StarVLA

StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing

StarVLA Community & Von Neumann Institute, HKUST

Abstract

Building generalist embodied agents requires integrating perception, language understanding, and action, which are core capabilities addressed by Vision-Language-Action (VLA) approaches based on multimodal foundation models, including recent advances in vision-language models and world models. Despite rapid progress, VLA methods remain fragmented across incompatible architectures, codebases, and evaluation protocols, hindering principled comparison and reproducibility. We present **StarVLA**, an open-source codebase for VLA research. StarVLA addresses these challenges in three aspects. First, it provides a modular backbone–action-head architecture that supports both VLM backbones (e.g., Qwen-VL) and world-model backbones (e.g., Cosmos) alongside four representative action-decoding paradigms, all under a shared abstraction in which backbone and action head can each be swapped independently. Second, it provides reusable training strategies, including cross-embodiment learning and multimodal co-training, that apply consistently across supported paradigms. Third, it integrates major benchmarks, including LIBERO, SimplerEnv, RoboTwin 2.0, RoboCasa-GR1, and BEHAVIOR-1K, through a unified evaluation interface that supports both simulation and real-robot deployment. StarVLA also ships simple, fully reproducible single-benchmark training recipes that, despite minimal data engineering, already match or surpass prior methods on multiple benchmarks with both VLM and world-model backbones. To our best knowledge, StarVLA is one of the most comprehensive open-source VLA frameworks available, and we expect it to lower the barrier for reproducing existing methods and prototyping new ones. ***StarVLA is being actively maintained and expanded; we will update this report as the project evolves.*** The code and documentation are available at github.com/starVLA/starVLA.

Date: April 2026
Project Page: `https://starvla.github.io`

1 Introduction

Embodied AI is advancing toward general-purpose agents that integrate perception, language understanding, and action in the physical world, driven in part by recent breakthroughs in large foundation models (OpenAI, 2023; Bai et al., 2025a; Gao et al., 2025). Vision-Language-Action (VLA) models have emerged as a dominant paradigm for this goal, with a diverse range of design choices. Existing approaches can be broadly grouped into two families: *VLM-based methods*, which repurpose the language model's representational capacity for action decoding, and *world-model-based methods*, which employ generative architectures to jointly model action distributions and future observations. While both directions have shown strong promise, they are often developed in isolation, with different codebases, interface assumptions, and evaluation protocols, making it challenging to systematically compare them and understand the trade-offs between different design choices.

Fragmentation hinders systematic exploration. Despite this progress, VLA research remains hindered by fragmentation at multiple levels. At the *architecture* level, existing approaches (Kim et al., 2025; Brohan et al., 2022, 2023; Bjorck et al., 2025; Black et al., 2024; Intelligence et al., 2025a; Wu et al., 2026; Li et al., 2026) adopt diverse action-decoding designs, from VLM-native methods (autoregressive tokenization, parallel regression) to generative-model-based methods (diffusion, flow matching), making systematic comparison

across paradigm families difficult. At the *system* level, methods are released with tightly coupled assumptions on model architecture, data processing, and training pipelines, limiting component reuse across projects. At the *evaluation* level, results are reported on disjoint subsets of benchmarks with inconsistent protocols, making fair comparison infeasible. Together, these issues create a "Tower of Babel" for VLA research, where ideas are difficult to compare, reproduce, or recombine. We attribute this fragmentation to the *lack of a unified abstraction for VLA systems*. Existing codebases (Bjorck et al., 2025; Black et al., 2024) are largely method-specific and do not support (i) modular composition across different action-decoding paradigms, (ii) reusable training across heterogeneous data sources, or (iii) standardized evaluation and deployment across benchmarks and embodiments.

StarVLA: a unified platform for exploring embodied intelligence. We introduce **StarVLA**, an open-source research platform that brings VLM-based and world-model-based VLA paradigms into a unified modular framework. The core design is a *backbone–action-head decomposition*, where a shared vision-language backbone encodes the scene and instruction, and a pluggable action head maps the resulting representation to motor commands. This formulation is flexible enough to support a wide range of existing approaches, including autoregressive tokenization, parallel regression, flow-matching denoising, and dual-system reasoning, with re-implementations that match or in some cases exceed reported performance. In practice, StarVLA provides three core capabilities:

- **Unified VLA frameworks:** StarVLA implements four representative paradigms under the shared backbone–action-head abstraction (Section 2): StarVLA-FAST (autoregressive tokenization), StarVLA-OFT (parallel regression), StarVLA-π (flow-matching denoising), and StarVLA-GR00T (dual-system reasoning). Crucially, both VLM backbones (e.g., Qwen3-VL) and world-model backbones (e.g., Cosmos-Predict2) are supported as drop-in alternatives, enabling direct comparison between VLM-based and world-model-based research paths under identical training and evaluation conditions. All variants share the same data interface and downstream infrastructure; only the backbone or the action head differs, enabling researchers to isolate the effect of any single design choice while holding all others constant.
- **Flexible training recipes:** StarVLA treats cross-embodiment learning and multimodal co-training as reusable, paradigm-agnostic configurations rather than method-specific add-ons. The same training infrastructure supports supervised action learning, co-training with web-scale vision-language data to preserve multimodal reasoning, and cross-embodiment pretraining across heterogeneous robot datasets. Every recipe applies uniformly to all supported paradigms, making it straightforward to study how training strategies interact with different architectural choices.
- **Broad benchmark integration:** StarVLA integrates five mainstream benchmarks (LIBERO, SimplerEnv, RoboTwin 2.0, RoboCasa-GR1, and BEHAVIOR-1K) through a unified server-client testing interface, enabling controlled comparison across environments and embodiments. For each benchmark, we provide simple, fully reproducible training recipes with minimal data engineering that already achieve competitive or state-of-the-art performance under both VLM and world-model backbones, lowering the barrier for the community to build upon. The same interface supports both simulation evaluation and real-robot deployment without code changes, closing the gap between research exploration and practical deployment.

To position StarVLA within the existing ecosystem, we compare it with representative open-source VLA systems across key capabilities in Table 1. To the best of our knowledge, StarVLA is the first platform to bring these capabilities together within a unified interface. Leveraging the controlled comparisons enabled by this framework, StarVLA achieves competitive, and in some cases state-of-the-art, performance across multiple benchmarks with both VLM and world-model backbones, demonstrating that the platform serves not only as a research toolkit but also as a provider of strong, easy-to-reproduce baselines.

A generalized VLA perspective. Beyond its engineering utility, StarVLA also suggests a broader perspective on unifying diverse VLA approaches. Empirically, we find that a single backbone–action-head abstraction can accommodate VLM-based decoding, generative-model-based decoding, and dual-system architectures, all within a shared data pipeline, training loop, and evaluation protocol. This observation indicates that VLM-based and world-model-based methods may be better understood not as fundamentally distinct paradigms, but as variations within a common structural framework, differing primarily in the form of auxiliary learning signals (e.g., language-aligned reasoning or future observation prediction). We refer to this as the *generalized VLA* perspective. Rather than a purely conceptual viewpoint, it arises from the practical unification enabled by StarVLA: when differences in infrastructure are minimized, underlying commonalities become more apparent. We hope this perspective encourages more systematic and cumulative exploration of robotic foundation models.

Table 1: Comparison of representative open-source VLA systems. **Modular Action Heads**: action heads are plug-and-play on a shared backbone. **Modular VLM**: supports swapping the VLM backbone. **Modular WA**: supports world model as VL backbone. **Mixture DS**: built-in mixture dataloader for heterogeneous data sources. **Open-Source MM Co-train**: open-source multimodal co-training support. **Open-Source X-Emb. Co-train**: open-source cross-embodiment co-training support. **#Sim Bench**: number of integrated simulation benchmarks with evaluation code. **Multi-Bench Co-train**: joint all benchmarks into one model.

Framework	Modular Action Heads	Modular VLM	Modular WA	Mixture DS	Open-Source MM Co-train	Open-Source X-Emb. Co-train	#Bench	Multi-Bench Co-train
OpenPI Intelligence et al. (2025a)	✗	✗	✗	✗	✗	✗	2	✗
Isaac-GR00T Bjorck et al. (2025)	✗	✗	✗	✓	✗	✓	6	✗
OpenVLA-OFT Kim et al. (2025)	✗	✗	✗	✗	✗	✓	1	✗
Dexbotic Contributors (2025)	✗	✓	✗	✗	✓	✗	5	✗
X-VLA Zheng et al. (2025a)	✗	✗	✗	✓	✗	✓	5	✗
StarVLA (Ours)	✓	✓	✓	✓	✓	✓	7	✓

2 Unified Framework for VLA Systems

The rapid evolution of Vision-Language-Action (VLA) models has led to a wide range of heterogeneous designs, with varying preprocessing pipelines, model boundaries, and inference assumptions. While this diversity enables rapid exploration, it often hinders reproducibility and makes fair comparison difficult. To address this, StarVLA adopts a *unified framework abstraction* at the system level: each method is implemented as a modular component with explicit training and inference interfaces, such that algorithmic differences are isolated to a minimal set of interchangeable modules.

Abstraction of VLA systems. Beyond the system-level abstraction, we introduce a unified *policy-centric formulation* of VLA models. Prior work often distinguishes between VLM-based policies (VLA) and world-model-based approaches (WAM); here, we place them under a common perspective centered on action generation.

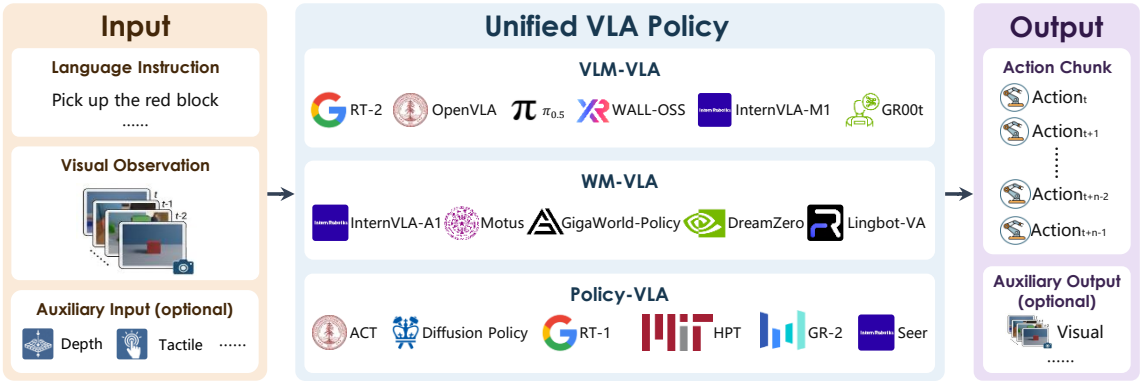

Figure 1: Conceptual view of the unified VLA formulation adopted in StarVLA. A policy π maps visual observations and a language instruction to a future action chunk. The training objective decomposes as $\mathcal{L} = \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{aux}}$, where different model families correspond to different forms of $\mathcal{L}_{\text{aux}}$.

As illustrated in Fig. 1, we model a VLA system as a policy that maps vision-language (VL) inputs to future action (A) sequences and optional auxiliary outputs:

$$\pi(\mathbf{a}_{t:t+k}, \mathbf{y}_{\text{aux}} \mid \mathbf{x}_{\leq t}, \ell), \tag{1}$$

where:

- $\mathbf{x}_{\leq t} = \{o^{\text{vis}}_{\leq t}, o^{\text{depth}}_{\leq t}, o^{\text{tactile}}_{\leq t}, \ldots\}$ denotes the multimodal observation history up to time t, which may include visual observations, depth maps, tactile feedback, proprioceptive states, or other sensor modalities;
- ℓ is the language instruction describing the task;
- $\mathbf{a}_{t:t+k}$ represents the predicted k-step action chunk from time t to $t+k$;

- $\mathbf{y}_{\text{aux}}$ denotes optional auxiliary outputs over the future horizon, such as predicted future visual observations $o^{\text{vis}}_{t+1:t+k}$, intermediate language reasoning or sub-goal descriptions ℓ_{plan}, or other modality predictions.

This formulation abstracts away intermediate representations and implicitly marginalizes over latent predictions when present, allowing both direct policies and model-based approaches to be expressed within a common interface.

