STARVLA: A LEGO-LIKE CODEBASE FOR VISION-LANGUAGE-ACTION MODEL DEVELOPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot learning continues to explore different pretrained models, approaches to action modeling, and sources of supervision. These choices require infrastructure that accommodates new methods while providing reusable training and evaluation workflows. We present STARVLA, an open-source infrastructure that gives the robot learning community accessible, reusable models, training tools, and benchmark resources. Model-level frameworks let researchers compose pretrained components, including vision-language models (VLMs), with action heads and modify their connections and learning objectives. The infrastructure provides 27 registered policy implementations spanning visuomotor, VLM-based, and world-model-based policies. Scalable, extensible dataloaders and configurable dataset mixtures integrate robot data, human-centered data, and multimodal data. Shared training tools support robot post-training, cross-embodiment training, VLM training, and multimodal co-training. The 13 simulation benchmark integrations cover 10 robot embodiments, spanning single-arm, bimanual, humanoid, and mobile manipulation, as well as navigation. Through these integrations, STARVLA makes baseline training recipes, evaluation entry points, and at least 41 policy checkpoint repositories available for research. A shared policy-serving path supports simulation and real-robot clients while keeping environment dependencies separate from model execution. Community research builds on these resources to extend policy architectures and conduct representation pretraining. STARVLA provides infrastructure for researchers to develop their own models, training procedures, and evaluations.

1 INTRODUCTION

Robot learning is advancing along several design routes. Visuomotor policies, policies based on vision-language models (VLMs), and world-model-based policies connect perception to action in different ways (Zhao et al., 2023; Chi et al., 2025; Kim et al., 2025b; Ye et al., 2026). Their components, action heads, and representation connections remain active research choices (Zitkovich et al., 2023; Black et al., 2025b; NVIDIA et al., 2025), and training draws on mixed supervision from robot demonstrations, human recordings, and vision-language data (Open X-Embodiment Collaboration, 2024; Kareer et al., 2025; Black et al., 2025a). Studying any one of these choices requires repeatedly adjusting how a model is built, what data it uses, and how it is trained.

Open-source frameworks address parts of this problem. OpenVLA, OpenPI, and GR00T release policy implementations with pretrained weights (Kim et al., 2025b; Black et al., 2025b; NVIDIA et al., 2025), LeRobot connects robot interfaces, data tools, and inference (Cadene et al., 2026), and projects such as Dexbotic, LingBot-VLA, XPolicyLab, and OpenWAM add further implementations and tooling (Xie et al., 2025; Wu et al., 2026a; XPolicyLab Community et al., 2026; Wang et al., 2026f). These systems are usually organized around a specific policy family or recipe, so adding a new backbone, action head, or supervision source often means editing code that is tied to the reusable tooling. A clear boundary between editable method computation and reusable data preparation, training, and evaluation tools lets researchers revise models and supervision without rebuilding the surrounding workflow. Implemented models, recipes, and checkpoints provide practical starting points.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

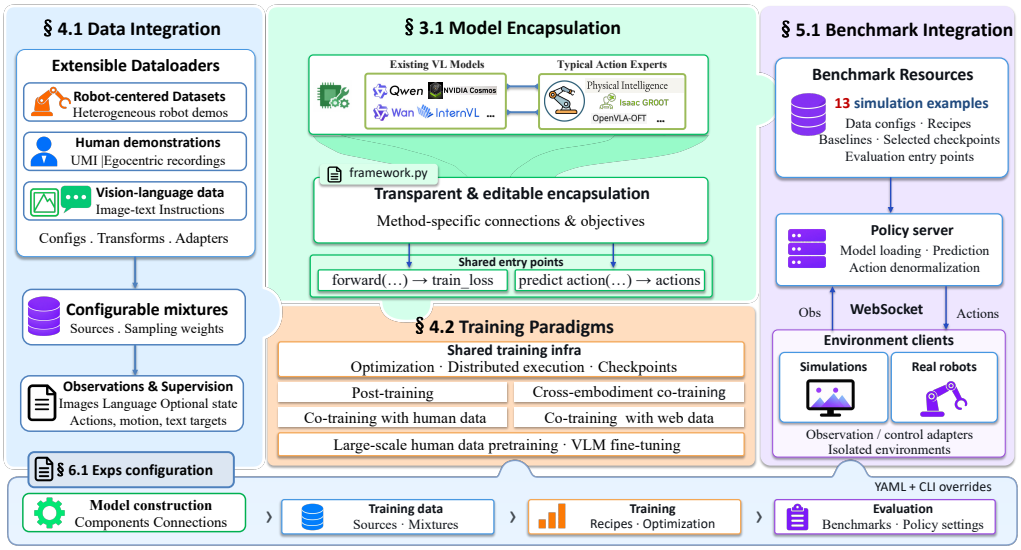


Figure 1: Overview of the StarVLA research infrastructure. Model frameworks expose method-specific computation while reusing pretrained components and action heads. Extensible dataloaders and configurable training recipes support multiple data sources and training paradigms. Benchmark integrations provide reference configurations, policy checkpoints, and evaluation entry points, with shared policy serving for simulation and real robots. Experiment configurations connect these stages through reusable settings and saved snapshots. Interlocking boundaries indicate relationships between modules.

We present StarVLA, an open-source infrastructure that gives the robot learning community accessible, reusable models, training tools, and benchmark resources. It separates editable method computation from reusable data preparation, training, and evaluation tools. Within the workflow illustrated in Figure 1, model-level *frameworks* support policy construction from pretrained VLMs, world models, and visual components. Researchers can modify these components, their connections to action heads, and the learning objectives within the framework boundary. Shared training and prediction interfaces connect the method to reusable experimental tools. Scalable, extensible dataloaders and configurable dataset mixtures organize robot data, human-centered data, and multimodal data for shared training tools. Benchmark integrations then connect trained policies to environment-specific evaluation workflows. A shared policy-serving path supports simulation and real-robot clients while keeping their environment dependencies separate from model execution.

