Learning Hierarchical Sensorimotor Representations for Dexterous Robotic Manipulation via Unsupervised Predictive Architectures

Guglielmo Montone
montone.guglielmo@gmail.com

December 10, 2025

Abstract

Current Vision-Language-Action (VLA) architectures for robotic control face significant limitations: they require extensive supervised training on teleoperated trajectories; they cannot perform planning or control loops at different time scales; and, similarly to LLMs, they struggle with long-horizon reasoning.

We propose a novel architecture that extends the Video Joint Embedding Predictive Architecture (V-JEPA) to create hierarchical, entangled sensorimotor representations for goal-directed robotic manipulation. Our approach learns from raw sensory streams in an unsupervised manner, encoding sensory and motor information at multiple temporal and spatial scales within a unified framework. The architecture enables high-level planning by clamping pose-encoding nodes in abstract representations, while decoders translate these plans into fine-grained motor commands.

We present a three-step roadmap for evaluating this architecture on dexterous manipulation tasks, focusing on: (1) determining optimal action datasets for learning action-oriented representations, (2) constructing hierarchical encodings that encode information at different levels of abstraction and across different time scales, and (3) demonstrating planning capabilities through mental simulation of action sequences.

By coupling perception and action within shared representational spaces and supporting control loops at multiple hierarchical levels, our architecture aims to achieve more data-efficient, generalizable, and adaptable robotic behavior compared to existing VLA systems.

1 Motivation and goal

1.1 State of the art solutions and their limitations

Designing a robot capable of performing complex, goal-directed tasks—such as cleaning a kitchen, organizing clothes, or working in a garden—requires the integration of three key functionalities: sensing the state of the environment, planning actions, and executing them. The most prevalent framework currently used to implement these functionalities is the **Vision-Language-Action (VLA)** architecture. Building on the recent advances in multimodal reasoning achieved by **Vision-Language Models (VLMs)**, researchers have extended multimodal learning to robotic control, giving rise to VLA systems.

VLA models (e.g., OpenVLA Kim et al. [2024], Octo Team et al. [2024], Nora Hung et al. [2025], Otter Huang et al. [2025], Evo-1 Lin et al. [2025]) take as input visual observations (such as images or video frames) and natural language instructions, and output corresponding actions or control commands for a robot. Typically, a VLA architecture employs a VLM backbone that jointly encodes the observed visual scene and the received instruction. This strategy affords a degree of visual generalization, improving robustness to variations in background, lighting, and phrasing of instructions.

VLA models are trained in a supervised, end-to-end fashion using trajectories obtained through teleoperation or guided demonstrations. During training, each trajectory consists of pairs of visual observations and linguistic commands, with the corresponding robot actions serving as labels for supervised learning.

In a recent architecture proposed by Google, Gemini Robotics 1.5 Team et al. [2025], a VLA model works in conjunction with a VLM to enhance the system's planning capabilities. In this architecture, the VLM functions as a planner that decomposes complex plans into simple action sequences, while the VLA is responsible for executing these simpler commands.

Despite their impressive performance, current VLA architectures—including combined VLM–VLA systems—exhibit several critical limitations.

1. **Limitations in planning.** In humans and animals, planning involves the ability to mentally simulate and evaluate alternative action sequences before execution. Large Language Model (LLM)-based planners can approximate this process but remain limited in tasks requiring extended, long-horizon reasoning—such as solving complex puzzles or performing multi-step navigation tasks. A likely cause of this limitation is the lack of an effective mechanism for revising or backtracking once a plan is committed. In contrast, simpler or more specialized architectures have demonstrated superior performance in such scenarios, consistent with the hypothesis presented in the paper "Less is more: Recursive reasoning with tiny networks" Jolicoeur-Martineau [2025]. In the Gemini Robotics 1.5 architecture, moreover, the planner operates without direct awareness of the robot's physical embodiment, resulting in limited optimization of action sequences with respect to the robot's body dynamics and constraints.