The training objective takes the general form

$$\mathcal{L} \;=\; \mathcal{L}_{\text{action}} \;+\; \mathcal{L}_{\text{aux}}, \tag{2}$$

where $\mathcal{L}_{\text{action}}$ supervises the predicted actions, and $\mathcal{L}_{\text{aux}}$ serves as an inductive bias that shapes the learned representation. Different VLA paradigms can then be interpreted as instantiations of this formulation with distinct learning signals:

- **Direct VLA Modeling** sets $\mathcal{L}_{\text{aux}} = 0$, optimizing actions alone.
- **VLM-based VLA** introduces language-aligned auxiliary objectives, such as sub-task planning, spatial grounding, or structured reasoning supervision, requiring the model to generate language tokens as auxiliary outputs.
- **WM-based VLA** incorporates future observation prediction (e.g., images or videos), either as an auxiliary objective or as an implicit latent structure that supports action generation, where the model must predict visual states as auxiliary outputs.

Under this view, seemingly different paradigms such as VLM-based, world-model-based, and direct policies can be understood as variations of a shared policy formulation with different inductive biases. This perspective simplifies comparison while remaining compatible with both step-wise execution and multi-step open-loop control.

2.1 Background: VL Foundation Models for Embodied Intelligence

Embodied agents interact continuously with the physical world, where vision serves as the primary modality for perceiving scene structure, object identity, spatial relations, and interaction affordances.

Vision-language foundation models. This central role of vision has driven advances in visual representation learning, from supervised models such as ResNet (He et al., 2016) and Vision Transformers (Dosovitskiy et al., 2021) to scalable self-supervised approaches (Oquab et al., 2023) and video pretraining that captures temporal structure. Building on these backbones, language-aligned pretraining (Radford et al., 2021; Zhai et al., 2023) enables shared vision–language representations, while promptable systems such as SAM (Kirillov et al., 2023; Liu et al., 2023b) extend open-world perception. Together with instruction-tuned VLMs (Liu et al., 2023a; Chen et al., 2023; Karamcheti et al., 2024; Bai et al., 2025b; OpenAI, 2024; Gemini Team, Google, 2024) and generative video models (Gao et al., 2025; Google DeepMind, 2025), these advances significantly enhance perceptual grounding. However, perception alone is insufficient: embodied agents must also reason over language-conditioned goals and predict environment dynamics under action. Existing models are not inherently designed for action generation or visuomotor control (Zhao et al., 2023; Dhariwal and Nichol, 2021; Ze et al., 2024).

Vision-language modeling for robotic perception and reasoning. Vision-language pretraining grounds perception in language, providing a scalable interface for task specification and high-level reasoning (Radford et al., 2021; Zhai et al., 2023). Extending this paradigm, Vision-Language-Action (VLA) models incorporate action supervision to unify perception, language, and control (Brohan et al., 2022, 2023; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025b; Bjorck et al., 2025). Early works (Nair et al., 2022; Xiao et al., 2022) show that strong visual priors improve control, while VLA models directly map observations and instructions to actions via behavior cloning or policy learning. By transferring large-scale semantic knowledge into control, they improve instruction following and cross-task generalization, often outperforming prior robotic policies (Zhao et al., 2023; Chi et al., 2024a; Ze et al., 2024). Recent work extends these models to humanoid loco-manipulation (Wei et al., 2026). To preserve reasoning capabilities, subsequent approaches explore multimodal co-training (Driess et al., 2025; Ye et al., 2026a; Chen et al., 2025c; Zeng et al., 2024; Yang et al., 2025c; Zhou et al., 2025), while others scale teleoperated datasets (Collaboration et al., 2023; Khazatsky et al., 2024; Wu et al., 2024; AgiBot, 2025; Ebert et al., 2021; Duan et al., 2024). However, these datasets remain limited in task, language, and scene diversity (Shi et al., 2025; Chi et al., 2024b), motivating portable data collection (Generalist AI, 2025; Liu et al., 2024b; Chi et al., 2024b). Meanwhile, tightly coupled pipelines hinder reproducibility, modularity, and scalability, highlighting the need for unified frameworks.

Video-based world model for robotic dynamics and interaction. Orthogonal to language-based scaling, video-based world models learn physical dynamics via visual prediction. Video captures motion, contact, and causality more effectively than static data. Early methods augment VLA policies with predictive latent modeling (Zheng et al., 2025b; Bjorck et al., 2025; Ye et al., 2025), while large-scale video pretraining enables planning with minimal robot data (Assran et al., 2025; Jang et al., 2025). Later work treats video as a primary policy substrate, either unifying policy, simulation, and evaluation or decoupling planning from control (Du et al., 2023; Ko et al., 2024; Pai et al., 2025; Chen et al., 2025a). Action-conditioned world models further support policy evaluation and improvement: imagined rollouts achieve strong performance (Wu et al., 2023), while recent systems enable counterfactual replay and safety evaluation (1X World Model Team, 2025; Team et al., 2025). Other approaches use controllable world models for trajectory generation, reinforcement learning, or scalable data synthesis (Guo et al., 2025; Jiang et al., 2026; GigaWorld Team et al., 2025; GigaBrain Team et al., 2026; Qiu et al., 2026). Recent work emphasizes causal consistency, controllability, and closed-loop efficiency by integrating action and value prediction into pretrained video models or jointly learning dynamics and control (Kim et al., 2026; Li et al., 2026; Cai et al., 2026; Gao et al., 2026; Zhu et al., 2025; Yuan et al., 2025). Some approaches formulate joint video–action prediction as policy learning or analyze gains from test-time imagination versus co-training (Ye et al., 2026b; Yuan et al., 2026). Additionally, human video provides scalable motion priors, with egocentric pipelines enabling transferable behaviors across tasks and embodiments (Hoque et al., 2025; Yang et al., 2025b; Zheng et al., 2026).

Above vision-language pretraining and video-based world modeling scale embodied intelligence along complementary but largely fragmented axes, motivating StarVLA's design choice to separate what should vary across methods from what should remain stable across training, evaluation, and deployment.

2.2 Building VLA Frameworks on VL Foundation Models

While the foundation models surveyed above provide powerful visual-linguistic representations, they are not natively designed for action generation. A key design goal of StarVLA is to make these VL foundation models *VLA-ready*: we provide a unified I/O interface contract and a compositional architecture that allow diverse action decoding strategies to be flexibly composed on top of the same VL backbone.

Unified I/O Interface. All framework modules in StarVLA inherit from a common base class and expose two methods that share a unified *input/output (I/O) interface*: both training and inference consume raw, environment-level observations identical to what the robot receives at deployment time.

- `forward({raw images, str, ...})` $\rightarrow$ `{raw images, str, ...}`: the training entry point. It receives a batch of raw samples, each containing multi-view RGB images, a natural-language instruction, and an action chunk, and returns a loss dictionary.
- `predict_action({raw images, str, ...})` $\rightarrow$ `{normalized_actions, ...}`: the inference entry point. It accepts the same observation format (minus ground-truth actions) and returns predicted action chunks.

By deliberately adopting this unified I/O interface, where training inputs mirror real deployment observations rather than relying on heavily preprocessed dataloader tensors, we minimize train/test distribution mismatch, a common source of silent performance degradation in VLA systems.

This design choice reflects a deeper invariant of robotic deployment: regardless of how different VL foundation models are pretrained—what tokenization scheme they adopt, how they resize or partition images, or what auxiliary objectives they optimize during pretraining—at inference time every model must ultimately accept the same raw sensor streams that the physical robot provides and produce executable motor commands. The unified I/O interface codifies this *deployment-time invariant* as the system's first-class contract, ensuring that any VL model whose inference path can consume raw observations is immediately compatible with StarVLA, without requiring users to reverse-engineer or replicate model-specific preprocessing pipelines. Crucially, this same invariant-driven principle extends naturally to the internal architecture, as we describe next.

Compositional framework. Applying the same principle internally, we decompose every VLA method into two explicitly separated components connected by a standardized representation contract: a **VL backbone** (e.g., Qwen2.5-VL, z) that consumes raw multimodal observations and exposes hidden-state representations through a common output specification, and a **pluggable action head** that reads those representations through a corresponding input specification and converts them into motor commands. Each framework assembles itself through the same two-step composition (first loading the backbone, then attaching an action head), with

both components configured declaratively via YAML. Because the outer system boundary (raw observations $\rightarrow$ actions) and the inner backbone–head boundary (multimodal inputs $\rightarrow$ hidden states $\rightarrow$ actions) are both governed by standardized contracts, StarVLA achieves *bidirectional modularity*: backbone and action head can each be replaced independently without affecting the other or any surrounding infrastructure.

This modularity provides flexibility across different stages of VLA development. **For researchers**, it supports rapid experimentation in multiple directions. New action decoding paradigms can be prototyped by implementing and registering an action-head module, while new vision-language backbones—such as instruction-tuned VLMs (e.g., Qwen2.5-VL (Bai et al., 2025b), InternVL (Cai et al., 2026)) or video-native models (e.g., Cosmos (Kim et al., 2026))—can be integrated through a lightweight adapter that conforms to a shared representation interface. Once integrated, these backbones can be evaluated across different action heads without requiring per-method modification. **For training infrastructure**, the standardized interfaces allow much of the upstream and downstream stack (e.g., training pipelines, benchmark harnesses, and deployment services) to remain largely *backbone- and action-head-agnostic*, reducing the need for method-specific code paths as new paradigms or models are introduced. **For deployment**, switching between different backbones or action paradigms can be handled through configuration changes, without requiring code-level modifications.

2.3 Representative VLA Instantiations

Under this unified abstraction, we implement four paradigms spanning the major action decoding families in the current VLA literature, as illustrated in Fig. 2. All variants share the same VL backbone, the same base class, and the same `forward`/`predict_action` contract, differing only in *how they extract actions from the backbone's representations*:

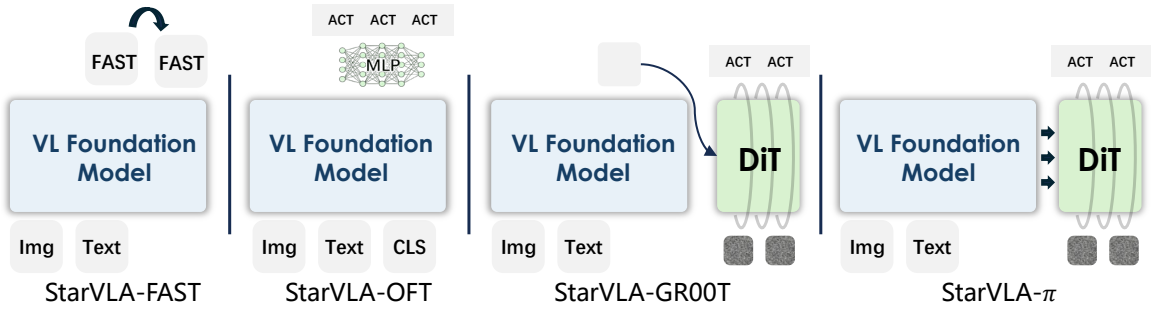

Figure 2: Overview of four representative approaches for adapting Vision-Language Models into Vision-Language-Action frameworks in StarVLA (FAST, OFT, π, and GR00T) under a unified interface.

- **StarVLA-FAST** (π_{fast}): Appends a FAST tokenizer (Pertsch et al., 2025) to the VL backbone and *autoregressively* generates discrete action tokens via next-token prediction, using the LLM's own vocabulary space.
- **StarVLA-OFT:** Attaches a lightweight MLP that reads the hidden states of predefined action tokens and *regresses* continuous actions in parallel (L1 loss), following OpenVLA-OFT (Kim et al., 2025)—the simplest form of pluggable head.i
- **StarVLA-π** (π_0): Integrates a layer-wise cross-DiT flow-matching action expert, conditioned on multi-layer VL hidden states via cross-attention, and predicts continuous actions through iterative *denoising*, following π_0 (Black et al., 2024).
- **StarVLA-GR00T:** Adopts a dual-system design where the VL backbone serves as *System 2* (slow reasoning) and a DiT-based flow-matching module serves as *System 1* (fast action generation), consistent with GR00T N1.5 (Bjorck et al., 2025). This variant demonstrates that even fundamentally different inference-time compute patterns can coexist under the same interface.