StarVLA provides the community with 27 registered policy implementations and 13 simulation benchmark integrations covering 10 robot embodiments spanning single-arm, bimanual, humanoid, and mobile manipulation, as well as navigation. These integrations make baseline training recipes, evaluation tools, and at least 41 policy checkpoint repositories available for research. Training recipes and reported baseline results provide reference settings for using these resources. Researchers can use this infrastructure to evaluate an existing policy, adapt it to a task, or build their own model and training procedure.

Community research illustrates how StarVLA supports further method development. LangForce introduces latent action queries and uses the StarVLA training pipeline (Lian et al., 2026c). VLAct uses its model, data, and training components for representation-centric continued pretraining (Yang et al., 2026c). These examples show how the infrastructure supports changes to both policy computation and learning.

Our contributions to robot learning infrastructure are:

- **Flexible model encapsulation.** We enable researchers to construct and extend robot policies from pretrained components while reusing shared training and prediction interfaces. The framework described in Section 3 keeps action heads, representation connections, and learning ob-

Table 1: Comparison of open-source VLA frameworks. ✓: implemented; E: a named external or companion integration; P: paper-reported, but a corresponding public implementation has not been confirmed; –: unconfirmed, not necessarily unsupported. VL co-training combines vision-language supervision with action supervision. WM denotes a world model used as the policy backbone, not as the environment. RL denotes reinforcement learning, and API denotes application programming interface. Sim Bench counts distinct named integrations, including documented external adapters. $\geq$ indicates a confirmed lower bound.

	Model design		Learning workflows		Environment ecosystem	
Framework	**Configurable architecture**	**WM-based policies**	**VL co-training**	**RL**	**Sim Bench (#)**	**Real-robot integration**
OpenVLA / OFT (Kim et al., 2025b;a)	VLM / action-head configs	–	–	E	1	Robot examples
OpenPI (Black et al., 2025b)	π-family configs	–	–	E	2	Shared serving
Isaac-GR00T (NVIDIA et al., 2025)	Model / pipeline plugins	–	–	E	6	Shared policy API
RoboVLMs (Li et al., 2026f)	Component composition	–	✓	–	2	Paper only
LeRobot (Cadene et al., 2026)	Policy plugins + configs	✓	✓	✓	≥ 11	Hardware APIs
Dexbotic (Xie et al., 2025)	Component composition	–	✓	✓ / E	8	Hardware workflows
LingBot-VLA family (Wu et al., 2026a;b)	VLM / action-head configs	–	–	E	2	Serving + configs
VLA Foundry (Mercat et al., 2026)	Component composition	–	P	–	1	–
FluxVLA Engine (Li et al., 2026h)	Component composition	✓	–	–	≥ 4	Serving + drivers
OpenWAM (Wang et al., 2026f)	WAM composition	✓	–	–	8	Serving + E
Complementary learning and deployment infrastructure						
RLinf (Yu et al., 2026c)	Model / worker plugins	E	–	✓	≥ 12	Robot workflows
XPolicyLab (XPolicyLab Community et al., 2026)	Policy adapters	E	–	–	≥ 2	Shared adapters
StarVLA	Component composition	✓	✓	E	**13**	Shared serving

jectives editable. The 27 registered policy implementations in Table 2 span visuomotor, VLM-based, and world-model-based policies and provide starting points for further development.

- **Versatile dataloader and trainer integration.** We enable researchers to build and revise training recipes across heterogeneous supervision while reusing a common training infrastructure. The integration described in Section 4 connects robot data, human-centered data, and multimodal data to shared training tools through source-specific preparation and configurable dataset mixtures. Table 4 summarizes the four widely used training paradigms supported by StarVLA: robot post-training, cross-embodiment training, VLM training, and multimodal co-training.
- **Comprehensive benchmark integration and reference resources.** We make benchmark-specific resources reusable starting points for policy development by connecting data preparation, baseline training, checkpoints, and evaluation into integrated workflows. We provide 13 simulation benchmark integrations covering 10 robot embodiments, with at least 54 benchmark-specific reference configurations and 41 policy checkpoint repositories, as summarized in Table 5. The shared serving boundary in Section 5 isolates environment-specific complexity and allows simulation and real-robot clients to reuse model deployment.

2 Related Work

Robot Policy Architectures. Robot policies use different representations and approaches to action modeling. Action Chunking with Transformers (ACT) and Diffusion Policy learn action chunks from visual observations (Zhao et al., 2023; Chi et al., 2025). RT-2 and OpenVLA adapt pretrained vision-language models through autoregressive action-token prediction (Zitkovich et al., 2023; Kim et al., 2025b), while π_0 and GR00T couple VLMs with flow-matching action heads (Black et al., 2025b; NVIDIA et al., 2025). Video-generative pretraining provides another starting point for policy development, as explored by GR-1 and RynnVLA-001 (Wu et al., 2024; Jiang et al., 2025). DreamZero uses a video-generation backbone to jointly predict future observations and actions (Ye et al., 2026). These approaches motivate support for different pretrained components, action computations, and connections within a common development infrastructure.

Open-Source Frameworks for Robot Learning. OpenVLA, OpenPI, and GR00T provide policy implementations and pretrained weights for robot learning (Kim et al., 2025b; Black et al., 2025b; NVIDIA et al., 2025). LeRobot connects robot interfaces, data tools, policy implementations, and inference in an end-to-end library (Cadene et al., 2026). Projects including Dexbotic (Xie et al., 2025), LingBot-VLA (Wu et al., 2026a), XPolicyLab (XPolicyLab Community et al., 2026), and OpenWAM (Wang et al., 2026f) also provide infrastructure for policy development. StarVLA



Figure 2: **Research adoption and community-driven evolution of StarVLA.** From October 2025 to September 2026, a growing body of research has built on StarVLA (top), while community-supported development has continuously expanded its models, training workflows, benchmark coverage, and deployment capabilities (middle). Integrations with external frameworks (bottom) further extend its ecosystem across scalable training, reinforcement learning, policy evaluation, and deployment.

contributes model-level frameworks, data and training recipes, and benchmark integrations that researchers can use together to develop their own policies. This paper describes their organization, available implementations and checkpoints, and use in model and training research. Table 1 summarizes the framework comparison.