2. **Lack of Unsupervised or Self-Supervised Learning.** As noted above, VLA models are trained on teleoperated robot trajectories, a process that presents multiple challenges. First, collecting such datasets is both costly and time-consuming. To mitigate this, researchers have begun aggregating data from multiple robots, as exemplified by the Open X-Embodiment (OXE) dataset used to train VLA models such as RT-2 and Octo. While this approach promotes generalization across robot morphologies (22 different robot embodiments are present in OXE dataset), it also drives a decoupling of sensory, motor, and planning representations. This separation, however, may not be advantageous. In biological systems, for instance, perception and action are tightly coupled, it is well known that in humans and monkeys for example observing an object automatically activates motor representations of potential interactions within the premotor cortex.

   Furthermore the method of data collection also introduces constraints. Because the robot is teleoperated by a human, it does not make much sense to equip the robot with sensors whose output the operator cannot perceive. For example, tactile sensors would provide no benefit during teleoperation, since the human controller lacks direct access to their feedback and therefore cannot adapt the robot's movement accordingly.

   Finally, despite the significant effort invested in collecting and labeling such data, it is evident that supervised learning alone is insufficient for achieving strong generalization. These models still struggle to adapt to novel situations. For instance, it has been reported that the Evo model's performance in a pick-and-place task drops sharply when the target object is displaced by as little as 3 cm from the trained position; similar limitations have been observed in the Gemini Robotics 1.5 architecture.

3. **Absence of explicit hierarchical sensorimotor representation.** Effective robotic control requires planning at multiple levels of abstraction, each operating at its own time scale. This, in turn, presupposes the existence of representations that encode both the state of the world and the available actions at different hierarchical levels. High-level representations enable long-term planning with little prediction error, while low-level representations provide detailed control over action execution.

   Current VLA architectures, however, do not impose such hierarchical structure. Instead, sensory, motor, and planning representations are expected to emerge implicitly through end-to-end training on large datasets. As a result, planning the next action often requires running the entire model—an increasingly expensive step as models grow in size. This coupling of all decision-making to a single, large forward pass not only limits efficiency but also prevents VLAs from supporting the fast control loops needed for real-time robotic behavior.

1.2 Characteristics of a new architecture and main advantages

In contrast to the approach of VLA, a new approach proposed by LeCun and his team. V-JEPA (Video Joint Embedding Predictive Architecture) is an architecture designed to build an abstract representation of video observations using a self-supervised training strategy. It combines a Vision Transformer–based encoder with a predictor network. During training, two versions of the encoder are used: one processes all video patches except a subset that is masked, while the other processes the masked patches. The predictor learns to take the output of the first encoder and predict the representations produced by the second. The first encoder is updated at every training step via gradient descent, whereas the second encoder's weights are maintained as an exponential moving average of the first. This setup allows the model to learn rich, physics-aware representations of the world without direct supervision. In their paper, LeCun and colleagues demonstrate that such representations can be fine-tuned on robot trajectories, enabling the model to perform motor planning tasks with good generalization and performance.

In this project, we propose a roadmap for extending the V-JEPA framework to design a novel architecture capable of generating and executing motor plans aimed at achieving a desired target state, specified as an image. The core innovation of our approach lies in the creation of a **hierarchical, entangled sensorimotor representation**, and an architecture that can use such representation to build a motor plan and execute it. The main characteristics of the proposed architecture are as follows:

1. **Hierarchical sensorimotor representation and control.** The architecture will encode sensory

and motor information at multiple levels of granularity within a unified framework. Higher-level representations will capture abstract, temporally extended aspects of perception and action, supporting long-horizon planning and task decomposition. In contrast, lower-level representations will govern fine-grained motor control and execution. The distinct levels of representation will be explicitly identifiable within the model's structure.

2. **Entanglement of sensory and motor representations.** Unlike traditional architectures where perception and action are learned separately, our approach will couple them within shared representational spaces. This entanglement allows sensory inputs to directly inform the generation of motor plans—effectively enabling perception to "suggest" or bias specific actions based on learned affordances.

The motivation for imposing a hierarchical structure also stems from the need to support control loops that operate at different temporal scales. Although this text does not focus on the control mechanisms themselves, the architecture is explicitly designed so that each layer can host its own control loop—fast, fine-grained loops at lower levels, and slower, more abstract loops at higher levels.

As in V-JEPA, the proposed model will be trained in an unsupervised manner, learning from raw sensory streams without explicit action labels or supervision.