This spectrum, from VLM-native decoding (autoregressive tokenization, parallel regression) to generative-model-based decoding shared with world-model architectures (iterative flow-matching denoising, dual-system reasoning), shows that the proposed compositional architecture and unified interface are broadly applicable. Adding further paradigms requires only implementing and registering a new action head; the backbone, training loop, and evaluation pipeline remain unchanged.

3 Unified System Pipeline for Model Training and Testing

The StarVLA codebase supports several practical training regimes for VLA policies, ranging from standard supervised fine-tuning (SFT) on downstream robot datasets to multi-objective co-training with vision–language (VLM) web data and cross-embodiment co-training on mixed robot embodiments. All training pipelines are implemented in explicit PyTorch loops built on Accelerate + DeepSpeed for distributed execution, while preserving a unified YAML configuration interface across methods. Figure 3 summarizes the supported training modes and how data streams connect to the unified model framework.

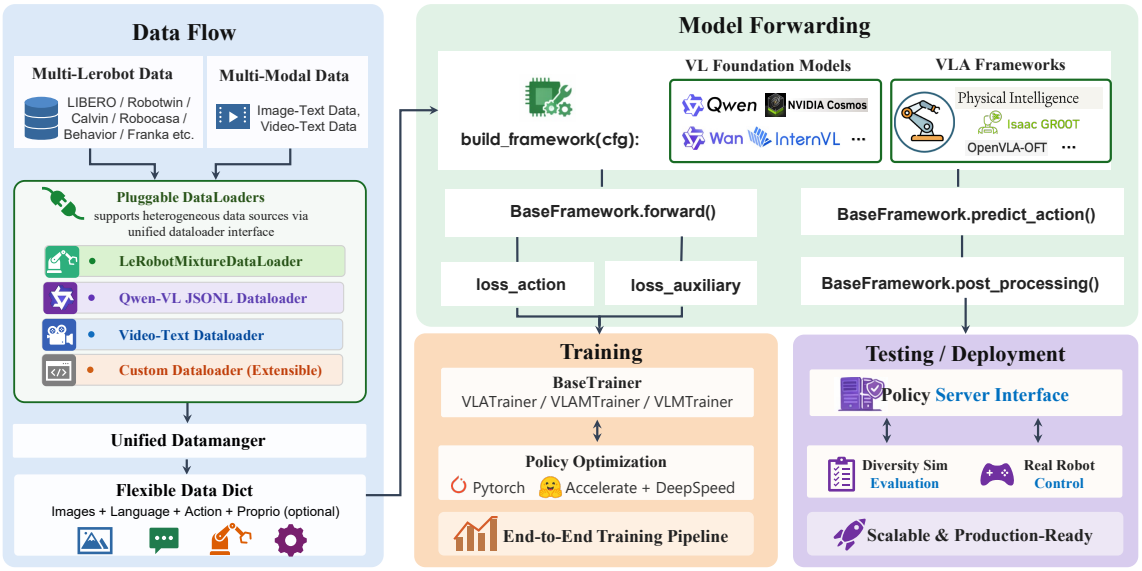

Figure 3: Overview of the StarVLA framework. We present a unified and modular pipeline that connects heterogeneous data sources, pluggable dataloaders, and flexible data representations with a standardized model forwarding interface. The framework supports diverse vision-language foundation models and VLA architectures, enabling end-to-end training and deployment.

3.1 Training Paradigms

3.1.1 Supervised Learning for Behavior Cloning

The most direct training mode is robot-only supervised learning, where the policy is trained to predict continuous actions from observations and language instructions. In our codebase, this training path is implemented in `starVLA/training/train_starvla.py`. The objective is the action modeling loss returned by the framework `forward()` method (e.g., `action_loss` in the output dict).

Optimization setup. We support (i) full-parameter fine-tuning and (ii) selective freezing of submodules via `trainer.freeze_modules` (comma-separated module paths). To stabilize training across heterogeneous components, the optimizer can use multiple parameter groups with different learning rates (e.g., separate LR for `qwen_vl_interface` and the action model) configured by `trainer.learning_rate`. Training uses bfloat16 autocast, gradient accumulation, gradient clipping, and a cosine schedule with a minimum learning rate.

3.1.2 Multi-Objective Co-Training for Embodied Reasoning

Robot-only SFT can over-specialize the VLM backbone to a narrow instruction distribution. To preserve general-purpose visual reasoning and language grounding while learning action prediction, StarVLA supports a co-training regime that interleaves robot action learning with a VLM loss on multimodal web data. This mode is implemented in `starVLA/training/train_starvla_cotrain.py`.

Dual-loader multi-objective training scheme. Co-training uses two dataloaders (VLA and VLM) and performs two forward/backward passes per optimization step: (i) a VLA forward pass through the framework

`forward()` to obtain `action_loss`, and (ii) a VLM forward pass through `qwen_vl_interface` to obtain the language modeling loss. The VLM loss is scaled by `trainer.loss_scale.vlm` in config, enabling a controlled trade-off between action learning and VLM capability retention.

3.1.3 Cross-Embodiment Co-Training with Robot Data Mixtures

To support cross-embodiment generalization, the codebase provides a unified LeRobot mixture dataset interface that allows training on heterogeneous robot datasets with different embodiments, action conventions, and camera setups. In config, users select a named mixture through `datasets.vla_data.data_mix`, which maps to a list of (dataset name, sampling weight, robot type) tuples. At runtime, the mixture is materialized as a `LeRobotMixtureDataset`, which samples trajectories across datasets according to the specified weights and tracks embodiment tags based on robot type. This design makes "cross-embodiment pretraining" an operational configuration choice, rather than a bespoke training script.

3.1.4 Reinforcement Learning Fine-Tuning

Beyond supervised and co-training regimes, we plan to support reinforcement learning (RL) fine-tuning as an extension of the same framework abstraction, collaborating with the RLinf project (`https://github.com/RLinf/RLinf`). At the time of writing, RL fine-tuning is an ongoing integration effort; the current public codebase focuses on supervised and co-training pipelines to build up a strong robotic foundation model.

3.2 Evaluation and Deployment

3.2.1 Unified Server-Client Evaluation Across Benchmarks

StarVLA adopts a thin server–client testing abstraction so that benchmark-side evaluation code remains close to the official implementations, while model-side inference is standardized. In practice, a checkpoint is loaded by `baseframework.from_pretrained()` and hosted as a lightweight WebSocket policy server in the StarVLA runtime environment. The benchmark evaluator, which may live in a different conda environment with its own simulator dependencies, interacts with the model through a small client wrapper rather than importing framework code directly. This decoupling is particularly useful for benchmarks such as LIBERO, SimplerEnv, and RoboTwin, whose official evaluators each carry different dependency stacks and control loops.

Inference interface. All framework variants expose the same inference entry point, `Framework.predict_action()`, and the server forwards incoming payload dictionaries to this method with minimal routing logic. The benchmark-side client packages observations into a single dictionary, typically containing `image` (single- or multi-view RGB observations), `lang` (task instruction), and optional fields such as `state`, timestamps, or episode metadata. The payload is serialized with `msgpack` and sent to the policy server, which returns a dictionary containing model outputs such as `normalized_actions`. Because the communication contract is action-head-agnostic, switching from OFT to FAST, π, or GR00T does not require modifying benchmark code.

Benchmark-specific adapters. In StarVLA, benchmark differences are isolated in lightweight interface files such as `model2libero_interface.py`, `model2simpler_interface.py`, and `model2robotwin_interface.py`. These adapters translate raw environment observations into the common StarVLA example format and post-process returned actions into the benchmark's native control API. Typical responsibilities include resizing images to the training resolution, reading `dataset_statistics.json` from the checkpoint directory for action unnormalization, converting chunked normalized predictions into executable actions, applying action ensembling, and handling benchmark-specific conventions such as sticky grippers or delta/relative-to-absolute action conversion. This design keeps the core policy server benchmark-agnostic while preserving faithful evaluation under each official protocol.

3.2.2 Deployment on Real Robots

The same client-server contract also supports real-robot or hosted-benchmark deployment. In this setting, the robot controller plays the role of the benchmark client: it captures camera observations, assembles the same example dictionary used in simulation, queries the remote policy server, and executes the returned action on hardware. As a result, the control loop, safety logic, and device-specific middleware remain outside the StarVLA model runtime, while the model service remains unchanged.

Deployment interface. This separation makes deployment much less intrusive. The model stack can stay in a GPU-oriented inference environment, whereas the robot-side process can remain integrated with vendor SDKs, ROS nodes, or hosted evaluation platforms such as RoboChallenge. More importantly, the exact same checkpoint can be reused across simulation and real-robot settings as long as the client provides observations in the agreed dictionary format and applies the appropriate benchmark- or robot-specific post-processing. In this sense, StarVLA treats deployment as a continuation of the same testing paradigm rather than as a separate engineering path.

4 Multiple Benchmark Integration

Recent vision-language-action (VLA) research has made rapid progress across a wide range of benchmarks. However, most existing methods are evaluated on only a limited number of environments, and their implementations often differ substantially in preprocessing pipelines, policy interfaces, and evaluation protocols. These inconsistencies hinder fair cross-paper comparison and weaken reproducibility.

4.1 Unified Benchmark Integration Interface

StarVLA aims to provide *simple and reproducible baselines* across a diverse benchmark suite by: (i) adhering as closely as possible to the official training and evaluation workflows of each benchmark, with minimal data engineering and environment-specific modifications, and (ii) standardizing the policy-side interface. Concretely, all StarVLA variants expose a unified lightweight WebSocket service, enabling different benchmark runners to interact with a shared inference endpoint. This design facilitates seamless integration and simplifies scaling to additional benchmarks.

To facilitate reproducibility, StarVLA defines a unified integration interface for benchmark onboarding. Specifically, each benchmark integration is structured around three aligned components: **(i)** a checkpoint package containing the saved `config.yaml` and `dataset_statistics.json`, **(ii)** a runnable training entry (YAML configuration and launch script under `examples/<BENCH>/train_files/`), and **(iii)** a runnable evaluation workflow that launches a policy server and invokes the official benchmark evaluator (typically under `examples/<BENCH>/eval_files/`). This design ensures that benchmark-specific workflows remain reproducible while maintaining a consistent policy interface across environments.

4.2 Supported Benchmark Suite

StarVLA integrates a diverse set of manipulation benchmarks spanning different simulators, embodiments, and protocols, including *LIBERO*, *SimplerEnv*, *RoboTwin 2.0*, *RoboCasa GR1 Tabletop Tasks*, and *BEHAVIOR-1K*. The experiments section reports detailed results and comparisons under each benchmark's official evaluation protocols.

LIBERO. LIBERO (Liu et al., 2024a) is a widely used benchmark for language-conditioned robot manipulation and lifelong robot learning. It contains 130 manipulation tasks organized into four suites: Spatial, Object, Goal, and Long, each targeting a different form of generalization, including spatial variation, object-centric manipulation, goal-conditioned execution, and long-horizon dependencies. A standard training protocol uses 50 demonstrations per task, resulting in approximately 6.5K trajectories in total. LIBERO provides a standardized evaluation protocol and serves as a comprehensive testbed for instruction following, compositional generalization, and multi-task policy learning.

LIBERO-Plus. LIBERO-Plus (Fei et al., 2025) is a robustness-oriented benchmark built on top of LIBERO for systematically evaluating the generalization ability of vision-language-action models under distribution shifts. It expands the original benchmark by introducing perturbations over seven factors, including object layout, camera viewpoints, robot initial states, language instructions, lighting, background textures, and sensor noise. The final benchmark is a test-only evaluation set with 10,030 tasks spanning 7 perturbation factors and 21 low-level components.

SimplerEnv. SimplerEnv (Li et al., 2024b) is a simulation-based evaluation benchmark designed as a scalable proxy for real-world robot evaluation. It provides standardized simulated environments corresponding to common real-robot platforms, including the WidowX (BridgeData V2) and the Google Robot (RT-series) setups. The benchmark defines fixed evaluation protocols such as Visual Matching and Variant Aggregation, as well as standardized success-rate aggregation rules. Although it does not specify a fixed number of tasks

or dataset size, it is widely used to evaluate policies trained on real-world data under reproducible simulated conditions, with prior work showing strong correlation between simulated and real-world performance.