Evolving Training Recipes for Robot Policies. Training recipes differ in action representation, supervision, and optimization. FAST uses compressed action tokens for autoregressive learning (Pertsch et al., 2025). The recipes of $\pi_{0.5}$ and InternVLA-M1 combine robot action supervision with visual question answering (Black et al., 2025a; Chen et al., 2025b). ECoT adds textual reasoning before action prediction (Zawalski et al., 2025). World-model-based policies can include future-observation prediction (Ye et al., 2026), while SimpleVLA-RL extends policy learning through interaction (Li et al., 2026b). These approaches illustrate why data sources, targets, and update procedures need to remain adaptable. Section 4 describes how StarVLA organizes source-specific data preparation, dataset mixtures, and training recipes for the policy implementations described here.

3 Model Encapsulation and Extensibility

As robot policy architectures evolve, their input preparation, internal computation, and prediction targets also change. StarVLA organizes these choices through the framework abstraction in Figure 1. Methods can use shared training and deployment tools while retaining their own computation. This abstraction supports visuomotor, VLM-based, and world-model-based policies.

3.1 UNIFIED MODEL ENCAPSULATION

STARVLA defines a *framework* as a model-level abstraction that encapsulates a method's computation. A dedicated file defines the top-level model class and assembles components from reusable modules. Shared training and prediction entry points establish the external boundary between the method and the surrounding infrastructure.

Within this abstraction, a policy predicts an action chunk and may produce auxiliary outputs, conditioned on its observations and task instruction:

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

Here, $\mathbf{x}_{\leq t}$ denotes the observations available to the policy, with modalities and history determined by the method. This notation can describe visual, proprioceptive, depth, or tactile inputs. The instruction ℓ specifies the task when language conditioning is used. The action chunk $\mathbf{a}_{t:t+k-1}$ contains k predicted actions starting at time t. Optional auxiliary outputs $\mathbf{y}_{\text{aux}}$ can include future observations or language plans used in learning or action prediction.

Each framework defines how observations are transformed into actions and, where applicable, auxiliary outputs. Researchers can compose pretrained components with action heads and revise their connections and learning objectives. Shared training and prediction entry points connect these method-specific computations to the surrounding infrastructure. Appendix A.2 describes component organization and interface details.

3.2 MODEL COVERAGE

Building on this abstraction, STARVLA provides 27 registered policy implementations. Table 2 groups them into two visuomotor, 19 VLM-based, and 6 world-model-based policies.

Available VLM components include Qwen, Gemma 4, MiniCPM-V, and PaliGemma. World-model-based policies use Cosmos-Predict2 and Wan2.2 video-generation components with regression or flow-matching action heads. Additional visual components can also enter the computation. For example, QwenDual combines VLM features with features from a separate visual encoder before action prediction.

Action modeling spans continuous regression, autoregressive action-token prediction, and diffusion or flow matching. The implementations also differ in how action heads use pretrained representations. QwenOFT regresses action chunks from action-query token representations, whereas QwenFast generates action tokens through the VLM. QwenGR00T conditions an action head on final-layer VLM features. QwenPI instead uses features from multiple VLM layers. These implementations illustrate choices in both action modeling and the representations used for action prediction.

Table 2: Registered policy implementations by implementation group. One entry name is shown per class.

Group	Classes	Registered entry names
Visuomotor (VM4A)	2	`ACT`, `DiffusionPolicy`
VLM-based (VLM4A)	19	`CosmosGR00T`, `EgoVLA`, `Gemma4GR00T`, `Gemma4PI`, `LangForce`, `InternVLA-M1`, `MiniCPMGR00T`, `MiniCPMPI`, `MiniCPMRobotManip`, `PI0`, `PI05`, `QwenAdapter`, `QwenDiscreteDiffusion`, `QwenDual`, `QwenFast`, `QwenGR00T`, `QwenOFT`, `QwenPI`, `QwenPI_v3`
World-model-based (WM4A)	6	`CosmoPredict2GR00T`, `CosmoPredict2OFT`, `CosmoPredict2PI`, `WanGR00T`, `WanOFT`, `WanPI`

3.3 RESEARCH EXTENSIONS

Beyond the policy implementations maintained within STARVLA, more than 50 community research works have built on the infrastructure to develop new models and learning methods (Figure 2). This adoption goes beyond evaluating or fine-tuning released checkpoints. Researchers

modify method-specific computation and supervision while reusing or adapting the surrounding model components, data pipelines, training infrastructure, and evaluation workflows.

Representative works illustrate the depth of these extensions. LangForce (Lian et al., 2026c) introduces latent action queries and paired vision-only and language-conditioned policy branches while reusing the training entry and dataset configurations. LaWAM (Chen et al., 2026b) extends the Qwen-GR00T architecture by conditioning action generation on predicted latent visual subgoals, while adapting existing dataloaders, training loops, and policy-serving workflows. Other works introduce new temporal representations and prediction objectives. EventVLA augments QwenOFT-based simulation policies with event-driven keyframe memory and jointly learns action and keyframe prediction (Yang et al., 2026a). W^2-VLA adds task-conditioned future-wrist prediction and jointly optimizes action, language-reasoning, and wrist-prediction objectives (Pan et al., 2026). These methods change policy representations, internal computation, and learning signals rather than merely selecting among existing pretrained policies.

Community use also extends beyond model architecture. Works such as VLAct (Yang et al., 2026c) and PhysBrain (Lian et al., 2026b) reuse model, data, and training components for representation pretraining, continued pretraining, and robot adaptation. More broadly, Figure 2 places the community research built on StarVLA alongside the concurrent expansion of its model, training, benchmark, and deployment capabilities and its connections to external frameworks. Together, these works provide external evidence that the framework boundary described in Section 3.1 supports method development beyond the implementations maintained within StarVLA itself. Appendix F summarizes six selected research examples and their scope of reuse. The full paper index appears in Appendix G.