We anticipate that such an architecture will offer several significant advantages:

- **Reduced reliance on labeled data.** By leveraging unsupervised learning, the model will require far fewer teleoperated trajectories, thereby lowering the cost and effort of dataset collection.
- **Accelerated exploration of the action space.** The entanglement between sensory and motor representations will allow the system to rapidly identify promising motor strategies directly from sensory cues.
- **More efficient and scalable planning.** Hierarchical abstraction will enable the model to plan over longer temporal horizons while maintaining low prediction error, supporting efficient adaptation to novel tasks and environments.

In summary, the proposed architecture aims to bridge the gap between perception and action by combining hierarchical organization, sensorimotor coupling, and unsupervised predictive learning. This integration is expected to yield more generalizable, data-efficient, and adaptable robotic behavior compared to current VLA-based systems.

2 Roadmap

To evaluate the proposed architecture, we identify dexterous object manipulation as the ideal experimental domain. This choice is motivated by three key considerations:

1. **Rich multimodal sensory–motor complexity.** Robotic hands and arms used for dexterous manipulation typically exhibit high-dimensional action spaces (20–30+ DoF) and provide diverse sensory modalities, including proprioception, tactile feedback, and visual input. This makes them an excellent testbed for learning hierarchical and entangled sensorimotor representations.

2. **Availability of extensive datasets and benchmarks.** The manipulation domain benefits from a wide variety of established tasks and datasets—ranging from simple pick-and-place to in-hand manipulation, assembly, reorientation, and tool use—each with clear quantitative evaluation metrics. This breadth allows for systematic assessment of performance across different levels of difficulty and abstraction.

3. **Need for multi-level planning.** Dexterous manipulation inherently requires both high-level planning (e.g., deciding grasp strategy, sequencing subtasks) and low-level motor control (e.g., finger placement, slip compensation, force modulation). This naturally aligns with our proposed architecture's hierarchical structure, making it possible to test whether the model can effectively separate and coordinate planning across temporal and physical scales.

Experimental Scenario: Learning Complex Dexterous Skills We propose to conduct our initial experiments in a simulated environment where a robot is equipped with:

- A multi-DoF arm and dexterous hand (20–30 DoF)
- Visual sensors: one or more RGB or RGB-D cameras providing observations of the workspace
- Proprioceptive sensors: joint positions, velocities, and actuator efforts
- Tactile sensors: fingertip or whole-hand tactile arrays enabling contact-rich manipulation

Because real robots often lack comprehensive sensing capabilities, we will run our experiments in simulation. The simulated setup will allow us to vary the robot's DoF, geometry, and sensor suite, enabling systematic tests of both generalization across embodiments and the role of individual sensors or structural features in dexterous manipulation.

2.1 Step 1: Defining the action dataset needed for and 'action-oriented' representation

In this part of the project, the main contribution is not modifying the V-JEPA architecture, but determining which robot experiences (i.e., which actions) should be collected to develop an action-oriented representation of the world. To investigate this, we will fine-tune the V-JEPA model using actions collected from a single robot. The robot will perform predefined actions. Such actions will involve: moving the end-effector without any object involved, simple interaction with objects (accidentally touching, pushing, pulling), complex object manipulation (piling objects, precision picking).

All robot sensors (proprioceptive and exteroceptive) will be recorded while the robot performs the actions.

The V-Jepa architecture will be fine-tuned using the same training paradigm used in V-JEPA Assran et al. [2025]. As shown in the following figure 1

- the sensor input $S(t)$ and $S(t+1)$ will be passed to the encoder to create a representation of the sensory input $L(t)$ and $L(t+1)$;

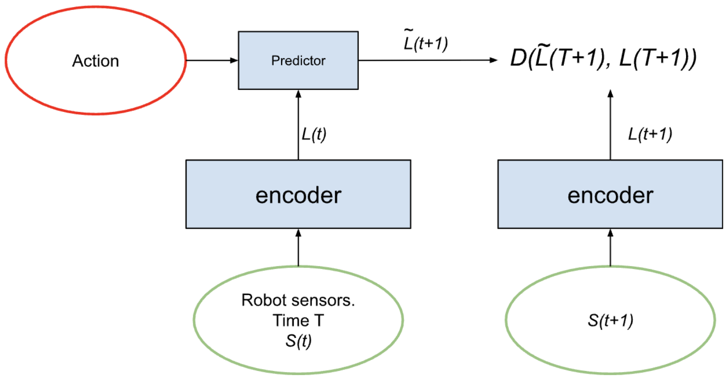

Figure 1: V-JEPA architecture

- a predictor will be trained to predict the next representation of the sensors $\tilde{L}(t+1)$ using as input the action performed and the encoding of the current sensors.