RoboCasa-GR1. RoboCasa-GR1 (Nasiriany et al., 2024; Bjorck et al., 2025) is a tabletop manipulation benchmark built on the RoboCasa simulation framework, commonly used to evaluate humanoid-style manipulation policies. Compared with standard single-arm setups, it introduces more complex embodiments and household interaction scenarios involving articulated objects and multi-stage tasks. The benchmark contains 24 tasks, with approximately 1,000 demonstrations per task, resulting in around 24K trajectories in total.

RoboTwin 2.0. RoboTwin 2.0 (Chen et al., 2025b) is a large-scale benchmark for bimanual robotic manipulation, focusing on dual-arm coordination across diverse scenarios. It contains 50 tasks with two evaluation setups: clean and randomized. Each task includes 50 clean demonstrations together with 500 randomized demonstrations, resulting in approximately 550 trajectories per task and 27.5K trajectories in total. The randomized data is generated via structured domain randomization, including variations in scene clutter, backgrounds, table height, and lighting, providing a challenging testbed for both coordination and robustness. For evaluation, each task is tested for 100 episodes under each setup. In total, this results in 50 tasks $\times$ 2 setups $\times$ 100 episodes, equals to 10,000 evaluation trials.

BEHAVIOR-1K. BEHAVIOR-1K (Li et al., 2023) is a large-scale benchmark for human-centered embodied AI, built around everyday activities. It defines 1,000 activities across 50 interactive scenes with more than 9,000 objects, covering environments such as homes, offices, and restaurants. Built on OmniGibson, it supports realistic physics for rigid, deformable, and liquid objects, and emphasizes long-horizon interaction requiring perception, navigation, and manipulation. An active evaluation setting is the BEHAVIOR Challenge, which selects 50 household tasks from the activity set and provides 10,000 teleoperated demonstrations (over 1,200 hours), with 200 demonstrations per task released for training. For evaluation, each task includes 20 additional instances with varying initial conditions, of which 10 are used for reporting, and each instance is evaluated once with a fixed timeout. Performance is measured by the average task success rate across all tasks, with partial credit based on goal completion.

CALVIN. CALVIN (Mees et al., 2022) is a benchmark for long-horizon language-conditioned manipulation, designed to evaluate whether a single policy can execute sequences of natural-language instructions from visual observations. It comprises four environments (A, B, C, and D) and 34 manipulation tasks involving articulated objects and stateful scene elements such as drawers, sliding doors, lights, and switches. The standard evaluation follows the ABC$\rightarrow$D setting, where policies are trained on A–C and tested on D using 1,000 task sequences of length 5. Performance is reported by the average length of successfully completed subtask sequences.

5 Single-Benchmark Training Examples

In this section, we report *single-benchmark SFT results* to establish transparent, reproducible reference points under official evaluation protocols. To provide the community with the cleanest possible baselines, we deliberately avoid any VLA-specific pretraining (e.g., large-scale robot pretraining mixtures), data augmentation, or online refinement techniques such as DAgger. Every model is initialized from publicly released VL pretrained weights and fine-tuned exclusively on the benchmark's standard demonstration dataset. These minimal-assumption results serve as reliable *anchor points* for future research: they make it straightforward to measure the marginal value of additional pretraining data, augmentation strategies, or co-training recipes.

5.1 Results on LIBERO

LIBERO (Liu et al., 2024a) is a widely used tabletop manipulation benchmark comprising four task suites of increasing difficulty: **Spatial**, **Object**, **Goal**, and **Long**. We treat it as the first worked example of our single-benchmark pipeline and walk through every step—data loading, training, and evaluation—so that readers can fully reproduce our numbers.

Training data format. To maintain a simple and reproducible baseline, we adopt minimal data engineering and follow the benchmark's native schema.

- `Input: a raw sample dict loaded directly from the LeRobot-format dataset, containing the primary (third-person) RGB view and the wrist-camera RGB view. We do not use proprioceptive state, history stacking, or image augmentation for this baseline.`

- `Output: a continuous end-effector (EEF) control action vector following the LIBERO action definition with action chunking`$= 8$.

Training setup. We train the LIBERO baseline using distributed training with 8 A100 GPUs (via `accelerate` + DeepSpeed ZeRO-2). Unless otherwise specified, the per-device batch size is 16 and training runs for 100K optimization steps. Checkpoints are saved every 10K steps, with periodic logging and evaluation during training. For transparency and exact reproducibility (full command line, YAML configuration, and environment variables),we provide the complete training scripts under `examples/LIBERO/train_files/`. We train a single policy jointly on four LIBERO suites (Spatial, Object, Goal, and LIBERO-10) using the corresponding LeRobot-format datasets: They are available as a public collection at `https://huggingface.co/collections/IPEC-COMMUNITY/libero-benchmark-dataset`.

Evaluation protocol. We evaluate on the four suites (Spatial, Object, Goal, and `LIBERO-Long`) using the official LIBERO evaluation scripts and report success rate. We periodically evaluate checkpoints (every 10K steps by default) and report the earliest checkpoint that achieves the best average success rate. For each suite, we run 10 tasks with 50 episodes per task (500 trials total) and report the mean success rate over all trials. To ensure reproducibility without modifying benchmark logic, we provide the complete evaluation scripts and launch instructions under `examples/LIBERO/eval_files/`.

Results and analysis. Table 2 summarizes the LIBERO baseline performance. Using only 30K steps ($\sim$10 epochs), StarVLA already matches or surpasses several strong published baselines. For instance, OpenVLA-OFT trains for 175K steps (223 epochs) to reach 97.1% average, whereas StarVLA-OFT achieves 96.6% (Qwen3-VL) and 95.8% (Cosmos-Predict2-2B) with $6\times$ fewer steps and $23\times$ fewer epochs. π_0+FAST and GR00T-N1.5 score 85.5% and 86.5% respectively, both considerably below our variants. Notably, replacing the VL backbone from Qwen3-VL-4B to Cosmos-Predict2-2B yields comparable performance (average $\geq$95.2% across all action heads), demonstrating that StarVLA generalizes well across different VL backbones. These comparisons suggest that the StarVLA pipeline is highly data-efficient on LIBERO.

Table 2: Comparison of different VLA models on LIBERO. We train one policy for all 4 suites. All scores are averaged over 500 trials for each task suite (10 tasks × 50 episodes).

Model	Steps	Epochs	Spatial	Object	Goal	Long	Avg
π_0+FAST Pertsch et al. (2025)	-	-	96.4	96.8	88.6	60.2	85.5
OpenVLA-OFT Kim et al. (2025)	175K	223	97.6	98.4	97.9	94.5	97.1
π_0	-	-	96.8	98.8	95.8	85.2	94.1
GR00T-N1.5 Bjorck et al. (2025)	20K	203	92.0	92.0	86.0	76.0	86.5
VL = Qwen3-VL-4B							
StarVLA-FAST	30K	9.54	97.3	97.4	96.3	90.6	95.4
StarVLA-OFT	30K	9.54	97.8	98.6	96.2	93.8	96.6
StarVLA-π	30K	9.54	98.8	99.6	95.8	88.4	95.7
StarVLA-GR00T	30K	9.54	97.8	98.8	97.4	92.0	96.5
VL = Cosmos-Predict2-2B							
StarVLA-OFT	30K	9.54	98.6	97.6	95.0	91.8	95.8
StarVLA-π	30K	9.54	98.9	98.3	94.4	90.4	95.5
StarVLA-GR00T	30K	9.54	97.4	98.0	95.1	90.4	95.2

5.2 Results on SimplerEnv

Training setup. All models are trained with full-parameter fine-tuning using distributed training on 16 A100 GPUs. Unless otherwise specified, the per-device batch size is 16 and training runs for 100K optimization steps. Checkpoints are saved every 10K steps, with periodic logging and evaluation during training. For transparency and exact reproducibility (full command line, YAML configuration, and environment variables), we provide the complete training scripts under `examples/SimplerEnv/train_files/`. We train SimplerEnv baselines on a merged mixture of Bridge and Fractal datasets in LeRobot format: `https://huggingface.co/datasets/IPEC-COMMUNITY/bridge_orig_lerobot` and `https://huggingface.co/datasets/IPEC-COMMUNITY/fractal20220817_data_lerobot`.

Evaluation protocol. We evaluate using the official SimplerEnv evaluation workflow and report the task success rate. We present detailed per-task results under two standard SimplerEnv settings: (i) WidowX

robot with Visual Matching (VM) in Table 3, and (ii) Google Robot in Table 4. We strictly follow the official protocol for per-task repeats/episodes and success-rate aggregation without modifying benchmark logic. Since SimplerEnv evaluation can exhibit non-trivial variance, we run each reported setting five times (each time rerunning the full official evaluation) and report the mean success rate. To ensure reproducibility without modifying benchmark logic, we provide the complete evaluation scripts and launch instructions under `examples/SimplerEnv/eval_files/`.

Results. Tables 3 and 4 summarize the SimplerEnv performance. On WidowX (VM), StarVLA with Qwen3-VL-4B achieves a strong average success rate (up to 65.3%), while the Cosmos-Predict2-2B backbone also delivers competitive results (up to 61.6%), confirming that StarVLA generalizes across different VL backbones. Both configurations show consistently high performance on the most structured task and a remaining gap on object placement tasks. On Google Robot, StarVLA is competitive with or better than strong recent baselines under the Visual Matching setting, and remains comparable under Variant Aggregation, suggesting that the policy transfers robustly across standardized simulation evaluation settings.

Table 3: Detailed results on the SimplerEnv WidowX benchmark (Visual Matching). Steps denote optimization steps; all numbers are success rates (%).

WidowX Robot	Method	Steps	Put Spoon on Towel	Put Carrot on Plate	Stack Green Block on Yellow Block	Put Eggplant in Yellow Basket	Average
SIMPLERENV Visual Matching	RT-1-X Brohan et al. (2022)	-	0.0	4.2	0.0	0.0	1.1
	Octo-Base Octo Model Team et al. (2024)	-	15.8	12.5	0.0	41.7	17.5
	Octo-Small Octo Model Team et al. (2024)	-	41.7	8.2	0.0	56.7	26.7
	OpenVLA Kim et al. (2024)	-	4.2	0.0	0.0	12.5	4.2
	CogACT Li et al. (2024a)	-	71.7	50.8	15.0	67.5	51.3
	SpatialVLA Qu et al. (2025)	-	16.7	25.0	29.2	**100.0**	42.7
	π_0 Black et al. (2024)	-	29.1	0.0	16.6	62.5	27.1
	π_0-FAST Pertsch et al. (2025)	-	29.1	21.9	10.8	66.6	48.3
	GR00T N1.5 Bjorck et al. (2025)	-	75.3	54.3	**57.0**	61.3	61.9
	Magma Yang et al. (2025a)	-	37.5	31.0	12.7	60.5	35.8
	VL = Qwen3-VL-4B						
	StarVLA-FAST	15K	18.8	31.3	4.2	71.9	31.6
	StarVLA-OFT	65K	**90.3**	38.5	29.7	**100.0**	64.6
	StarVLA-π	40K	78.1	46.9	30.2	88.5	60.9
	StarVLA-GR00T	20K	83.0	59.4	18.8	**100.0**	65.3
	VL = Cosmos-Predict2-2B						
	StarVLA-OFT	30K	66.8	62.6	25.3	90.2	61.2
	StarVLA-π	30K	81.4	55.2	25.1	73.0	58.7
	StarVLA-GR00T	30K	80.4	65.4	20.0	80.6	61.6

5.3 Results on RoboCasa-GR1

Training setup. We train the RoboCasa-GR1 baselines with distributed full-parameter fine-tuning on 16 A100 GPUs. Unless otherwise specified, the per-device batch size is 16 and training runs for up to 100K optimization steps. Checkpoints are saved every 10K steps, with periodic logging and evaluation during training. For the specialist setting, we use the official RoboCasa-GR1 tabletop release and train one model jointly across all 24 tasks from this benchmark only. This keeps the policy architecture fixed while treating RoboCasa as a multi-task humanoid-style manipulation suite rather than 24 separate single-task runs.

Evaluation protocol. We follow the official RoboCasa-GR1 evaluation workflow and report average success rate over the 24 tasks. For the architecture comparison in this section, each model is evaluated with 50 rollouts per task. Table 6 further reports the task-level success rates for representative baselines and StarVLA variants.