4 Dataloader and Trainer Integration

StarVLA connects dataloaders and trainers with model frameworks to support different data sources and training paradigms. Dataloaders prepare observations and supervision, while training recipes specify how the selected framework learns from them through objectives and parameter updates.

4.1 Flexible Data Integration

StarVLA supports heterogeneous robot data, human-centered data, and multimodal data through extensible dataloaders. Table 3 summarizes their coverage and typical roles in embodied learning. The multimodal sources include vision-language samples such as image-text pairs and visual instruction data. Local registries, data configurations, and source-specific adapters allow new datasets and recording formats to be added as coverage grows. Configurable dataset mixtures associate each source with its preparation settings and a dataset sampling weight. Sources with different supervision can also retain separate data streams, with training recipes specifying how their objectives are combined, as described in Section 4.2. Appendix B.1 describes the loader and sampling details.

4.2 Supported Training Paradigms

StarVLA supports the four widely used training paradigms in robot learning listed in Table 4. Within these paradigms, each training recipe specifies the data sources, learning objectives, and optimization settings for a run. Selected sources can form dataset mixtures within one or more data streams. Dataset sampling weights control source sampling, while objective weights control contributions to model parameter updates.

These recipes share tools for optimization, distributed execution, logging, and checkpointing. The example configurations in Appendix B.3 provide practical starting points for LIBERO training, Bridge/RT-1 dataset mixtures, and multimodal co-training with robot and vision-language data. Sections 6.1 and 6.2 describe experiment configurations and execution efficiency, respectively.

Research use. VLAct uses StarVLA model, data, and training components for representation-centric continued pretraining, releasing representations for subsequent policy learning (Yang et al.,

Table 3: Data-source coverage and training roles. Counts are lower bounds on dataset entries, including task, split, and conversion variants.

Data sources	Dataset entries	Typical role in embodied learning
Robot data	≥ 300	Embodiment-specific action learning and task adaptation.
Human-centered data	≥ 20	Large-scale pretraining from human activity and manipulation recordings.
Multimodal data	≥ 30	Visual grounding, semantic understanding, and instruction following.

Table 4: Training paradigms and supported data inputs. Checkmarks indicate supported input categories, selected through each run's data configuration.

	Dataloader inputs		
Training paradigm	Robot	Human-centered	Multimodal
Robot post-training	✓	–	–
Cross-embodiment training	✓	✓	–
VLM training	–	–	✓
Multimodal co-training	✓	✓	✓

2026c). For robot adaptation, PhysBrain follows StarVLA training configurations to adapt VLMs pretrained on human egocentric data to action prediction (Lin et al., 2025).

5 Benchmark Integration and Reference Resources

StarVLA brings benchmark data preparation, baseline training recipes, and policy evaluation into a shared workflow. Integrated resources provide starting points for new experiments, while a shared policy-serving path connects compatible policies to simulation and real-robot clients.

5.1 Benchmark Coverage and Reference Resources

StarVLA provides 13 simulation benchmark integrations covering 10 robot embodiments. These integrations span single-arm, bimanual, humanoid, and mobile manipulation, as well as navigation, across hundreds of tasks. Task coverage includes tabletop and household activities, dynamic manipulation, long-horizon sequences, and robustness under changed conditions. Each integration connects data configurations, baseline training recipes, and evaluation entry points while retaining the benchmark's task definitions and metrics. Appendix C.3 details the individual integrations, with coverage summarized in Table 5.

These integrations organize at least 54 benchmark-specific reference configurations and 41 available policy checkpoint repositories, including community-contributed resources. These reference resources let researchers train a baseline, evaluate an available policy, or continue training for their own tasks. Appendix C.2 describes the policy and initialization resources. Figure 3 summarizes reference performance for selected policies and recipes. Individual results and evaluation conditions are provided in Appendix C.4.

5.2 Environment Isolation and Shared Deployment

StarVLA separates model execution from benchmark environments through a policy server and environment-specific clients. The server handles model loading and prediction, while clients exchange observations and actions over WebSocket. Each client maintains its simulator dependencies and adapts observations and controls to the policy interface. Task execution, success checks, and metric aggregation remain with the benchmark evaluator. This organization allows different environments to reuse policy-serving tools while keeping their dependencies and control conventions local to their integrations.

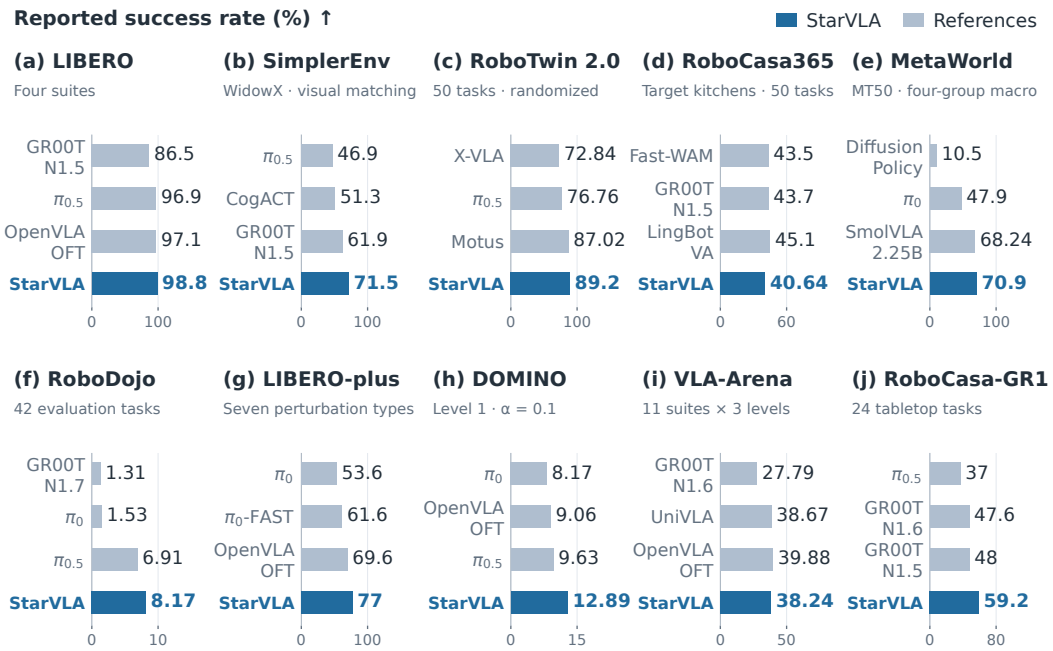


Figure 3: Reported success rates on ten selected benchmark settings. Each panel shows one StarVLA configuration and three reference methods. Training recipes and aggregation conventions can differ. Selected variants and evaluation conditions are listed in Appendix C.4.