A key component of this study is to determine which types of robot experiences should be included in the dataset to form an action-oriented representation. We consider three data-collection strategies:

1. Random robot movements These are the easiest to collect, but it is unclear whether they improve representation quality. In fact, prior VLA work suggests that random actions may degrade downstream performance.

2. Actions generated by simple algorithmic policies: These refer to policies that, for example, move the gripper/robotic hand toward objects in the scene, creating more structured and object-oriented interactions. More advanced options could include using trajectories generated by existing trained VLA models. The advantage of these actions is that they naturally produce meaningful interactions with objects, which would constitute valuable training for our V-Jepa architecture even if the action is not successful (eg. robot fails to correctly grasp the target object).

3. Tele-operated actions. Incorporating tele-operated trajectories allows us to measure how important expert demonstrations are for forming an action-oriented representation. However, collecting such data is comparatively expensive, so part of the goal is to determine how little tele-operation is actually needed.

A major part of the study will be to quantify how much of each action type is required in the dataset to produce a robust acting-in-the-world representation. This includes exploring different ratios of random actions, algorithmic actions, and tele-operated trajectories, and assessing their impact on the quality of the learned representation.

Another outcome of this research is to define what constitutes an “action-oriented” representation and to establish metrics and benchmarks to evaluate whether the robot has acquired such a representation. Relevant criteria may include:

- distinguishing graspable vs. non-graspable objects
- distinguishing reachable vs. unreachable objects or regions
- inferring heavy vs. light objects (i.e., what moves when touched)

- grouping objects by required grasp type
- visual presentation of an object triggering the appropriate proprioceptive/action priors
- recognizing actions performed by others

2.2 Step 2: Building a hierarchical representation of the sensory space

The outcome of this part of the work would be:

1. Define an architecture where we enforce a hierarchical representation of the sensor space
2. Define an architecture that can be used for planning at different time scales.

2.2.1 Step 2.1: Use an implicit representation of action to predict the next status

In this part of the work we will use the same dataset collected in the previous step of the work. As a reminder the dataset will consist of actions performed by the robots where all the sensors (proprioception and exteroception) were recorded.

Differently from the previous experiments we will not use an explicit representation of the action as an input to the predictor, but the predictor will receive as an input the encoding (the output of the Encoder 1 in the figure 2) of the sensory state for the previous N time steps and will use them to predict the representation at time T+1. See the figure 2 (A).

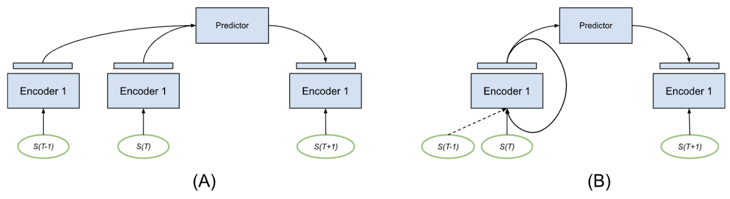

Figure 2: In this architecture the prediction of the next hidden state is obtained using the hidden states of the previous N time steps

In this part of the work we wish to test if it is possible to use such an architecture and keep all the benefits of the architecture used in step 1 in terms of action-oriented representation, good representation of the world and intuitive understanding of physics.

We would also like to test an architecture where the encoder becomes a recurrent architecture see figure 2 (B).

2.2.2 Step 2.2: Hierarchical architecture

In the following figure we present an idea to scale the architecture presented in the previous paragraph so that we will have a more abstract representation, such representation changing on a longer time scale.

The architecture overlaps to the architecture presented in figure 2 (B) a second architecture which receives as input: the encoding of lower level architecture and its own encoding at the previous step.

