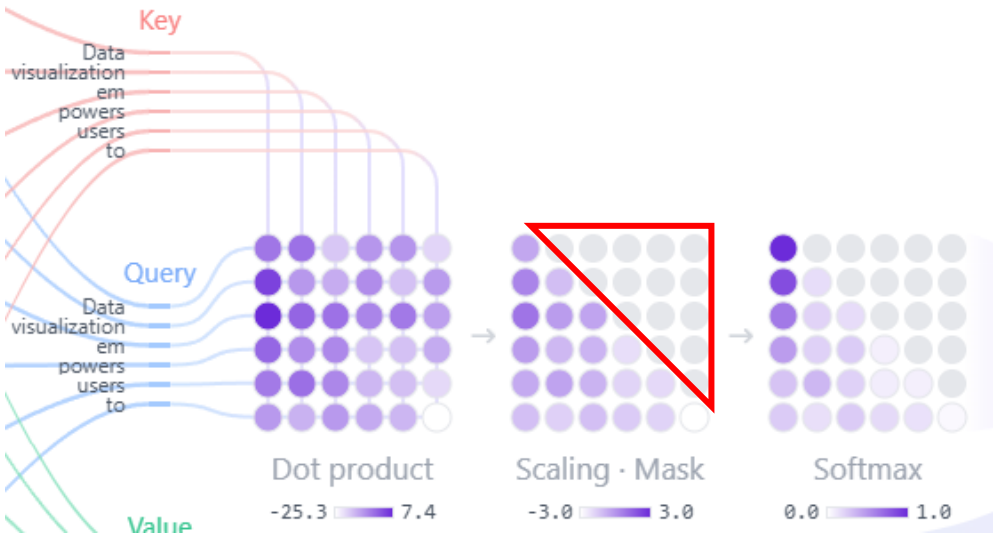
Multi-head cross-attention

- Attention between encoder and decoder
- KV comes from encoder, Q from decoder



- Mask to forbid dependence on the future

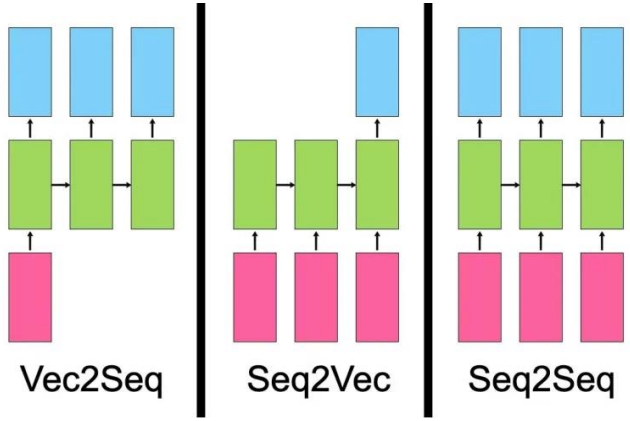




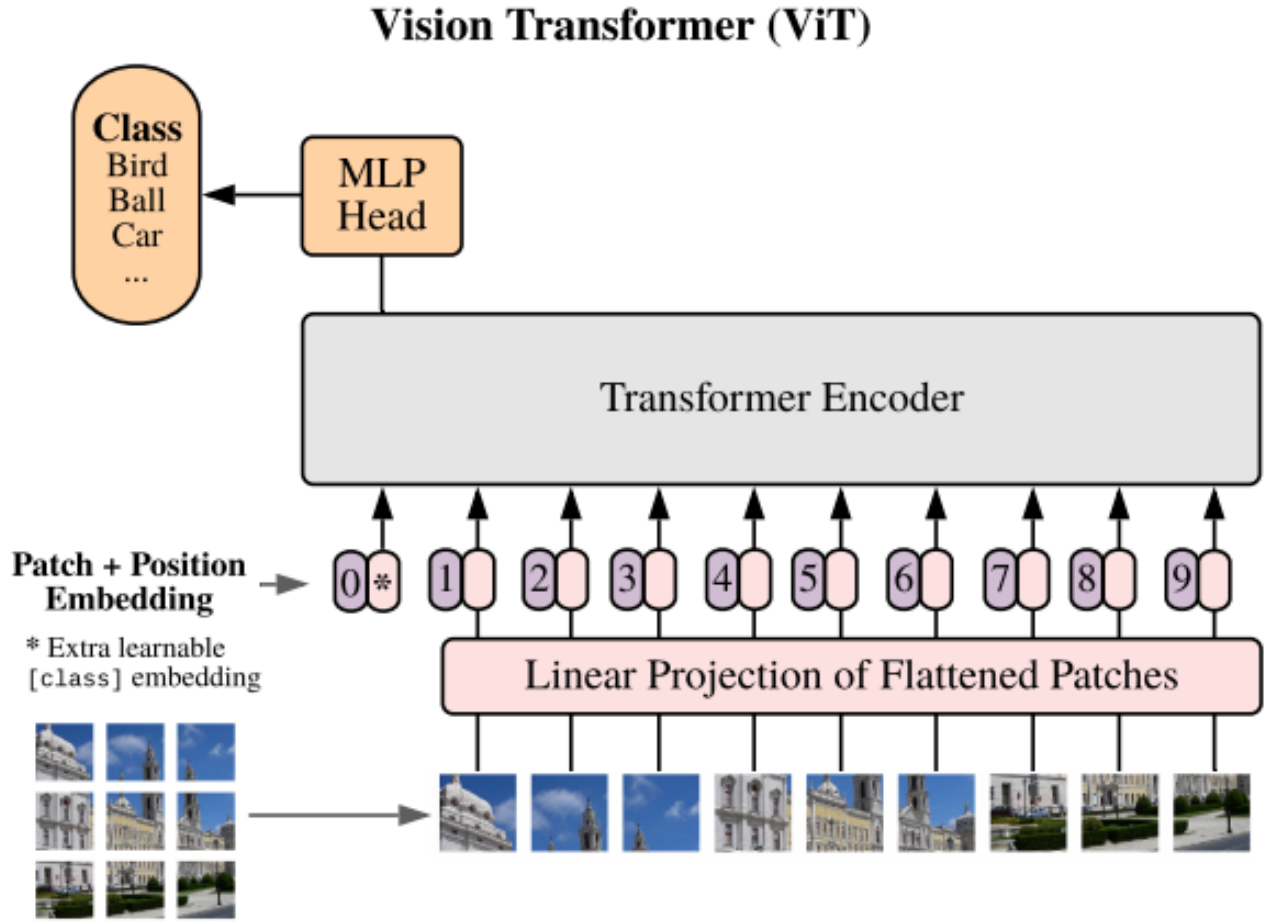
https://poloclub.github.io/transformer-explainer/

Transformer applications

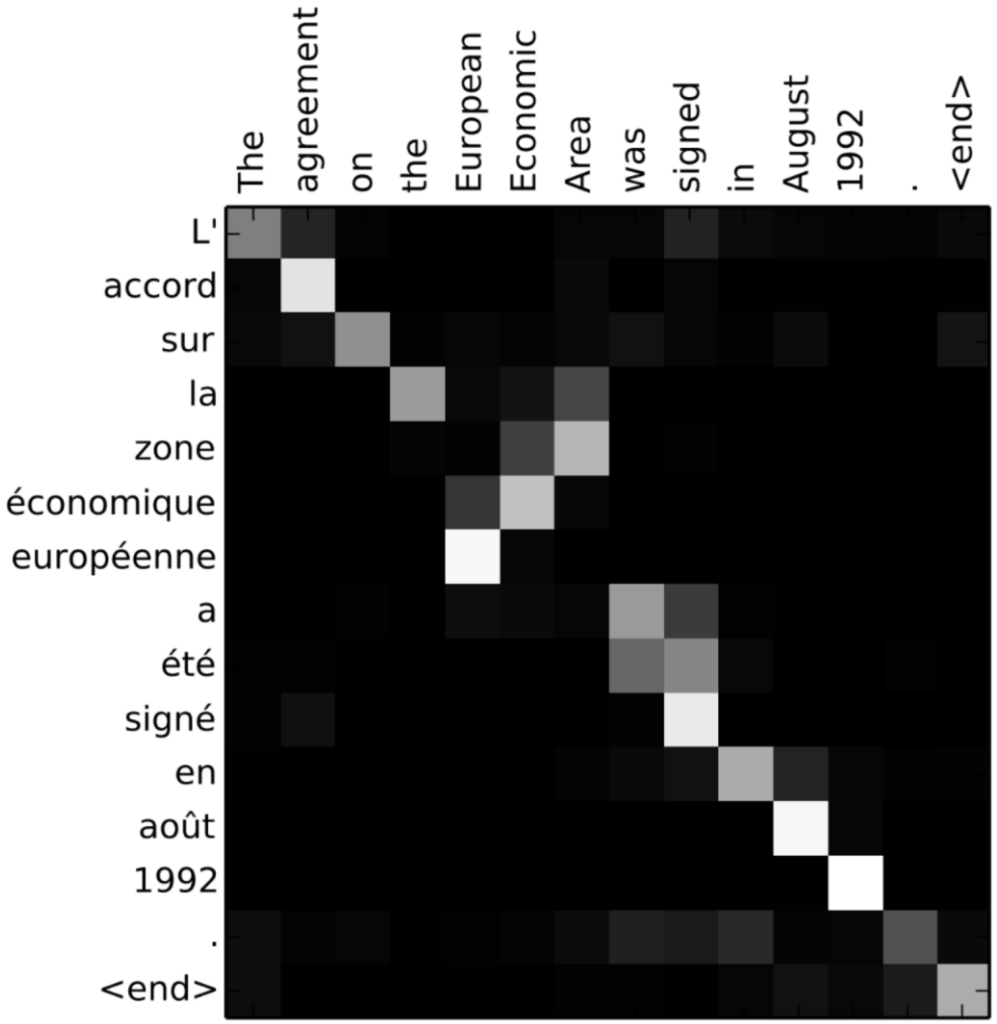
- Originally applied to text
  - Classification (Seq2Vec)
  - Traduction, answering (Seq2Seq)
  - Generation (Vec2Seq)



- Can be applied to image by splitting images in patches

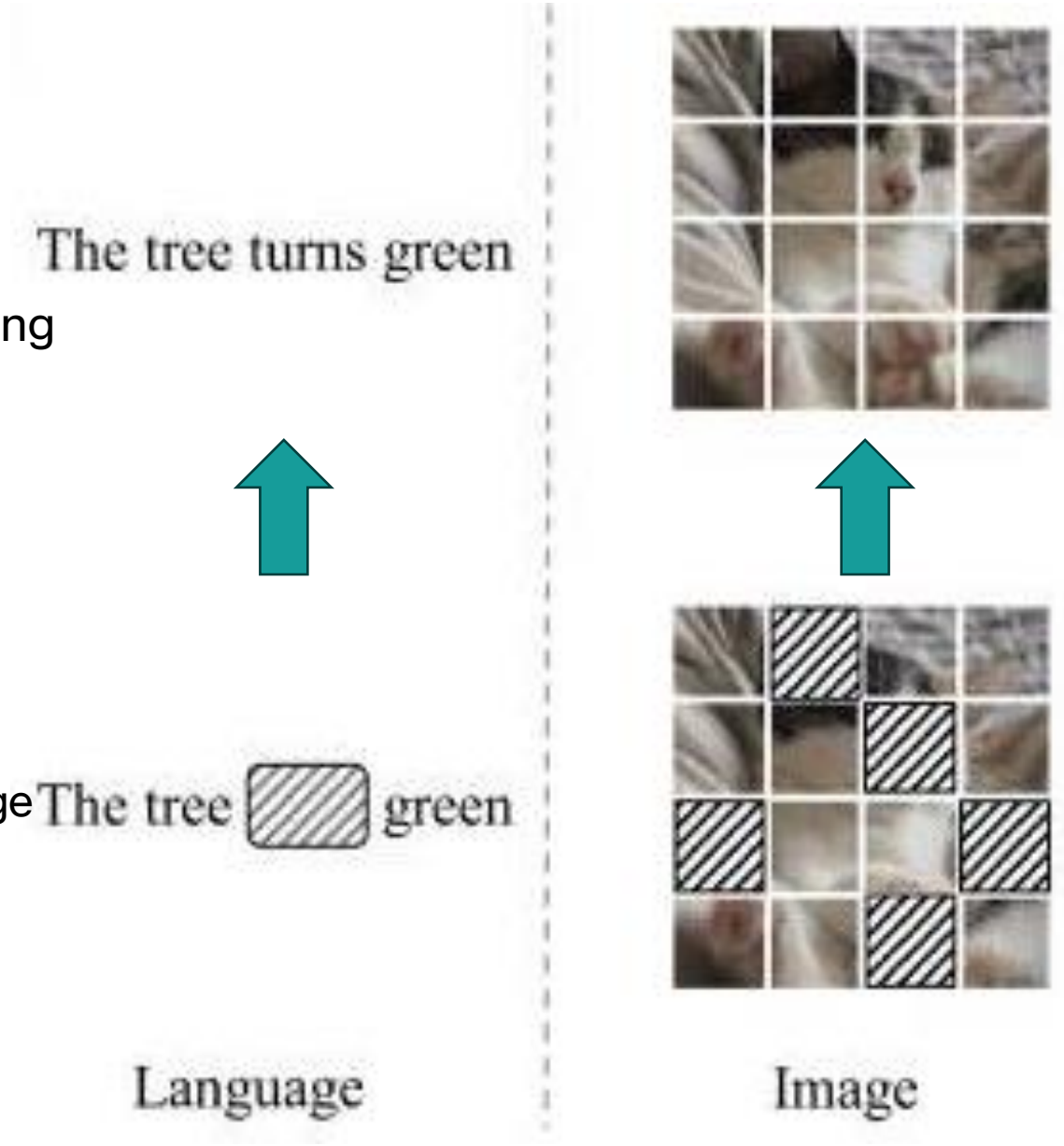


Transformer attention visualization



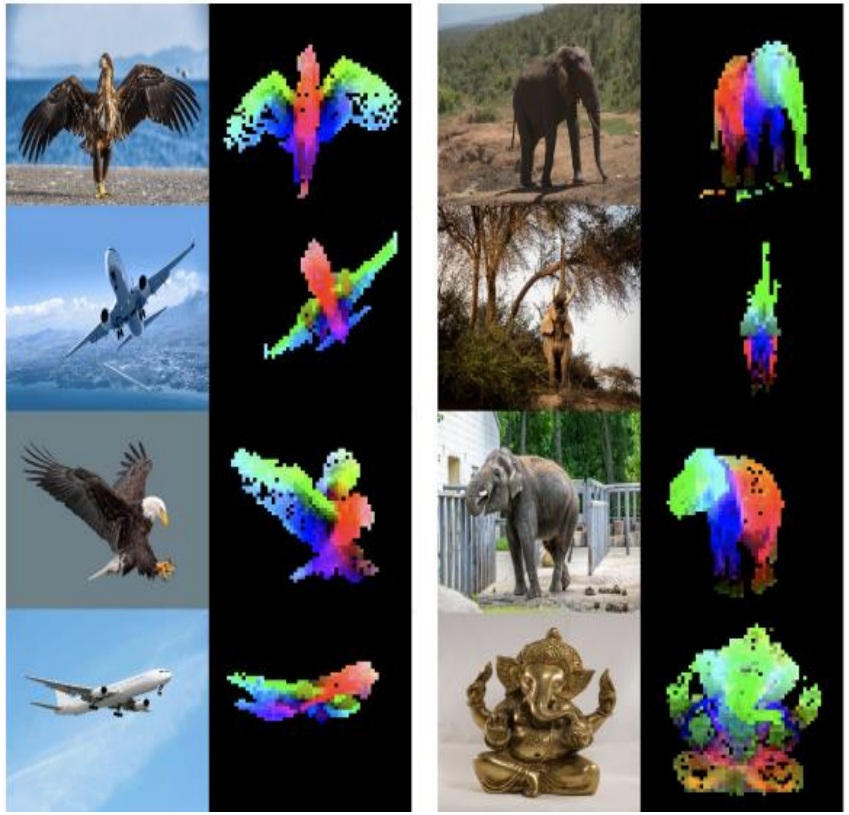
Unsupervised training

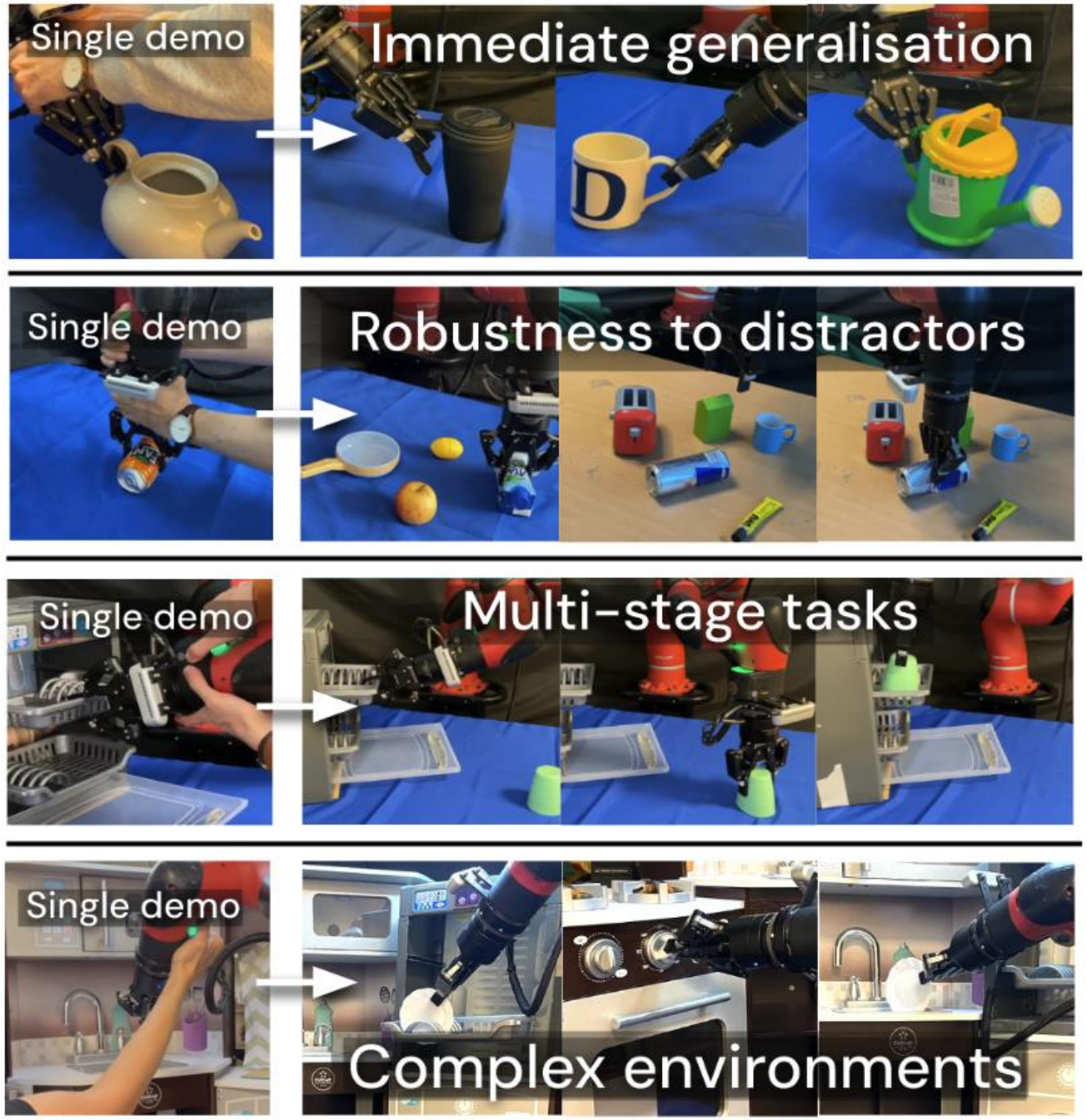
- Models can/should be trained without human labeling
- Learns good features that can be used for many tasks
- Various approaches including
  - Next token prediction
  - Masked auto-encoder
  - Predicting similar features for two distortion of an image
- Applied to text
  - BERT, GPT, …
- Applied to images
  - DINOv2, ViT…



Unsupervised training

- Ex : DINOv2 features
  - Robust correspondences between instances/poses

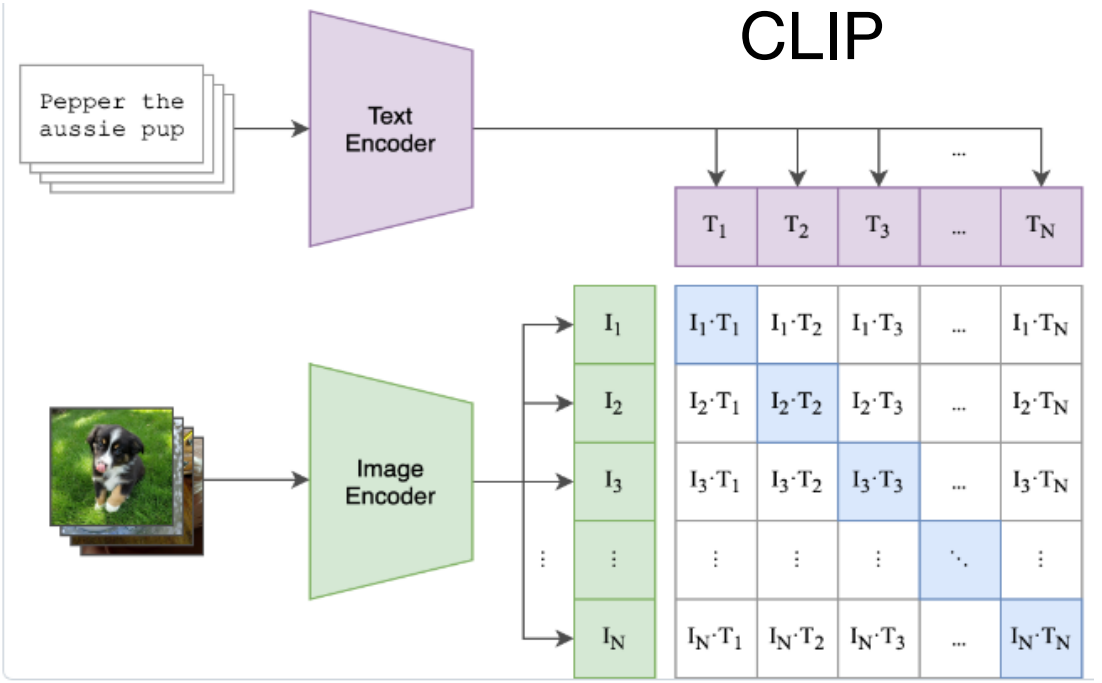


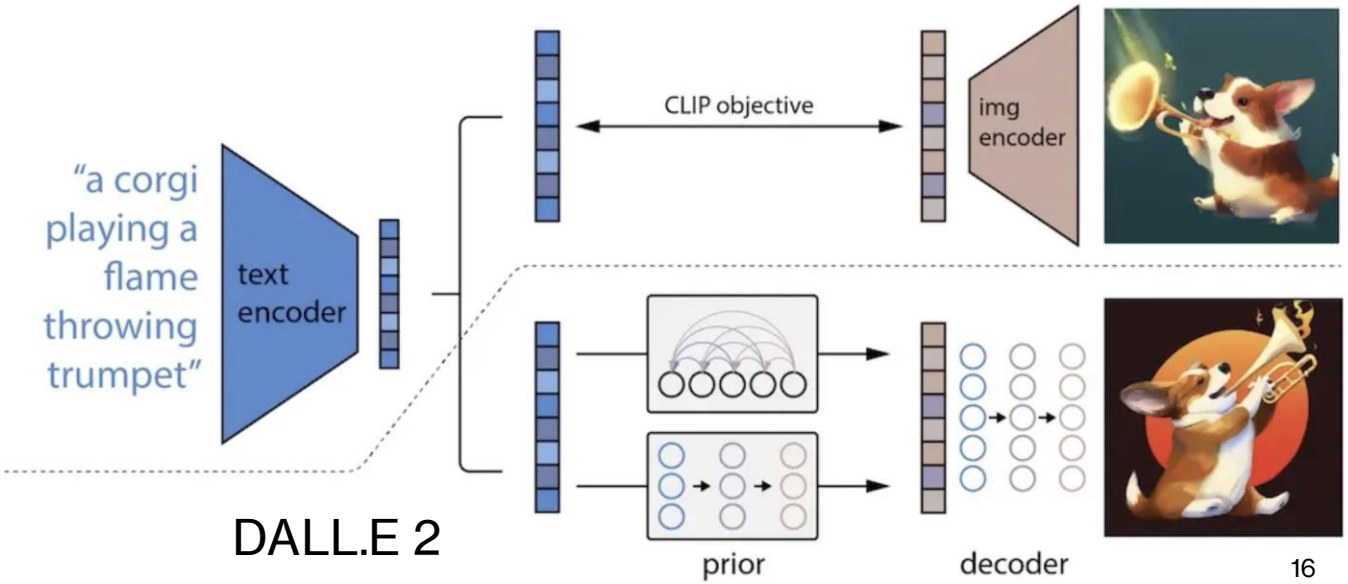


DINOBot, Di Paolo et al., ICRA24

Multimodal models

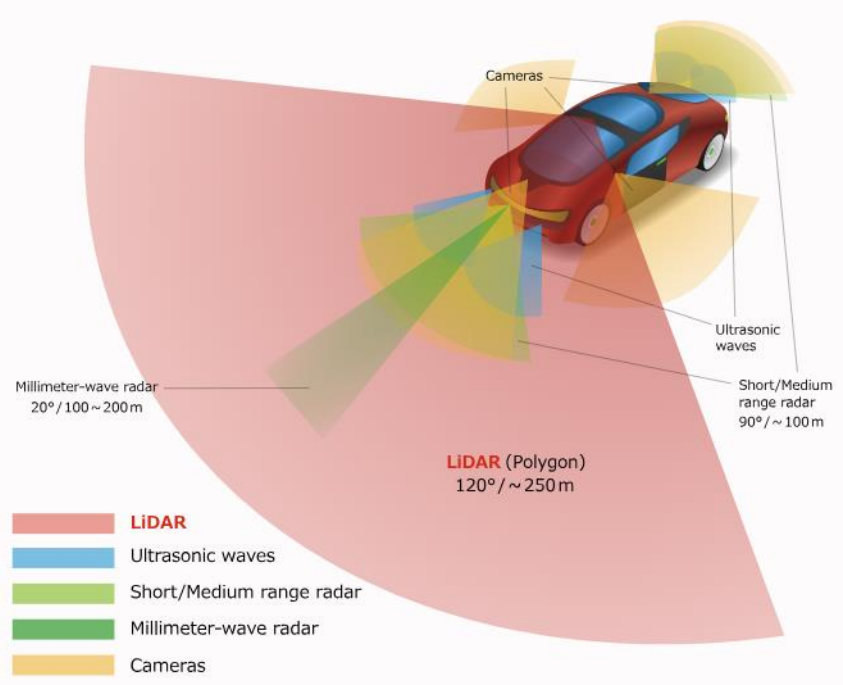
- Transformers make it easy to merge modalities represented with different tokens
- For example CLIP is trained to provide similar encoding for images and their captions
- Can be leveraged to caption images, generate image from text, answer questions about images, ...