Results. Table 5 summarizes the average RoboCasa-GR1 performance for the single-benchmark setting. This benchmark is noticeably harder than LIBERO and SimplerEnv, and the choice of action head matters more: the discrete StarVLA-FAST baseline reaches 39.0%, while the continuous-action variants improve to 43.9–48.8%. Among the StarVLA variants, StarVLA-OFT performs best with a 48.8% average success rate, slightly exceeding StarVLA-GR00T (47.8%) and outperforming $\pi_{0.5}$ by 11.8 points. Detailed task-level results are reported in Table 6. We defer cross-benchmark generalist results to Sec. 7.

5.4 Results on Robotwin 2.0

Training setup. We train the RoboTwin 2.0 baseline using distributed training with 48 A100 GPUs (via `accelerate` + DeepSpeed ZeRO-2). Unless otherwise specified, the per-device batch size is 4

Table 4: Detailed results on the SimplerEnv Google Robot benchmark. Numbers are officially reported unless marked with $*$, which denotes our reimplementation. We report StarVLA-OFT with Qwen3-VL-4B as a representative configuration due to the high evaluation cost on this platform.

Google Robot	Models	Pick Coke Can	Move Near	Open/Close Drawer	Open Top Drawer and Place Apple	Avg
Visual Matching	RT-1 Brohan et al. (2022)	85.7	44.2	**73.0**	6.5	52.4
	RT-1-X Collaboration et al. (2023)	56.7	31.7	59.7	21.3	42.4
	RT-2-X Brohan et al. (2023)	78.7	77.9	25.0	3.7	46.3
	OpenVLA Kim et al. (2024)	18.0	56.3	63.0	0.0	34.3
	CogACT Li et al. (2024a)	91.3	85.0	71.8	50.9	74.8
	SpatialVLA Qu et al. (2025)	86.0	77.9	57.4	-	75.1
	π_0 Black et al. (2024)	72.7	65.3	38.3	-	58.8
	π_0-FAST Pertsch et al. (2025)	75.3	67.5	42.9	-	61.9
	GR00T N1.5* Bjorck et al. (2025)	51.7	54.0	27.8	7.4	35.2
	Magma Yang et al. (2025a)	83.7	65.4	56.0	6.4	52.9
	StarVLA-OFT	**95.3**	75.0	68.8	66.1	76.0
Variant Aggregation	RT-1 Brohan et al. (2022)	89.8	50.0	32.3	2.6	43.7
	RT-1-X Collaboration et al. (2023)	49.0	32.3	29.4	10.1	30.2
	RT-2-X Brohan et al. (2023)	82.3	79.2	35.3	20.6	54.4
	OpenVLA Kim et al. (2024)	60.8	67.7	28.8	0.0	39.3
	CogACT Li et al. (2024a)	89.6	80.8	28.3	46.6	61.3
	SpatialVLA Qu et al. (2025)	88.0	**82.5**	41.8	-	70.7
	π_0 Black et al. (2024)	75.2	63.7	25.6	-	54.8
	π_0-FAST Pertsch et al. (2025)	77.6	68.2	31.3	-	59.0
	GR00T N1.5 Bjorck et al. (2025)	69.3	68.7	35.8	4.0	44.5
	Magma Yang et al. (2025a)	68.8	65.7	53.4	18.5	51.6
	StarVLA-OFT	91.3	75.1	55.0	59.4	70.2

Table 5: Average success rate on RoboCasa-GR1 (24 tasks) under the single-benchmark training setting.

Method	**Avg (%)**	**Method**	**Avg (%)**
$\pi_{0.5}$ Intelligence et al. (2025a)	37.0	GR00T-N1.6 Bjorck et al. (2025)	47.6
StarVLA-FAST	39.0	StarVLA-π	43.9
StarVLA-GR00T	47.8	StarVLA-OFT	**48.8**

and training runs for 150K optimization steps. Checkpoints are saved every 10K steps, with periodic logging and evaluation during training. For transparency and exact reproducibility (full command line, YAML configuration, and environment variables), we provide the complete training scripts under `examples/Robotwin/train_files/`. We train RoboTwin 2.0 baselines on official clean and randomized datasets in LeRobot format: `https://huggingface.co/datasets/StarVLA/RoboTwin-Clean` and `https://huggingface.co/datasets/StarVLA/RoboTwin-Randomized`.

Evaluation protocol. We evaluate on the 50 tasks using the official RoboTwin 2.0 evaluation scripts and report success rate. We periodically evaluate checkpoints (every 10K steps by default) and report the earliest checkpoint that achieves the best average success rate. For each suite, we run 50 tasks with 100 episodes per task under clean and randomized condition (10000 trials total) and report the mean success rate over all trials. To ensure reproducibility without modifying benchmark logic, we provide the complete evaluation scripts and launch instructions under `examples/Robotwin/eval_files/`.

Results. Table 7 summarizes the RoboTwin baseline performance. Under Qwen3-VL-4B backbones, all four StarVLA variants achieve strong average success rates when trained as a single unified policy over 50 tasks, demonstrating that our end-to-end baseline pipeline (data $\rightarrow$ training $\rightarrow$ evaluation) is reliable and reproducible.

Table 6: RoboCasa GR1 Tabletop Tasks Evaluation Results. A single model was trained for all 24 tasks. Results are reported over 50 rollouts per task (average success rate with 250 rollouts: 48.97%).

Task	GR00T-N1.6	StarVLA-GR00T	StarVLA-π	StarVLA-OFT	StarVLA-FAST
PnPBottleToCabinetClose	**51.5**	46.0	26.0	**30.0**	38.0
PnPCanToDrawerClose	13.0	80.0	62.0	76.0	44.0
PnPCupToDrawerClose	8.5	54.0	42.0	**44.0**	**56.**0
PnPMilkToMicrowaveClose	14.0	48.0	**50.0**	**44.0**	44.0
PnPPotatoToMicrowaveClose	41.5	28.0	**42.0**	**32.0**	14.0
PnPWineToCabinetClose	16.5	**46.0**	32.0	**36.0**	14.0
PnPNovelFromCuttingboardToBasket	**58.0**	48.0	40.0	**50.0**	54.0
PnPNovelFromCuttingboardToCardboardbox	**46.5**	40.0	46.0	**40.0**	42.0
PnPNovelFromCuttingboardToPan	68.5	68.0	**70.0**	70.0	58.0
PnPNovelFromCuttingboardToPot	**65.0**	52.0	40.0	54.0	58.0
PnPNovelFromCuttingboardToTieredbasket	46.5	**56.0**	44.0	**38.0**	40.0
PnPNovelFromPlacematToBasket	**58.5**	42.0	44.0	**32.0**	36.0
PnPNovelFromPlacematToBowl	57.5	**44.0**	52.0	**58.0**	38.0
PnPNovelFromPlacematToPlate	**63.0**	48.0	50.0	**52.0**	42.0
PnPNovelFromPlacematToTieredshelf	**28.5**	18.0	28.0	**24.0**	18.0
PnPNovelFromPlateToBowl	57.0	**60.0**	52.0	**60.0**	52.0
PnPNovelFromPlateToCardboardbox	43.5	**50.0**	40.0	**50.0**	30.0
PnPNovelFromPlateToPan	51.0	**54.0**	36.0	**66.0**	48.0
PnPNovelFromPlateToPlate	78.7	**70.0**	48.0	**68.0**	50.0
PnPNovelFromTrayToCardboardbox	**51.5**	38.0	34.0	44.0	28.0
PnPNovelFromTrayToPlate	**71.0**	56.0	64.0	**56.0**	34.0
PnPNovelFromTrayToPot	**64.5**	50.0	44.0	**62.0**	46.0
PnPNovelFromTrayToTieredbasket	**57.0**	36.0	50.0	**54.0**	36.0
PnPNovelFromTrayToTieredshelf	**31.5**	16.0	28.0	**30.0**	16.0
Average	**47.6**	**47.8**	**43.9**	**48.8**	39.0

Table 7: Detailed results on the RoboTwin 2.0 benchmark. We report different StarVLA model architecture on this platform.

Method	Clean	Random	Method	Clean	Random
π_0 Black et al. (2024)	65.9	58.4	$\pi_{0.5}$ Intelligence et al. (2025a)	82.7	76.8
X-VLA Zheng et al. (2025a)	72.9	72.8	Lingbot-VLA Wu et al. (2026)	88.6	86.7
StarVLA-FAST	72.5	83.2	StarVLA-OFT	88.2	88.3
StarVLA-GR00T	88.0	88.5	StarVLA-π	88.1	88.8

6 Multimodal Co-Training Examples

Beyond single-benchmark supervised fine-tuning, StarVLA natively supports *multimodal co-training*, in which the VLM backbone is jointly optimized on both robot action data and auxiliary vision-language tasks (e.g., spatial grounding, visual question answering, and captioning). The motivation is twofold: (i) action-only fine-tuning can rapidly degrade the pre-trained multimodal representations, undermining instruction comprehension and spatial reasoning; and (ii) co-training with carefully curated auxiliary data can align the optimization dynamics of perception and control, leading to better-performing policies.

When a pre-trained VLM is fine-tuned exclusively on action prediction, it tends to "forget" pre-trained visual and linguistic capabilities within thousands of steps. This manifests as degraded object grounding, instruction following, and scene understanding, all of which are prerequisites for robust manipulation. Co-training with multimodal grounding data counteracts this forgetting by maintaining the gradient flow through perception-relevant pathways.

6.1 Experimental Setup

Co-training setup. StarVLA provides built-in support for mixing heterogeneous data sources during training. Users can specify arbitrary combinations of action datasets and VLM-style QA datasets in a single configuration file; the framework handles tokenization, loss masking, and gradient accumulation transparently across data types. This makes it straightforward to reproduce co-training recipes, for instance, mixing OXE action data with RefCOCO spatial grounding or LLaVA-style visual QA data, without modifying the training loop.

Evaluation and baselines. To illustrate the effect, we summarize a spatially guided co-training study built on the StarVLA codebase (Ye et al., 2026a). This study compares three training strategies: (1) *Vanilla VLA*, which fine-tunes only on action data, (2) *Vanilla co-training VLA*, which jointly optimizes on spatial grounding and action data, and (3) *Spatially guided training VLA*, which additionally incorporates spatial pre-training and spatial prompting during co-training.

6.2 Main Results for Multimodal Co-training

Figure 4 visualizes the interaction between spatial perception (measured by IoU@0.5 on RefCOCO-g) and manipulation performance (WidowX success rate) across training steps. Vanilla VLA suffers rapid perception degradation: RefCOCO-g performance drops to near-random levels within 20K steps. Vanilla co-training partially preserves perception but exhibits unstable oscillations. The spatially guided StarVLA (ST4VLA (Ye et al., 2026a)) variant achieves the best balance, maintaining $\sim$70% of original grounding performance while reaching strong manipulation success.

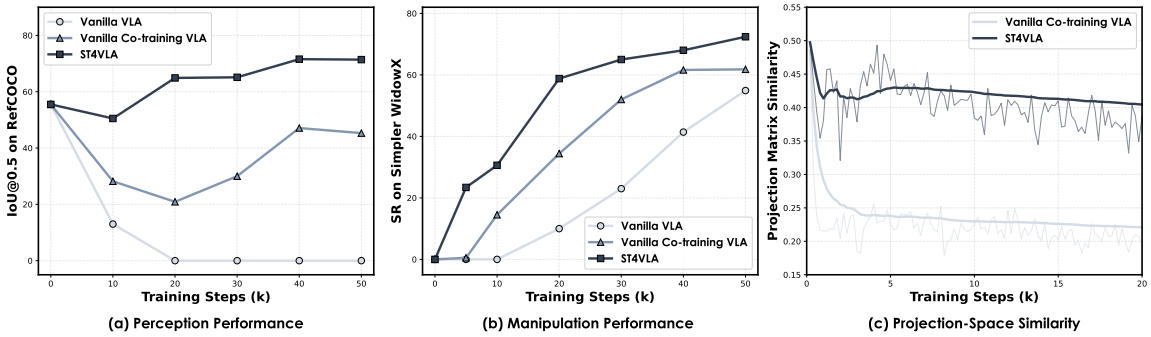

Figure 4: Perception–action co-optimization dynamics under different co-training strategies (reproduced from a StarVLA-based spatially guided co-training study ST4VLA (Ye et al., 2026a)). From left to right: (a) spatial grounding performance (IoU@0.5 on RefCOCO-g); (b) manipulation success rate (WidowX); (c) gradient subspace alignment (PSS) between spatial grounding and action objectives under vanilla co-training vs. spatially guided co-training.