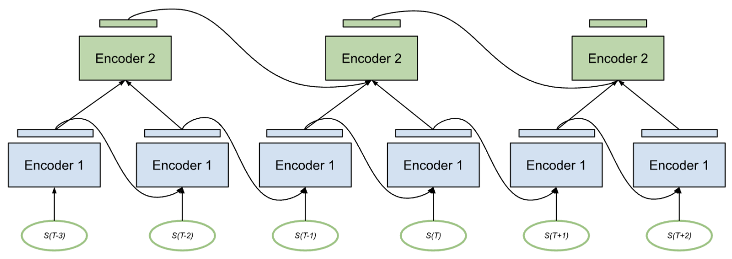

Figure 3: A second encoder (Encoder 2) is used to encode the state of the world at a longer time scale than Encoder 1

We will train such architecture on the same dataset used before. We will then define a set of metrics to validate that the representation is indeed hierarchical, meaning that the higher level encoder (Encoder 2 in the figure) would encode the state of the world using less details and with a representation that varies slowly in time compared to the representation of Encoder 1 which will be more rich in details and would change more often.

We would also like to validate that in the representation generated by the Encoder 1 and Encoder 2 would it still be possible to distinguish the part of the architecture that mostly encodes the proprioceptive sensors and the part that encodes the exteroceptive sensors. We would expect that this is the case to some extent because the 2 types of sensor will vary on a big dataset quite independently. The possibility to recognize nodes that are mostly dedicated to encode the proprioception would be essential for using the architecture to plan future scenarios.

In this phase of the study, some work would be dedicated to the best way to train the 2 encoders. If the training should happen in parallel or sequentially.

2.3 Step 3 Use the previous architecture for planning

2.3.1 Step 3.1 Clamping the pose-enoding nodes to predict the outcome of an action

Here we propose a way to use the architecture to build a high level motor planning. Our approach assumes that, within the high-level representation (the output of Encoder 2), we can separate the component that encodes the relative positions of the robot's body parts (shown in red in the figure 4) from the component that encodes the external world (shown in green). In the following we will refer to the component that encodes the relative positions of the robot's body parts as the "pose-encoding nodes".

Planning would work as follows: we will pass to the two encoders (1 and 2) the current status of the world and we will obtain a first high level representation as output of the encoder 2 (represented as the vector $H1$ in the picture). We will afterwards clamp the pose-encoding nodes of $H1$ vector obtaining the vector $H1^*$ and let the vector pass through the Encoder 2 to define the expected status of the exteroceptive sensors associated with the clamped pose-enconding nodes.

In order to validate that the previous paradigm can be effectively used for planning we will test in the following way. We will collect a new dataset in a way similar to how we collected the dataset that we

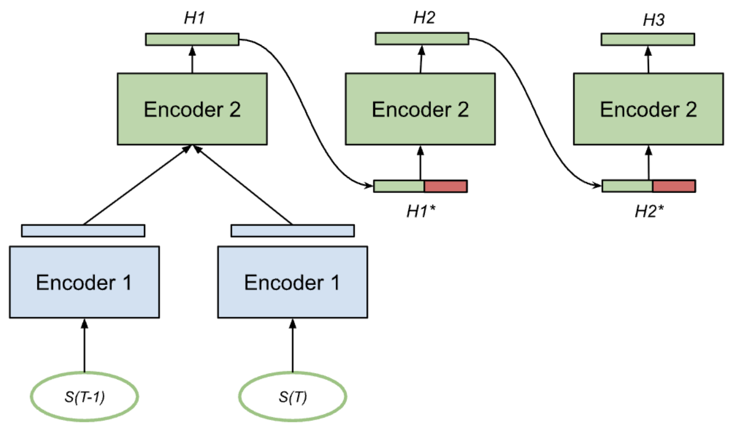

Figure 4: Clamping the pose-encoding nodes of $H1$ vector obtaining the vector $H1^*$ and let the vector pass through the Encoder 2 to define the expected status of the exteroceptive sensors associated with the clamped pose-enconding nodes.

used so far: robot moving hands performing simple and more complex action sequences. While the robot performs such actions we will record the value of all the sensors and the encodings (all the output of the two encoders) for different time stamps.

In order to test the quality of the planning we will for every action sequence give the sensor input collected for time $T-1$ and T, we will pass it to Encoder 1 and Encoder 2 and we will obtain the first encoding of level 2. We will afterward clamp the pose-encoding nodes to the status that was observed while the robot was actually performing the action. We will pass the clamped vector to the Encoder 2 once again and we will finally compare the representation of the world ($H2$ in the picture) with the one that was observed while the robot was actually performing the action.

It is important here to stress that the paradigm described for testing the quality of the planning could be also used to fine-tune the architecture to improve the architecture motor planning skills.