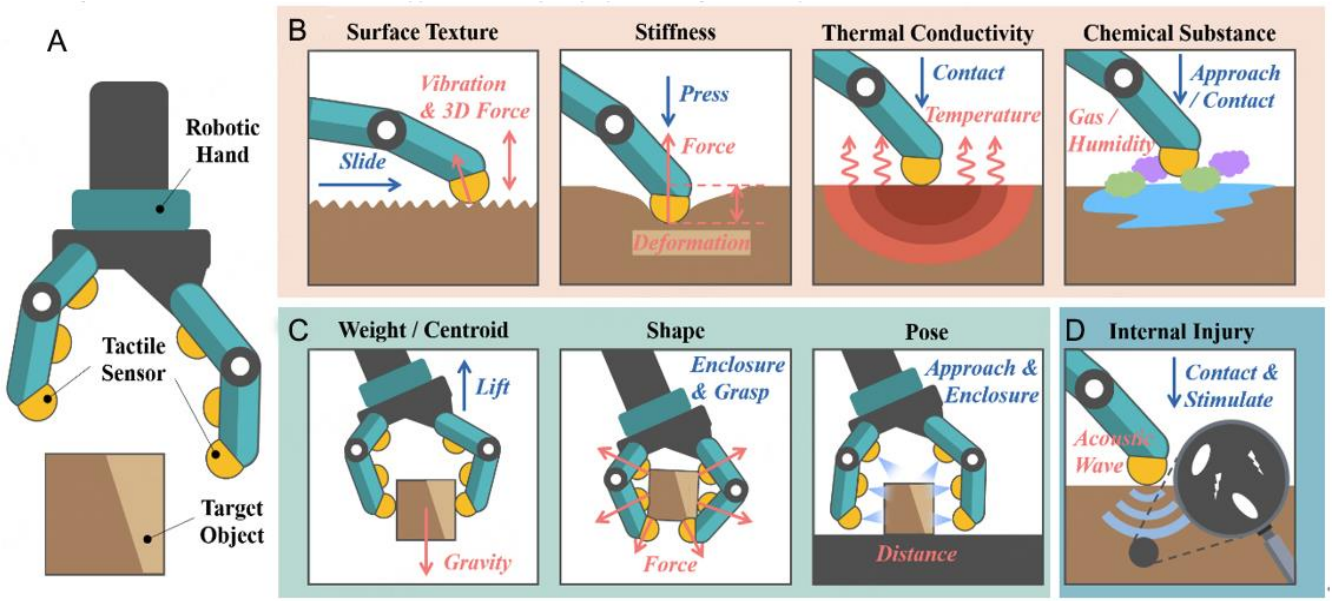




Note: Multimodality in robotics

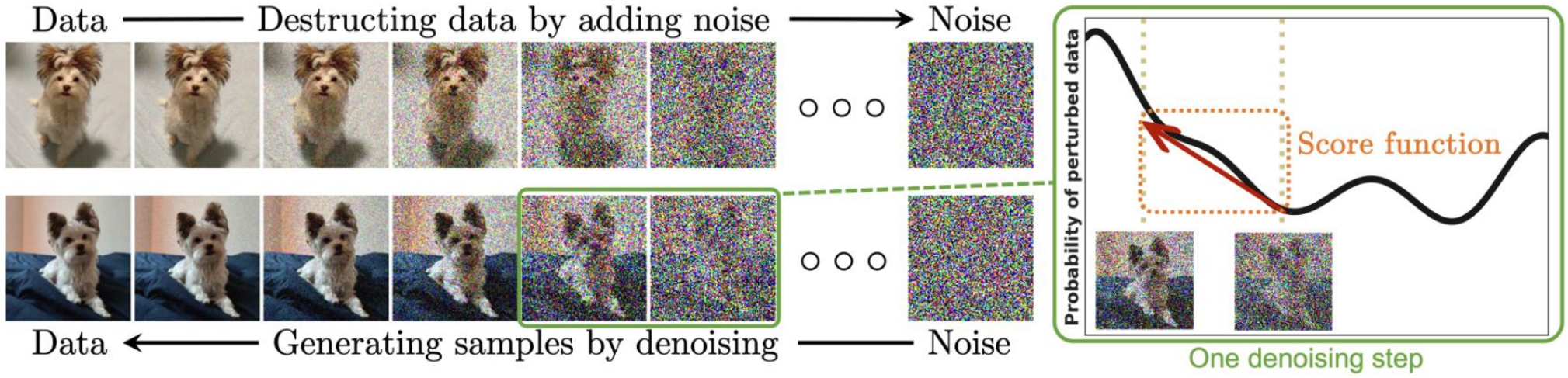
- Multimodal in AI mostly means Text + Image (+video)
- Multimodality is key in robotics, but with a much wider meaning
  - Images
  - LiDAR, radar
  - Proprioception, IMU
  - Touch, force
  - Text, dialog
  - Actions, trajectories ...
- Large multimodal models can be useful
  - Connecting language / robot for HRI
  - Improving perception/sensor fusion
  - Improving control/planning

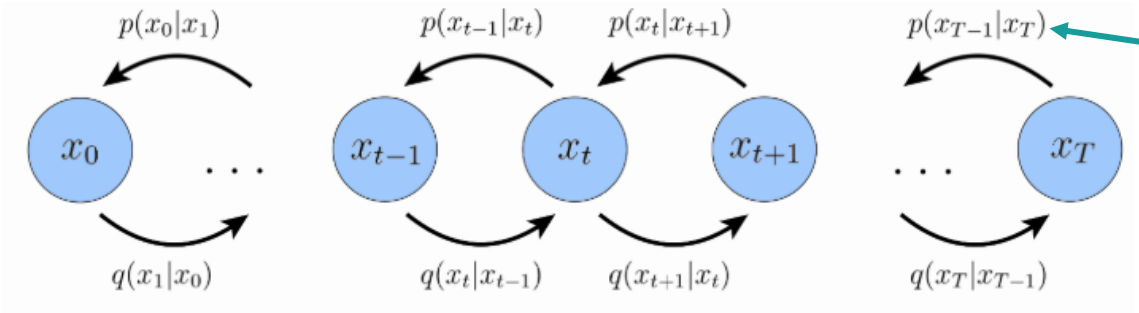




Diffusion models

- Models used to generate data following a training dataset distribution
- Learn a denoising network from data with artificially added noise

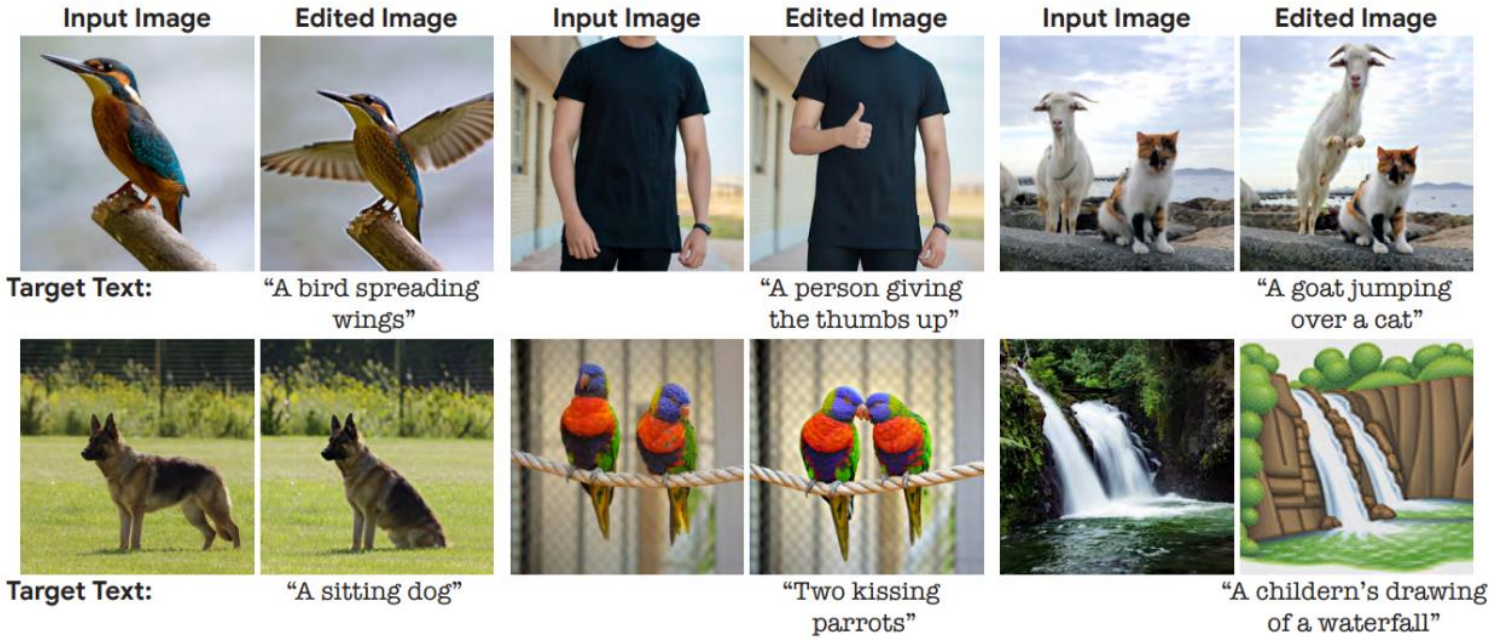




Learned denoising

Diffusion models

- Denoising network can be conditioned on class, text, images ...

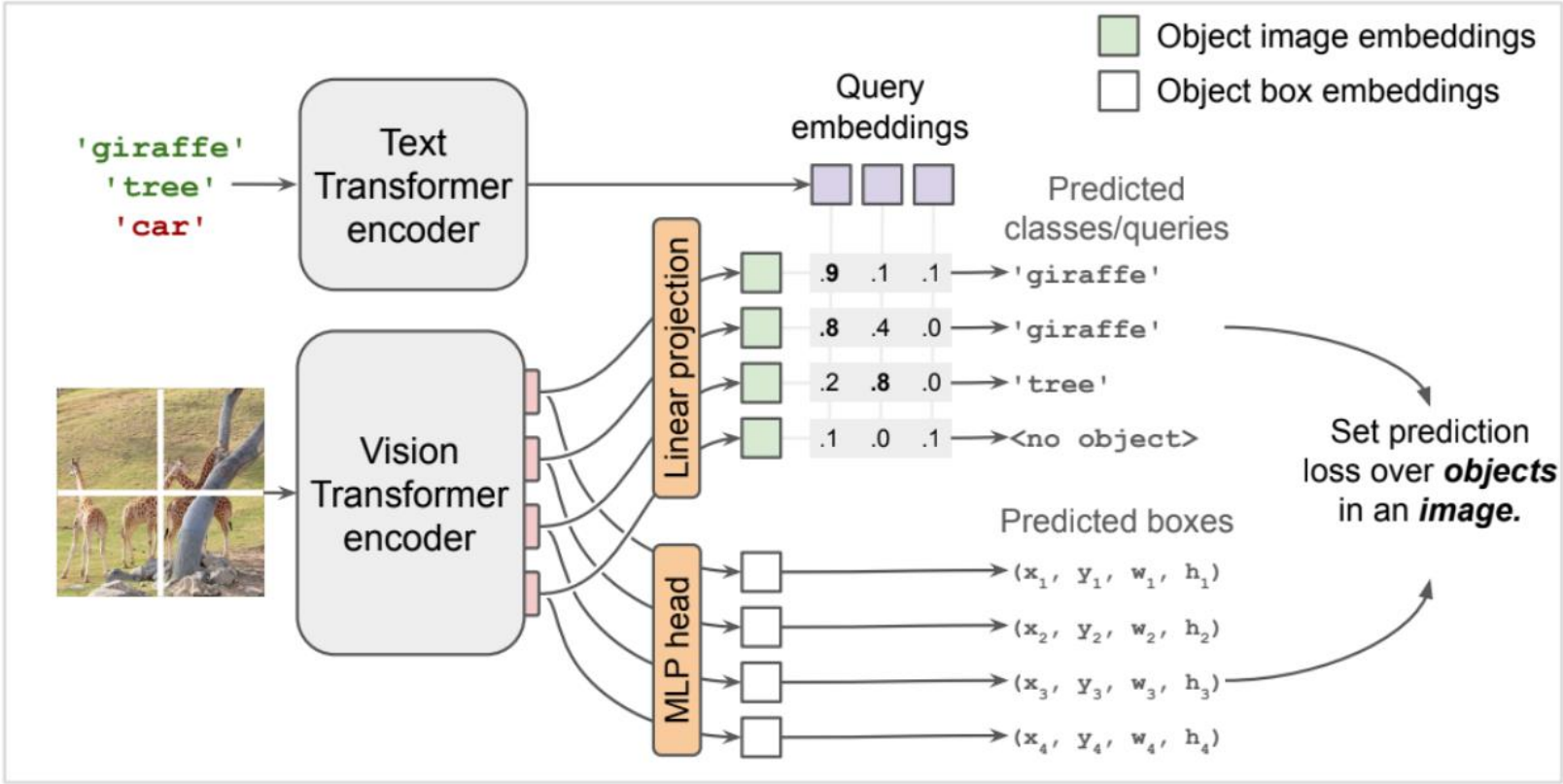


- Can be applied to point clouds, DNA, trajectories ...

USING LARGE MODELS IN ROBOTICS SYSTEMS

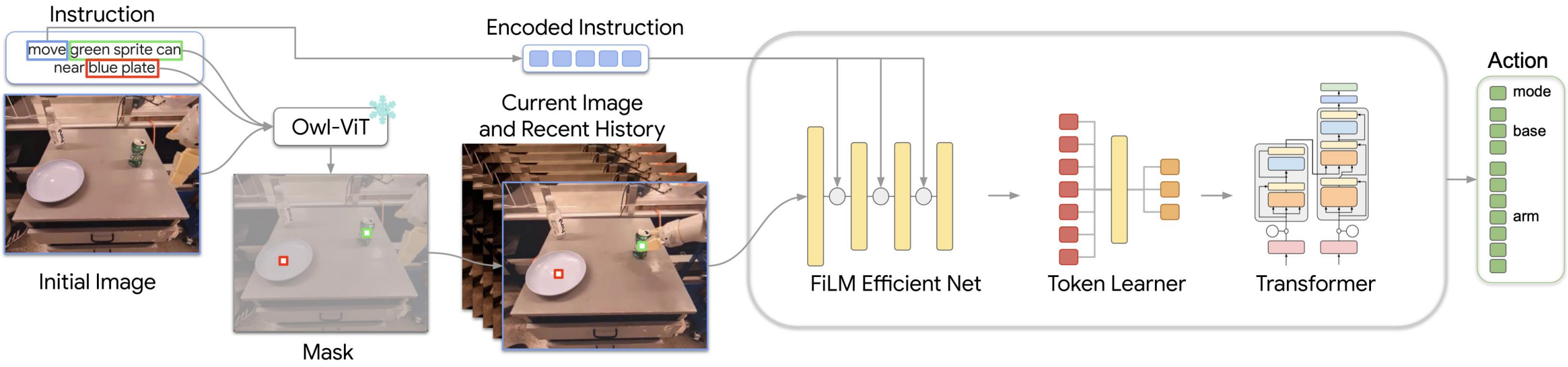
Using Vision-Language-Models

- Open vocabulary perception
  - Detect object from text description, not only class id
  - e.g., OWL-ViT
  - Pretrained to align image/text representation (~ CLIP)
  - Train to predict per-token description score and bounding boxes



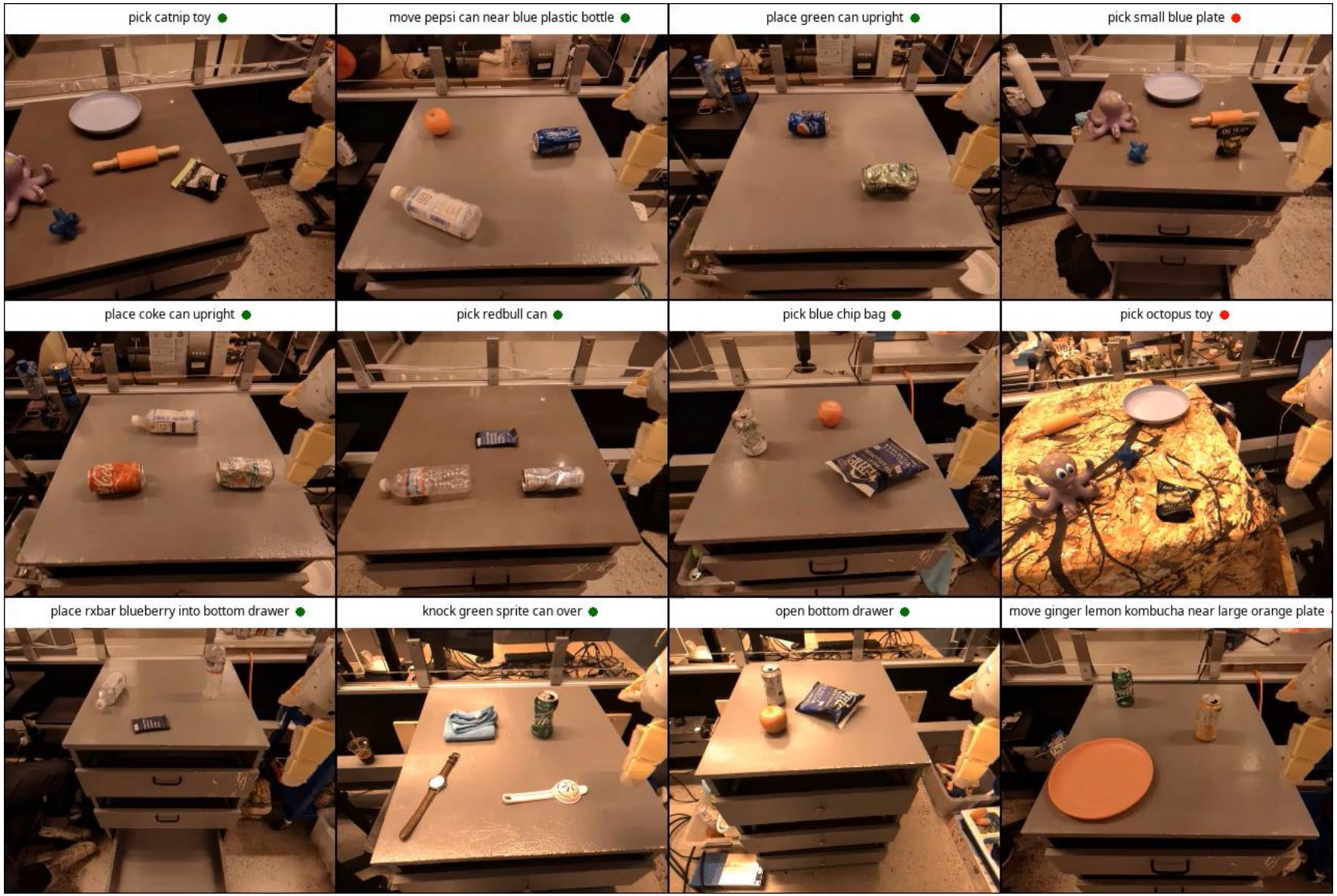
Using Vision-Language-Models

- Open vocabulary perception
  - Used for object localisation
  - Here associated with a VLA for control (see latter), but could be with a classical pipeline



https://robot-moo.github.io/

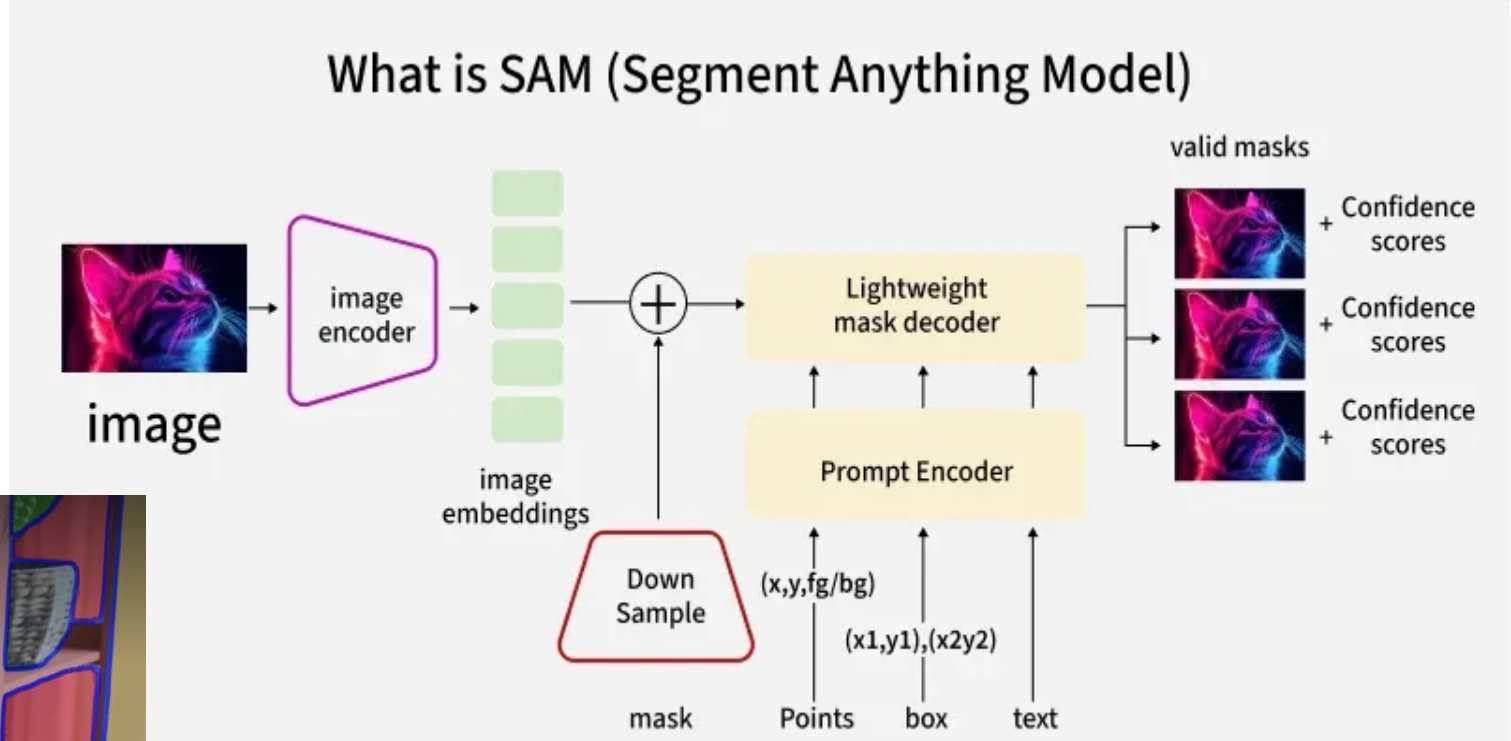
Using Vision-Language-Models



https://robot-moo.github.io/

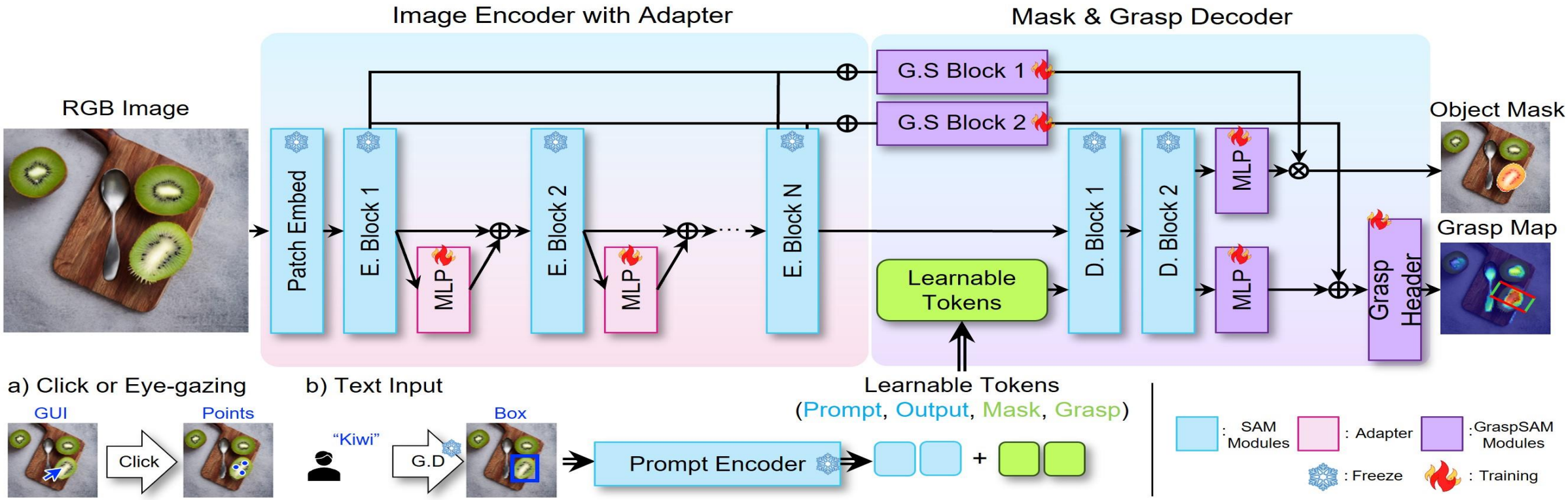
Using Vision-Language-Models

- Semantic scene understanding
  - e.g., Segment Anything (SAM)
  - Can detect object from text, or refine object detection (point or box)



Using Vision-Language-Models

- Semantic scene understanding
  - e.g., GraspSAM : an extension of SAM for arbitrary object grasping



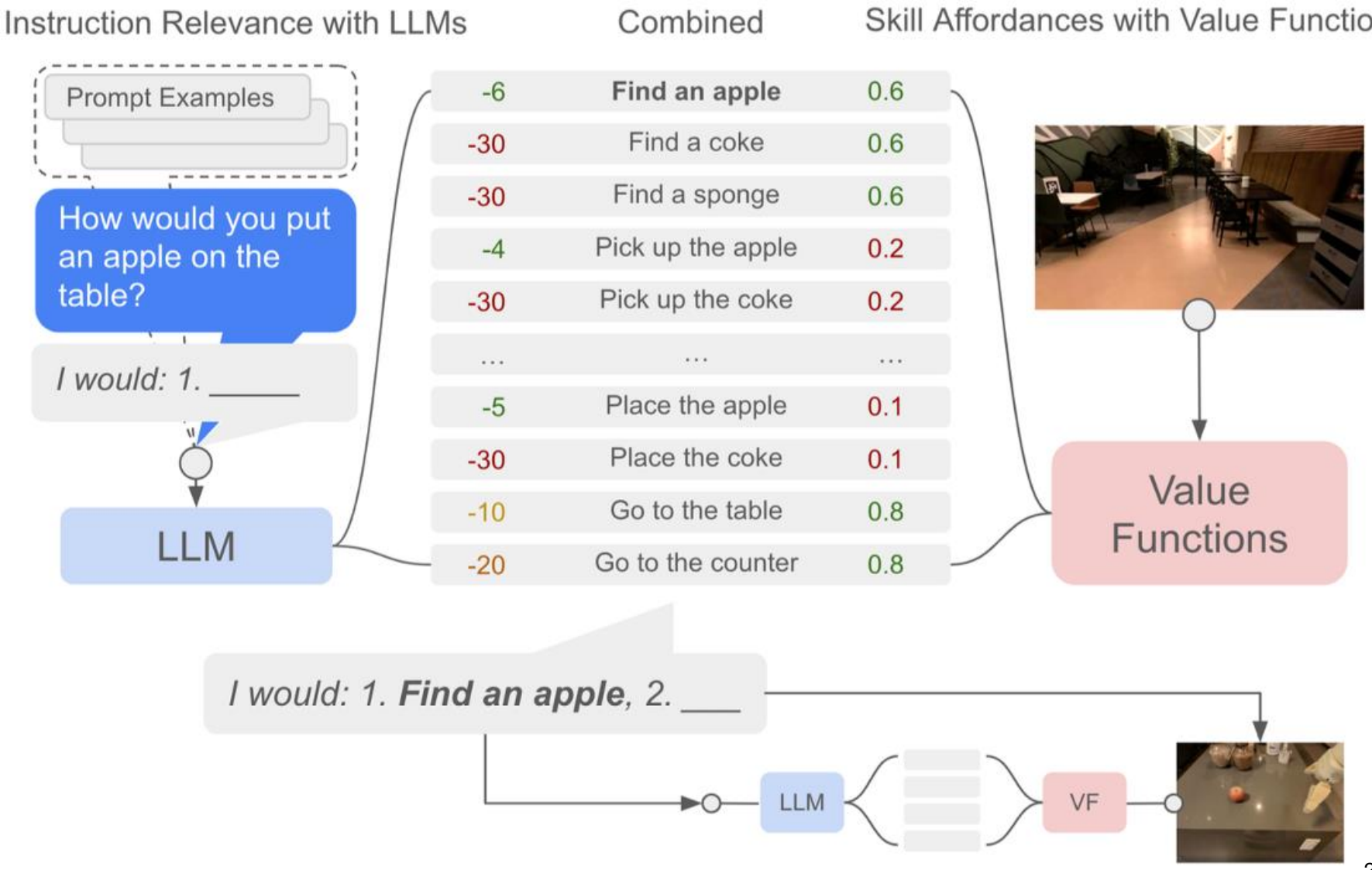
Using Large Language Models

- High level planning
  - E.g., SAY-CAN [Brohan et al., 2022]

Figure 1: LLMs have not interacted with their environment and observed the outcome of their responses, and thus are not grounded in the world. SayCan grounds LLMs via value functions of pretrained skills, allowing them to execute real-world, abstract, long-horizon commands on robots.