Real-robot clients use the same serving boundary, adapting hardware observations and controls to compatible policies. Franka and Realman workflows illustrate how robot clients use this shared policy-serving path. A new environment can connect through its client and evaluation adapter without embedding its simulator or hardware dependencies in the model implementation. Appendix D.1 describes the serving interfaces and adapters, and Appendix D.2 details the robot workflows.

6 Additional Infrastructure

Reusable experiment configurations connect model and data choices to training runs and checkpoints. Reported execution measurements complement these resources by characterizing training efficiency and scaling for selected configurations.

6.1 Flexible Experiment Configuration

StarVLA uses a hierarchical configuration system built around YAML files to organize model, data, and training settings. These files provide reusable experiment configurations that can be extended as researchers settings relevant to its computation. Command-line overrides allow selected parameters to vary across runs while reusing the same YAML file. Method-specific options can accompany new computations without requiring changes to a fixed global parameter schema. Researchers can therefore adapt an existing experiment file to a new model or training recipe. For example, a new action-head option or objective weight can be added to the corresponding group and read by the component that implements it.

Configuration snapshots. StarVLA saves a configuration snapshot of the experiment parameters accessed during a run. The snapshot supports reusing these parameters with selected overrides and reconstructing the model when loading a checkpoint. Appendix E.1 describes parameter grouping, override precedence, and extension of configuration fields. Figure 7 illustrates how these settings enter method-specific computation while retaining shared training tools.

6.2 Training Efficiency and Scaling

Execution measurements characterize the compute requirements and scaling of selected training configurations. Figure 4(a) compares reported step latency on eight graphics processing units (GPUs), while panel (b) shows throughput scaling from 8 to 256 GPUs. The A100 measurements use StarVLA-GR00T with Qwen3-VL-4B on RoboCasa-GR1. At 256 GPUs, throughput reaches 2,200 samples/s with a reported scaling efficiency of 79.1%. On eight A100 GPUs, increasing batch per GPU from 2 to 24 raises reported throughput from 22.7 to 79.9 samples/s. The separate GPU-scaling series holds batch at eight per GPU, with step latency increasing from 0.735 to 0.931 seconds. Global batch therefore grows from 64 to 2,048 across 8–256 GPUs. Appendix E.3 provides batch-size effects, step latency across GPU counts, and measurement details.

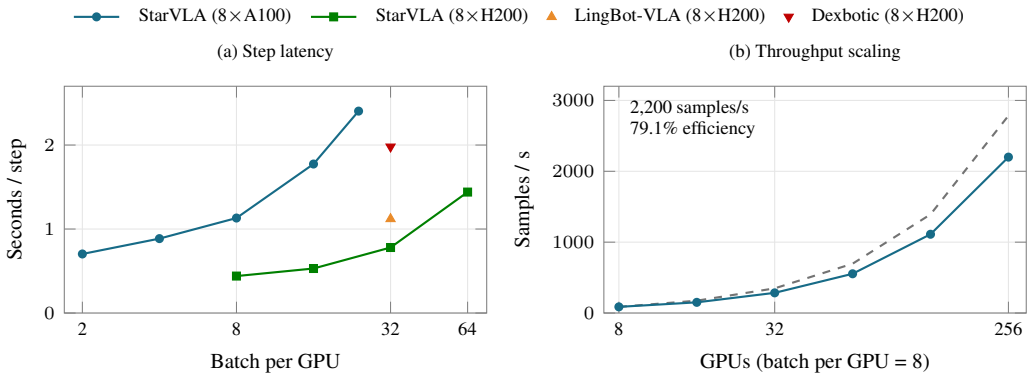


Figure 4: Training efficiency. (a) Step latency across reported eight-GPU configurations, with the legend specifying hardware. (b) Throughput on 8–256 A100 GPUs at batch eight per GPU, with a dashed linear reference from this series' eight-GPU baseline. Global batch grows with GPU count.

7 Discussion and Conclusion

Design tradeoffs. StarVLA combines reusable research resources with room to modify the methods that use them. Frameworks define representation connections and learning objectives, while data configurations and environment clients retain source-specific and task-specific conventions. This organization supports changes to one part of an experiment while preserving applicable training and deployment tools. The method and its adapters remain responsible for matching representations, supervision, and controls. Shared interfaces do not establish their semantic compatibility by themselves.

Scope and further evaluation. The policy implementation catalog, training recipes, benchmark results, and research uses document different aspects of the infrastructure's utility. Performance and throughput measurements apply to the specified models, recipes, and evaluation conditions. Resource completeness varies across integrations, and the available evidence does not quantify development effort or reproduction success across the release. Evaluating additional extensions and checkpoint reproduction under fixed settings would further characterize these aspects of usability.

Conclusion. StarVLA provides open-source infrastructure for building, training, and studying robot learning models. Its policy implementations combine pretrained components with different action heads, making it possible to reuse common representations while exploring alternative action-generation mechanisms. Data integration and training recipes support learning from heterogeneous robots, datasets, and supervision sources. Benchmark integrations, reference policies, and selected policy checkpoints provide practical resources for evaluating, reproducing, and adapting these models. Together, these capabilities let researchers begin with existing implementations and resources, while retaining the flexibility to develop their own model architectures, training procedures, and experimental settings.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript organization and LaTeX editing. The authors review all AI-assisted material and take full responsibility for the scientific claims, experimental results, and artifacts.