Table 8 further quantifies the impact of co-training on multimodal understanding, spatial grounding, and robotic manipulation. Compared to the vanilla VLA, vanilla co-training already improves manipulation performance (+4.1% Google Robot VM, +6.4% WidowX) while recovering multimodal capabilities. The spatially guided StarVLA variant pushes the results further, achieving 84.6%/75.9% on Google Robot VM/VA and 73.2% on WidowX, while simultaneously preserving strong spatial grounding (71.2 IoU@0.5 on RefCOCO-g).

Table 8: Effect of co-training strategies on multimodal understanding, spatial grounding, and robotic manipulation (from a StarVLA-based spatially guided co-training study Ye et al. (2026a)).

	Multi-modal Understanding				**Spatial Grounding**		**Robotic Manipulation**	
Training Strategy	MME	MMVet	TextVQA	POPE Acc	RefCOCO-g IoU@0.5	RoboRefIt Acc@0.5	Google Robot VM / VA	WidowX VM
Vanilla VLA	–	–	–	–	–	–	66.1 / 63.5	54.7
+ Co-training	1106	19.2	20.5	78.0	47.1	66.7	70.2 / 66.5	61.1
+ Spatially guided	1374	23.0	28.4	84.6	68.1	72.5	78.8 / 70.0	67.4
+ Spatially pretrained	1411	23.3	28.6	86.2	71.2	74.3	84.6 / 75.9	73.2

Takeaways. These results demonstrate that StarVLA's co-training infrastructure enables significant gains over action-only fine-tuning. By preserving multimodal understanding during policy learning, co-training yields more generalizable agents. For a comprehensive treatment of spatially guided co-training, including the full training recipe, gradient alignment analysis, and extensive real-world experiments, we refer readers to the ST4VLA Ye et al. (2026a), a study paper based on StarVLA.

Table 9: **Performance comparison between generalist and specialist settings.** Specialist represents multiple models trained separately on each benchmark-specific dataset, while Generalist represents a single model jointly trained across all datasets.

Settings	Method	LIBERO					SimplerEnv			RoboTwin 2.0			RoboCasa-GR1
		Spatial	Object	Goal	Long	avg	WidowX	Google VA	Google VM	clean	clean*	random*	(avg of 24 tasks)
Specialist	$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9	46.9	68.4	72.7	**60.2**	82.7	76.8	37.0
	GR00T-N1.6	97.5	98.5	97.5	94.4	94.1	67.8	41.5	35.2	–	–	–	47.6
	StarVLA-π	98.0	99.2	98.2	93.6	98.1	65.9	72.8	76.6	50.8	88.1	**88.8**	48.9
	StarVLA-GR00T	98.9	99.6	98.4	**95.3**	98.7	65.3	70.7	75.3	48.8	88.0	88.5	52.8
	StarVLA-OFT	**99.0**	**99.8**	98.5	94.1	**98.8**	64.6	70.2	76.0	53.4	88.2	88.3	53.8
Generalist	StarVLA	98.7	99.7	**98.6**	94.2	97.8	**70.2**	**73.8**	**79.3**	–	**88.7**	87.8	**57.3**

7 Cross-Benchmark Training Examples

Building on the benchmark-wise specialist baselines in Sec. 5, we next evaluate a stricter setting for embodied generalization: *one model jointly trained across benchmarks and robot embodiments*. StarVLA natively supports co-training on heterogeneous datasets under a unified framework, which makes this all-in-one setting a natural case study for generalist VLA training.

Existing evaluation patterns. The Embodied AI community shares a common ambition: to develop a generalist agent that can seamlessly operate across diverse tasks, environments, and robots. In practice, however, the research landscape remains fragmented. Many state-of-the-art systems are tuned for specific benchmarks, and their performance can drop substantially when transferred to different environments or embodiments. This makes it difficult to measure true generalization ability.

7.1 Experimental Setups

Training setup. In this setting, we jointly train one model on the merged training sets from LIBERO, SimplerEnv, RoboTwin 2.0, and RoboCasa-GR1, and then directly evaluate on each benchmark under its official protocol. No additional benchmark-specific fine-tuning is applied. We set the learning rate to 1×10^{-4}, the total batch size to 256, and train jointly on the merged benchmark datasets. To handle action-space differences across embodiments, we avoid task-specific action heads and apply a simple unified padding strategy that expands lower-DoF actions to a shared 32-dimensional action vector.

Evaluation protocol as a generalist. A practical way to test generalization is to require one model to handle diverse benchmarks simultaneously. Following this principle, we evaluate StarVLA under a unified multi-benchmark setting, where a single policy is trained once and evaluated across suites without benchmark-specific fine-tuning.

Baselines. To further demonstrate the effectiveness of our method and the proposed setting, we report both specialist results, where models are trained only on individual datasets, and results from the generalist training setting. In addition to comparing with our model, we also evaluate several state-of-the-art methods, such as $\pi_{0.5}$ and GR00T-N1.6.

7.2 Main Results as a Generalist

As shown in Table 9, we compare our generalist model (jointly trained across datasets) with specialist models trained per benchmark. The generalist model remains competitive across most benchmarks and improves RoboCasa-GR1 from the best specialist average of 48.8% to 57.3% on the 24-task average. These results support the feasibility of a single policy that transfers across tasks and embodiments under a unified training/evaluation setting.

Takeaways This section focuses on a direct capability demonstration rather than ablation analysis: StarVLA can jointly train on heterogeneous, cross-embodiment benchmark datasets and produce a single model that remains competitive across diverse evaluation suites. We view this as evidence that all-in-one multi-benchmark training is a practical path toward large-scale cross-embodiment pretraining for future generalist VLA systems.

8 Computation Efficiency

This section reports the training efficiency of StarVLA using the public profiling measurements collected in issue `https://github.com/starVLA/starVLA/issues/158`. Our goal is to provide actionable scaling guidance for practitioners, while keeping the reported metrics aligned with common distributed-training bottlenecks (compute and communication).

Table 10: Single-node training efficiency (8 × A100). Sample throughput is derived from the measured time per 100K steps and the global batch size.

Per-GPU batch	Global batch	Time / 100K steps	Seconds / step	Samples / s	GPU util
2	16	19:32:17	0.703	22.7	74%
4	32	24:35:59	0.886	36.1	89%
8	64	31:25:38	1.131	56.6	92%
16	128	49:15:53	1.774	72.2	91%
24	192	66:47:02	2.404	79.9	96%

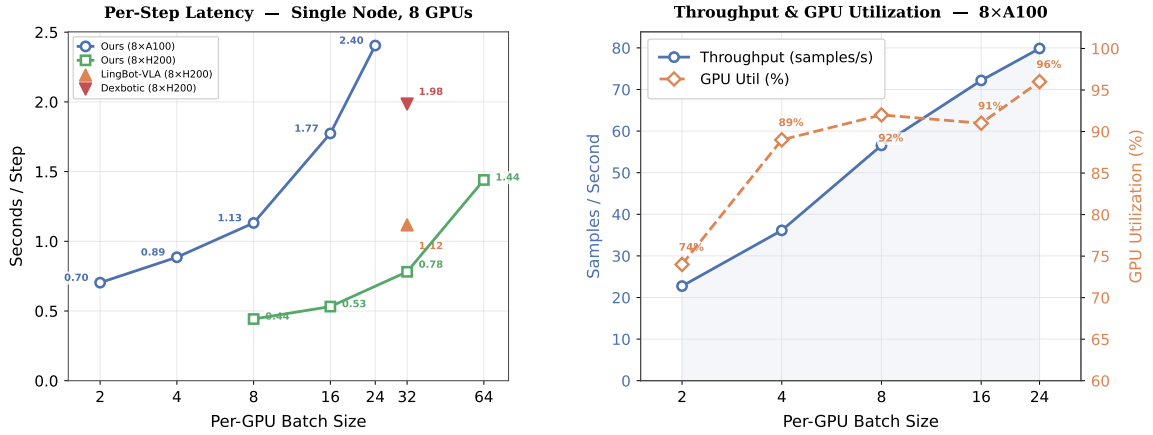

Figure 5: **Per-step latency and throughput on a single 8-GPU node.** Left: step latency as a function of per-GPU batch size for our method on A100 and H200, compared with LingBot-VLA and Dexbotic (both on 8×H200). Right: training throughput and GPU utilization on 8×A100 across batch sizes.

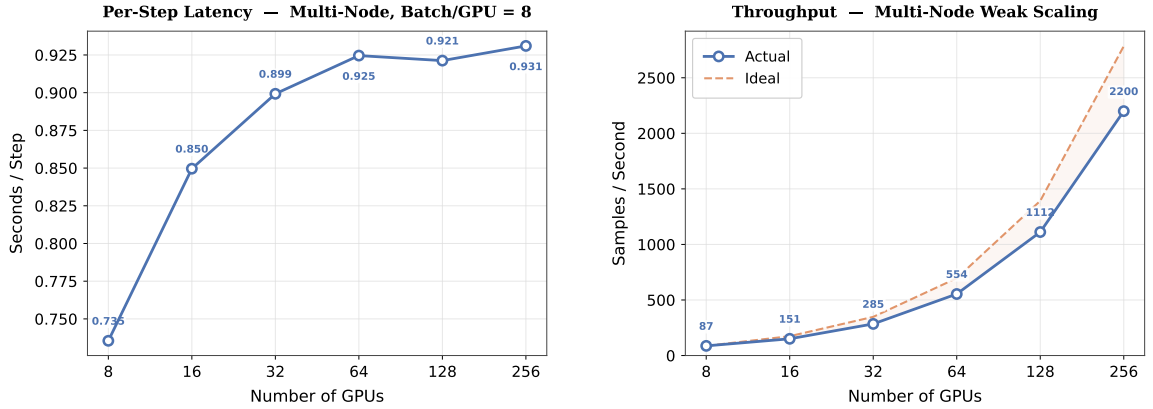

Figure 6: **Multi-node scaling efficiency.** Left: per-step latency rises noticeably from 8 to 32 GPUs due to inter-node communication overhead, then plateaus between 64 and 256 GPUs. Right: measured sample throughput versus ideal linear scaling; parallel efficiency stabilizes around 79–80% beyond 32 GPUs.

Table 11: Multi-node training efficiency (per-GPU batch = 8). "Ideal" scaling assumes linear growth of samples/s from the 8-GPU baseline.

# GPUs	Global batch	Time / 100K steps	Seconds / step	Samples / s	Scaling eff.
8	64	20:25:48	0.735	87.0	100%
16	128	23:36:00	0.850	150.7	86.7%
32	256	24:58:45	0.899	284.7	81.9%
64	512	25:40:59	0.925	553.8	79.6%
128	1024	25:35:26	0.921	1111.5	79.9%
256	2048	25:51:41	0.931	2200.0	79.1%

Experimental setup. Unless otherwise specified, the measurements use StarVLA-GR00T with a Qwen3-VL-4B backbone trained on the RoboCasa-GR1 dataset on A100 80GB GPUs. We report wall-clock time per 100K optimization steps, which includes distributed communication and system overhead.

Efficiency metrics. We distinguish two throughput notions: (i) *step throughput* (lower seconds/step is better), and (ii) *sample throughput* (higher samples/s is better), where samples/s is computed as global batch$/$(seconds per step). This distinction is important because distributed scaling often decreases step throughput (due to synchronization) while increasing sample throughput (due to larger global batch).

8.1 Single-Node Training Efficiency

Table 10 summarizes a single-node sweep that varies the per-GPU batch size. We omit derived "24-hour" projections and focus on directly measured quantities and the implied sample throughput.

Figure 5 visualizes the main trade-off. Smaller per-GPU batches yield faster steps (e.g., 0.703 s/step at batch 2 vs. 2.404 s/step at batch 24), while larger per-GPU batches improve sample throughput (from 22.7 to 79.9 samples/s) at the cost of sharply increased step latency.