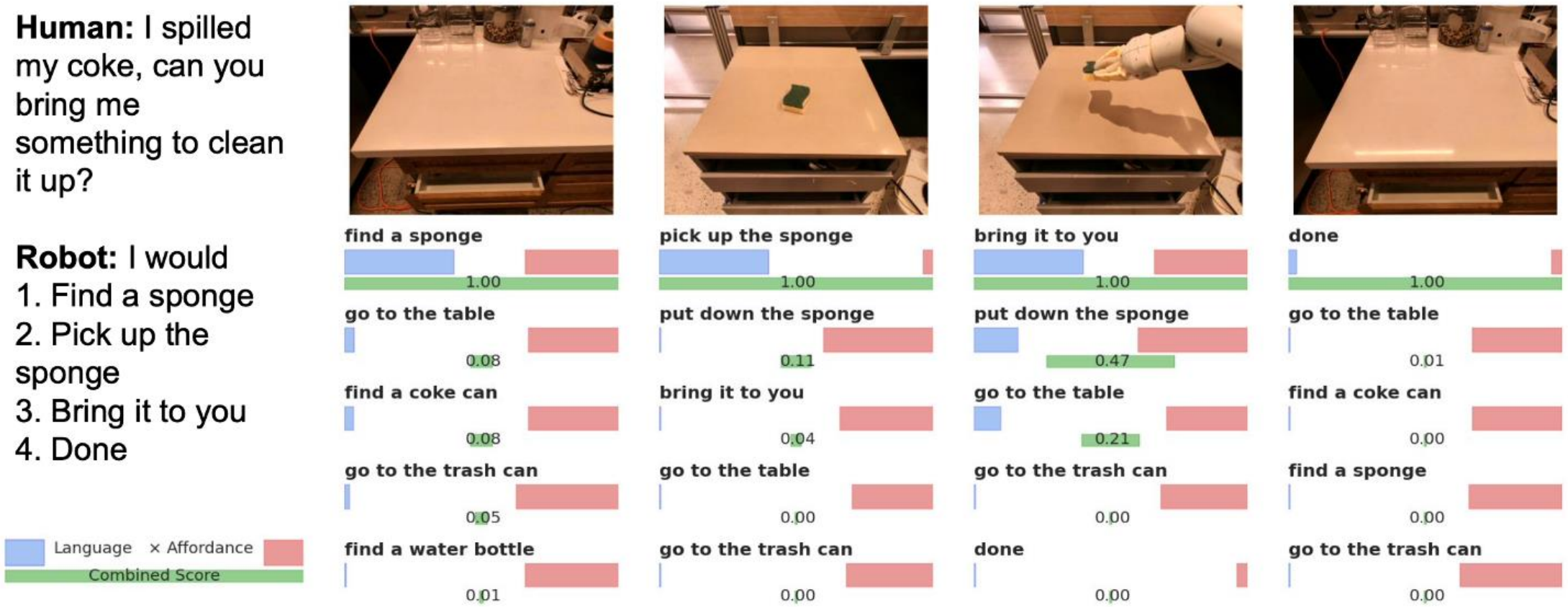
Using Large Language Models

- SAYCAN
  - A library of vision-based primitive actions are learned using RL
  - LLM give weight to primitives actions from prompt
  - Value function gives weight to primitive actions from images



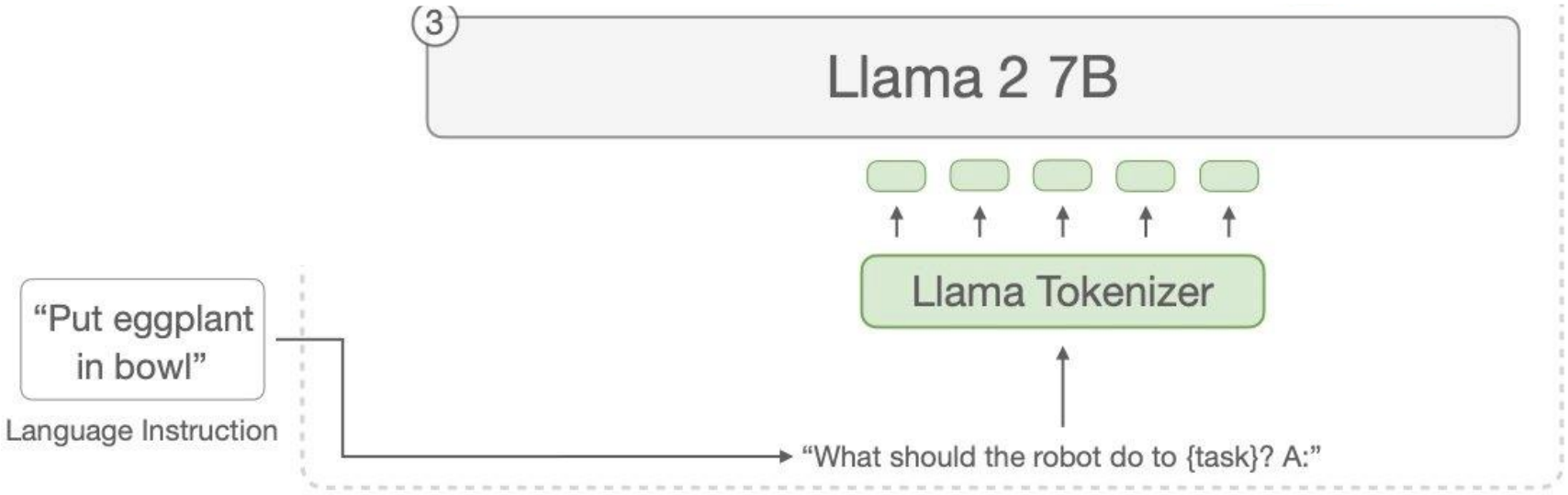
Using Large Language Models

- SAYCAN



VISION LANGUAGE ACTION MODELS

Vision Language Action models



LLM

VLM

VLA

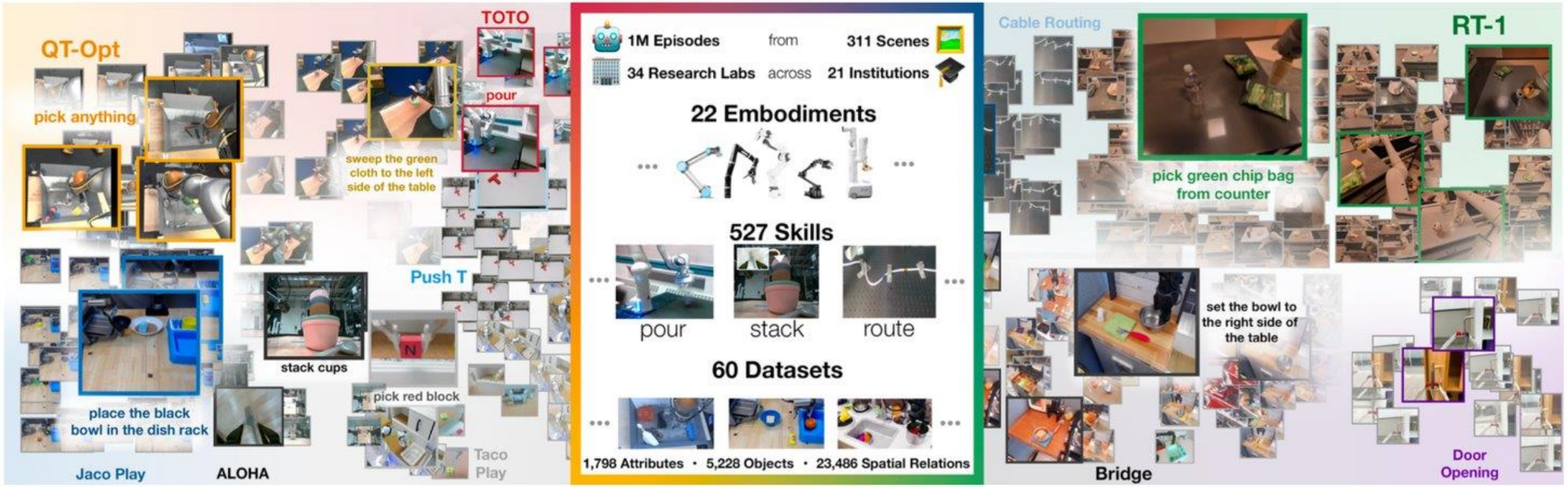
Can learn several task/robots

Needs a lot of demonstration data

Mostly trained by imitation

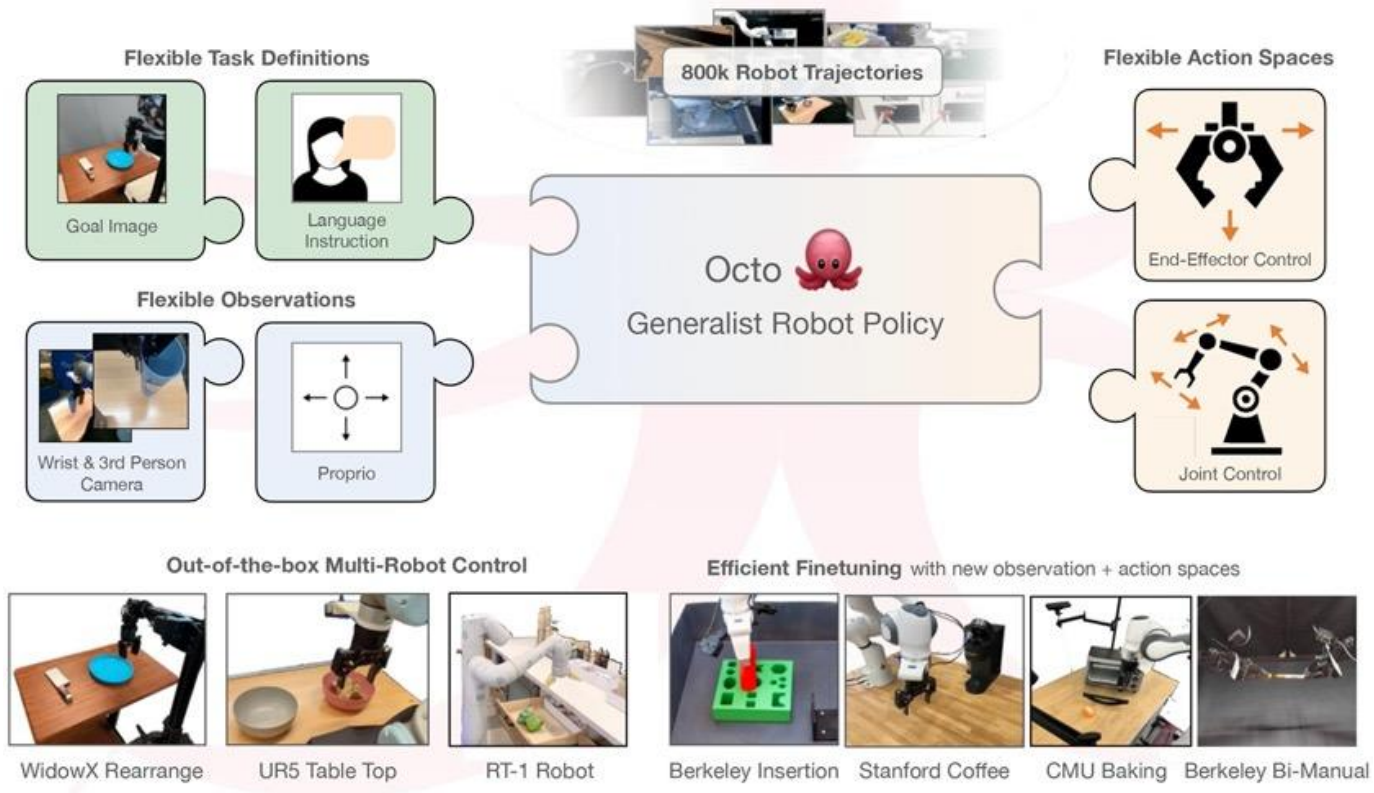
Large robotics dataset

- Open-X embodiment
  - Large collaborative academics/industry effort for recording demonstrations



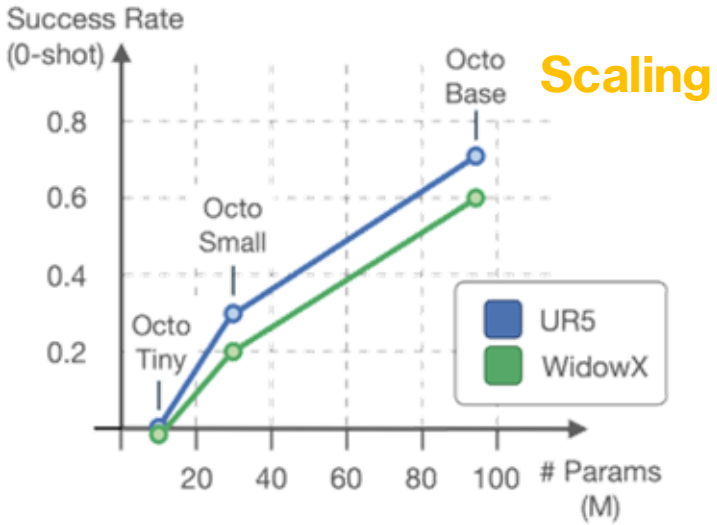
An example model : Octo

- Large generalist model
- Multi-task, multi-robot policy trained by imitation (supervised)

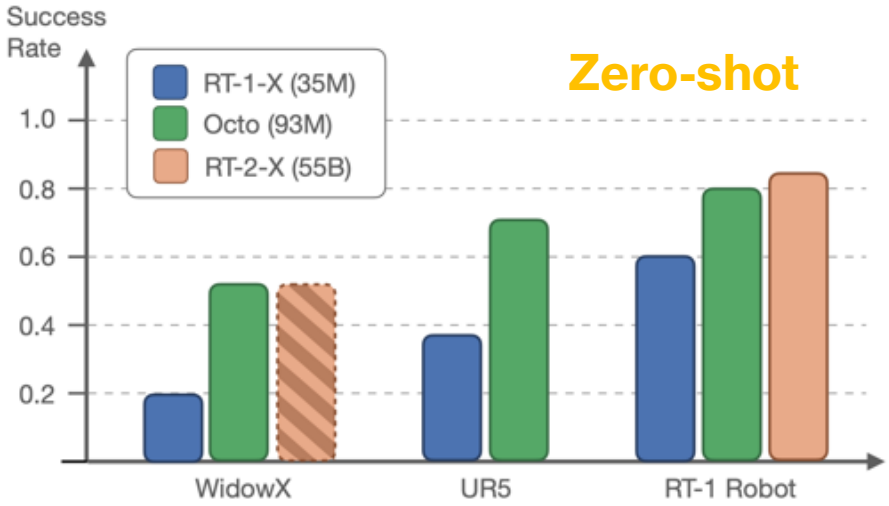


VLA in robotics manipulation

- Model capabilities (Ex. : Octo)
  - Control of multiple robots / tasks



Scaling



Zero-shot

		Language-conditioned		Goal-conditioned		
		Put carrot on plate	Put eggplant in pot	Put bread on plate	Put spoon on glove	Average
	Octo-small (Ours)	80%	90%	70%	90%	**83%**
DATA	RT-X dataset mix [67]	80%	80%	40%	40%	60%
	Single robot dataset (Bridge Data)	20%	70%	60%	20%	43%

- Efficient fine tuning on new robots / tasks - Advantage of multi-embodiment datasets

	Berkeley Insertion*	Stanford Coffee	CMU Baking	Berkeley Pick-Up†	Berkeley Coke	Berkeley Bimanual†	Average
ResNet+Transformer Scratch	10%	45%	25%	0%	20%	20%	20%
VC-1 [57]	5%	0%	30%	0%	10%	50%	15%
Octo (Ours)	**70%**	**75%**	**50%**	**60%**	**100%**	**80%**	**72%**

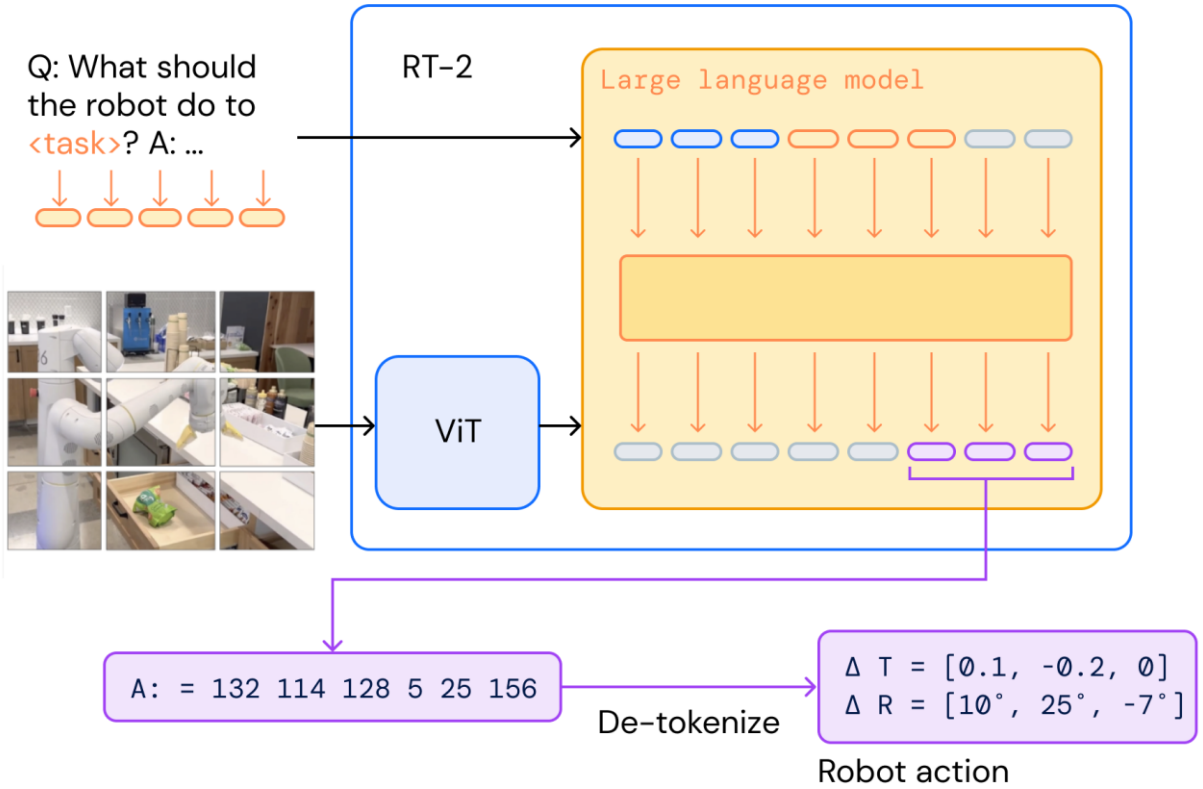
Fine tuning on 100 demonstrations

VLA models : RT-2 (Google, ~55B)

- Pretrained VLM (PaLM-E & PaLI-X)
- Discrete action tokens output:
  - Sentence to explain plan
  - Action tokens ~ output sentence

Prompt:
Given <img> Instruction:
Move all the objects
together.
Prediction:
Plan: move green can near
green rice chip bag.
Action: 1 128 126 127 135
123 119 127

Prompt:
Given <img> Instruction:
Pick the object that is
different from all other
objects
Prediction:
Plan: pick rxbar
chocolate. Action: 1 128
129 125 131 125 128 127

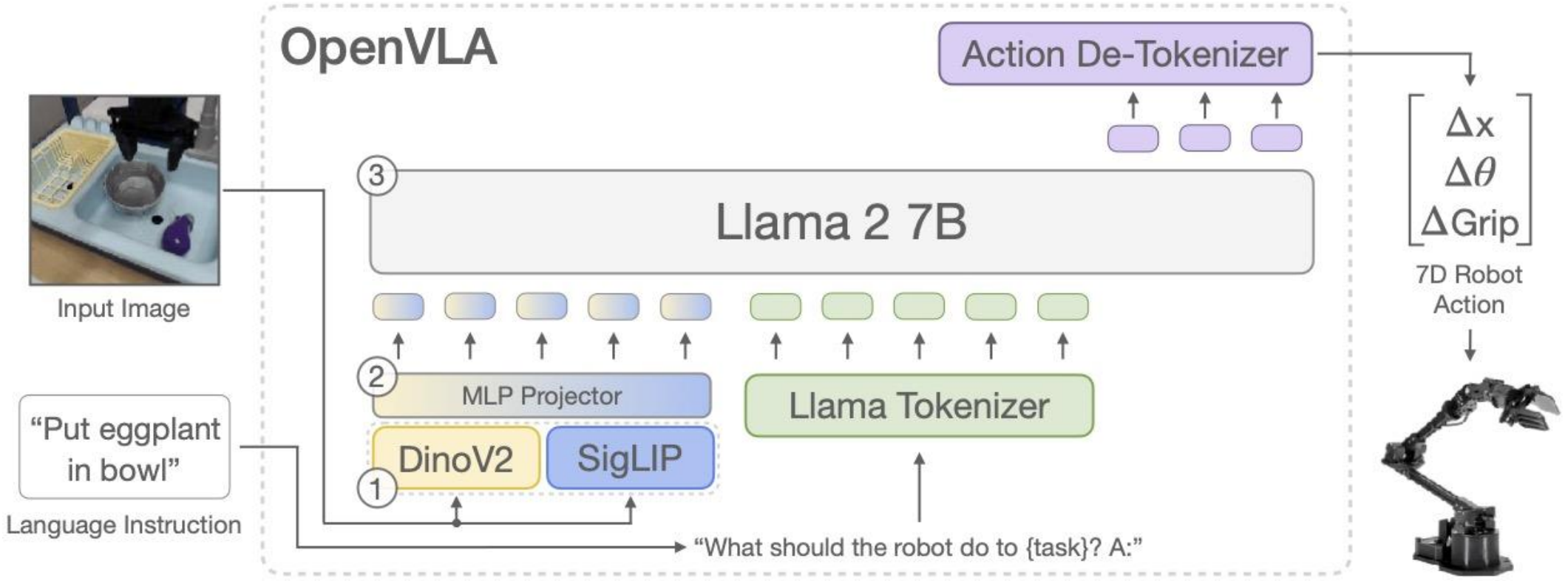


+ IK controller

https://learnopencv.com/vision-language-action-models-lerobot-policy/

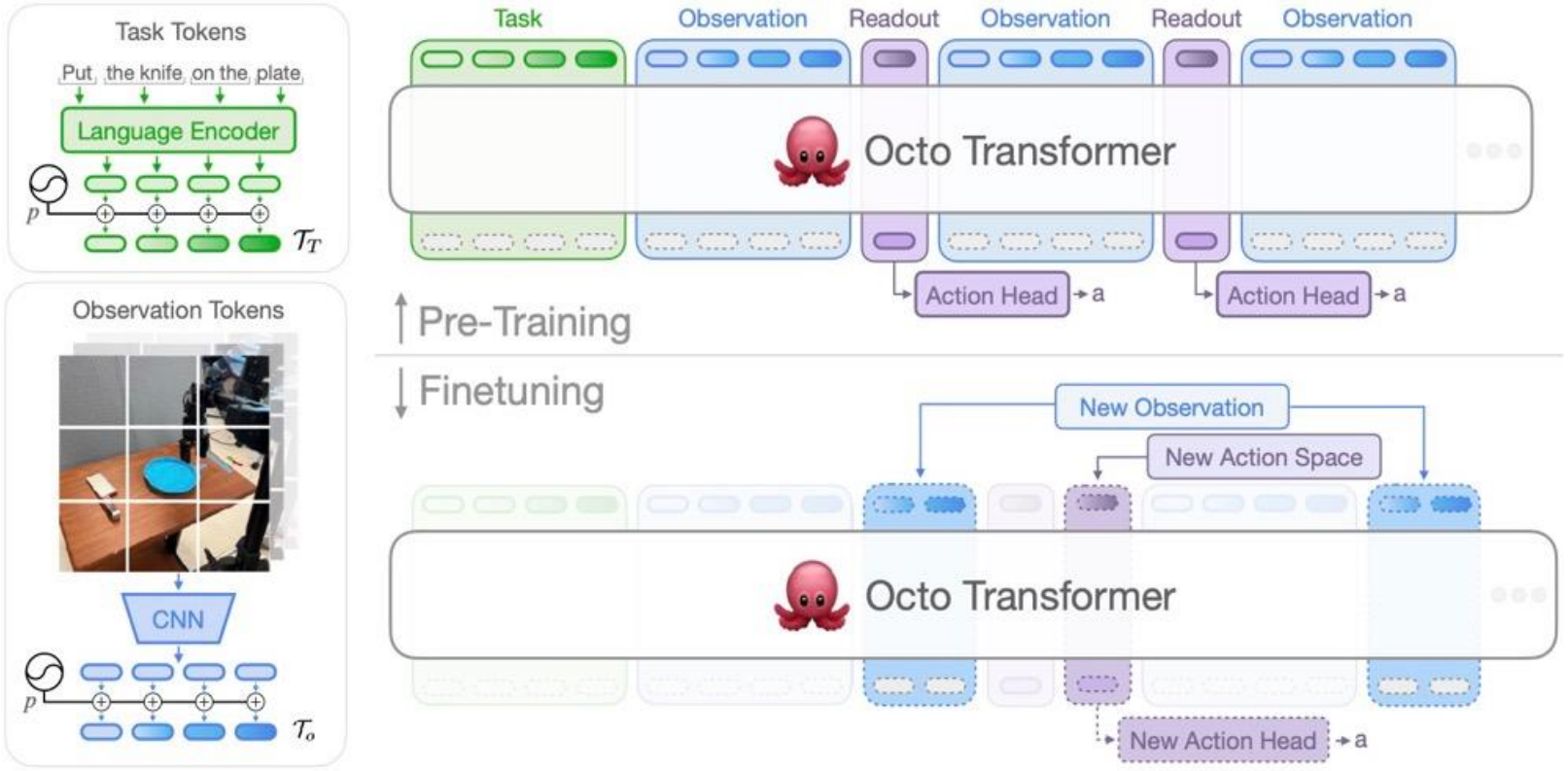
VLA models : OpenVLA (Stanford, ~7B)

- Structure similar to RT-2, open-source, better efficiency, but less generalization



VLA models : OCTO (Berkeley, ~93M)

- Smaller model, trained from scratch
- Masking to adapt to multiple sensor configurations
- Perf close to RT-2
- Diffusion policy to model multiple action possibilities

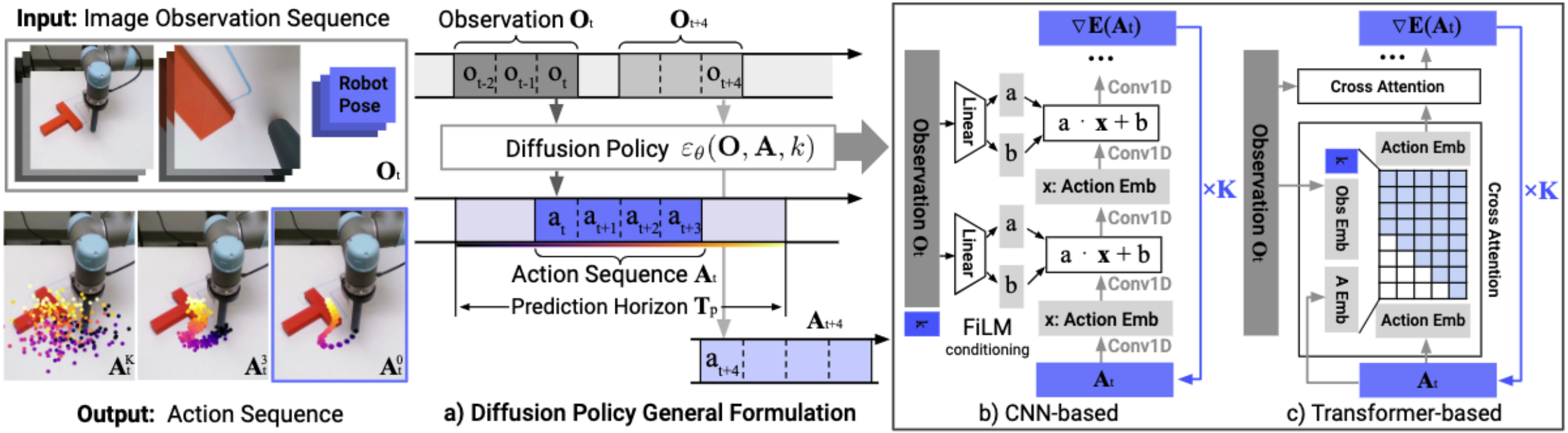


VLA models : OCTO

Diffusion Policies

[Chi et al., 23]

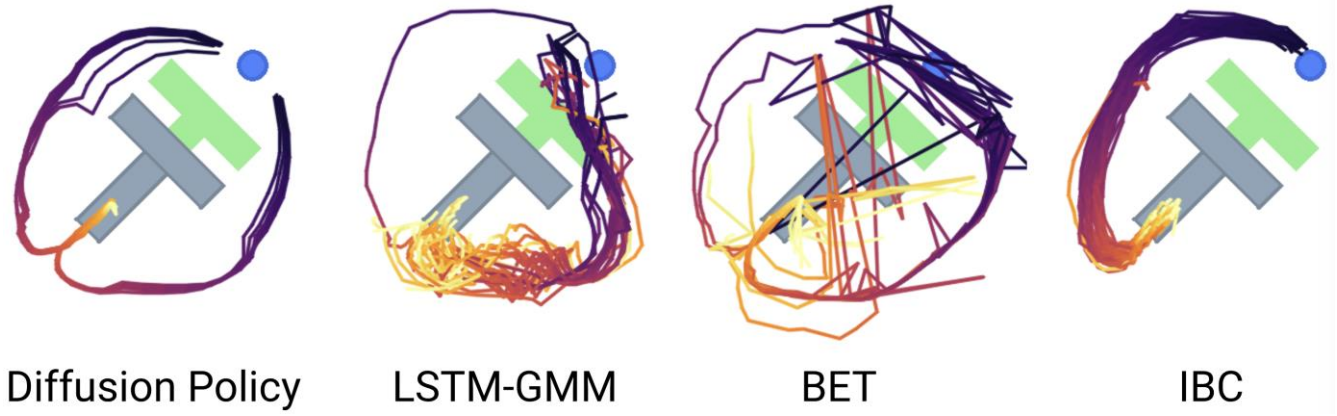
- Diffusion model for trajectory generation
  - Generate trajectories from current state, using diffusion starting from gaussian noise

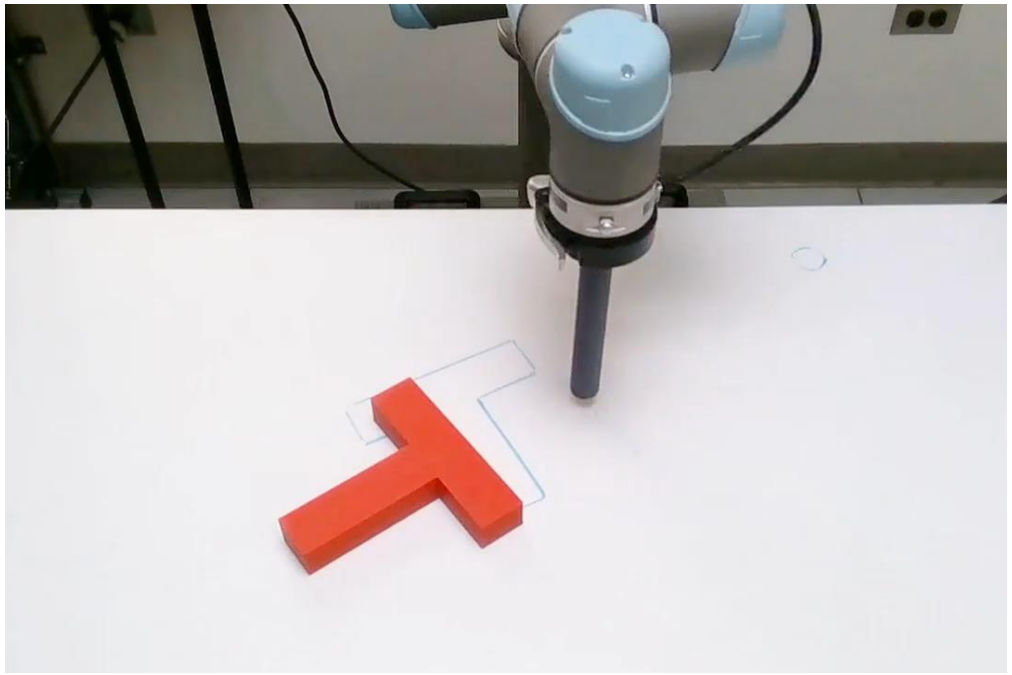


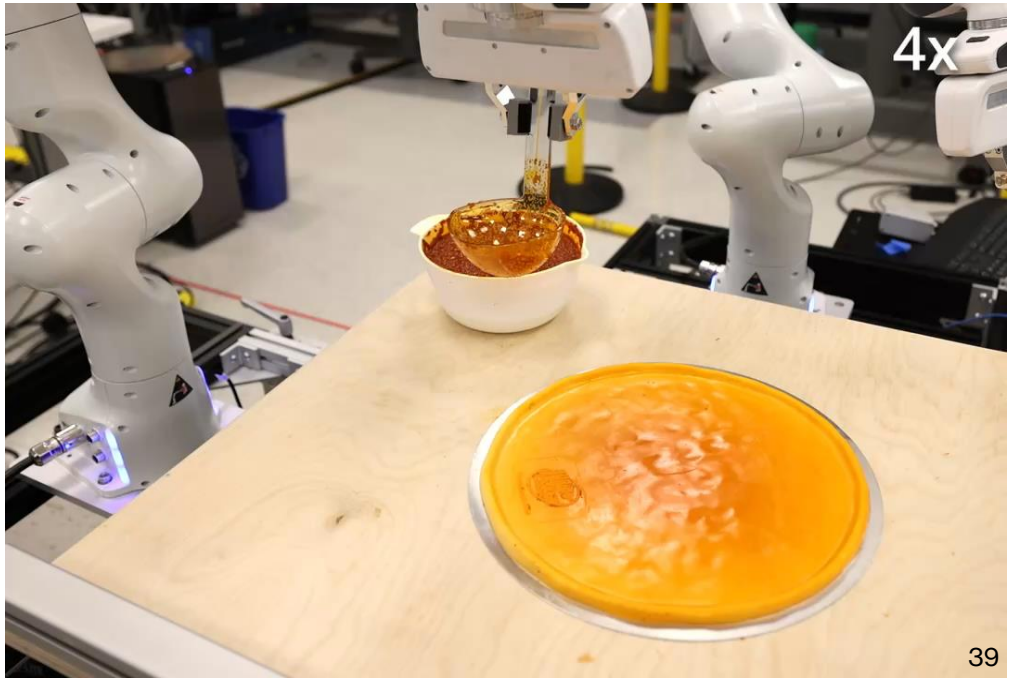
Diffusion Policies

[Chi et al., 23]