REPRODUCIBILITY STATEMENT

StarVLA supports reproducible experimentation through 27 registered policy implementations, reusable training configurations, and 41 available policy checkpoint repositories, including community-contributed resources. Table 2 summarizes the implementations, while Appendix C documents checkpoint resources, benchmark integration, and reported evaluation settings. Data preparation and representative training recipes are described in Appendix B; Appendix E.1 documents configuration settings and snapshots for reusing experiment parameters and loading checkpoints. Community research has also used and extended StarVLA for model development, training, and evaluation, as summarized in Figure 2. Appendix F details selected examples and their scope of reuse.

REFERENCES

Aniri, Chen Yilin, Jinhe Bi, Junfei Guo, Donglai Ran, Xu Bian, Zengjie Jin, Yujun Wang, Yijun Tian, Volker Tresp, Fei Shen, Tat-Seng Chua, and Yunpu Ma. FabriMAE I Trust Myself? Self-Evaluating VLA Action Generation with Markov Attention Entropy. *arXiv preprint arXiv:2608.16697*, 2026. URL https://arxiv.org/abs/2608.16697v1.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. *arXiv preprint arXiv:2602.01166*, 2026a. URL https://arxiv.org/abs/2602.01166.

Shuanghao Bai, Dakai Wang, Cheng Chi, Wanqi Zhou, Jing Lyu, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Lei Xing, Shanghang Zhang, et al. Reshaping action error distributions for reliable vision-language-action models. *arXiv preprint arXiv:2602.04228*, 2026b. URL https://arxiv.org/abs/2602.04228.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black et al. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Nils Blank, Paul Mattes, Maximilian Xiling Li, Jakub Suliga, Thomas Roth, Moritz Reuss, Pankhuri Vanjani, and Rudolf Lioutikov. SPARC: Reliable Spatial Annotations from Robot Demonstrations at Scale. *arXiv preprint arXiv:2606.13497*, 2026. URL https://arxiv.org/abs/2606.13497v1.

Xizhou Bu, Qingda Hu, Lei Zhou, Lingfeng Zhang, Yingbo Tang, Zihao Liu, Xinyi Tao, Zhiqiang Ma, Qingqiu Huang, Chufeng Tang, Hongbo Wang, Jing Zhang, Jiayi Ma, Hangjun Ye, Wei Li, and Xiaoshuai Hao. What Matters for Latent Actions in Robot Learning. *arXiv preprint arXiv:2608.19613*, 2026. URL https://arxiv.org/abs/2608.19613v1.

Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, Mustafa Shukor, Jess Moss, Alexander Soare, Dana Aubakirova, Quentin Lhoest, Quentin Gallouédec, and Thomas Wolf. LeRobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

Hanyang Chen, Hongliang Li, Jiarui Cao, Yang Li, Yang Jiang, Haonan Wen, Kaiyu Huang, Shengnan Guo, and Huaiyu Wan. Beyond English: Uncovering the Multilingual Gap in Vision-Language-Action Models. *arXiv preprint arXiv:2606.15714*, 2026a. URL https://arxiv.org/abs/2606.15714v1.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent world action models for efficient dynamics-aware robot policies. *arXiv preprint arXiv:2606.15768*, 2026b.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025a.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026c. URL https://arxiv.org/abs/2607.04434v3.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, et al. InternVLA-M1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025b.

Ziyang Chen, Shaoguang Wang, Weiyu Guo, Qianyi Cai, He Zhang, Pengteng Li, Yiren Zhao, and Yandong Guo. PHASER: Phase-Aware and Semantic Experience Replay for Vision-Language-Action Models. *arXiv preprint arXiv:2606.03598*, 2026d. URL https://arxiv.org/abs/2606.03598v2.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Yinwei Dai, Zhuofu Chen, Lijie Yang, and Ravi Netravali. Geometry Guided Self-Consistency for Physical AI. *arXiv preprint arXiv:2605.08638*, 2026. URL https://arxiv.org/abs/2605.08638v1.

Dwip Dalal, Shivansh Patel, Chahit Jain, Jeonghwan Kim, Utkarsh Mishra, Alex Baratian, Hyeonjeong Ha, Heng Ji, Svetlana Lazebnik, and Unnat Jain. Generalizable VLA Finetuning via Representation Anchoring and Language-Action Alignment. *arXiv preprint arXiv:2607.13429*, 2026. URL https://arxiv.org/abs/2607.13429v1.

Trung Dao, Sankalp Yamsani, Jaden Park, Joohyung Kim, and Yong Jae Lee. Think Like a World Model, Act Like a VLA: Distilling World-Model Representations into Compact Robot Policies. *arXiv preprint arXiv:2609.24682*, 2026. URL https://arxiv.org/abs/2609.24682v2.

Xingyu Ding, Yuzhong Zhao, Chunhai Zhao, Yinghuan Shi, Chaoyang Zhao, and Yifan Zhang. Temporal Forcing: 4D Representation Alignment for Vision-Language-Action Models. *arXiv preprint arXiv:2608.30643*, 2026. URL https://arxiv.org/abs/2608.30643v1.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. In *European Conference on Computer Vision (ECCV)*, pp. 314–333, 2026. doi: 10.1007/978-3-032-37592-6_18. URL https://link.springer.com/chapter/10.1007/978-3-032-37592-6_18.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, June 2026.

Xintong Hu, Xuhong Huang, Jinyu Zhang, Yutong Yao, Yuchong Sun, Qiuyue Wang, Mingsheng Li, Sicheng Xie, Yitao Liu, Junhao Chen, Yixuan Chen, Yingming Zheng, Shuai Bai, and Tao Yu. FineVLA: Fine-Grained Instruction Alignment for Steerable Vision-Language-Action Policies. *arXiv preprint arXiv:2605.27284*, 2026. URL https://arxiv.org/abs/2605.27284v3.

Zheng Huang, Mingyu Liu, Xiaoyi Lin, Muzhi Zhu, Canyu Zhao, Zongze Du, Ye Lin, Xiaoman Li, Yiduo Jia, Hao Zhong, et al. NoTVLA: Semantics-preserving robot adaptation via narrative action interfaces. *arXiv preprint arXiv:2510.03895*, 2025. URL https://arxiv.org/abs/2510.03895.