8.2 Multi-Node Scaling Efficiency

We next fix per-GPU batch size to 8 and scale the number of GPUs. As shown in Table 11, the time per step rises from 0.735 s (8 GPUs) to 0.899 s (32 GPUs) due to inter-node communication overhead, then plateaus at $\sim$0.93 s up to 256 GPUs. Despite this overhead, sample throughput scales from 87.0 to 2200.0 samples/s, which is the relevant metric when the training objective is to process a fixed amount of data quickly.

Figure 6 plots both step latency and sample throughput against GPU count, together with the ideal linear reference line. The results highlight a practical guideline: scaling out is most beneficial for data-volume-driven training, while fixed-step training does not become faster with more GPUs.

Takeaways. First, inter-node communication introduces a one-time latency overhead (0.735$\rightarrow$0.93 s/step), but sample throughput still scales near-linearly via larger global batch. Second, on a single node, a moderate per-GPU batch (e.g., 8) often provides the best balance between step latency and GPU utilization; very large batches (e.g., 24) maximize utilization (96%) but inflate step latency by $3.4\times$. Third, for large-scale training, once the system scales beyond 8 nodes (64 GPUs), the communication burden no longer grows further, maintaining a stable scaling efficiency of 79–80%. This indicates that practitioners can confidently scale to hundreds of GPUs without incurring additional parallel efficiency degradation.

Contents

References

1X World Model Team (2025). 1x world model: Evaluating bits, not atoms. Supplementary technical progress report. Contributed by Daniel Ho, Jack Monas, Juntao Ren, Christina Yu.

AgiBot (2025). Agibot official website. `https://www.agibot.com/`.

Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Komeili, M., Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zholus, A., Arnaud, S., Gejji, A., Martin, A., Hogan, F. R., Dugas, D., Bojanowski, P., Khalidov, V., Labatut, P., Massa, F., Szafraniec, M., Krishnakumar, K., Li, Y., Ma, X., Chandar, S., Meier, F., LeCun, Y., Rabbat, M., and Ballas, N. (2025). V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*.

Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al. (2025a). Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M.-H., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., and Lin, J. (2025b). Qwen2.5-VL technical report. *CoRR*, abs/2502.13923.

Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L., Fang, Y., Fox, D., Hu, F., Huang, S., et al. (2025). Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. (2024). $\backslash pi_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. (2023). Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. (2022). Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

Cai, J., Cai, Z., Cao, J., Chen, Y., He, Z., Jiang, L., Li, H., Li, H., Li, Y., Liu, Y., et al. (2026). Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*.

Chen, B., Zhang, T., Geng, H., Song, K., Zhang, C., Li, P., Freeman, W. T., Malik, J., Abbeel, P., Tedrake, R., Sitzmann, V., and Du, Y. (2025a). Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*.

Chen, D., Zhang, J., Mu, T., Tan, Q., Li, Y., Mao, J., Liu, X., Li, K., Qiao, Y., Xiao, F., Ling, Z., and Su, H. (2025b). Robotwin 2.0: Towards general robot policies with active data generation. *arXiv preprint arXiv:2504.13059*.

Chen, X., Chen, Y., Fu, Y., Gao, N., Jia, J., Jin, W., Li, H., Mu, Y., Pang, J., Qiao, Y., et al. (2025c). Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*.

Chen, X., Djolonga, J., Padlewski, P., Mustafa, B., Changpinyo, S., Wu, J., Riquelme Ruiz, C., Goodman, S., Wang, X., Tay, Y., Shakeri, S., Dehghani, M., Salz, D., Lucic, M., Tschannen, M., Nagrani, A., Hu, H., Joshi, M., Pang, B., Montgomery, C., Pietrzyk, P., Ritter, M., Piergiovanni, A., Minderer, M., Pavetic, F., Waters, A., Li, G., Alabdulmohsin, I., Beyer, L., Amelot, J., Lee, K., Steiner, A. P., Li, Y., Keysers, D., Arnab, A., Xu, Y., Rong, K., Kolesnikov, A., Seyedhosseini, M., Angelova, A., Zhai, X., Houlsby, N., and Soricut, R. (2023). PaLI-X: On scaling up a multilingual vision and language model.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. (2024a). Diffusion policy: Visuomotor policy learning via action diffusion.

Chi, C., Xu, Z., Pan, C., Cousineau, E., Burchfiel, B., Feng, S., Tedrake, R., and Song, S. (2024b). Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*.

Collaboration, O. X.-E., O'Neill, A., Rehman, A., Gupta, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., Tung, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Gupta, A., Wang, A., Kolobov, A., Singh, A., Garg, A., Kembhavi, A., Xie, A., Brohan, A., Raffin, A., Sharma, A., Yavary, A., Jain, A., Balakrishna, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Wulfe, B., Ichter, B., Lu, C., Xu, C., Le, C., Finn, C., Wang, C., Xu, C., Chi, C., Huang, C., Chan, C., Agia, C., Pan, C., Fu, C., Devin, C., Xu, D., Morton, D., Driess, D., Chen, D., Pathak, D., Shah, D., Büchler, D., Jayaraman, D., Kalashnikov, D., Sadigh, D., Johns, E., Foster, E., Liu, F., Ceola, F., Xia, F., Zhao, F., Frujeri, F. V., Stulp, F., Zhou, G., Sukhatme, G. S., Salhotra, G., Yan, G., Feng, G., Schiavi, G., Berseth, G., Kahn, G., Yang, G., Wang, G., Su, H., Fang, H.-S., Shi, H., Bao, H., Amor, H. B., Christensen, H. I., Furuta, H., Bharadhwaj, H., Walke, H., Fang, H., Ha, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Abou-Chakra, J., Kim, J., Drake, J., Peters, J., Schneider, J., Hsu, J., Vakil, J., Bohg, J., Bingham, J., Wu, J.,

Gao, J., Hu, J., Wu, J., Wu, J., Sun, J., Luo, J., Gu, J., Tan, J., Oh, J., Wu, J., Lu, J., Yang, J., Malik, J., Silvério, J., Hejna, J., Booher, J., Tompson, J., Yang, J., Salvador, J., Lim, J. J., Han, J., Wang, K., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Oslund, K., Kawaharazuka, K., Black, K., Lin, K., Zhang, K., Ehsani, K., Lekkala, K., Ellis, K., Rana, K., Srinivasan, K., Fang, K., Singh, K. P., Zeng, K.-H., Hatch, K., Hsu, K., Itti, L., Chen, L. Y., Pinto, L., Fei-Fei, L., Tan, L., Fan, L. J., Ott, L., Lee, L., Weihs, L., Chen, M., Lepert, M., Memmel, M., Tomizuka, M., Itkina, M., Castro, M. G., Spero, M., Du, M., Ahn, M., Yip, M. C., Zhang, M., Ding, M., Heo, M., Srirama, M. K., Sharma, M., Kim, M. J., Kanazawa, N., Hansen, N., Heess, N., Joshi, N. J., Suenderhauf, N., Liu, N., Palo, N. D., Shafiullah, N. M. M., Mees, O., Kroemer, O., Bastani, O., Sanketi, P. R., Miller, P. T., Yin, P., Wohlhart, P., Xu, P., Fagan, P. D., Mitrano, P., Sermanet, P., Abbeel, P., Sundaresan, P., Chen, Q., Vuong, Q., Rafailov, R., Tian, R., Doshi, R., Mart'in-Mart'in, R., Baijal, R., Scalise, R., Hendrix, R., Lin, R., Qian, R., Zhang, R., Mendonca, R., Shah, R., Hoque, R., Julian, R., Bustamante, S., Kirmani, S., Levine, S., Lin, S., Moore, S., Bahl, S., Dass, S., Sonawani, S., Tulsiani, S., Song, S., Xu, S., Haldar, S., Karamcheti, S., Adebola, S., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Ramamoorthy, S., Dasari, S., Belkhale, S., Park, S., Nair, S., Mirchandani, S., Osa, T., Gupta, T., Harada, T., Matsushima, T., Xiao, T., Kollar, T., Yu, T., Ding, T., Davchev, T., Zhao, T. Z., Armstrong, T., Darrell, T., Chung, T., Jain, V., Kumar, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Chen, X., Wang, X., Zhu, X., Geng, X., Liu, X., Liangwei, X., Li, X., Pang, Y., Lu, Y., Ma, Y. J., Kim, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Wu, Y., Xu, Y., Wang, Y., Bisk, Y., Dou, Y., Cho, Y., Lee, Y., Cui, Y., Cao, Y., Wu, Y.-H., Tang, Y., Zhu, Y., Zhang, Y., Jiang, Y., Li, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Ma, Z., Xu, Z., Cui, Z. J., Zhang, Z., Fu, Z., and Lin, Z. (2023). Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`.

Contributors, D. (2025). Dexbotic: Open-source vision-language-action toolbox. *arXiv preprint arXiv:2510.23511*.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net.

Driess, D., Springenberg, J. T., Ichter, B., Yu, L., Li-Bell, A., Pertsch, K., Ren, A. Z., Walke, H., Vuong, Q., Shi, L. X., et al. (2025). Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*.

Du, Y., Yang, M., Dai, B., Dai, H., Nachum, O., Tenenbaum, J. B., Schuurmans, D., and Abbeel, P. (2023). Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Duan, J., Yuan, W., Pumacay, W., Wang, Y. R., Ehsani, K., Fox, D., and Krishna, R. (2024). Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*.

Ebert, F., Yang, Y., Schmeckpeper, K., Bucher, B., Georgakis, G., Daniilidis, K., Finn, C., and Levine, S. (2021). Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*.

Fei, S., Wang, S., Shi, J., Dai, Z., Cai, J., Qian, P., Ji, L., He, X., Zhang, S., Fei, Z., Fu, J., Gong, J., and Qiu, X. (2025). Libero-plus: In-depth robustness analysis of vision-language-action models.

Gao, S., Liang, W., Zheng, K., Malik, A., Ye, S., Yu, S., Tseng, W.-C., Dong, Y., Mo, K., Lin, C.-H., Ma, Q., Nah, S., Magne, L., Xiang, J., Xie, Y., Zheng, R., Niu, D., Tan, Y. L., Zentner, K. R., Kurian, G., Indupuru, S., Jannaty, P., Gu, J., Zhang, J., Malik, J., Abbeel, P., Liu, M.-Y., Zhu, Y., and Linxi "Jim" Fan (2026). Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*.

Gao, Y., Guo, H., Hoang, T., Huang, W., Jiang, L., Kong, F., Li, H., Li, J., Li, L., Li, X., Li, X., Li, Y., Lin, S., Lin, Z., Liu, J., Liu, S., Nie, X., Qing, Z., Ren, Y., Sun, L., Tian, Z., Wang, R., Wang, S., Wei, G., Wu, G., Wu, J., Xia, R., Xiao, F., Xiao, X., Yan, J., Yang, C., Yang, J., Yang, R., Yang, T., Yang, Y., Ye, Z., Zeng, X., Zeng, Y., Zhang, H., Zhao, Y., Zheng, X., Zhu, P., Zou, J., and Zuo, F. (2025). Seedance 1.0: Exploring the boundaries of video generation models. *CoRR*, abs/2506.09113.

Gemini Team, Google (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Generalist AI (2025). Gen-0: Embodied foundation models that scale with physical interaction. `https://generalistai.com/blog/nov-04-2025-GEN-0`. Generalist AI Blog.

GigaBrain Team, Wang, B., Li, B., Ni, C., Huang, G., Zhao, G., Li, H., Li, J., Lv, J., Liu, J., Feng, L., Yu, M., Li, P., Deng, Q., Liu, T., Zhou, X., Chen, X., Wang, X., Wang, Y., Li, Y., Nie, Y., Li, Y., Zhou, Y., Ye, Y., Liu, Z., and Zhu, Z. (2026). Gigabrain-0.5m*: a vla that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*.

GigaWorld Team, Ye, A., Wang, B., Ni, C., Huang, G., Zhao, G., Li, H., Zhu, J., Li, K., Xu, M., Deng, Q., Wang, S., Qin, W., Chen, X., Wang, X., Wang, Y., Cao, Y., Chang, Y., Xu, Y., Ye, Y., Wang, Y., Zhou, Y., Zhang, Z., Dong, Z., and Zhu, Z. (2025). Gigaworld-0: World models as data engine to empower embodied ai. *arXiv preprint arXiv:2511.19861*.