- Policy execution
  - Follow beginning, then replan (~MPC)
- Good at representing multimodality

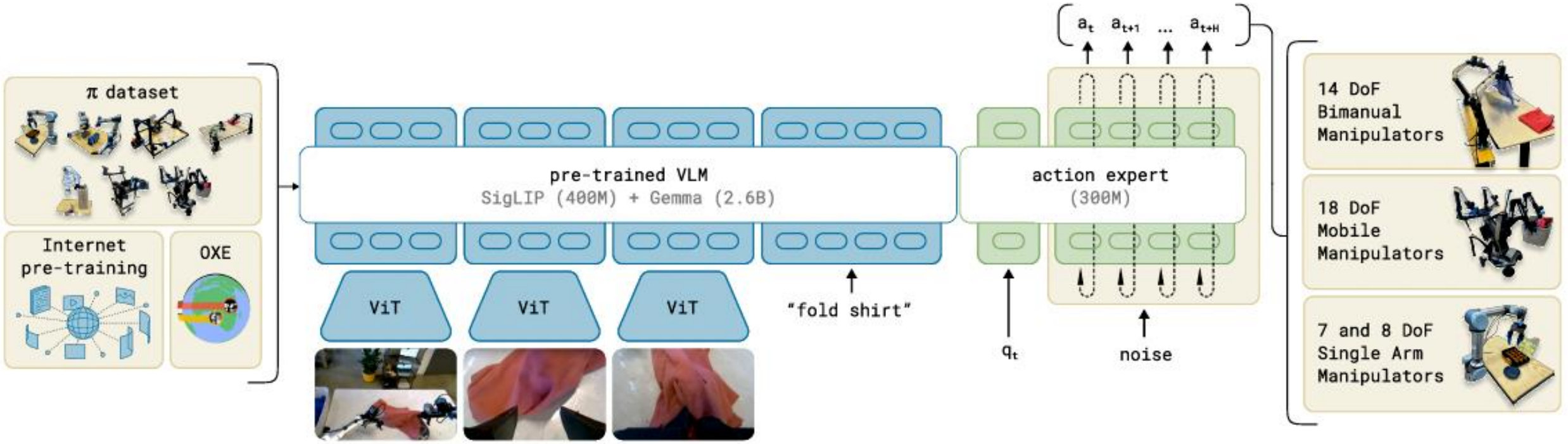






VLA models : π0 (Physicalintelligence, ~3,3B)

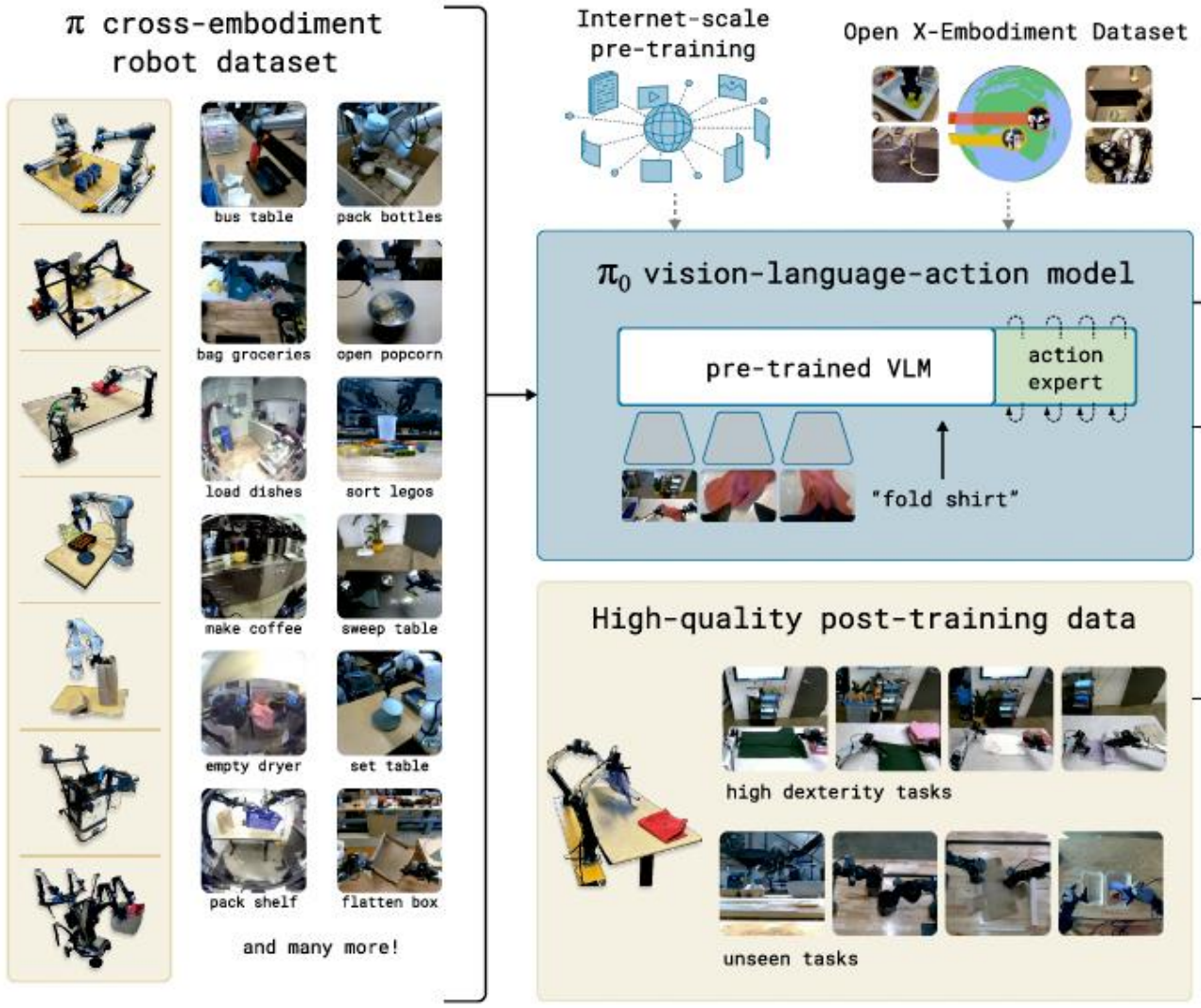
- Vision-Language-Action (VLA) Model with Flow Matching (~diffusion policies)
  - Backbone: Pre-trained VLM PaliGemma (3B parameters)
  - Action Generation: Flow matching for continuous robot control actions up to 50 Hz



https://www.physicalintelligence.company/blog/pi0

VLA models : π_0

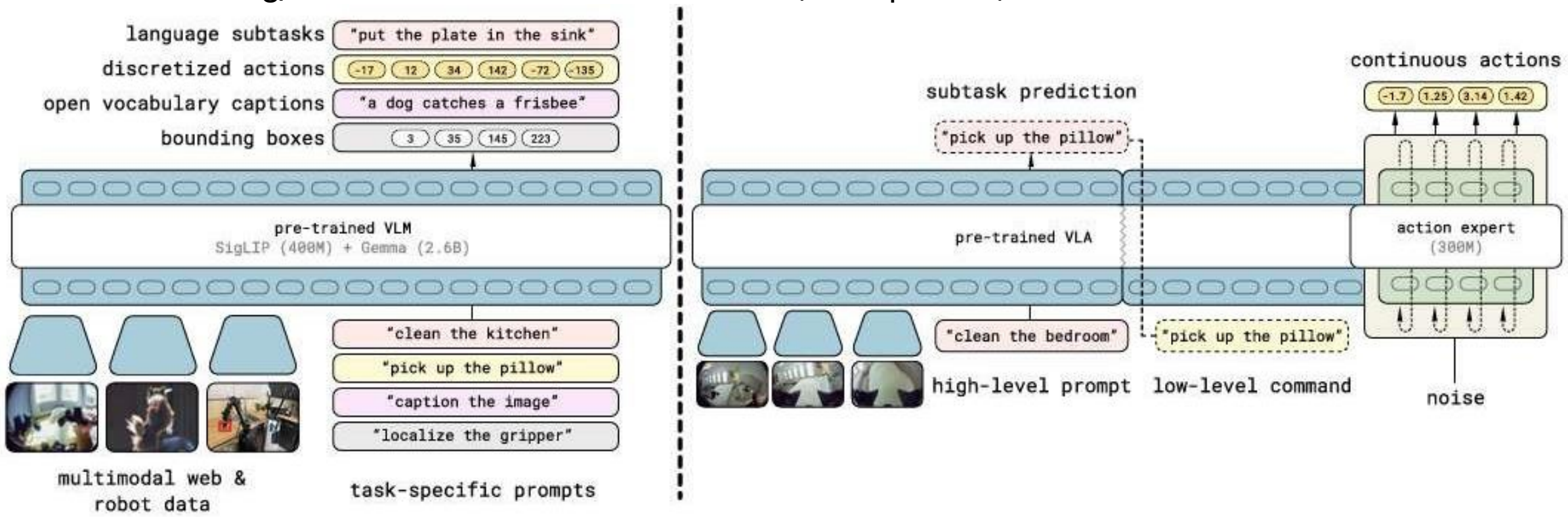
- Massive Cross-Embodiment Pretraining
  - Internal corpus: 10,000+h of robot data across 7 robots and 68 tasks (903M timesteps)
  - Open-source datasets: OXE (22 robots), Bridge v2, DROID
  - Cross-embodiment learning: Unified training across robot morphologies
- Post training
  - 5 h robot data for simplest tasks
  - 100+h for complex tasks



https://www.physicalintelligence.company/blog/pi0

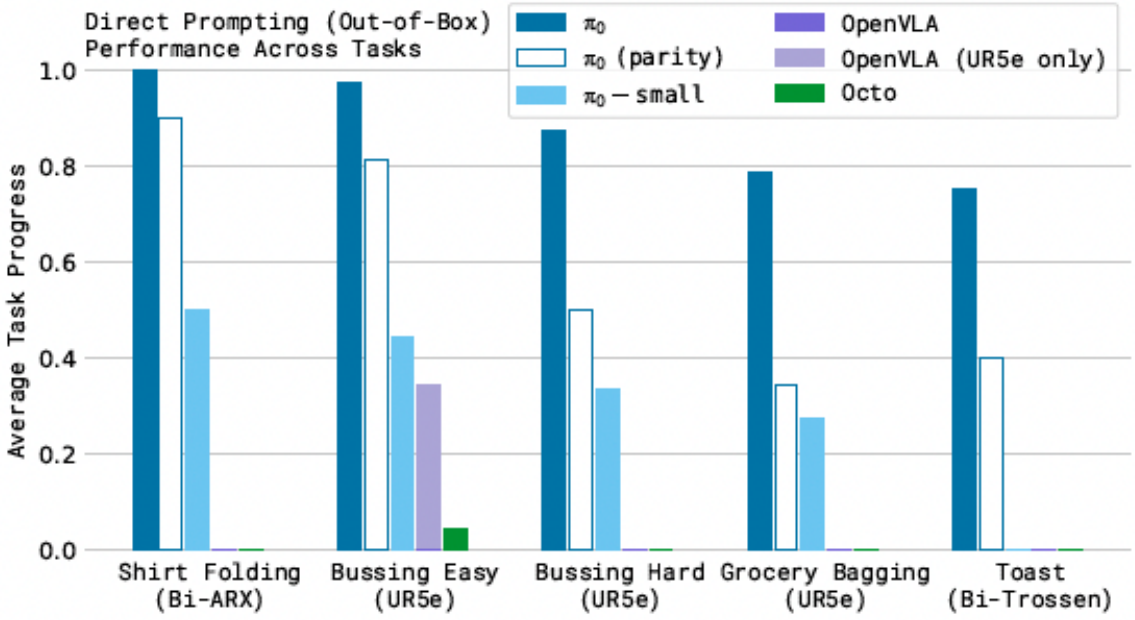
VLA models : π_0

- Mixing discrete and continuous actions
  - Discrete pre-training with several tasks / next subtask /discrete actions (faster / easier)
  - Post-training/inference with continuous actions (more precise)



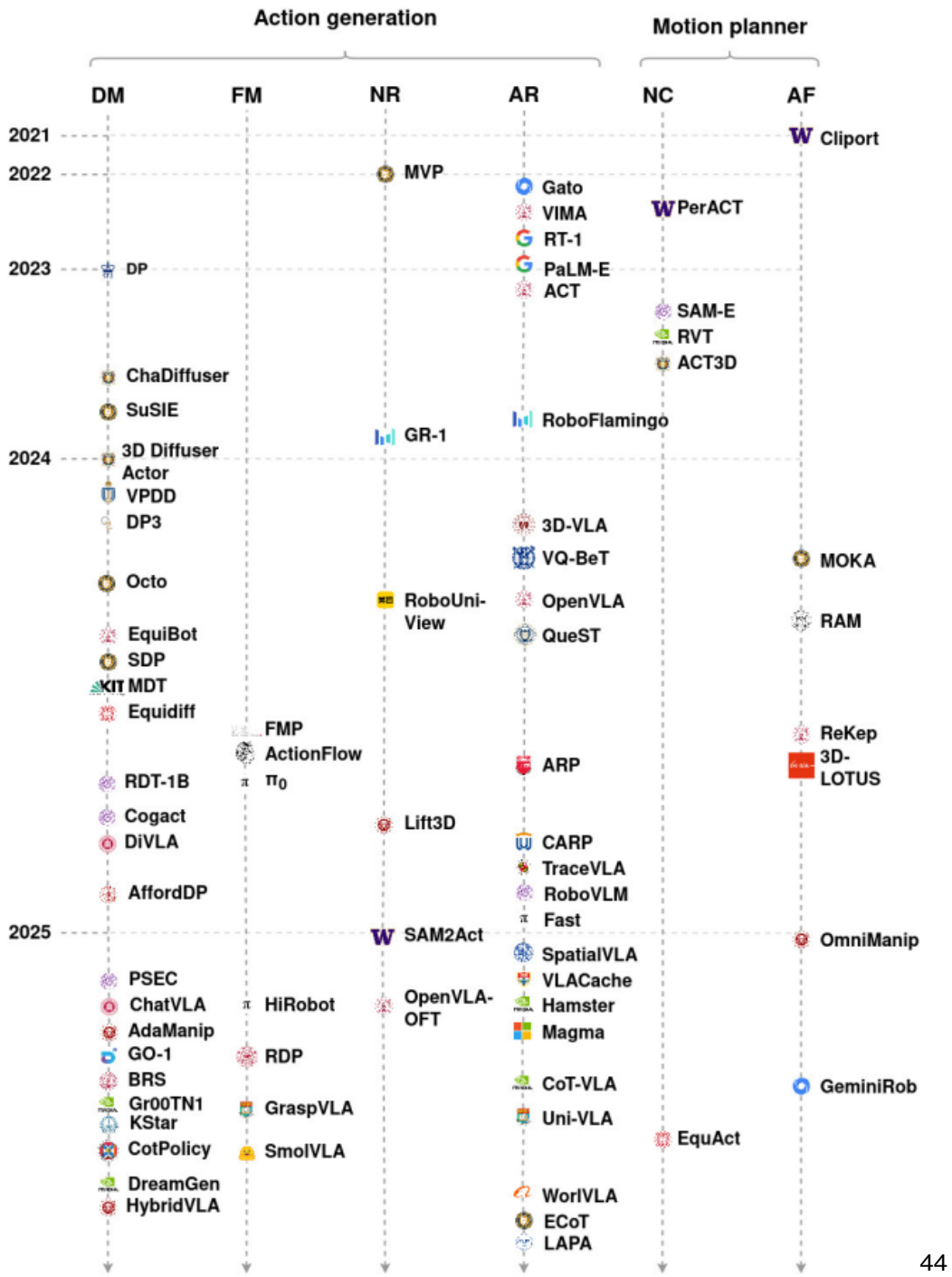
VLA models : $\pi_{0.5}$

- Good zero-shot performances
- Good generalization capabilities
- Importance of large high-quality data



VLA Models

- Most company/labs in robotics develop their VLA at the moment
- Data is still a bottleneck adressed by many companies
- Robustness can be quite high, and performance can be quite good, but still far from 99.9...% reliability expected in many applications
- Models still require quite large number of demonstrations on each task, even with efficient pretraining, much more that what a human would need



Zezeng Lee et al., https://arxiv.org/abs/2508.17449

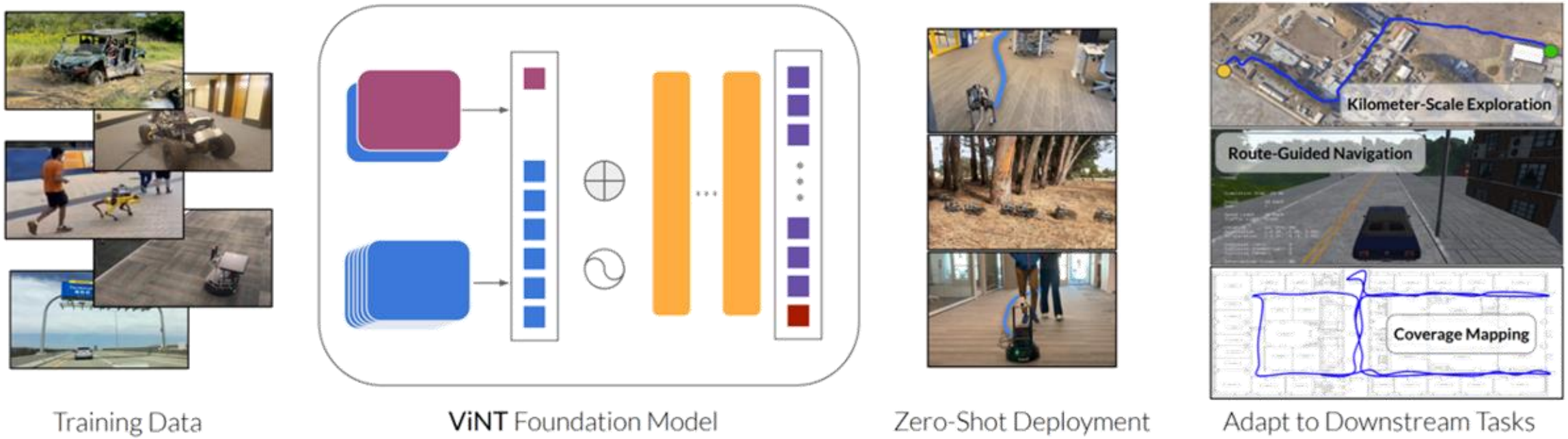
Fig. 2: Timeline of the models explored in this survey. Each model is classified by its control strategy as presented in Tab. I.

LARGE MODELS FOR NAVIGATION

ViNT (Berkeley, ~31M)

- A diffusion policy for navigation
  - Integration of 10 different navigation datasets from very different robots (no language)
  - Zero shot / fine tuning on different tasks/robots
  - Goal based / exploration with diffusion policies

[Shah et al., 23]



Training Data | **ViNT** Foundation Model | Zero-Shot Deployment | Adapt to Downstream Tasks

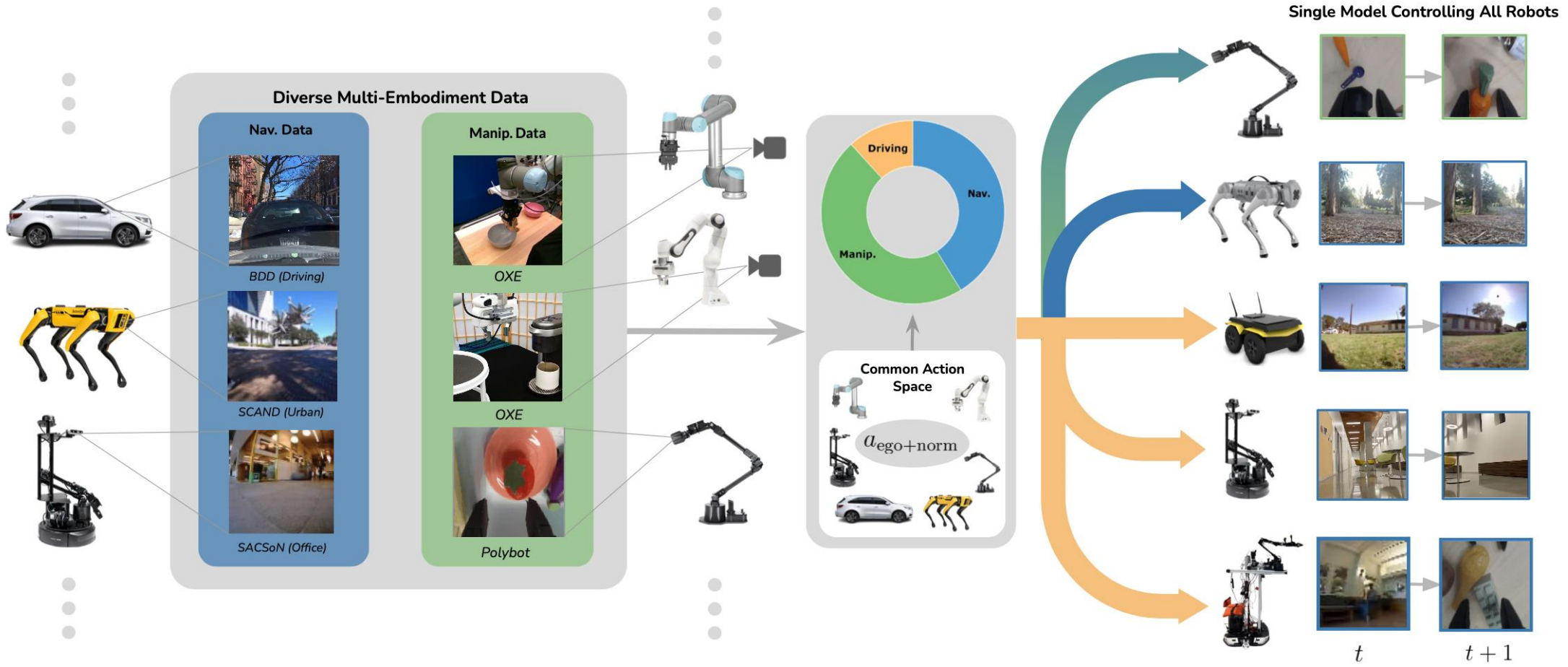
ViNT

Robust
But
Inefficient

Multi-task Multi-embodiment models ?

- One large models for all robots/tasks ?

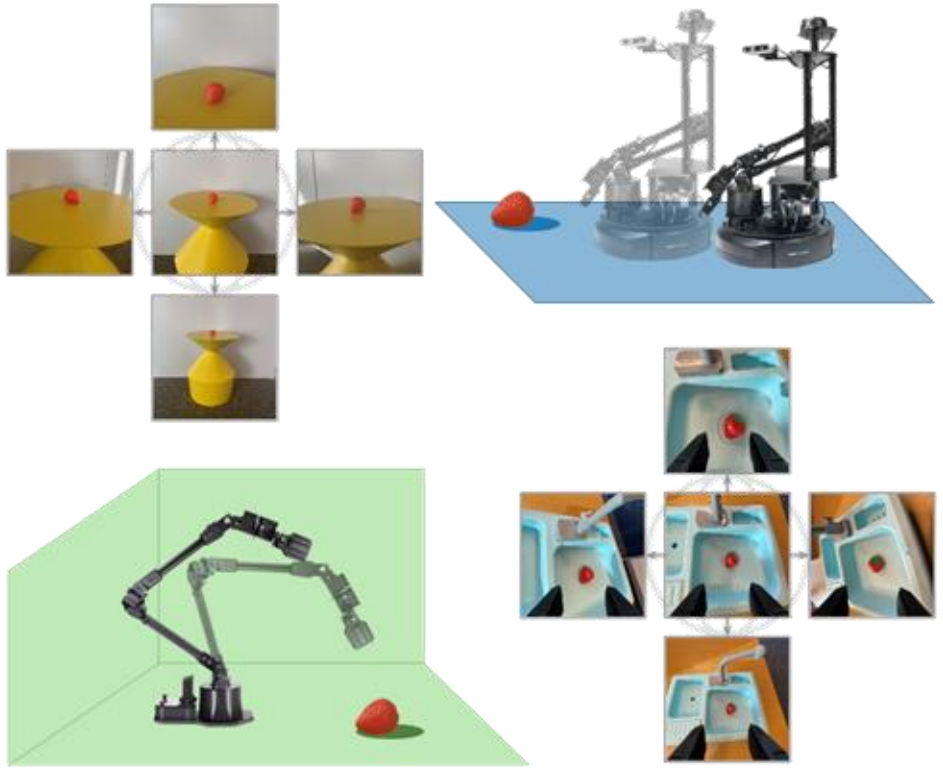
[Yang et al., 24]



Multi-task Multi-embodiment models ?

[Yang et al., 24]

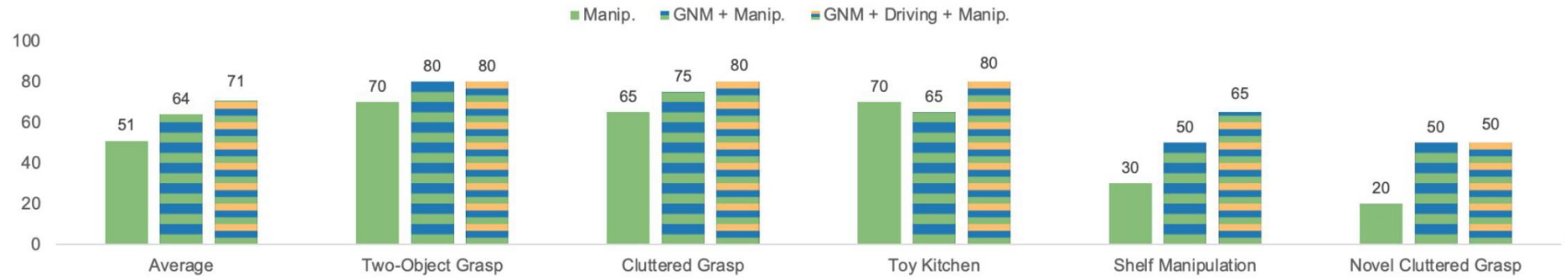
- Aligning datasets
  - Defining common representation for manipulation and navigation
  - Focusing on 1st person view
  - Normalised actions (left/right/up/down)
  - Applied to visual goal reaching task
- Large model
  - EfficientNet image encoding
  - Transformer
  - Diffusion policy action head



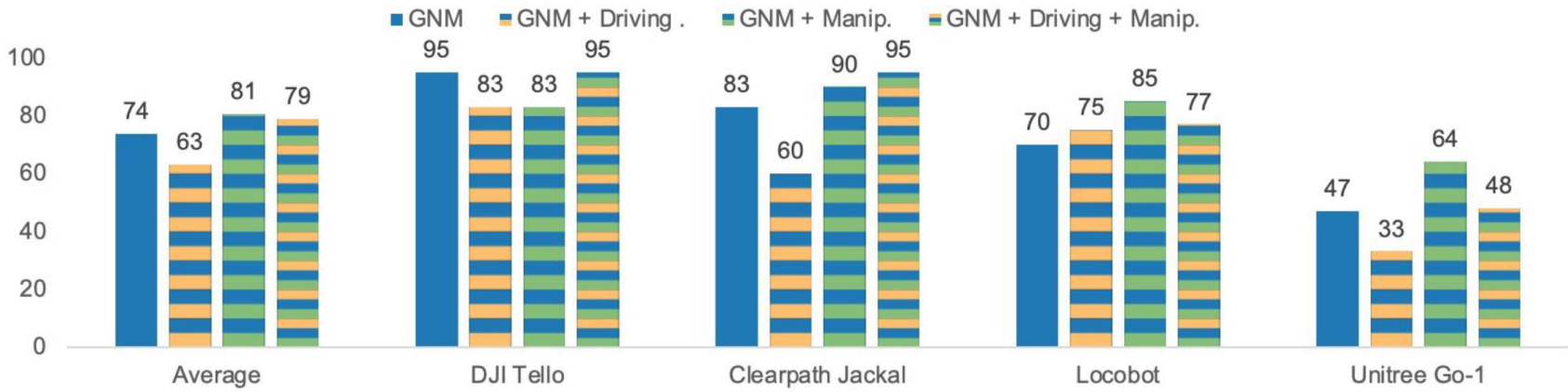
Multi-task Multi-embodiment models ?

[Yang et al., 24]

- Multi-embodiment benefits
  - Clear benefit for manipulation, in particular robustness



  - Less obvious for navigation (driving reduce navigation performances ?)