Yuming Jiang, Siteng Huang, Shengke Xue, Yaxi Zhao, Jun Cen, Sicong Leng, Kehan Li, Jiayan Guo, Kexiang Wang, Mingxiu Chen, Fan Wang, Deli Zhao, and Xin Li. RynnVLA-001: Using human demonstrations to improve robot manipulation. *arXiv preprint arXiv:2509.15212*, 2025.

Dong Jing, Tianqi Zhang, Jiaqi Liu, Jinman Zhao, Zelong Sun, Li Erran Li, Zhiwu Lu, and Mingyu Ding. Learning Action Priors for Cross-embodiment Robot Manipulation. *arXiv preprint arXiv:2606.26095*, 2026a. URL https://arxiv.org/abs/2606.26095v1.

Zhi Jing, Jinbin Qiao, Ouyang Lu, Jicong Ao, Shuang Qiu, Huazhe Xu, Yu-Gang Jiang, and Chenjia Bai. AssemLM: A spatial reasoning multimodal large language model for robotic assembly. *arXiv preprint arXiv:2604.08983*, 2026b. URL https://arxiv.org/abs/2604.08983.

Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. EgoMimic: Scaling imitation learning via egocentric video. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025. doi: 10.1109/ICRA55743.2025.11127989. URL https://ieeexplore.ieee.org/document/11127989.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, pp. 104–120, 2020. doi: 10.1007/978-3-030-58604-1_7. URL https://link.springer.com/chapter/10.1007/978-3-030-58604-1_7.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, et al. BEHAVIOR-1K: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.

Dachong Li, ZhuangZhuang Chen, Jin Zhang, and Jianqiang Li. CorridorVLA: Explicit spatial constraints for generative action heads via sparse anchors. *arXiv preprint arXiv:2604.21241*, 2026a. URL https://arxiv.org/abs/2604.21241.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. SimpleVLA-RL: Scaling VLA training via reinforcement learning. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=TQhSodCM4r.

Siyi Li, Yuchen Kang, Wuliang Wang, Zhengjie Zhang, Jiangpin Liu, Jianhao Yao, and Jie Chen. DeepInsight II: One Trace from Benchmark to Robot. *arXiv preprint arXiv:2608.16556*, 2026c. URL https://arxiv.org/abs/2608.16556v1.

Wei Li, Peijin Jia, Yuan Ma, Xuefeng Jiang, Titong Jiang, Sheng Sun, Yujian Li, Xin Wen, Han Hong, Zhikang Liu, Bailin Li, and Kun Zhan. FoMoVLA: Bridging Visual Foresight and Motion Guidance for Vision-Language-Action Models. *arXiv preprint arXiv:2607.14739*, 2026d. URL https://arxiv.org/abs/2607.14739v1.

Wenbo Li, Yiteng Chen, Wei Zhang, Wenhao Li, Jun Yang, and Qingyao Wu. SCULPT-VLA: Learning Structured Control through Staged Action Grounding. *arXiv preprint arXiv:2609.23275*, 2026e. URL https://arxiv.org/abs/2609.23275v1.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8(2):158–172, 2026f. doi: 10.1038/s42256-025-01168-7. URL https://www.nature.com/articles/s42256-025-01168-7.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Yibo Li, Enshen Zhou, Rui Chen, Yanjun Ding, Mengzhen Liu, Yi Han, Jiabo Zhan, Lipeng Wang, Shanghang Zhang, and Lu Sheng. ActiveArena: Benchmarking and Understanding Active Perception in Robotic Manipulation. *arXiv preprint arXiv:2609.24124*, 2026g. URL https://arxiv.org/abs/2609.24124v2.

Yinhao Li, Weixin Mao, Zihan Lan, Jikun Rong, Qirui Hu, Yiming Zhang, Weipeng Deng, Bowen Shen, Minzhao Zhu, Yiming Mao, et al. FluxVLA engine: A one-stop VLA engineering platform for embodied intelligence. *arXiv preprint arXiv:2609.17210*, 2026h.

Shijie Lian, Bin Yu, Xiaopeng Lin, Zhaolong Shen, Laurence Tianruo Yang, Yurun Jin, Haishan Liu, Changti Wu, Hang Yuan, Cong Huang, and Kai Chen. IntentVLA: Short-Horizon Intent Modeling for Aliased Robot Manipulation. *arXiv preprint arXiv:2605.14712*, 2026a. URL https://arxiv.org/abs/2605.14712v3.

Shijie Lian, Bin Yu, Xiaopeng Lin, Changti Wu, Hang Yuan, Xiaolin Hu, Zhaolong Shen, Yuzhuo Miao, Haishan Liu, Yuxuan Tian, Yukun Shi, Cong Huang, and Kai Chen. PhysBrain 1.0 Technical Report. *arXiv preprint arXiv:2605.15298*, 2026b. URL https://arxiv.org/abs/2605.15298v1.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T. Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. LangForce: Bayesian decomposition of vision language action models via latent action queries. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026c. URL https://arxiv.org/abs/2601.15197.

Tao Lin, Yuxin Du, Yiran Mao, Zewei Ye, Yilei Zhong, Bing Cheng, Yiming Wang, Jiting Liu, Yang Tian, Junchi Yan, Feiran Wu, Zenan Meng, Hu Wei, Yuqian Fu, Gen Li, and Bo Zhao. LA4VLA: Learning to Act without Seeing via Language-Action Pretraining. *arXiv preprint arXiv:2606.27295*, 2026a. URL https://arxiv.org/abs/2606.27295v2.

Weifeng Lin, Siyuan Huang, Hao Li, Tingwei Chen, Ruichuan An, Xinyu Wei, Jianbo Liu, and Hongsheng Li. Rethinking VLM Representation for VLA Initialization. *arXiv preprint arXiv:2605.25802*, 2026b. URL https://arxiv.org/abs/2605.25802v1.