Google DeepMind (2025). Veo 3 model card. Technical report, Google DeepMind. Published May 23, 2025.

Guo, Y., Shi, L. X., Chen, J., and Finn, C. (2025). Ctrl-world: A controllable generative world model for robot manipulation. *arXiv preprint arXiv:2510.10125*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hoque, R., Huang, P., Yoon, D. J., Sivapurapu, M., and Zhang, J. (2025). Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*.

Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., et al. (2025a). $pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*.

Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., et al. (2025b). Pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*.

Jang, J., Ye, S., Lin, Z., Xiang, J., Bjorck, J., Fang, Y., Hu, F., Huang, S., Kundalia, K., Lin, Y.-C., Magne, L., Mandlekar, A., Narayan, A., Tan, Y. L., Wang, G., Wang, J., Wang, Q., Xu, Y., Zeng, X., Zheng, K., Zheng, R., Liu, M.-Y., Zettlemoyer, L., Fox, D., Kautz, J., Reed, S., Zhu, Y., and Fan, L. (2025). Dreamgen: Unlocking generalization in robot learning through neural trajectories. *arXiv preprint arXiv:2505.12705*.

Jiang, Z., Zhou, S., Jiang, Y., Huang, Z., Wei, M., Chen, Y., Zhou, T., Guo, Z., Lin, H., Zhang, Q., Wang, Y., Li, H., Yu, C., and Zhao, D. (2026). Wovr: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*.

Karamcheti, S., Nair, S., Balakrishna, A., et al. (2024). Prismatic: A (nearly) universal vision-language model with fine-grained visual representations. In *International Conference on Machine Learning (ICML)*.

Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M. K., Chen, L. Y., Ellis, K., et al. (2024). Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*.

Kim, M. J., Finn, C., and Liang, P. (2025). Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*.

Kim, M. J., Gao, Y., Lin, T.-Y., Lin, Y.-C., Ge, Y., Lam, G., Liang, P., Song, S., Liu, M.-Y., Finn, C., and Gu, J. (2026). Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. (2024). Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollar, P., and Girshick, R. (2023). Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026.

Ko, P.-C., Mao, J., Du, Y., Sun, S.-H., and Tenenbaum, J. B. (2024). Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations (ICLR)*.

Li, C., Zhang, R., Wong, J., Gokmen, C., Srivastava, S., Martín-Martín, R., Wang, C., Levine, G., Lingelbach, M., Sun, J., et al. (2023). Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR.

Li, L., Zhang, Q., Luo, Y., Yang, S., Wang, R., Han, F., Yu, M., Gao, Z., Xue, N., Zhu, X., Shen, Y., and Xu, Y. (2026). Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*.

Li, Q., Liang, Y., Wang, Z., Luo, L., Chen, X., Liao, M., Wei, F., Deng, Y., Xu, S., Zhang, Y., et al. (2024a). Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*.

Li, X., Hsu, K., Gu, J., Pertsch, K., Mees, O., Walke, H. R., Fu, C., Lunawat, I., Sieh, I., Kirmani, S., et al. (2024b). Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. (2024a). Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023a). Visual instruction tuning. *CoRR*, abs/2304.08485.

Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., and Zhu, J. (2024b). Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., Zhu, J., and Zhang, L. (2023b). Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. (2022). Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. (2022). R3M: A universal visual representation for robot manipulation.

Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. (2024). Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*.

Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Xu, C., Luo, J., Kreiman, T., Tan, Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. (2024). Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands.

OpenAI (2023). Gpt-4 technical report. *arXiv:2303.08774*.

OpenAI (2024). GPT-4o system card. *CoRR*, abs/2410.21276.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. (2023). DINOv2: Learning robust visual features without supervision.

Pai, J., Achenbach, L., Montesinos, V., Forrai, B., Mees, O., and Nava, E. (2025). mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*.

Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., and Levine, S. (2025). Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*.

Qiu, Y., Zhao, Z., Li, W., Ziser, Y., Korhonen, A., Cohen, S. B., and Ponti, E. M. (2026). Self-improving world modelling with latent actions. *arXiv preprint arXiv:2602.06130*.

Qu, D., Song, H., Chen, Q., Yao, Y., Ye, X., Ding, Y., Wang, Z., Gu, J., Zhao, B., Wang, D., et al. (2025). Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Shi, L. X., Ichter, B., Equi, M., Ke, L., Pertsch, K., Vuong, Q., Tanner, J., Walling, A., Wang, H., Fusai, N., et al. (2025). Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*.

Team, G. R., Choromanski, K., Devin, C., Du, Y., Dwibedi, D., Gao, R., Jindal, A., Kipf, T., Kirmani, S., Leal, I., Liu, F., Majumdar, A., Marmon, A., Parada, C., Rubanova, Y., Shah, D., Sindhwani, V., Tan, J., Xia, F., Xiao, T., Yang, S., Yu, W., and Zhou, A. (2025). Evaluating gemini robotics policies in a veo world simulator.

Wei, S., Jing, H., Li, B., Zhao, Z., Mao, J., Ni, Z., He, S., Liu, J., Liu, X., Kang, K., Zang, S., Yuan, W., Pavone, M., Huang, D., and Wang, Y. (2026). Ψ_0: An open foundation model towards universal humanoid loco-manipulation. *arXiv preprint arXiv:2603.12263*.

Wu, K., Hou, C., Liu, J., Che, Z., Ju, X., Yang, Z., Li, M., Zhao, Y., Xu, Z., Yang, G., Fan, S., Wang, X., Liao, F., Zhao, Z., Li, G., Jin, Z., Wang, L., Mao, J., Liu, N., Ren, P., Zhang, Q., Lyu, Y., Liu, M., He, J., Luo, Y., Gao, Z., Li, C., Gu, C., Fu, Y., Wu, D., Wang, X., Chen, S., Wang, Z., An, P., Qian, S., Zhang, S., and Tang, J. (2024). Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*.

Wu, P., Escontrela, A., Hafner, D., Abbeel, P., and Goldberg, K. (2023). Daydreamer: World models for physical robot learning. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR.

Wu, W., Lu, F., Wang, Y., Yang, S., Liu, S., Wang, F., Zhu, Q., Sun, H., Wang, Y., Ma, S., et al. (2026). A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*.

Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. (2022). Masked visual pre-training for motor control.

Yang, J., Tan, R., Wu, Q., Zheng, R., Peng, B., Liang, Y., Gu, Y., Cai, M., Ye, S., Jang, J., et al. (2025a). Magma: A foundation model for multimodal ai agents. *arXiv preprint arXiv:2502.13130*.

Yang, R., Yu, Q., Wu, Y., Yan, R., Li, B., Cheng, A.-C., Zou, X., Fang, Y., Cheng, X., Qiu, R.-Z., Yin, H., Liu, S., Han, S., Lu, Y., and Wang, X. (2025b). Egovla: Learning vision-language-action models from egocentric human videos. *arXiv preprint arXiv:2507.12440*.

Yang, S., Li, H., Chen, Y., Wang, B., Tian, Y., Wang, T., Wang, H., Zhao, F., Liao, Y., and Pang, J. (2025c). Instructvla: Vision-language-action instruction tuning from understanding to manipulation. *arXiv preprint arXiv:2507.17520*.

Ye, J., Wang, F., Gao, N., Yu, J., Zhu, Y., Wang, B., Zhang, J., Jin, W., Fu, Y., Zheng, F., et al. (2026a). St4vla: Spatially guided training for vision-language-action models. *arXiv preprint arXiv:2602.10109*.

Ye, S., Ge, Y., Zheng, K., Gao, S., Yu, S., Kurian, G., Indupuru, S., Tan, Y. L., Zhu, C., Xiang, J., Malik, A., Lee, K., Liang, W., Ranawaka, N., Gu, J., Xu, Y., Wang, G., Hu, F., Narayan, A., Bjorck, J., Wang, J., Kim, G., Niu, D., Zheng, R., Xie, Y., Wu, J., Wang, Q., Julian, R., Xu, D., Du, Y., Chebotar, Y., Reed, S., Kautz, J., Zhu, Y., Fan, L., and Jang, J. (2026b). World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*.

Ye, S., Jang, J., Jeon, B., Joo, S., Yang, J., Peng, B., Mandlekar, A., Tan, R., Chao, Y.-W., Lin, B. Y., Liden, L., Lee, K., Gao, J., Zettlemoyer, L., Fox, D., and Seo, M. (2025). Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Yuan, C., Zhou, R., Liu, M., Hu, Y., Wang, S., Yi, L., Wen, C., Zhang, S., and Gao, Y. (2025). Motiontrans: Human vr data enable motion-level learning for robotic manipulation policies. *arXiv preprint arXiv:2509.17759*.

Yuan, T., Dong, Z., Liu, Y., and Zhao, H. (2026). Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*.

Ze, Y., Zhang, G., Zhang, K., Hu, C., Wang, M., and Xu, H. (2024). 3d diffusion policy. *arXiv preprint arXiv:2403.03954*.

Zeng, A., Florence, P., Yang, M., Du, Y., et al. (2024). Molmoact: Vision-language-action model for robotic manipulation. *arXiv preprint arXiv:2403.03368*.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. (2023). Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11975–11986.

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. (2023). Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*.

Zheng, J., Li, J., Wang, Z., Liu, D., Kang, X., Feng, Y., Zheng, Y., Zou, J., Chen, Y., Zeng, J., et al. (2025a). X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*.

Zheng, R., Niu, D., Xie, Y., Wang, J., Xu, M., Jiang, Y., Castañeda, F., Hu, F., Tan, Y. L., Fu, L., Darrell, T., Huang, F., Zhu, Y., Xu, D., and Fan, L. (2026). Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*.

Zheng, R., Wang, J., Reed, S., Bjorck, J., Fang, Y., Hu, F., Jang, J., Kundalia, K., Lin, Z., Magne, L., Narayan, A., Tan, Y. L., Wang, G., Wang, Q., Xiang, J., Xu, Y., Ye, S., Kautz, J., Huang, F., Zhu, Y., and Fan, L. (2025b). Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*.

Zhou, Z., Zhu, Y., Zhu, M., Wen, J., Liu, N., Xu, Z., Meng, W., Cheng, R., Peng, Y., Shen, C., et al. (2025). Chatvla: Unified multimodal understanding and robot control with vision-language-action model. *arXiv preprint arXiv:2502.14420*.

Zhu, L. Y., Kuppili, P., Punamiya, R., Aphiwetsa, P., Patel, D., Kareer, S., Ha, S., and Xu, D. (2025). Emma: Scaling mobile manipulation via egocentric human data. *arXiv preprint arXiv:2509.04443*.

9 Authors and Contributors for StarVLA v1.0

StarVLA thrives on the synergy between its dedicated core team and a vibrant open-source community. To accurately reflect the nature of involvement, we list contributors in two categories: **Authors** and **Community Contributors**. We extend our deepest gratitude to everyone who has helped shape and scale StarVLA.

Authors. Jinhui Ye, Ning Gao, Yilun Chen[†], Weiyu Guo, Zixuan Wang, Yuxing Chen, Fangjing Wang, Senqiao Yang, Chengyao Wang, Yuqi Liu, Meng Chu, Changsheng Lu, Pengguang Chen, Shu Liu[†], Jiaya Jia[†*]

Community Contributors. Junqiu Yu, Shuang Zeng, Shijie Lian, Hanwen Wan, Changjiu Zhang, Zhijie Song, Mingsheng Li, Qiuyue Wang, Sicheng Xie, Jinliang Zheng, Deyu Zhou, Jiaming Zhou, Lu Dai, Xiaorui Zhao

Contributor Policy. *Authors* constitute the core team of StarVLA. This group is responsible for continuously iterating on core features, maintaining the foundational framework, and providing ongoing, long-term support for the project. Researchers and developers who are interested in making sustained, structural contributions and wish to join the core author team are highly encouraged to contact us. *Community Contributors* are the vital force behind the project's broader ecosystem. We continuously receive invaluable support from the open-source community—ranging from new feature implementations (pull requests) and bug fixes to constructive feedback. We deeply appreciate these efforts, which allow StarVLA to evolve rapidly. The full and actively updated contributor history is maintained at starvla.github.io/contributors.

[†]Corresponding authors. $*$ Von Neumann Institute, HKUST