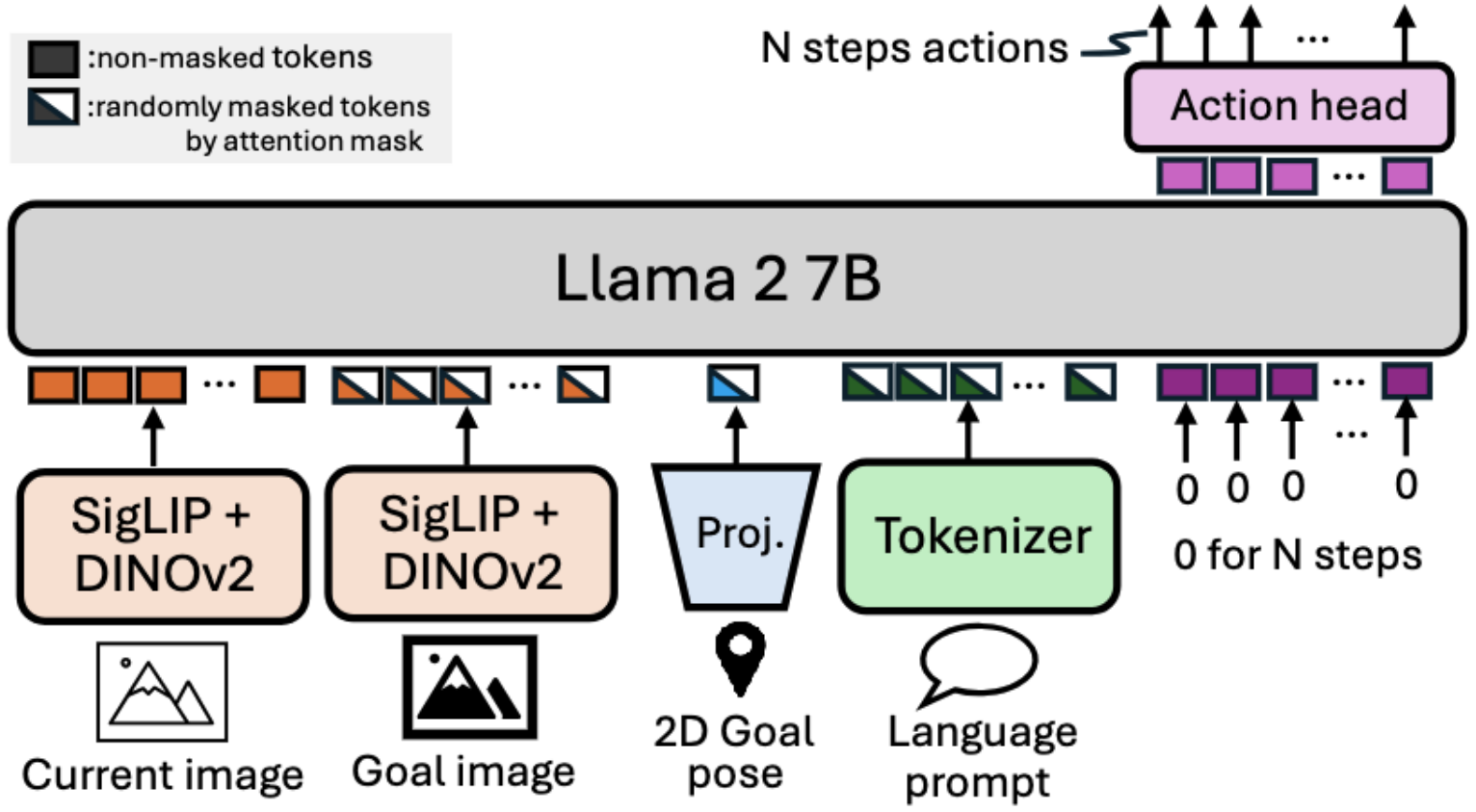


OmniVLA

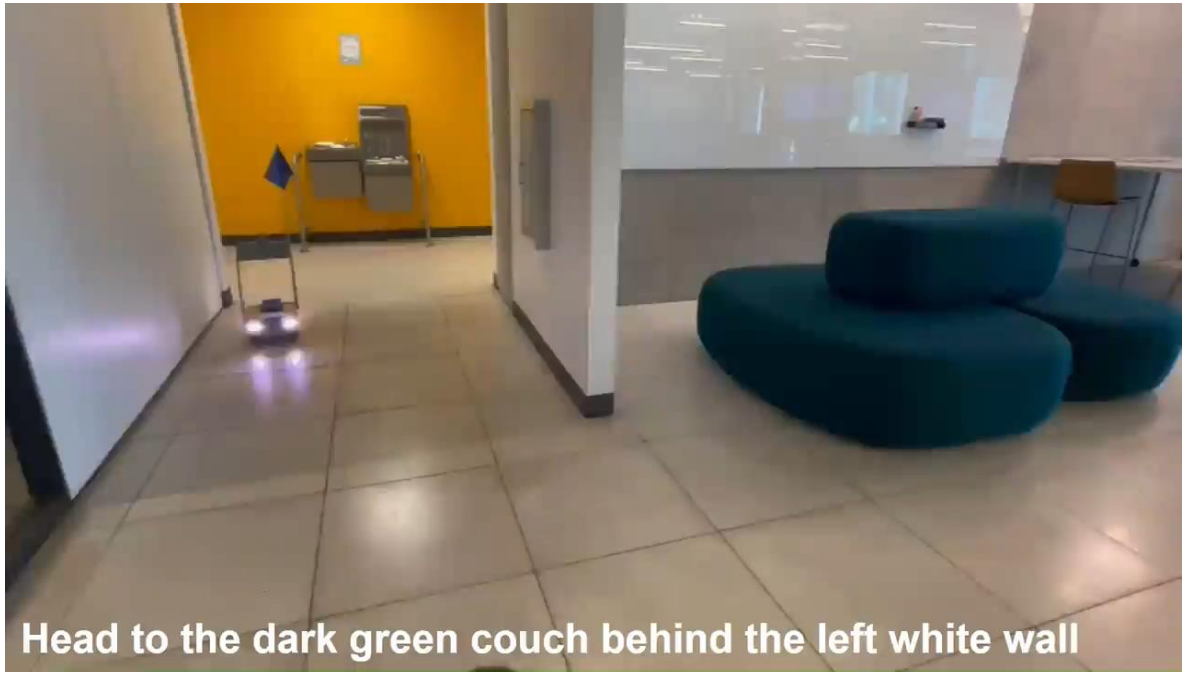
- Large scale training of a VLA for navigation
- Much larger dataset than ViNT
- Trained with several goal modalities : image, pose, prompt

OmniVLA (Berkeley, ~3B)

- Fine tuning of OpenVLA (based on Llama2) + diffusion policy
- Random masking to learn the 3 goal conditioning



OmniVLA



text goal

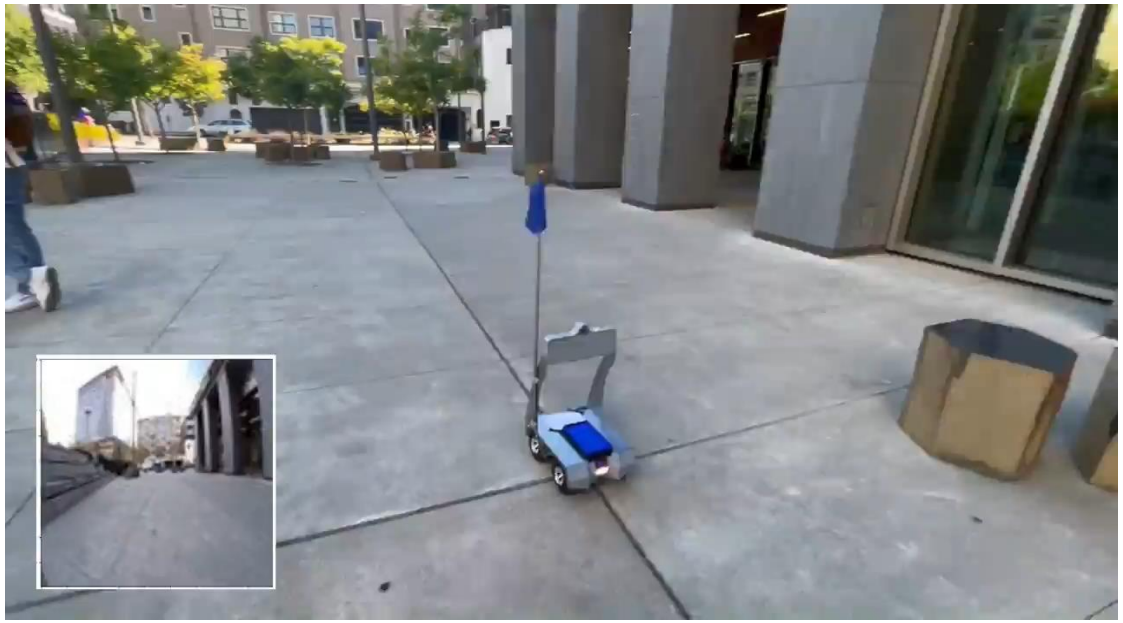
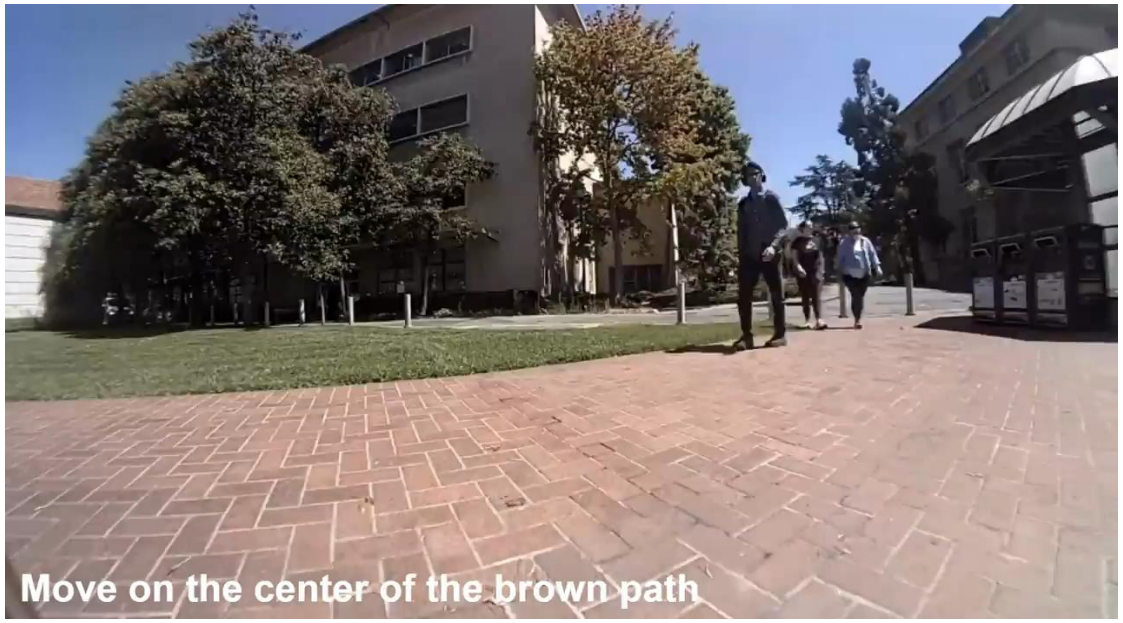


Image goal



2D + text goal

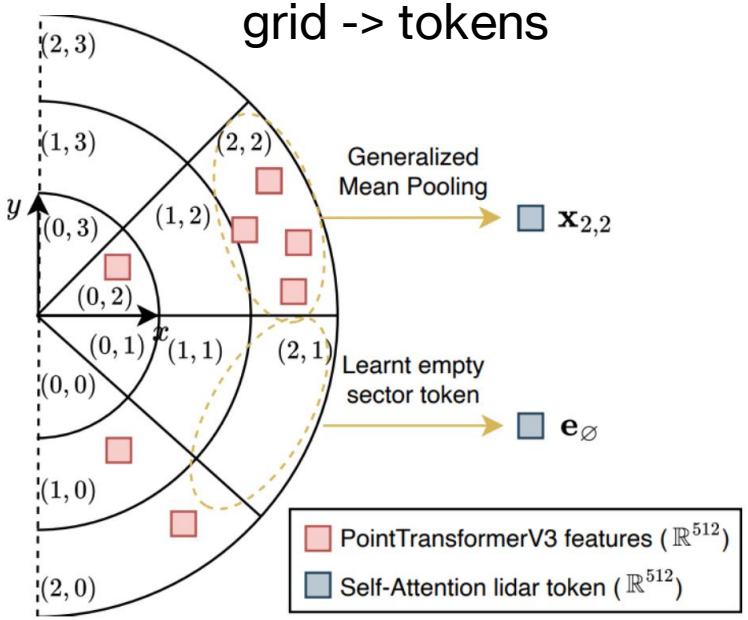
ViLiNT (AMIAD, ~31M)

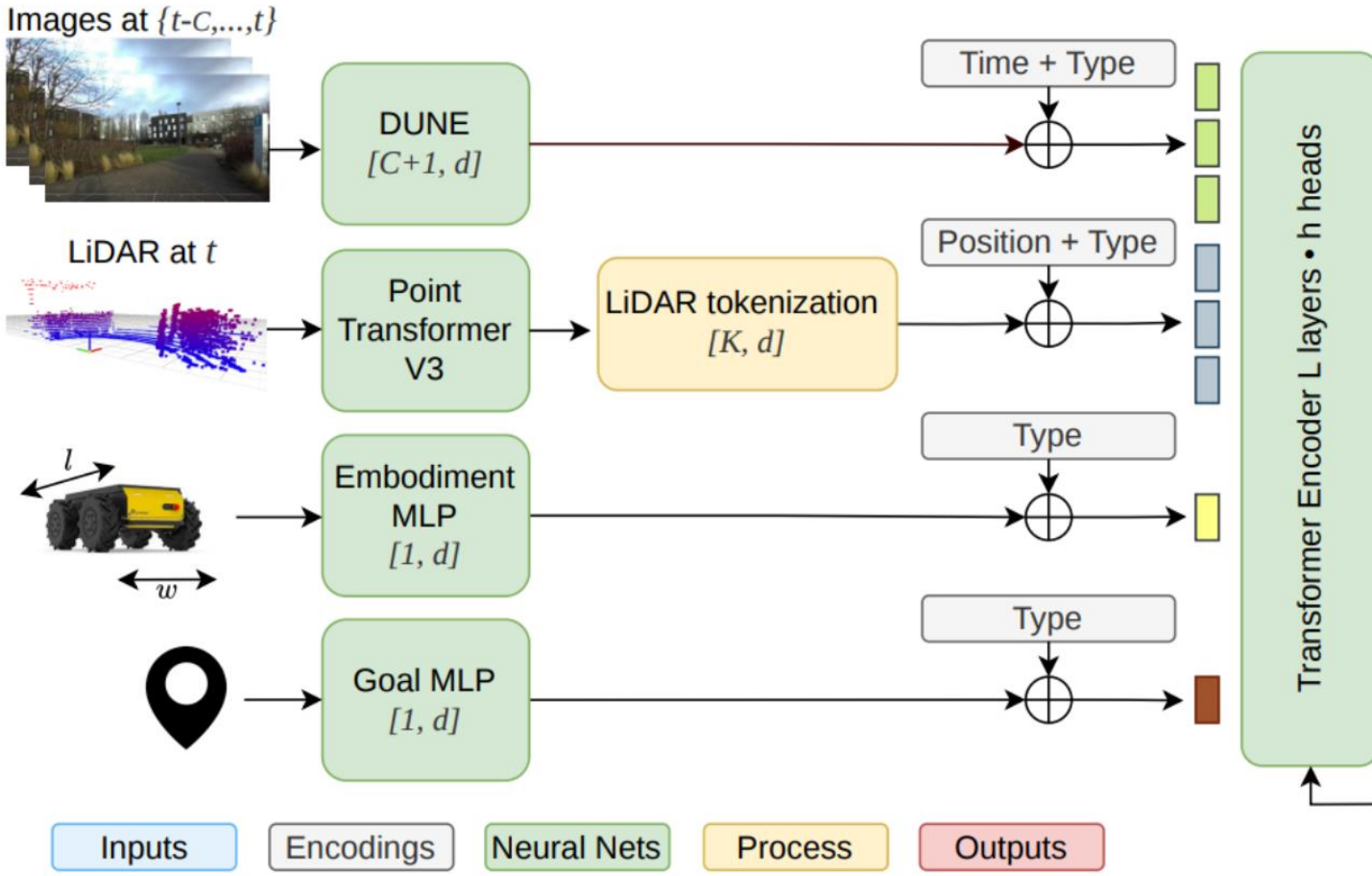
- Extension of Visual Navigation Transformer
  - Based on Transformer encoder (image + 2Dgoal) + diffusion policy head
- Added LiDAR as input for finer obstacle avoidance
  - Specific polar tokenization
  - Predict clearance around path
- Added size information
  - To generate robot specific trajectories
- Select collision free trajectory at run time
  - Using clearance prediction for ranking
  - MPC / MPPI style

With Louis Dezons (PhD), Quentin Picard (PostDoc), François Goulette (ENSTA)

ViLiNT

- Model LiDAR inputs
  - Features from pre-trained point cloud processing network (PtTransfv3)
  - Pooling of features in a cylindrical voxel grid -> tokens

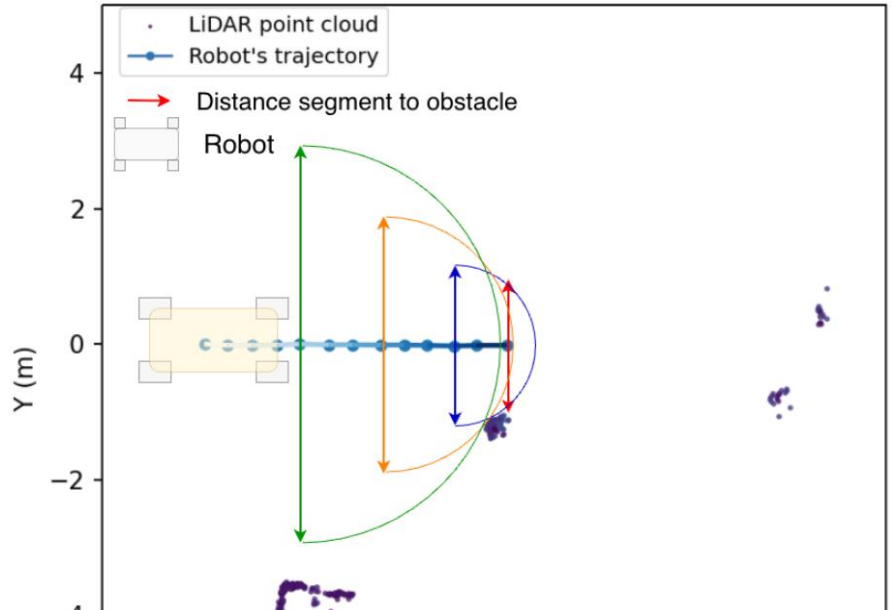


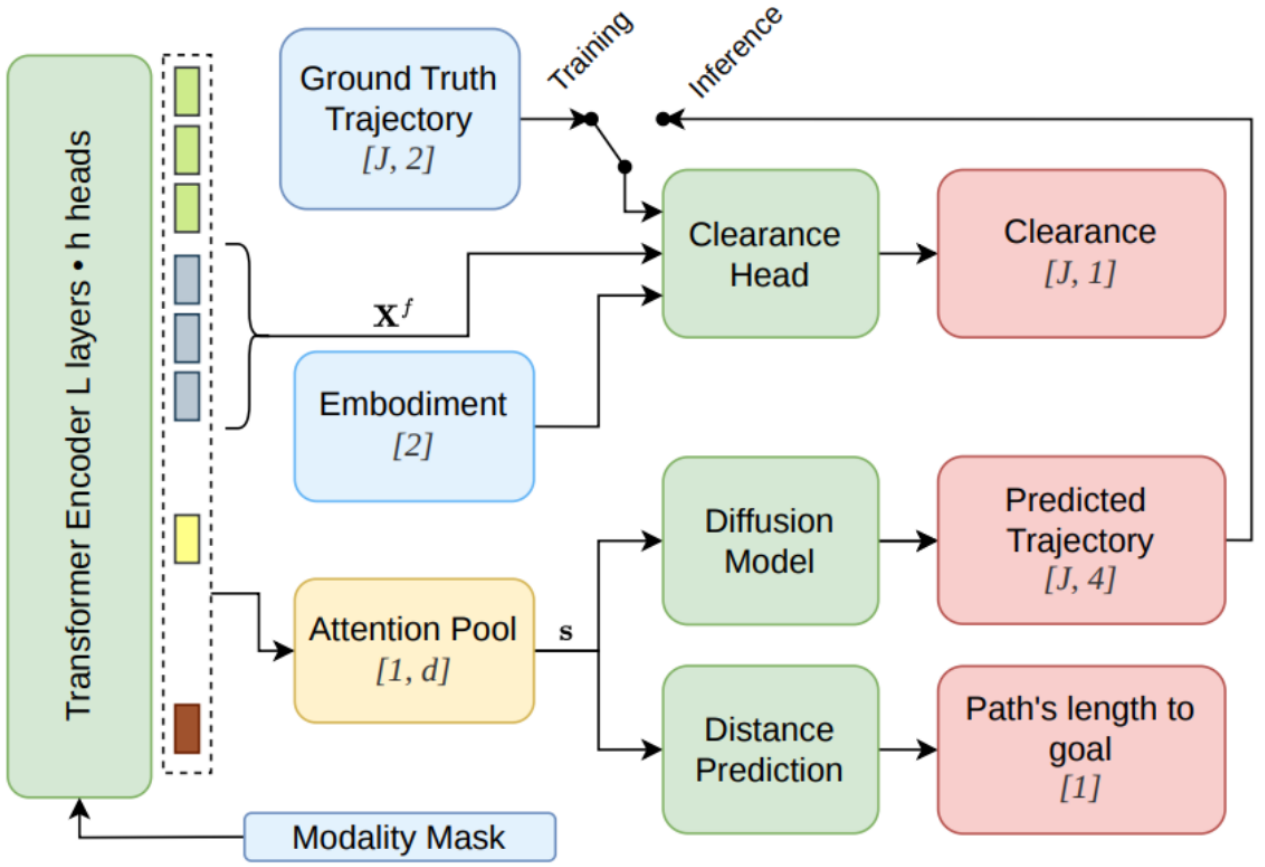


  - Add a position + type encoding

ViLiNT

- Model output
  - Diffusion head generates candidate trajectories
  - Clearance head estimate per waypoint free space around trajectories using LiDAR
  - Clearance computed as min (robot size, distance to closest point not on the ground)

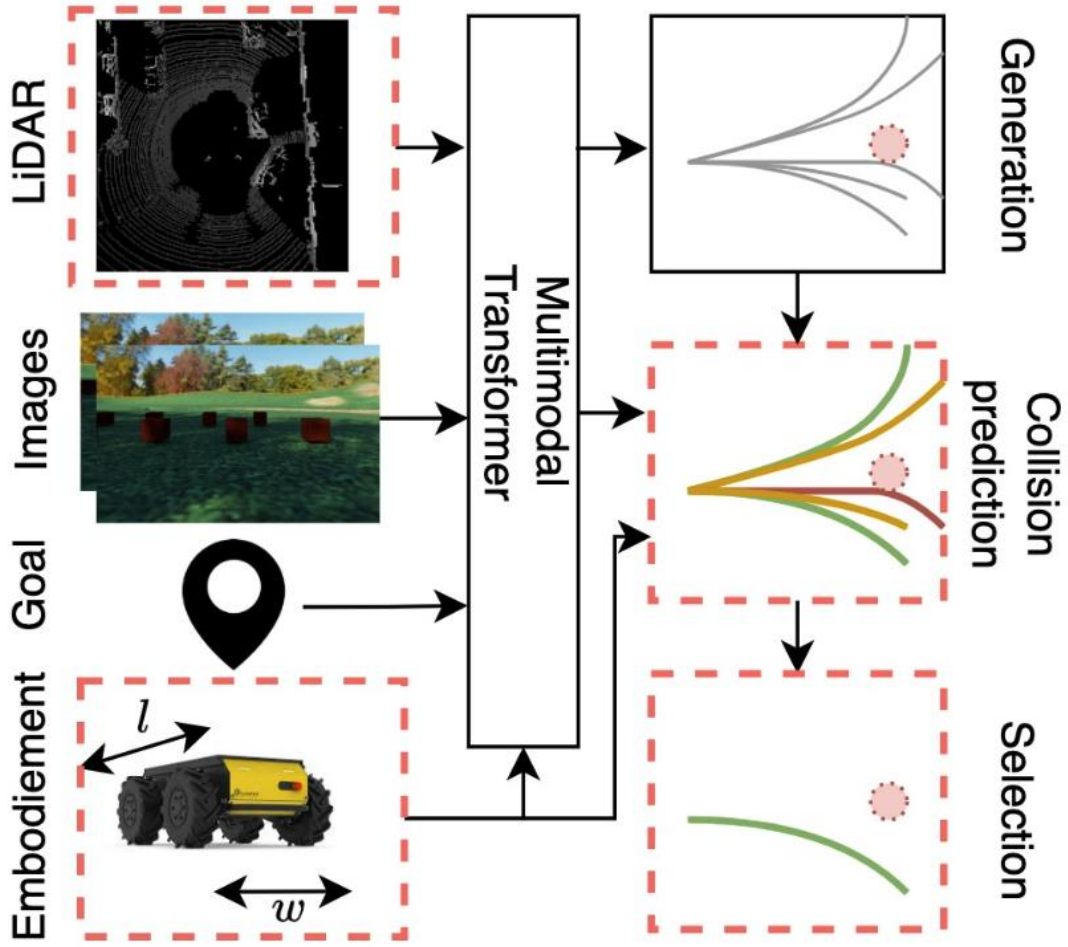




Takes into account "obstacles" that are traversed in the dataset (e.g., grass, bushes…)

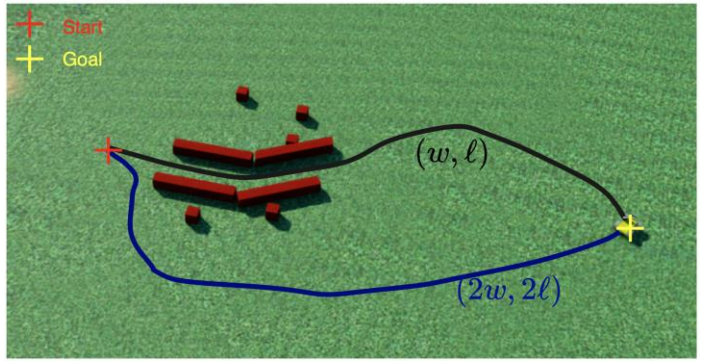
ViLiNT

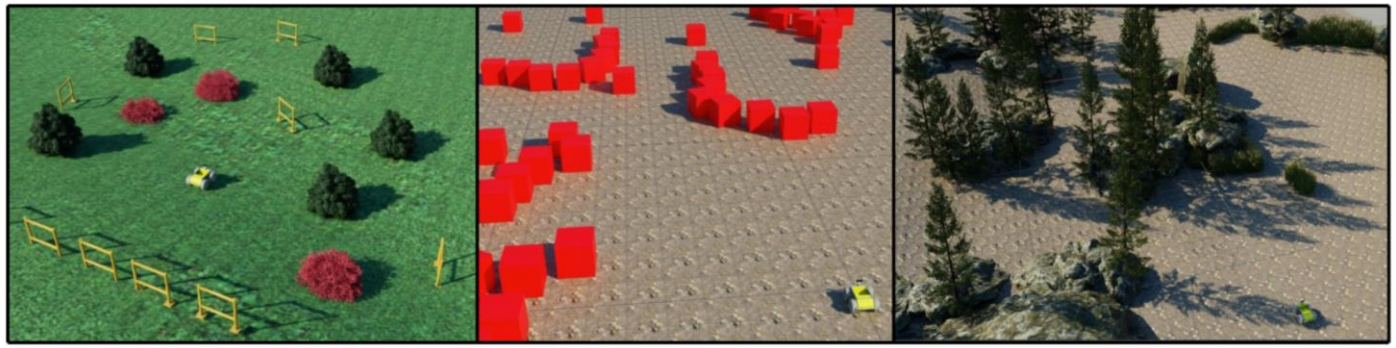
- Inference
  - Generate N goal-conditioned trajectories
  - Estimate ratio clearance / robot size
  - Follow the trajectory above threshold (3) that goes the closest to the goal
  - If no trajectory above threshold
    - Follow the one with highest ratio if > 1
    - Otherwise generate N trajectories with goal masking (exploration) and follow the safest
    - Come back to goal conditioning when a trajectory is possible