2.3.2 Step 3.2: Projecting a high-level motor plan to lower level motor command

In this paragraph, we explain how to execute a motor plan once it has been identified, as described in the previous section.

Identifying a motor plan is equivalent—according to the definition given earlier—to determining the optimal way to clamp the robot's pose-encoding nodes (the red nodes) so that the output of Encoder 2 matches the goal state as closely as possible. In this context, a plan corresponds to a sequence of vectors, such as the sequence H_2, H_3 illustrated in the figure 4.

Once an ideal plan has been identified, we need a mechanism to translate it into concrete motor commands. To achieve this, we introduce additional components into our architecture: decoders. These modules take an abstract representation from a higher level and produce a sequence of representations for the next lower level. In the figure below, the first decoder (Decoder 2) transforms the vector H_2 into an expected status of the lower-level vectors, we name them $\hat{L_3}$ and $\hat{L_4}$. A second decoder then transforms $\hat{L_3}$ and $\hat{L_4}$ into the expected sensor status $\hat{S}(t+1)$, $\hat{S}(t+2)$ see figure **??**. We assume that, once we have the raw proprioceptive sensor values, defining the corresponding motor commands is straightforward. In this way, we obtain the full sequence of motor commands needed to reach the desired end state.

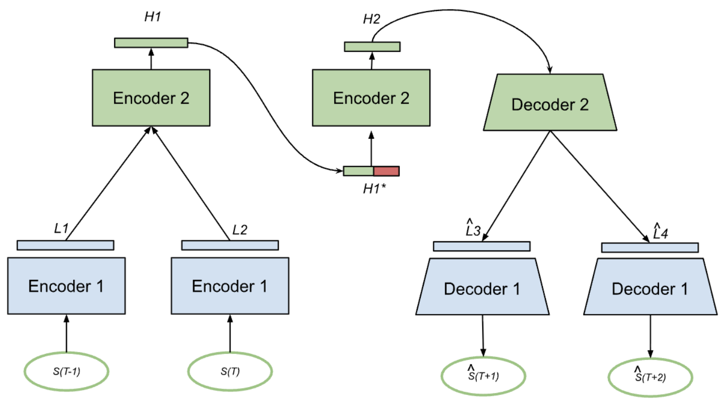

Figure 5: The first decoder (Decoder 2) transforms the vector H_2 into the lower-level vectors $\hat{L_3}$ and $\hat{L_4}$. A second decoder then transforms $\hat{L_3}$ and $\hat{L_4}$ into the expected sensory inputs $\hat{S}(t+1)$, $\hat{S}(t+2)$.

The training of the decoder would be executed using the same or similar dataset previously collected, with the robot performing all types of actions.

The reason we propose having a decoder for each encoder layer is that we want to enable a control loop at every level of the hierarchy. At each layer, the planned state can be compared with the actual observed state, allowing the system to adjust the plan at different time scales. This, in turn, allows the robot to execute fast control loops that rely on only a subset of the full architecture. To give a more concrete example a control loop of the lowest layer would probably looks like this: it would take as input the expected state of the world l3 and the observed state of the world after taking an action, and it would suggest changes to the l4 and hence to the next action to take.

3 Conclusions and potential further directions of development

We think that the architecture previously described can be effectively used to:

- Build a hierarchical representation of the motor and sensor space of the robot
- Allow for high level planning and for transforming a high level plan in more detailed motor command

The architecture offers the benefits that the motor, sensor and planning are highly connected, it also offers the advantage of being trained using only unsupervised training techniques.

We would expect that the main benefit of such architecture would be for the robot to be able to learn complex actions using an extremely small number of supervised data.

The most immediate next step that we see would be to add to the architecture a system that can find the ideal plan to reach a derided target state. Here we are assuming that

Also interesting directions could include: end-to-end training of the architecture and autonomous collection of new experience with some sort of curiosity driven paradigm.

References

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Huang Huang, Fangchen Liu, Letian Fu, Tingfan Wu, Mustafa Mukadam, Jitendra Malik, Ken Goldberg, and Pieter Abbeel. Otter: A vision-language-action model with text-aware visual feature extraction. *arXiv preprint arXiv:2503.03734*, 2025.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. *arXiv preprint arXiv:2511.04555*, 2025.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.