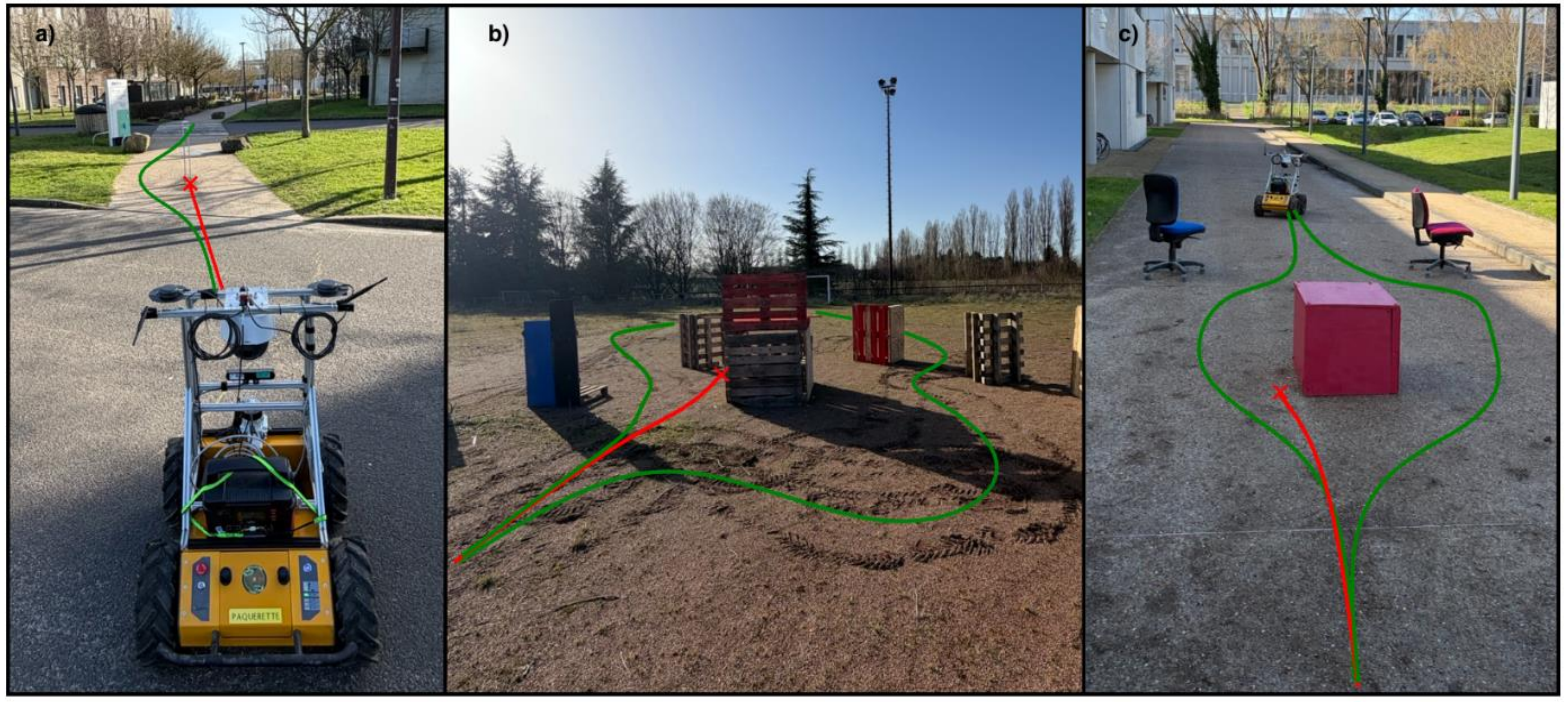
ViLiNT

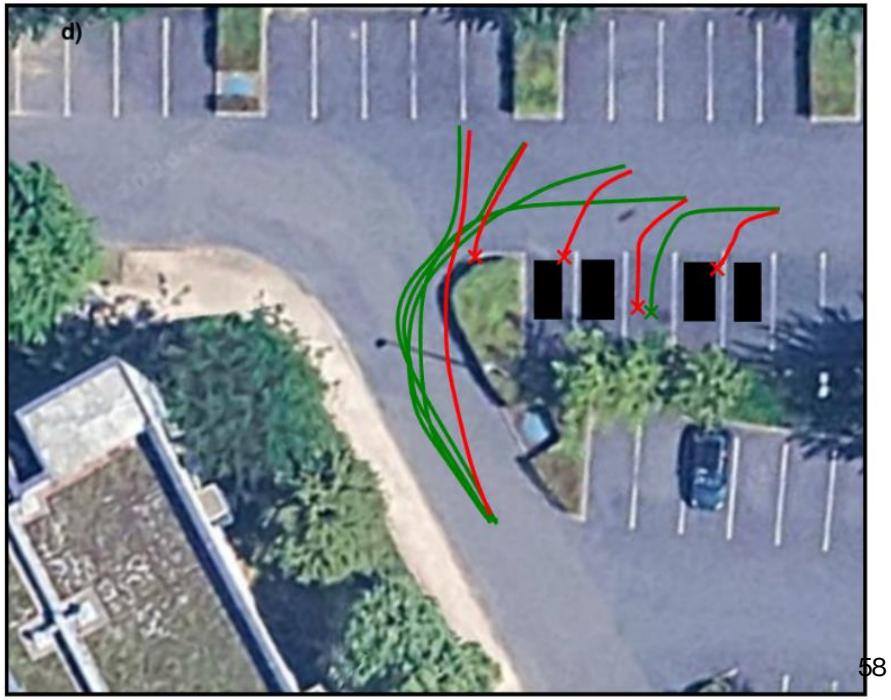
- Results





Method	Env. 1		Env. 2		Env. 3	
	SR ↑	CR ↓	SR ↑	CR ↓	SR ↑	CR ↓
NoMaD-FT [10]	0.33	0.67	0.37	0.63	0.12	0.88
NoMaD-Col	0.20	0.80	0.38	0.62	0.19	0.81
ViLiNT-LiDAR-Mask	0.17	0.83	0.37	0.63	0.20	0.80
ViLiNT-Image-Mask	0.72	0.28	**0.67**	**0.33**	0.23	0.77
ViLiNT (ours)	**0.79**	**0.21**	0.61	0.39	**0.76**	**0.24**
TEB-Elev	0.66	0.00	0.89	0.00	1.00	0.00





ViLiNT

Multimodal Embodiment Aware Navigation Transformer

IROS 2026 video submission

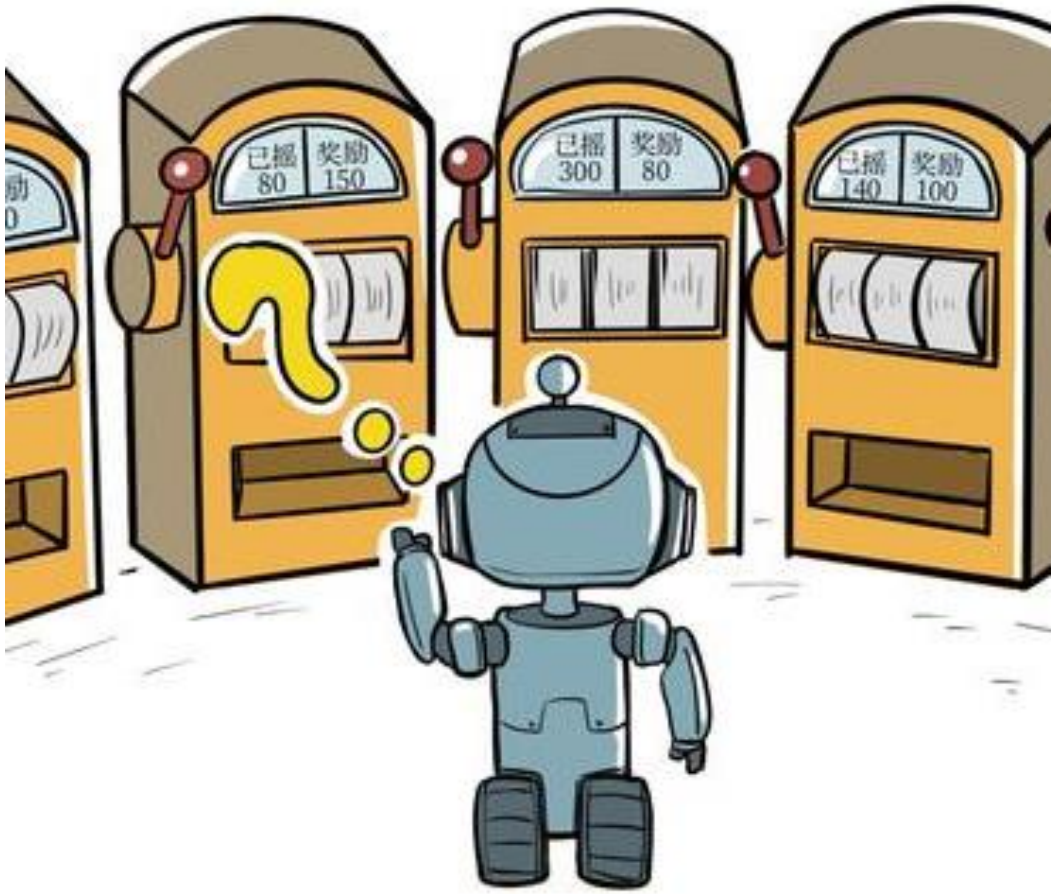
In this video we show our model **ViLiNT** deployed on a Clearpath Husky equipped with a Nvidia AGX orin embedded computer

- The paths and obstacles we avoid in the video are not in the training data specifically
- The goals are set 15 to 30 meters away on the initial robot's frame x axis

REINFORCEMENT LEARNING

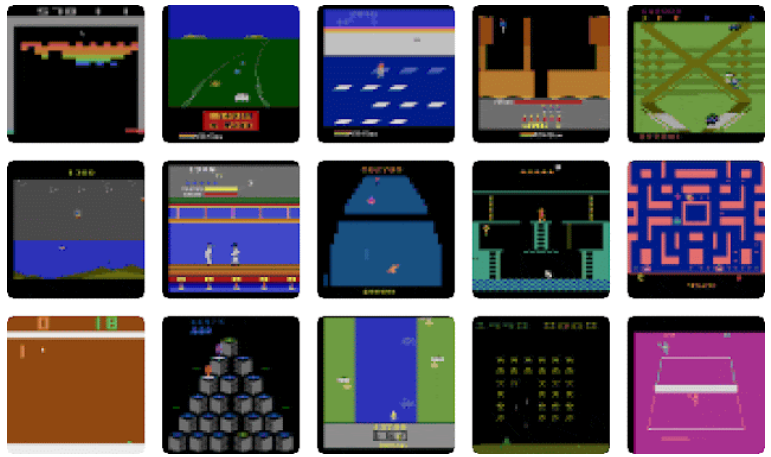
Reinforcement learning

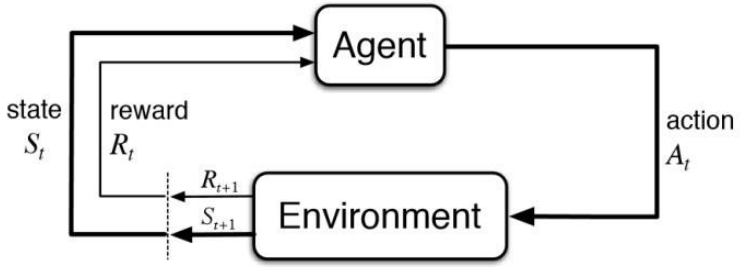
- Sequential decision problems
- No ‘output’ variable as in supervised learning, but a measure of answer/behaviour quality (**reward**)
- Simplest form : **bandit**
  - choose between alternative with different rewards
  - Find the best strategy to minimize regret (i.e., **explore/exploit** dilema)

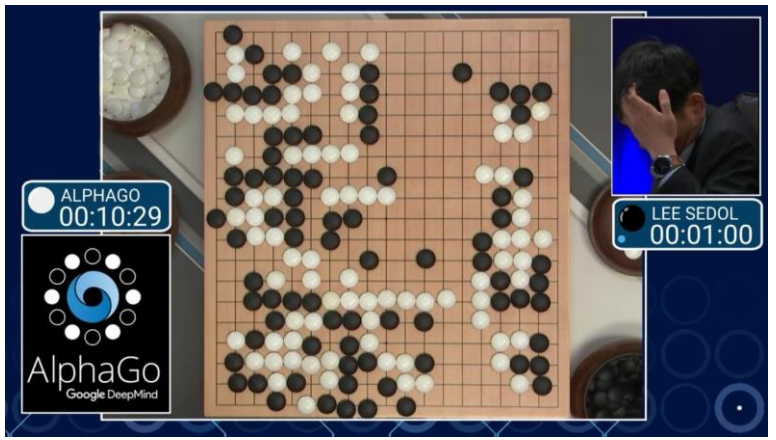


Reinforcement learning

- Problems with evolving state (Markov Decision Processes)
- Rewards can be delayed (e.g., win a game)
- Find a ‘policy’ mapping ‘state’ to ‘action’ maximizing sum of rewards
- Achieved brilliant performances in games (Atari, Go, ...)
- Conceptually well adapted to robotics







Value-Based Reinforcement Learning

Markov Decision Process, Policy & RL objective:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$$

a = π (s)

$$\pi^* \in \arg\max_{\pi} \mathop{\mathbb{E}}_{T \sim \pi, P^\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

$$T = (s, a, r, s+)$$

$$V^\pi(s) = \mathop{\mathbb{E}}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s; \right]$$

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s; a_t = a \right]$$

Bellman evaluation operator:

$$Q^\pi(s,a) = \mathcal{T}^\pi Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s+ \sim P^\pi(s+|s)} Q(s+, \pi)$$

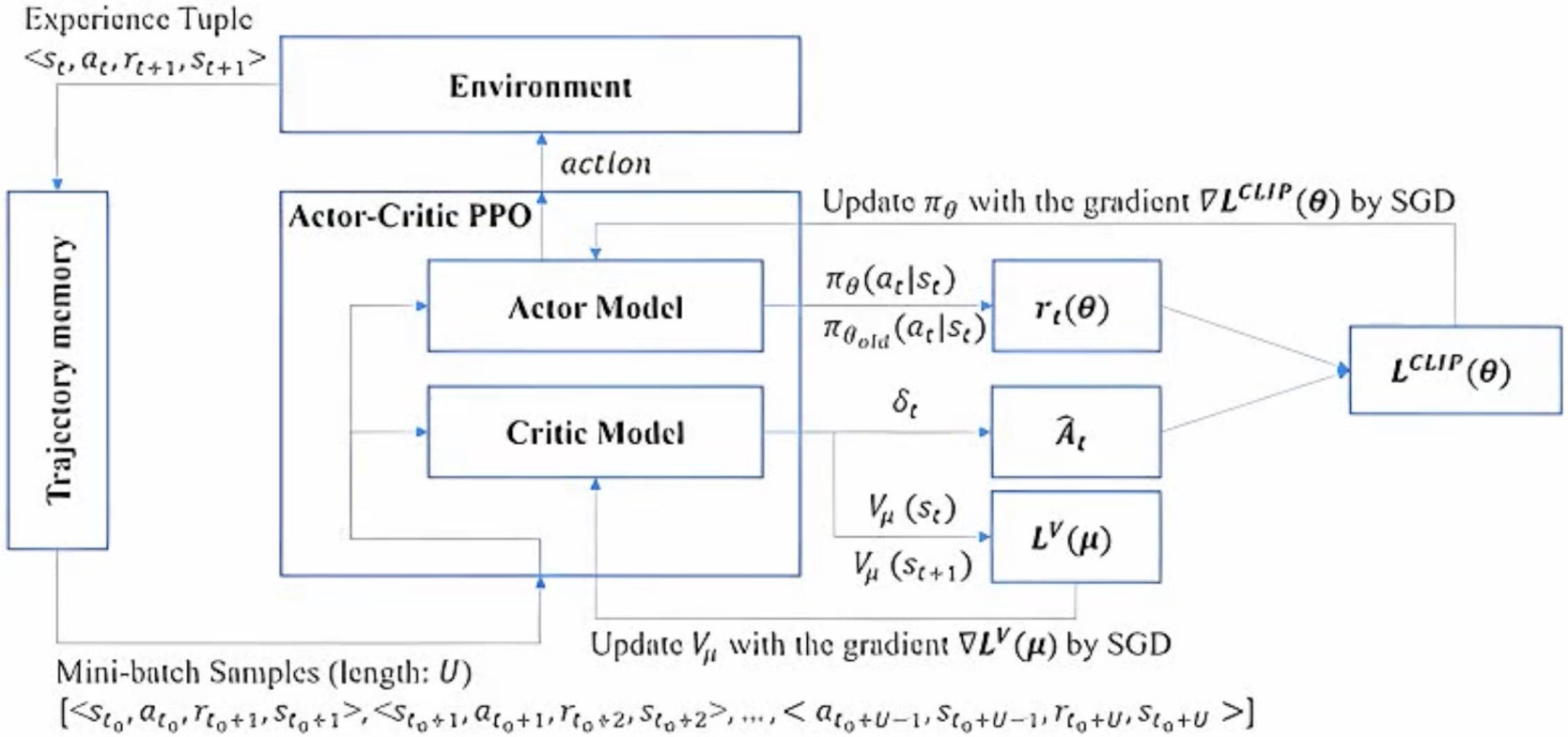
Bellman error and residuals can be used to train a model (TD-Learning)

$$(\mathcal{B}^\pi Q)(s,a) = Q^\pi(s,a) - \mathcal{T}^\pi Q^\pi(s,a)$$

$$\mathcal{L}_{\mathrm{BR}} = \mathbb{E}_T \left[||(\mathcal{B}^\pi Q)(s,a)||^2 \right]$$

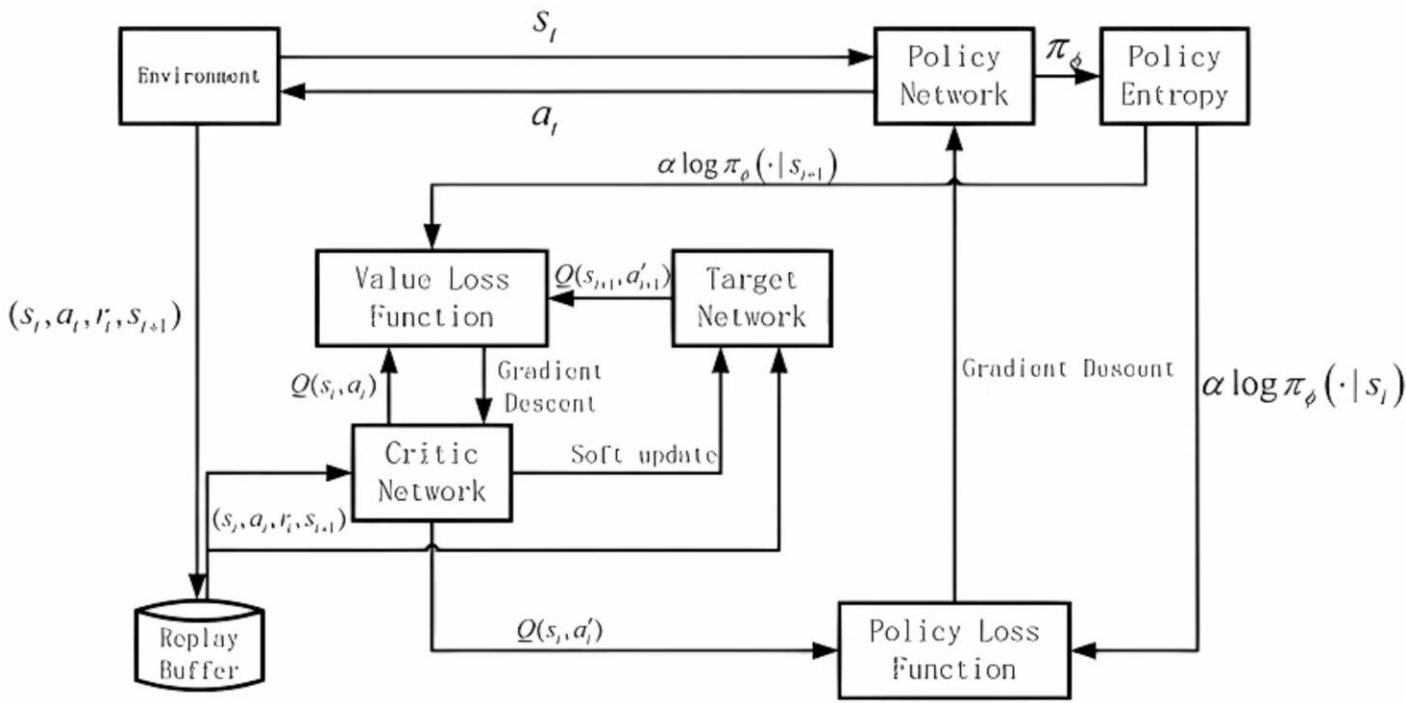
Some popular RL algorithms : PPO

- **Proximal Policy Iteration**
  - Direct policy optimization
  - Critic (value function) learned to compute policy gradient
  - Constraint to limit the policy modification magnitude
  - On-policy algorithm -> stable but slow



Some popular RL algorithms : SAC

- **Soft Actor-Critic**
  - Learn Critic / Locally optimize policy
  - Off policy algorithm (replay buffer)
  - Stochastic policy -> maximize reward and entropy
  - More sample efficient than PPO
  - Maybe less stable



Reinforcement Learning and Robotics

Main application

- Learning reactive controller
- Can learn more complex behavior with more complex architectures (memory, maps, ...)

- Potentially better than imitation learning
  - Don't require expert demonstrations
  - Directly optimize performances, thus not limited by demonstrator performances
- Can be coupled with imitation
  - Pretraining using imitation, then RL
  - Offline RL : use RL on recorded trajectories, balance optimisation/imitation

Reinforcement Learning and Robotics

Robotics constraints

- Data are expensive (vs games), robots are slow, break easily
- Search (behaviour) space are huge, enough (iid) data difficult to gather
- Ideally should be extended too incremental learning, multi-tasks learning ...

How to improve efficiency of reinforcement learning on real robots ?

- Learn compact representation to accelerate learning in real life
- Use auxiliary tasks to accelerate learning in real life
- Learn in simulation and transfer to real life

All of the above ?

LEARNING STATE REPRESENTATION

States ?

Often, robot controllers require simple, ‘high-level’, low dimension inputs (the ‘state’ of the robot/world)

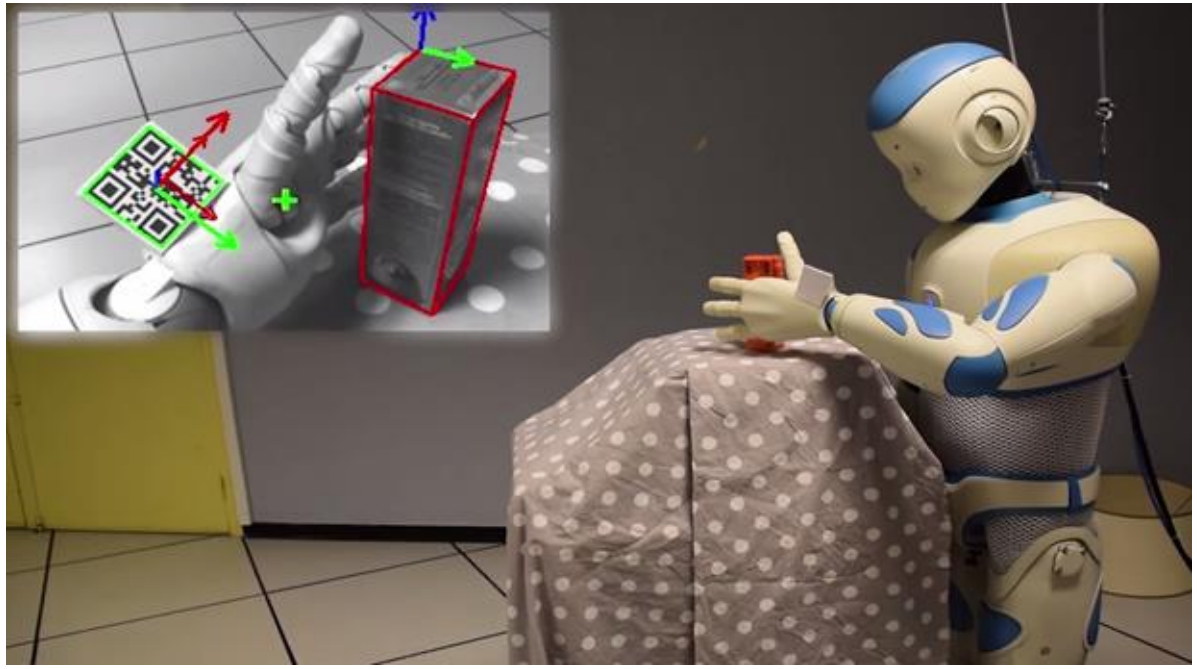
- E.g., grasping: object position, gripper position

  driving: road direction, obstacle positions, ...

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$$

Vision based control requires filtering to get this information

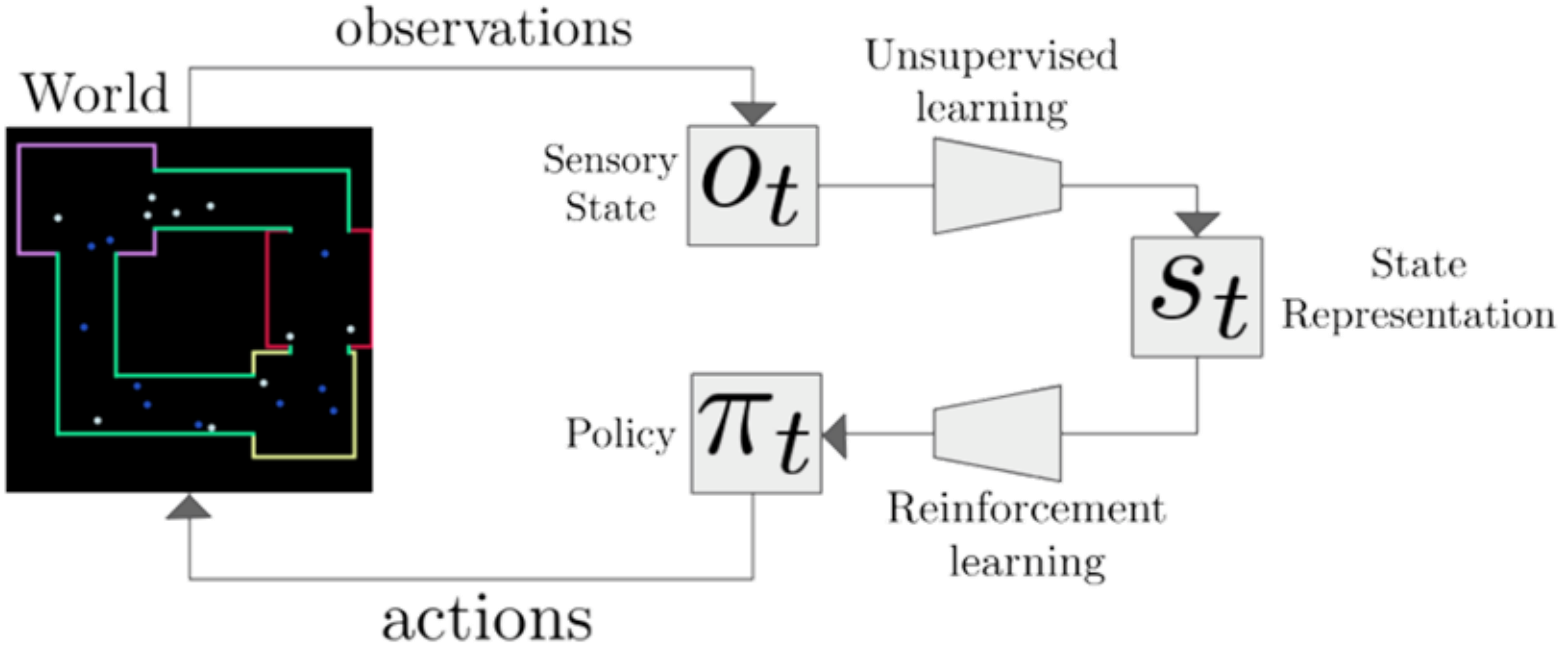
- Many solutions, often hand-crafted, or task specific
- Foundation models can be useful



State representation learning (SRL)

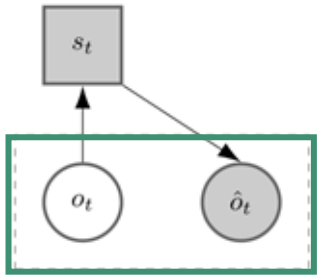
Finding the right vision-based representation without supervision

- Deep learning makes it possible to learn useful representation
- Representations can be specialized for robotics/control, capture scene geometry, dynamics, and intrinsic physical properties
- Exploit observations / actions / rewards sequences, avoid human supervision

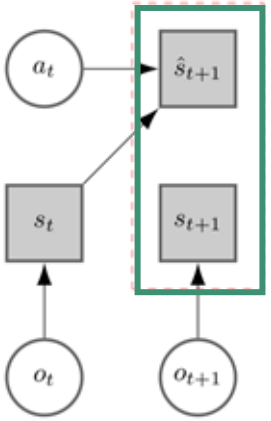


SRL approaches

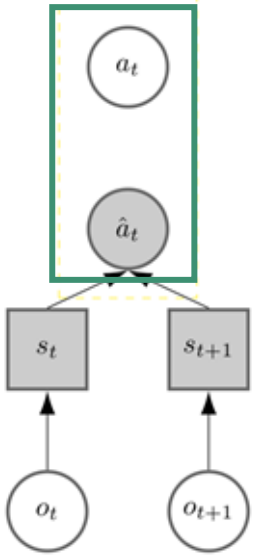
- Learning state representation using self-supervision
  - Several objectives can be exploited without human labelling
  - Objectives can be combined



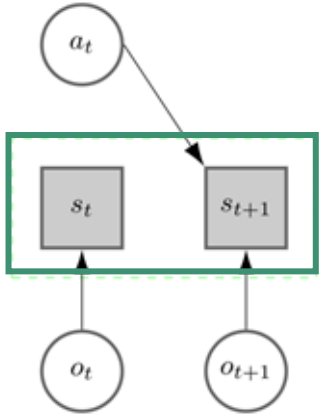
Autoencoders



Forward Models

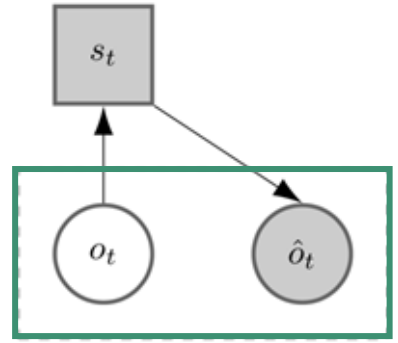


Inverse models



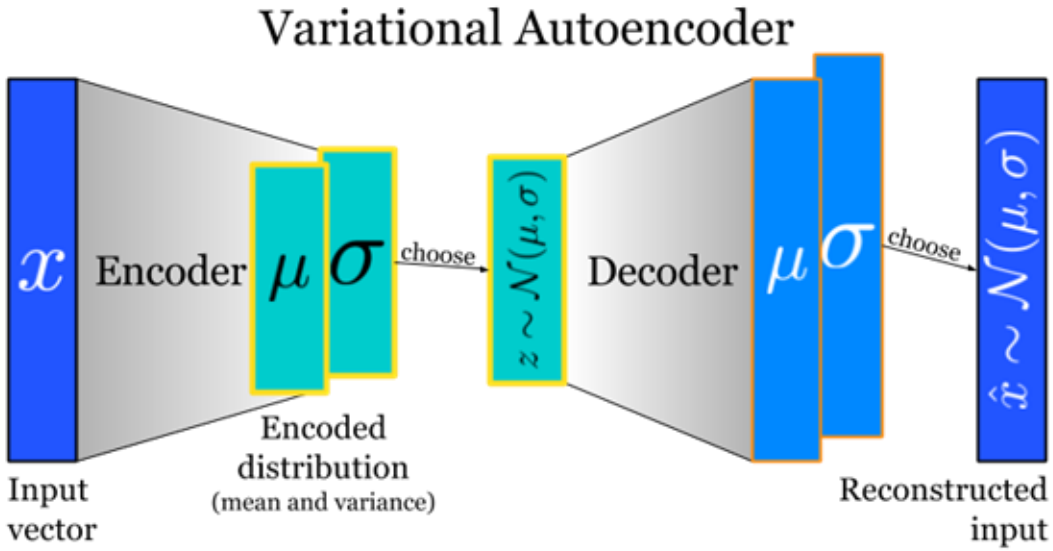
Priors

Reconstructing the observation



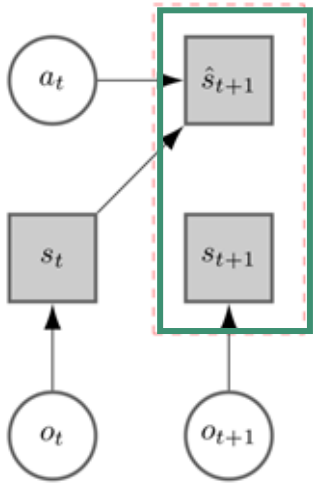
Train a state that is sufficient for reconstructing the input observation

- AE, DAE, VAE, ...
- (Bi)GANs, ...



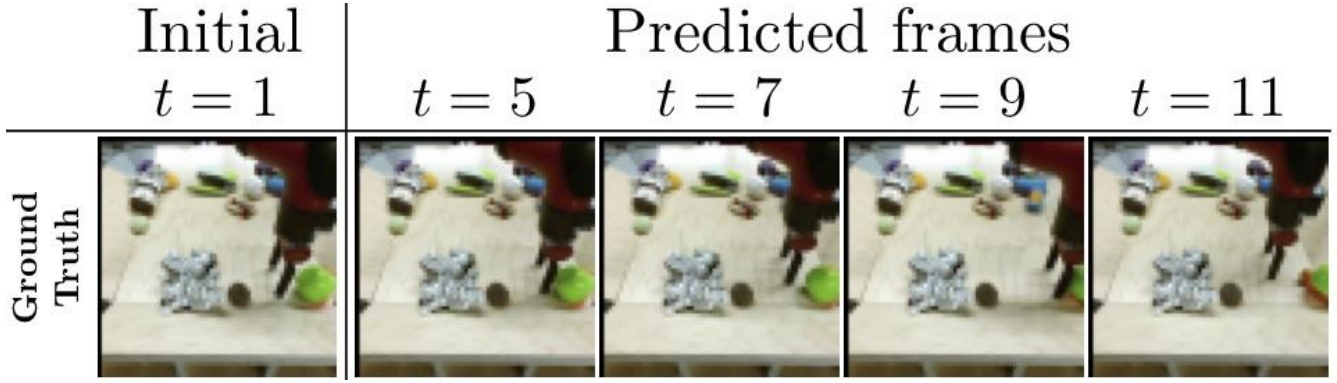
Downside: sensitive to irrelevant variations (wrt actions)

Forward models



Find state from which it is easy to predict next state

- Additional constraints to avoid fixed representations
  (AE, triplet loss...)
- Impose constraints on forward model (e.g., linear model)



Naturally discard irrelevant features

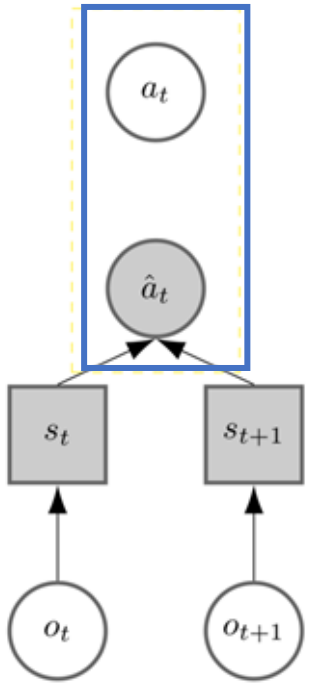
Model may be useful in model based RL / MPC / planning

Inverse models

Find a state sufficient to recover action from 2 observations

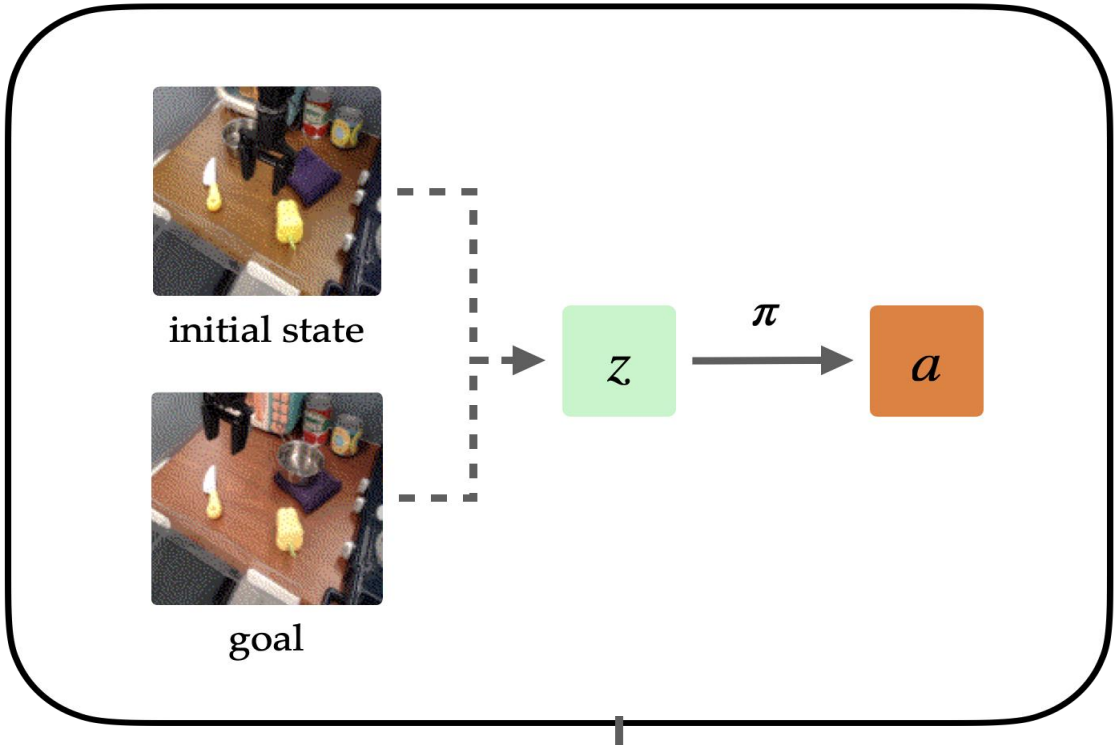
- Impose constraints on model (e.g., linear model)

Focus on states that can be controlled

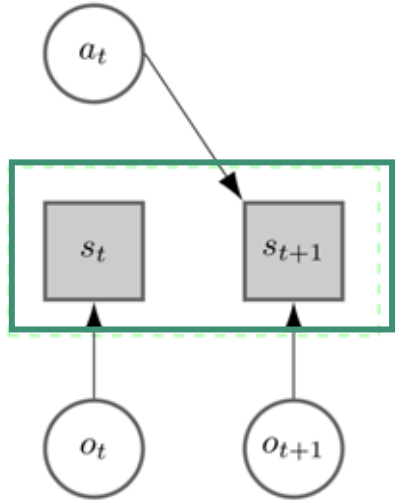


Useful for a direct control model

- E.g., goal conditioned policies
- a = π(s,g)



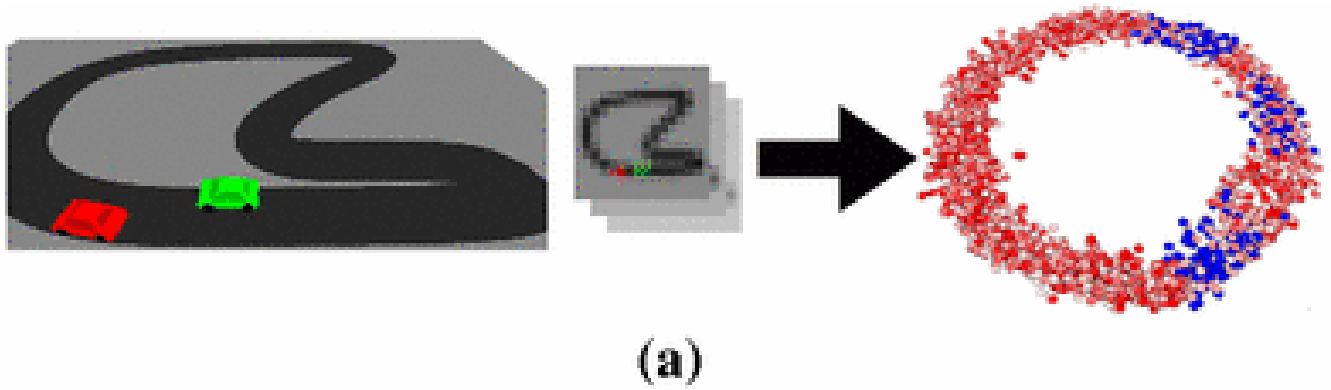
Prior models

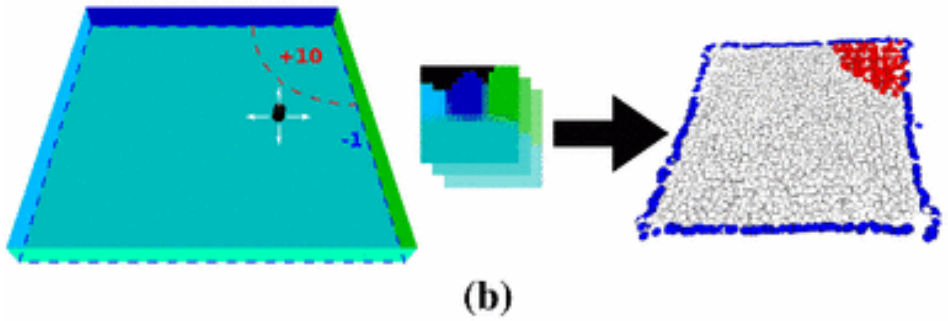


Encode high-level constraints on the states

- Temporal continuity
- Controllability
- Inertia
- etc....

May exploit rewards





Robotic Priors

Use *a priori* knowledge to learn representations relevant to the task

- **Temporal coherence Prior:** *Two states close to each other in time are also close to each other in the state representation space.*

$$L_{Temp}(D, \hat{\phi}) = \mathrm{E}[\| \Delta \hat{s}_t \|^2] \,, \tag{1}$$

- **Proportionality Prior:** *Two identical actions should result in two proportional magnitude state variations.*

$$L_{Prop}(D, \hat{\phi}) = \mathrm{E}[(\| \Delta \hat{s}_{t_2} \| - \| \Delta \hat{s}_{t_1} \|)^2 | a_{t_1} = a_{t_2}] \,, \tag{2}$$

- **Repeatability Prior:** *Two identical actions applied at similar states should provide similar state variations, not only in magnitude but also in direction.*

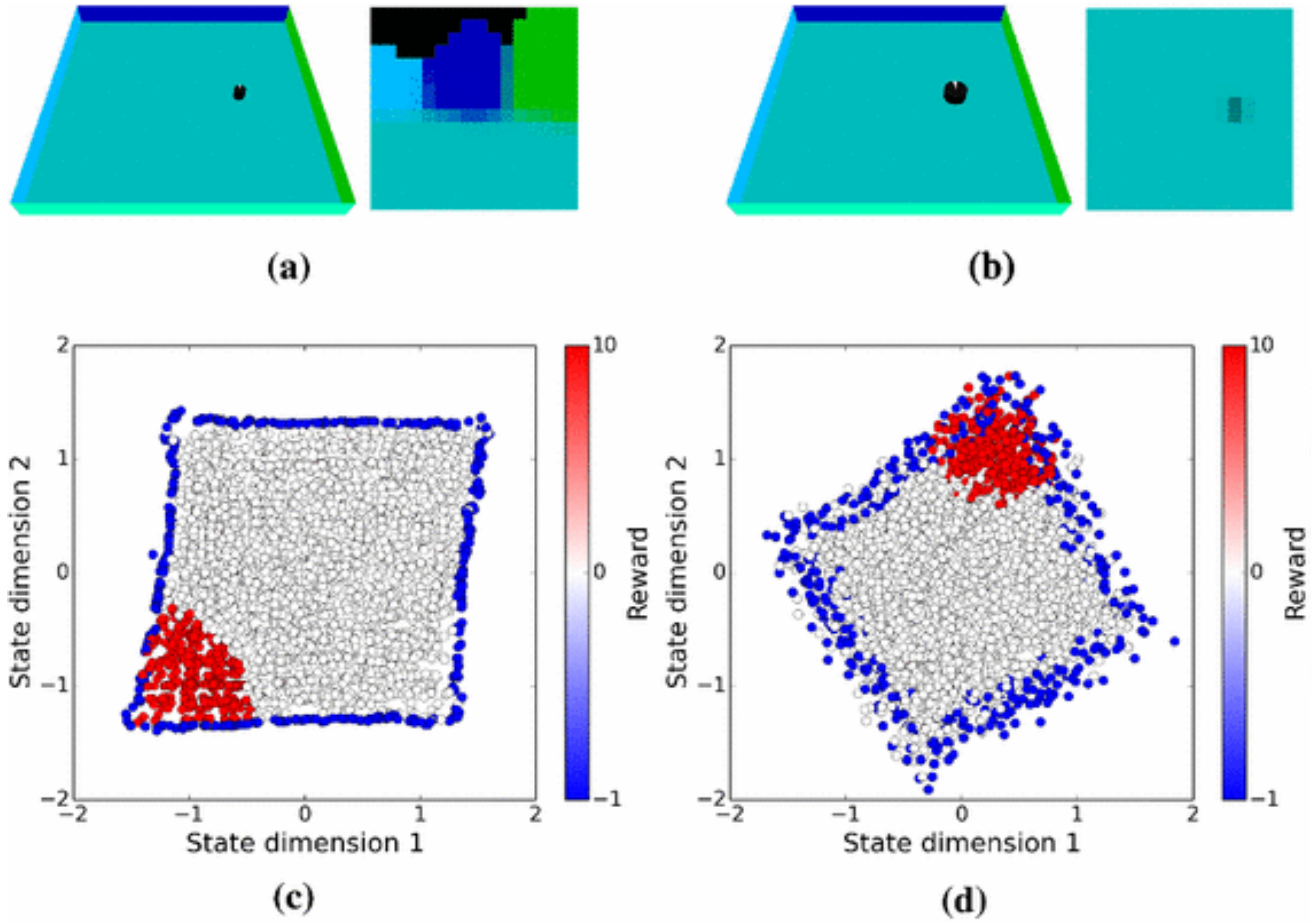
$$L_{Rep}(D, \hat{\phi}) = \mathrm{E}[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \| \Delta \hat{s}_{t_2} - \Delta \hat{s}_{t_1} \|^2 | \, a_{t_1} = a_{t_2}] \,, \tag{3}$$

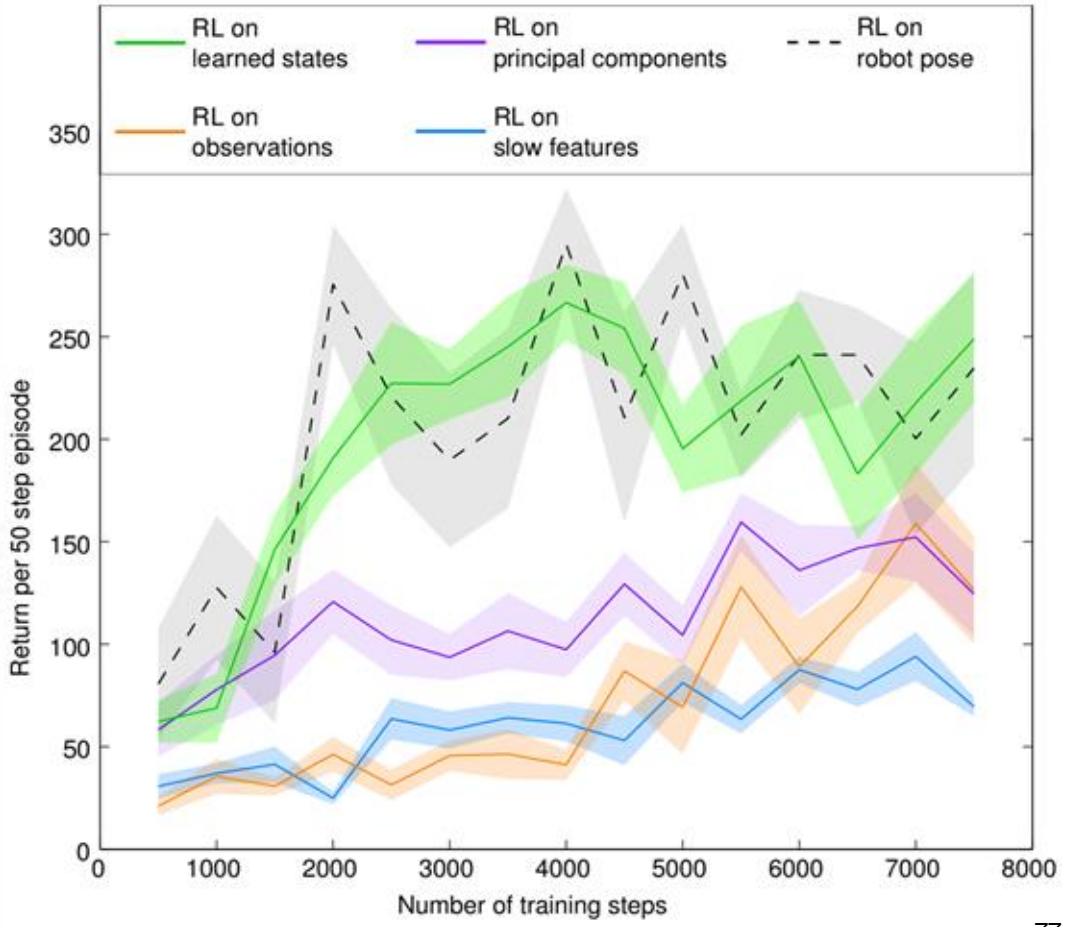
- **Causality Prior:** *If two states on which the same action is applied give two different rewards, they should not be close to each other in the state representation space.*

$$L_{Caus}(D, \hat{\phi}) = \mathrm{E}[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \mid a_{t_1} = a_{t_2}, r_{t_1+1} \neq r_{t_2+1}] \,, \tag{4}$$

Robotic Priors

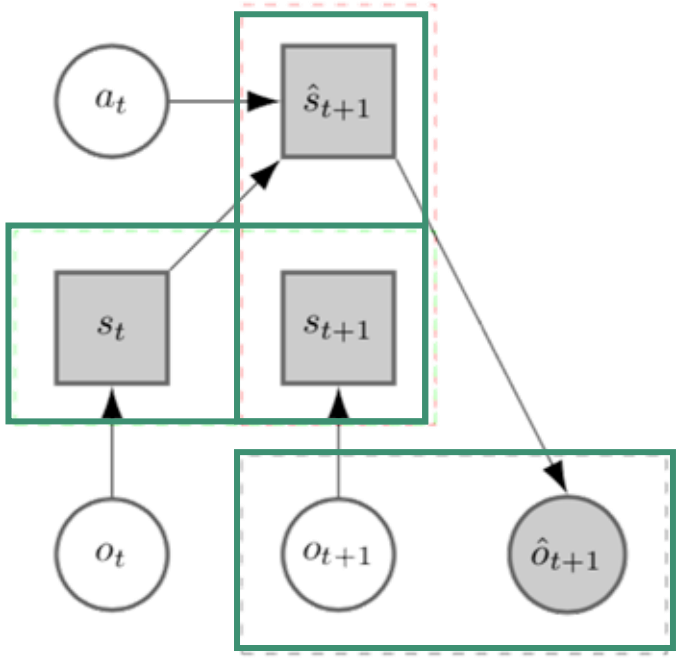
Learned state can make RL faster





Mixing objectives

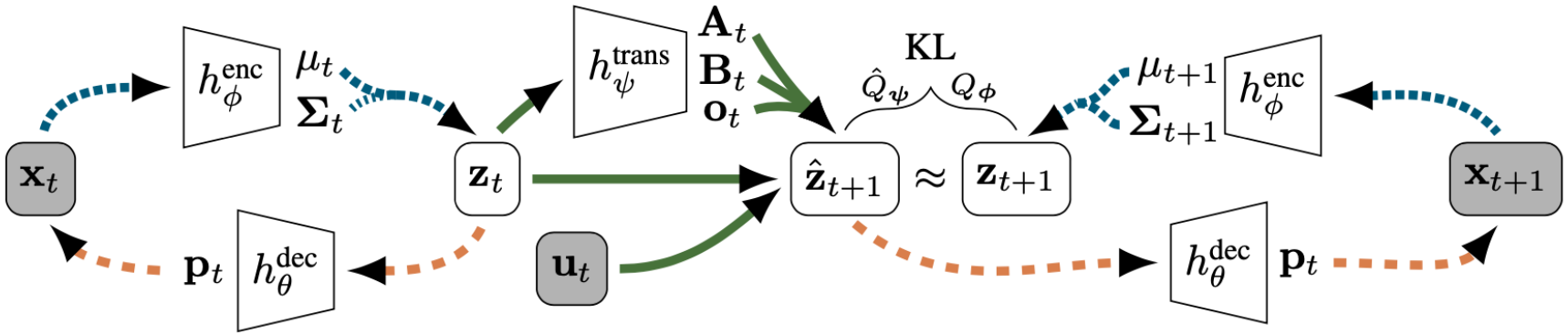
Integrating several approaches



Mixing objectives

E.g., Embed to control

- Watter, M., Springenberg, J., Boedecker, J., & Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. Advances in neural information processing systems, 28.



- Reconstruct observation using VAE
- Learn a locally linear forward model
- Exploit this forward model in optimal control setting

Mixing objectives

Embed to control

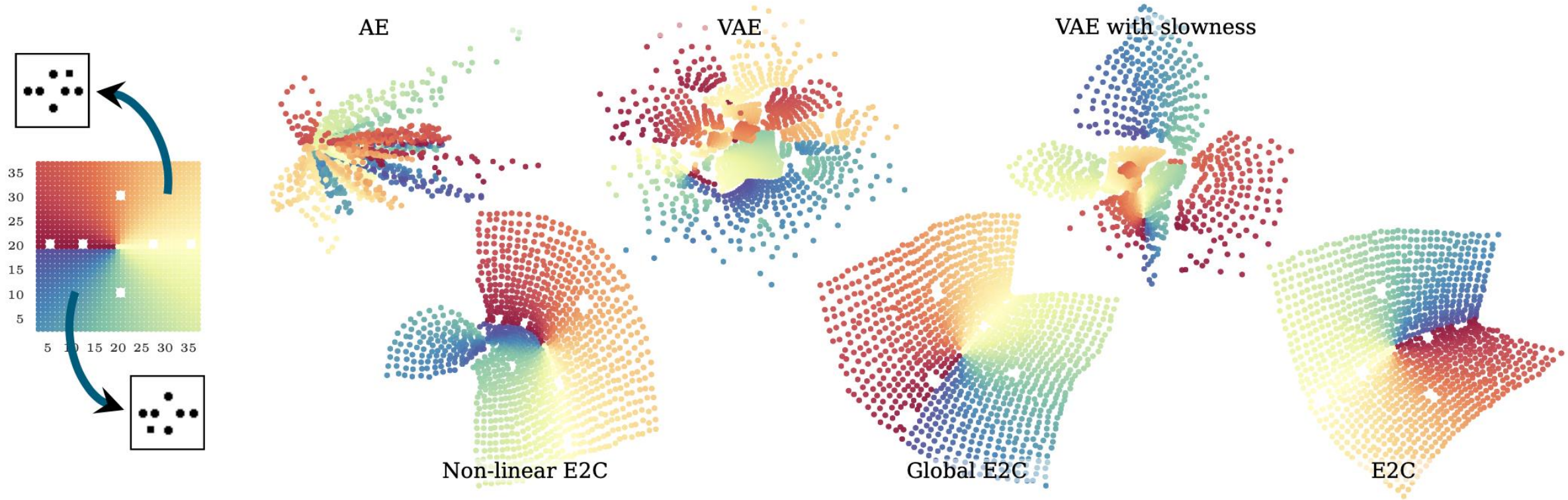
- Much more efficient than simple autoencoders

Algorithm	State Loss $\log p(x_t \mid \hat{x}_t)$	Next State Loss $\log p(x_{t+1} \mid \hat{x}_t, u_t)$	Trajectory Cost Latent	Trajectory Cost Real	Success percent
Planar System					
AE†	11.5 ± 97.8	3538.9 ± 1395.2	1325.6 ± 81.2	273.3 ± 16.4	0 %
VAE†	3.6 ± 18.9	652.1 ± 930.6	43.1 ± 20.8	91.3 ± 16.4	0 %
VAE + slowness†	10.5 ± 22.8	104.3 ± 235.8	47.1 ± 20.5	89.1 ± 16.4	0 %
Non-linear E2C	8.3 ± 5.5	11.3 ± 10.1	19.8 ± 9.8	42.3 ± 16.4	96.6 %
Global E2C	$\mathbf{6.9 \pm 3.2}$	$\mathbf{9.3 \pm 4.6}$	12.5 ± 3.9	27.3 ± 9.7	**100** %
E2C	7.7 ± 2.0	9.7 ± 3.2	10.3 ± 2.8	$\mathbf{25.1 \pm 5.3}$	**100** %
Inverted Pendulum Swing-Up					
AE†	8.9 ± 100.3	13433.8 ± 6238.8	1285.9 ± 355.8	194.7 ± 44.8	0 %
VAE†	7.5 ± 47.7	8791.2 ± 17356.9	497.8 ± 129.4	237.2 ± 41.2	0 %
VAE + slowness†	26.5 ± 18.0	779.7 ± 633.3	419.5 ± 85.8	188.2 ± 43.6	0 %
E2C no latent KL	64.4 ± 32.8	87.7 ± 64.2	489.1 ± 87.5	213.2 ± 84.3	0 %
Non-linear E2C	$\mathbf{59.6 \pm 25.2}$	$\mathbf{72.6 \pm 34.5}$	313.3 ± 65.7	37.4 ± 12.4	63.33 %
Global E2C	115.5 ± 56.9	125.3 ± 62.6	628.1 ± 45.9	125.1 ± 10.7	0 %
E2C	84.0 ± 50.8	89.3 ± 42.9	275.0 ± 16.6	$\mathbf{15.4 \pm 3.4}$	**90** %

Mixing objectives

Embed to control

- Learnt representation topology very close to real environment



Mixing objectives

Very similar ideas in the "World models"

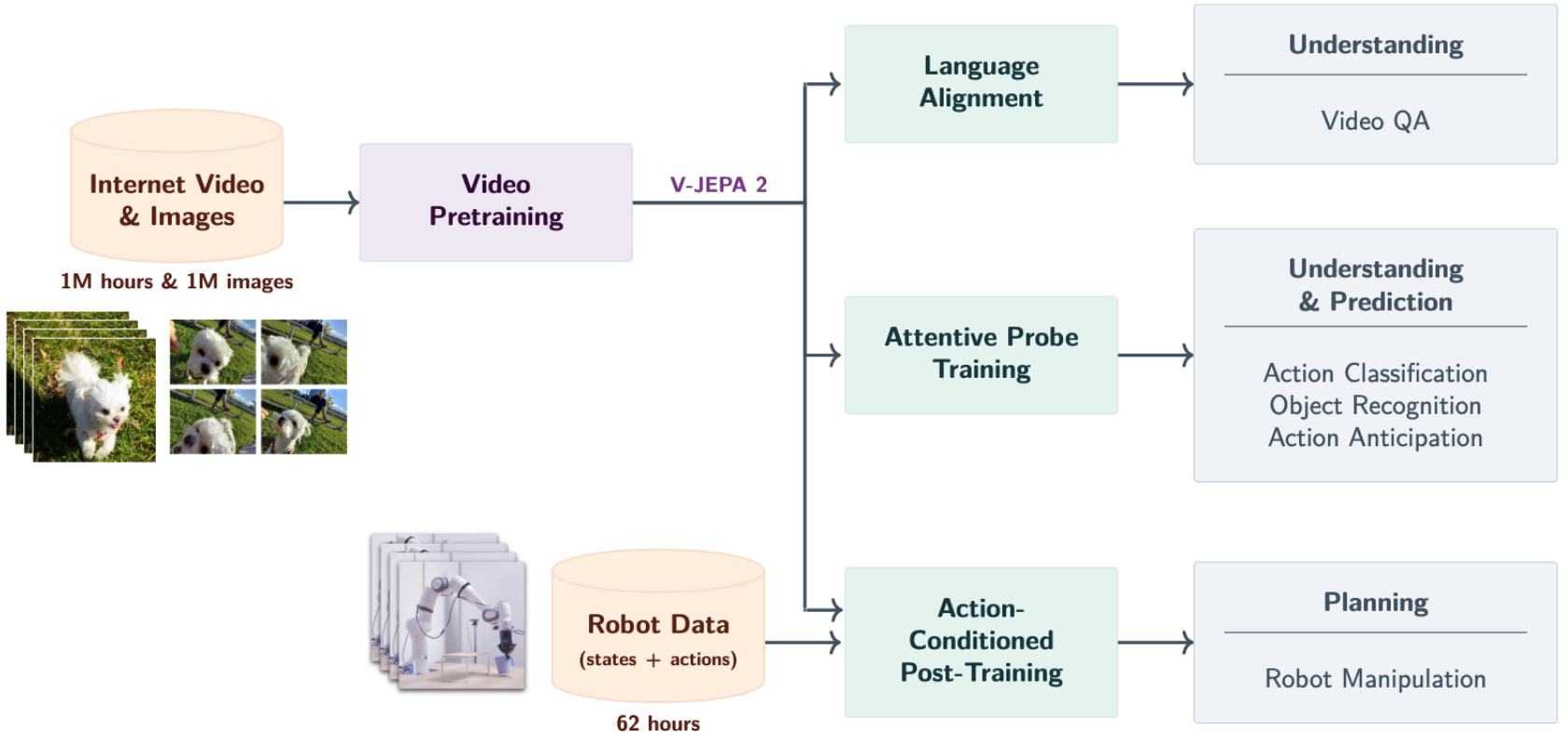
V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning

Mahmoud Assran[1,*], **Adrien Bardes**[1,*], **David Fan**[1,*], **Quentin Garrido**[1,*], **Russell Howes**[1,*], **Mojtaba Komeili**[1,*], **Matthew Muckley**[1,*], **Ammar Rizvi**[1,*], **Claire Roberts**[1,*], **Koustuv Sinha**[1,*], **Artem Zholus**[1,2,*], **Sergio Arnaud**[1,*], **Abha Gejji**[1,*], **Ada Martin**[1,*], **Francois Robert Hogan**[1,*], **Daniel Dugas**[1,*], **Piotr Bojanowski**[1], **Vasil Khalidov**[1], **Patrick Labatut**[1], **Francisco Massa**[1], **Marc Szafraniec**[1], **Kapil Krishnakumar**[1], **Yong Li**[1], **Xiaodong Ma**[1], **Sarath Chandar**[2], **Franziska Meier**[1,*], **Yann LeCun**[1,*], **Michael Rabbat**[1,*], **Nicolas Ballas**[1,*]

[1]FAIR at Meta, [2]Mila – Quebec AI Institute and Polytechnique Montréal

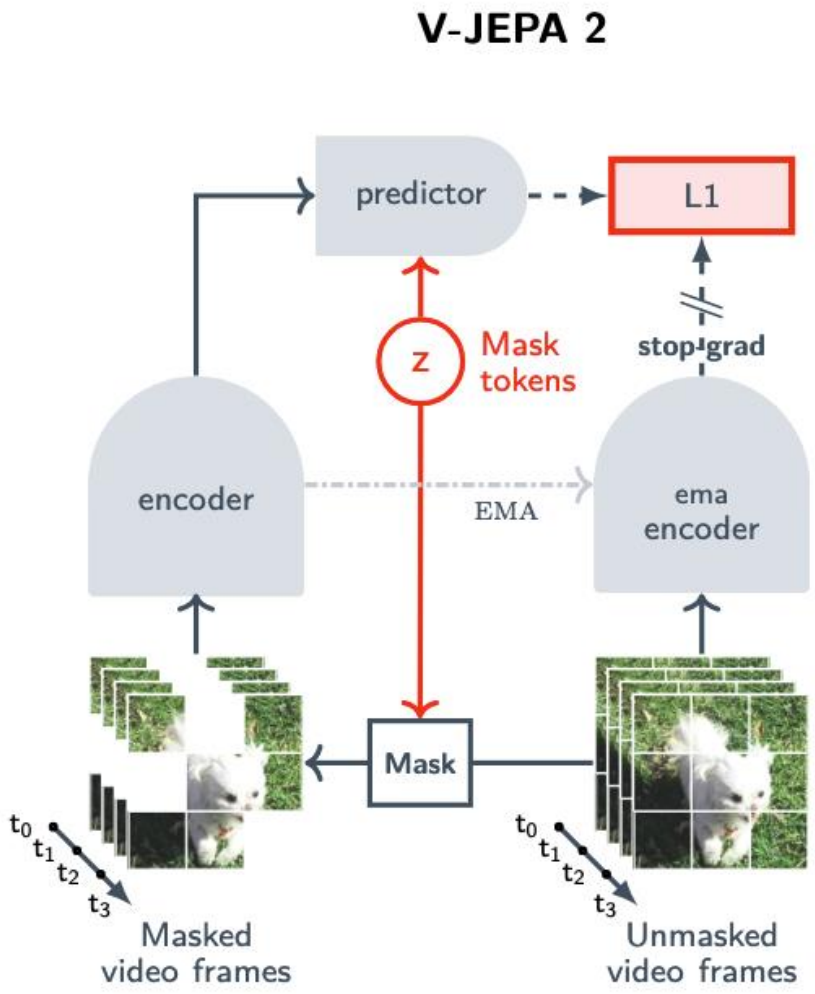
e.g., V-JEPA 2

- Pretraining with multiple tasks on uncontrolled video data
- Fine tuning with robot data to learn a forward model

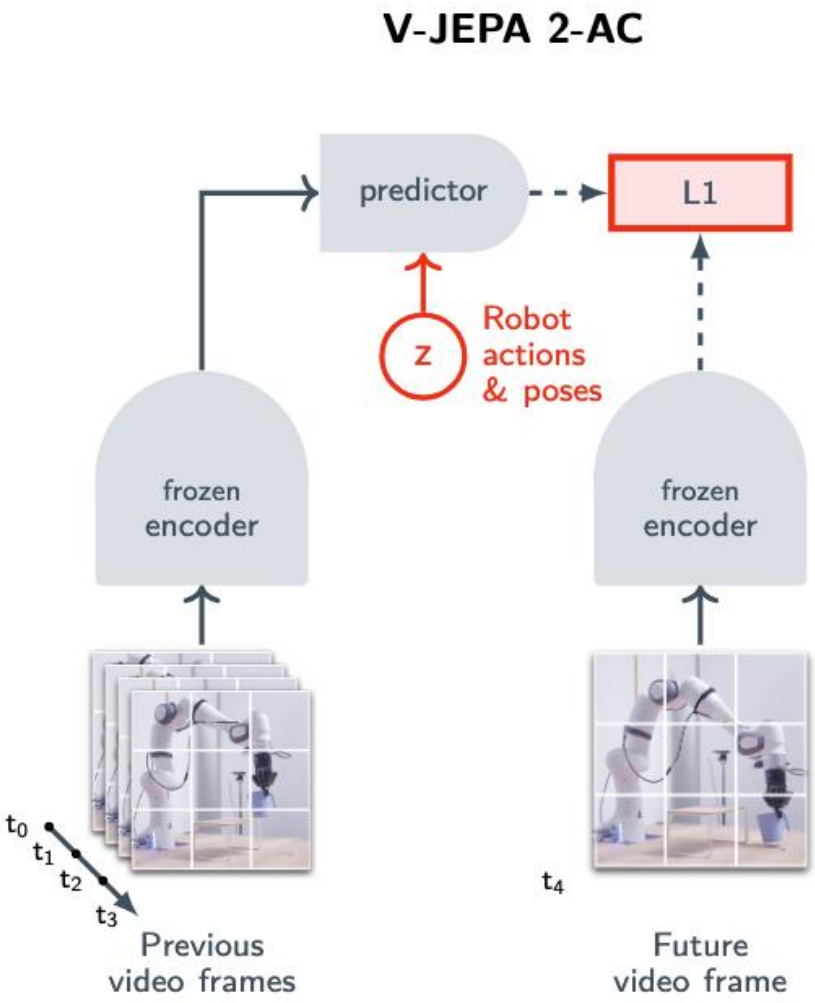


Mixing objectives

V-JEPA 2

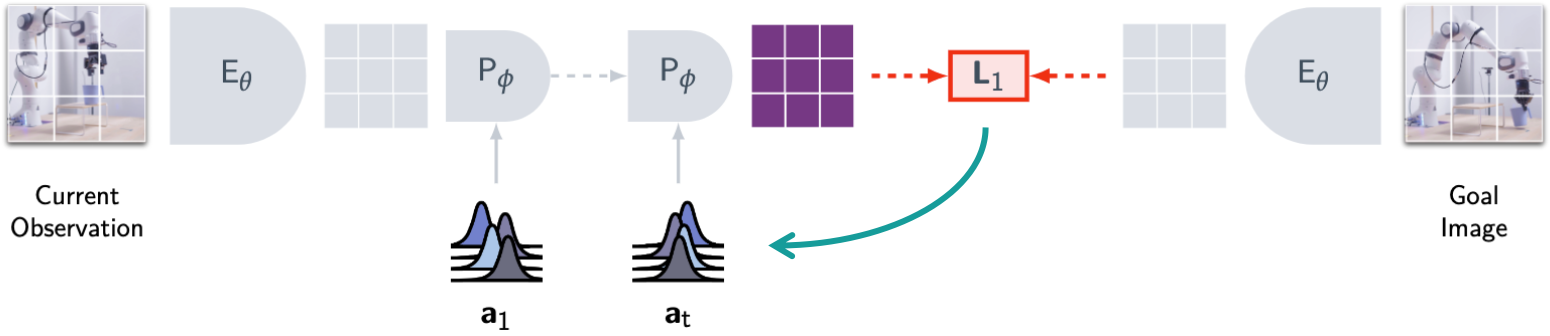


Latent state reconstruction - Semantics



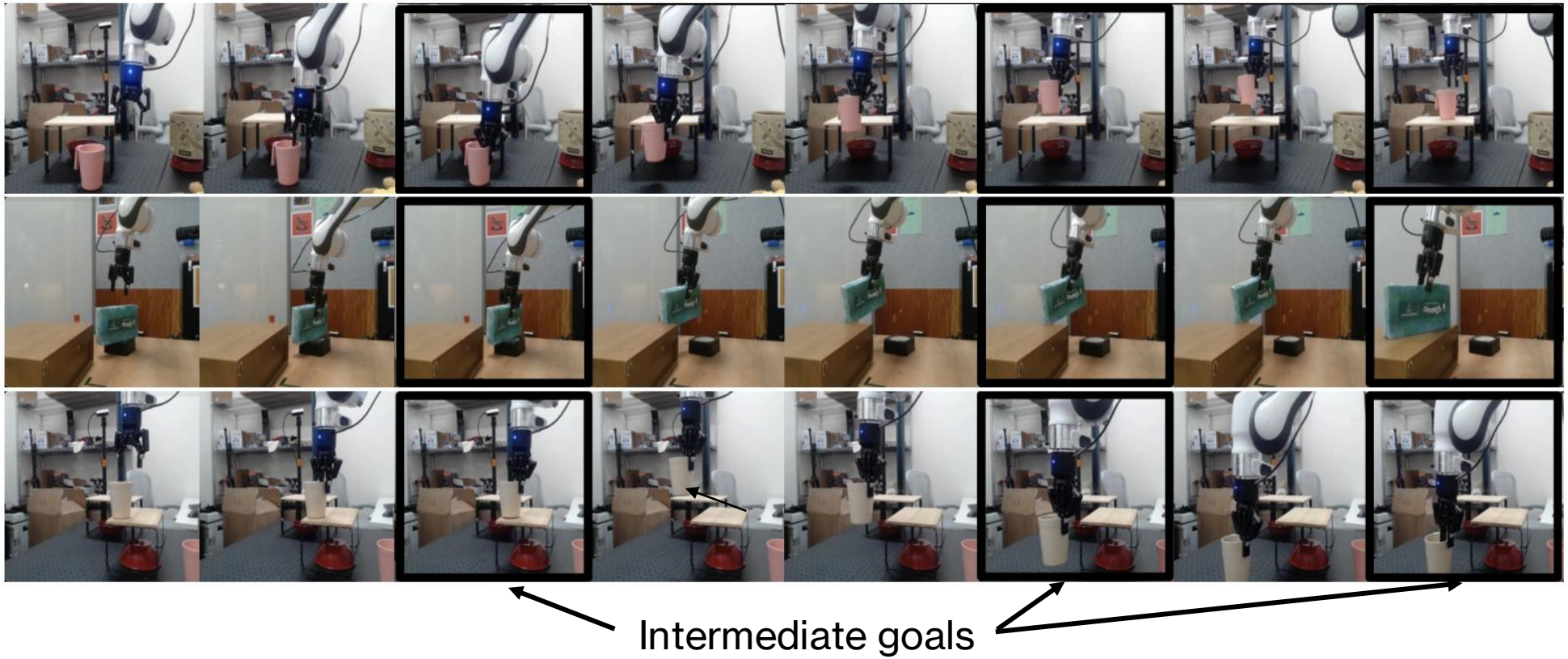
Forward model - dynamics

Mixing objectives



V-JEPA 2

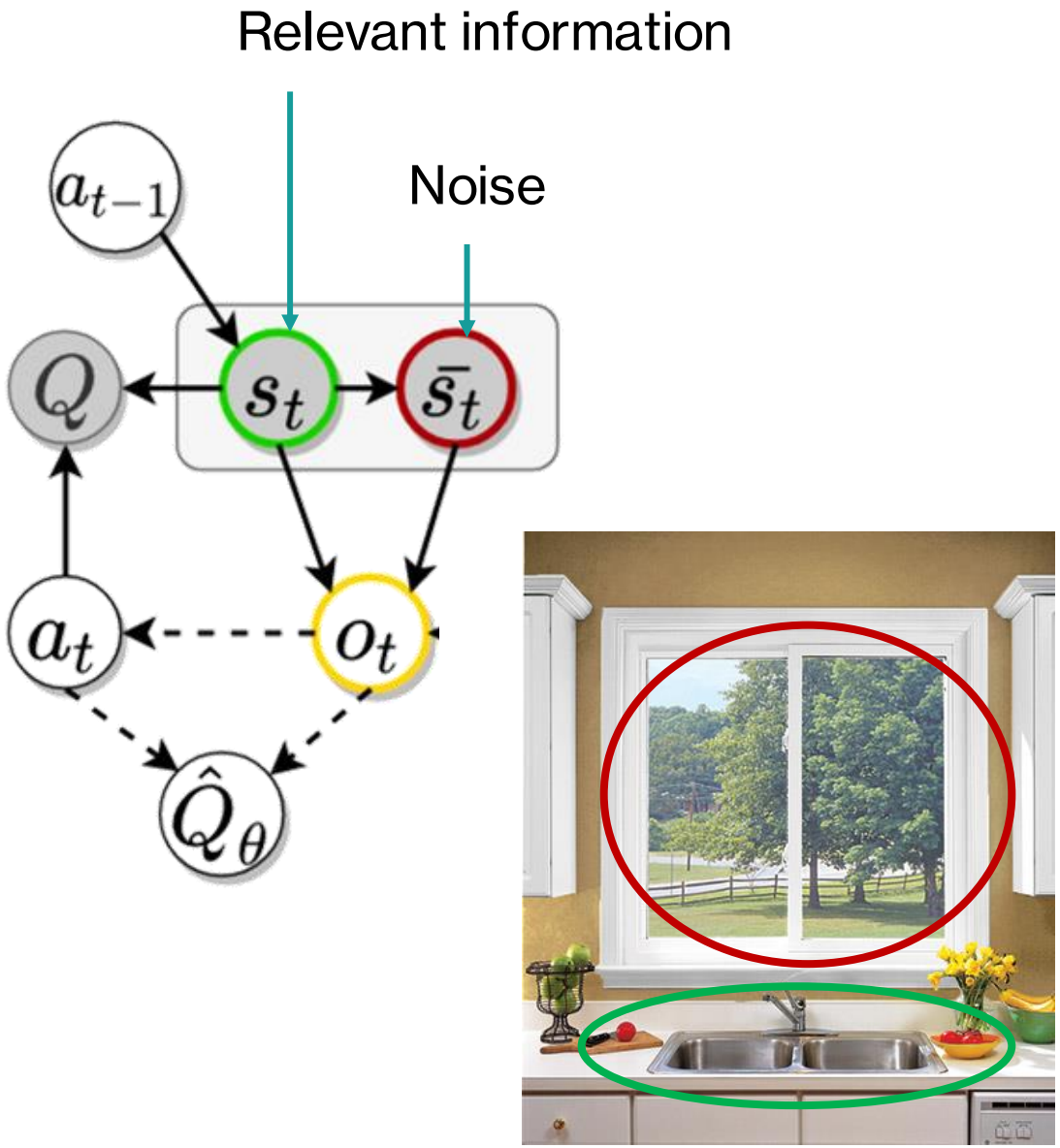
- Planning : optimise action sequence to reach goal



TRANSFERING SIMULATION BASED RL TO REALITY

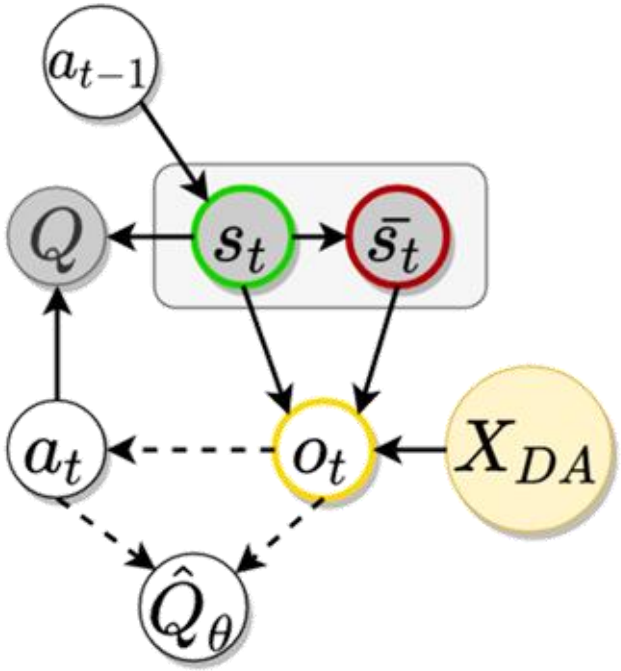
Boosting robustness in RL

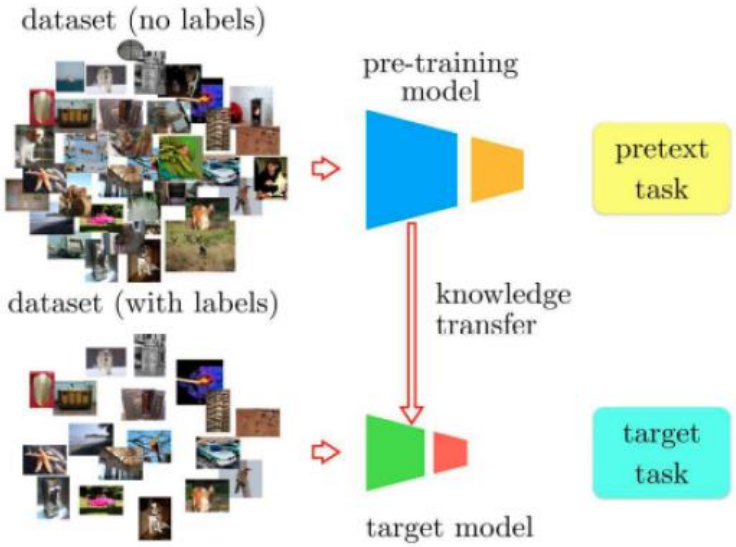
- Learning in simulation / transfer to reality (Sim2Real)
  - Because RL still very sample hungry
  - Need to face a large domain gap : Sim2Real challenge
  - Should rely on relevant info (e.g., 3D pose), and discard irrelevant features (e.g., specific appearance)
  - Existing solutions : Data augmentation, domain randomization ...



Data augmentation

- Train representations insensitive to added noise / missing info
  - Common in all deep learning models
  - Self-supervised representation learning in Deep Learning



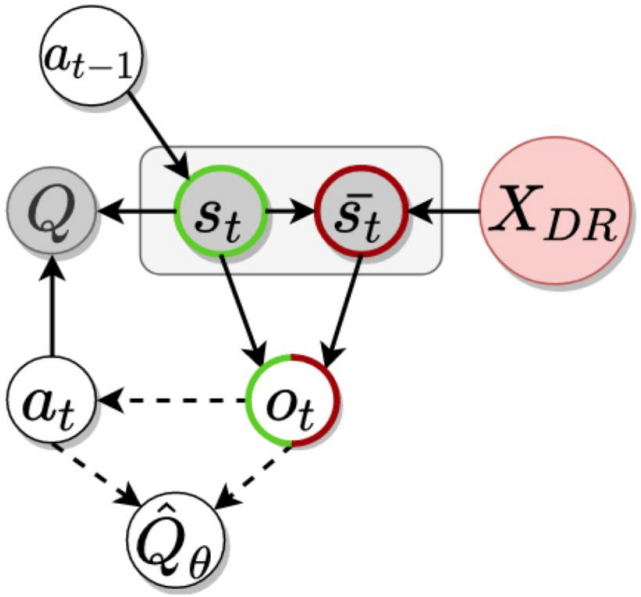


Many possible pretext tasks : rotation, localization, inpainting, remove data augmentation...

- Useful for RL but don't separate relevant/irrelevant features

Domain randomization

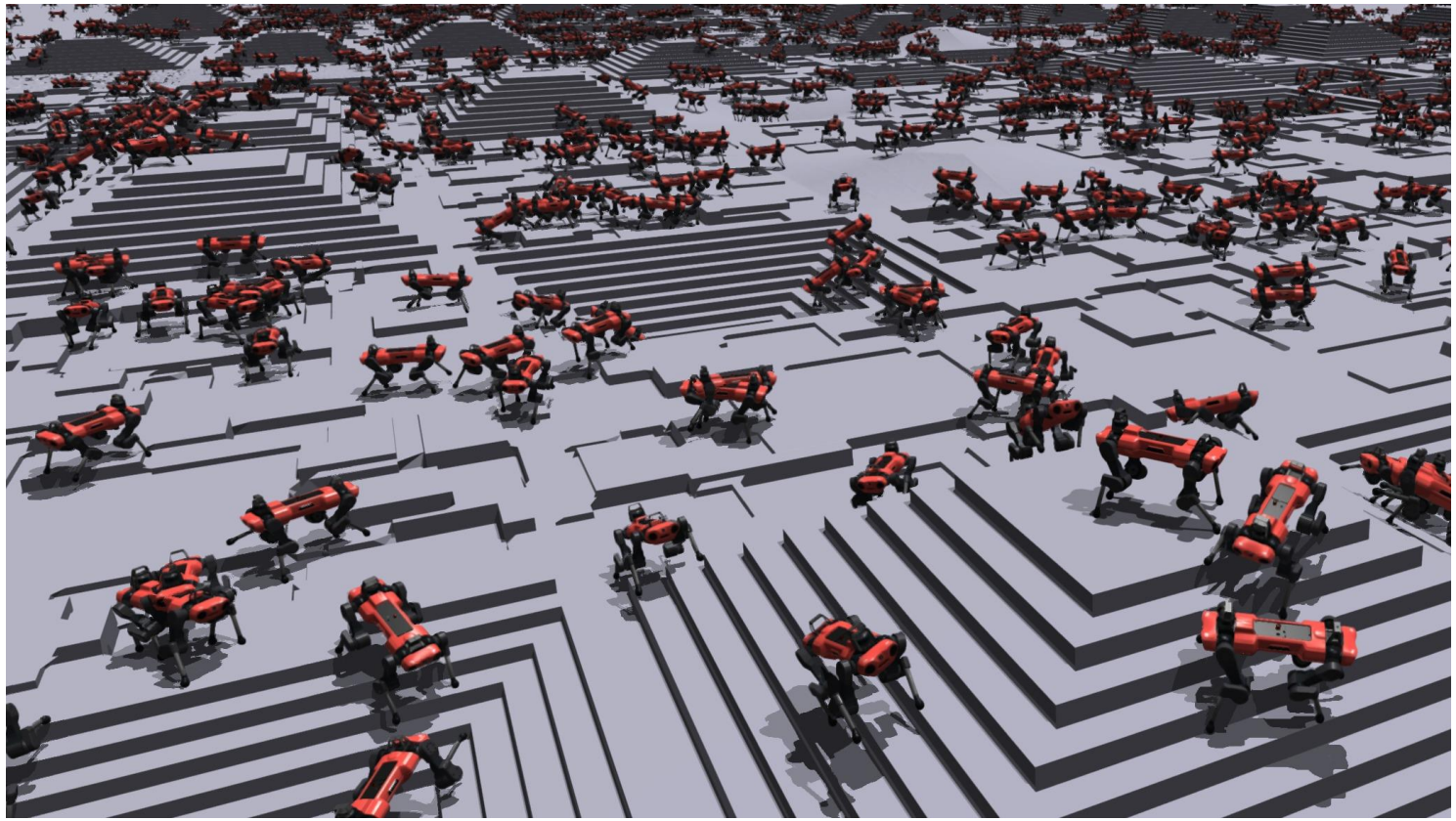
- Train representation insensitive to noise in the uncertain simulation parameters
  - Well adapted to RL and Sim2Real transfer
  - More information about underlying state than Data Augmentation
  - Can focus / control robustness to particular features
  - e.g., randomize friction to walk on any terrain



Learning to walk/move

Using sim to real tranfer

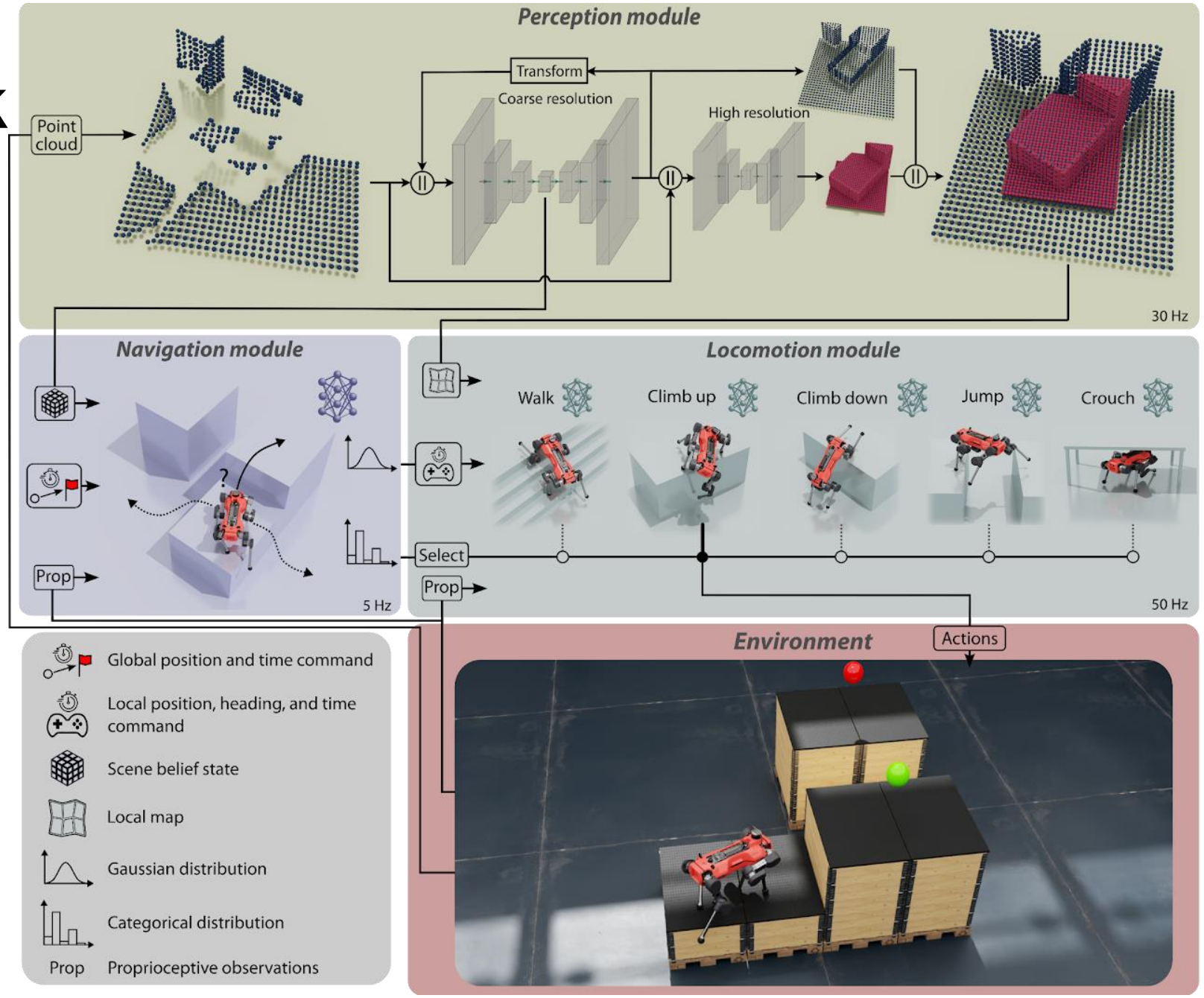
- Highly parallelized simulation on GPU (Isaac Sim)
- Randomize weight, friction, stating configuration, environment ...
- Works well for control (quadrupeds, humanoids...)
- More difficult when you include (visual) perception



Learning to walk

Sim to real Transfer

- E.g. : Anymal Parkour
- Behavior learning in simulation with RL (PPO)
- Hierarchical control : global policy and specific locomotion behaviors
- Perception module trained separately to fill-in LiDAR perception



Learning to walk/move

Sim to real Transfer

- E.g. : Anymal Parkour

SUMMARY - CONCLUSION

Summary

- From Modular Pipelines → More Integrated Learning Systems
  - Classical robotics (perception–planning–control) often rigid, engineered, and task-specific
  - Large models enable end-to-end learning and global optimization
- Foundation Models Bring Generalization
  - Transformers + large-scale pretraining → shared representations across tasks
  - Multimodality (vision, language, sensors, actions) is critical for robotics
  - Enables:
    - Open-vocabulary perception
    - Language-guided planning
    - Cross-task / cross-robot transfer
    - ...

Summary

- New Paradigm: VLA (Vision-Language-Action) Models
  - Unified models for perception + reasoning + control
  - Trained on large static datasets + robotics demonstration datasets
  - Capabilities:
    - Multi-task, multi-robot learning
    - Zero-shot / few-shot adaptation
    - Trajectory generation via diffusion policies
- Reinforcement Learning (RL) in Robotics
  - Learn a policy maximizing reward through interaction
  - Effective for robot control, often combined with imitation learning (pretraining + RL fine-tuning)
  - Data inefficient, costly on real robots, difficult exploration in large state/action spaces
  - Simulation (Sim2Real)/better state representations/hybrid learning approaches

Limitations and challenge

- Current Bottlenecks
  - Data scarcity & cost (robot data expensive, slow to collect)
  - Reliability gap vs industrial/application requirements
  - Still need quite a lot task-specific demonstrations
  - Sim2Real transfer remains difficult with perception (domain gap)
- Technical Challenges
  - Reward design in RL
  - Efficient reinforcement learning in real-world settings
  - Learning compact, actionable state representations (world models)
  - Aligning heterogeneous datasets & embodiments, using human data, ...
  - Balancing generalization vs efficiency

QUESTIONS ?