Xiaopeng Lin, Shijie Lian, Bin Yu, Ruoqi Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Yurun Jin, Yukun Shi, Jiyan He, Cong Huang, Bojun Cheng, and Kai Chen. PhysBrain: Human egocentric data as a bridge from vision language models to physical intelligence. *arXiv preprint arXiv:2512.16793*, 2025. URL https://arxiv.org/abs/2512.16793.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 44776–44791, 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-Horizon Manipulation via Trace-Conditioned VLA Planning. *arXiv preprint arXiv:2604.21924*, 2026a. URL https://arxiv.org/abs/2604.21924v1.

Jiahao Liu, Zhongpu Xia, Shuai Tian, Huangrui Li, Yuhang Zheng, Ning Ma, Xin Fu, Xiaotian Liu, Jing Li, Yixian Li, ShangQing Zhou, Zebin Xing, Linbo Wang, Chaoyue Li, Haoran Li, and Dongbin Zhao. WALA Learning Executable Latent Actions from Action-Labeled Demonstrations and Action-Free Videos. *arXiv preprint arXiv:2607.11397*, 2026b. URL https://arxiv.org/abs/2607.11397v1.

Zheng Lu, Haoran Liao, Wanqi Zhong, Yunhe Ni, Lijie Wang, Xingjie Fan, Zhisheng Chen, Yantang Qu, Meijia Chen, Tianyu Xin, Zirui Song, and Yiming Li. LayerRoute: Action-Conditioned Mixture-of-Layers Routing for Vision-Language-Action Policies. *arXiv preprint arXiv:2609.06079*, 2026. URL https://arxiv.org/abs/2609.06079v2.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. LDA-1B: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026. URL https://arxiv.org/abs/2602.12215.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. DiT4DiT: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026a. URL https://arxiv.org/abs/2603.10448.

Zipei Ma, Xiaofei Wei, Junzhe Jiang, Shunlin Lu, and Li Zhang. TacPAC: Tactile Prediction and Real-Time Action Correction in World-Action Models for Contact-Rich Manipulation. *arXiv preprint arXiv:2609.05266*, 2026b. URL https://arxiv.org/abs/2609.05266v1.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://ieeexplore.ieee.org/document/9788026.

Jean Mercat, Sedrick Keh, Kushal Arora, Isabella Huang, Paarth Shah, Haruki Nishimura, Shun Iwase, and Katherine Liu. VLA Foundry: A unified framework for training vision-language-action models. *arXiv preprint arXiv:2604.19728*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.050. URL https://www.roboticsproceedings.org/rss20/p050.html.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, 2026.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734v2.

Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://ieeexplore.ieee.org/document/10611477.

Yuhao Pan, Haosong Peng, Zhengshen Zhang, Zhengyang Yan, Yalun Dai, Fushuo Huo, Chujie Wang, Tianyu Qi, Xiucheng Wang, Nan Cheng, and Wenchao Xu. World-to-Wrist: Task-conditioned future wrist modeling for fine-grained robot manipulation. *arXiv preprint arXiv:2608.05369*, 2026. URL https://arxiv.org/abs/2608.05369.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

Boyang Shen, Kaixiang Yang, Hao Wang, Qiuyu Yu, Qiang Xie, Qiang Li, and Zhiwei Wang. LoopVLA: Learning Sufficiency in Recurrent Refinement for Vision-Language-Action Models. *arXiv preprint arXiv:2605.09948*, 2026. URL https://arxiv.org/abs/2605.09948v2.

Simple AI, Yuteng Wei, Jinming Ma, Jiawei Wang, Weitao Zhou, Yushen Zuo, Ke Rui, Minglei Li, Jinhao Zhang, Zhikang Pan, Xiang Wang, Haoran Jia, Huan Du, Zicheng Zeng, Jun Ma, Guiyu Qin, Di Zhang, and Xiaofei Li. HiFi-UMI: Learning Deployable Manipulation Policies from High-Fidelity UMI Data Alone. *arXiv preprint arXiv:2607.25895*, 2026. URL https://arxiv.org/abs/2607.25895v1.

Haoran Sun, Wentao Zhang, Junyang Hua, Hedan Yang, Yongjian Guo, Yifei Zhang, Xiaolong Xiang, Mingxi Luo, Jing Long, Chen Zhao, Chen Zhou, Wanting Xu, Qiming Yang, Hui Zhang, Song Wang, Xiaodong Bai, Shuai Di, Xu Chu, Xiaotie Deng, Yicheng Gong, and Junwu Xiong. JoyNexus: Service-Oriented Multi-Tenant Post-Training for VLA Models. *arXiv preprint arXiv:2607.16074*, 2026a. URL https://arxiv.org/abs/2607.16074v1.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026b. doi: 10.1007/978-3-032-37196-6_26. URL https://link.springer.com/chapter/10.1007/978-3-032-37196-6_26.

Jun Sun, Boyu Yang, Jiahao Zhang, Ning Ma, Chencheng Wu, Siqing Zhang, Yiou Huang, Qiufeng Wang, Shan Liang, and Yaran Chen. TempoFit: Plug-and-play layer-wise temporal KV memory for long-horizon vision-language-action manipulation. *arXiv preprint arXiv:2603.07647*, 2026c. URL https://arxiv.org/abs/2603.07647.

Luoyang Sun, Guoyang Xia, Fengfa Li, Lei Ren, Xinyu Cui, Haifeng Zhang, Fangxiang Feng, Kaike Zhang, Kun Zhan, Yan Xie, Jun Wang, and Cheng Deng. What Makes an Efficient VLA? Navigating Action-Head Design, Scaling, and Latency. *arXiv preprint arXiv:2609.13984*, 2026d. URL https://arxiv.org/abs/2609.13984v2.

Jingkai Wang, Zihan Tang, Gu Zhang, Mingyu Cao, Jiapeng Chen, Jingjiao Zhao, Xiansheng Chen, Pengwei Wang, Lemao Liu, and Dejing Dou. SLIM-0.5B: Learning Action-Grounded Predictive Latents for Robot Manipulation. *arXiv preprint arXiv:2608.09771*, 2026a. URL https://arxiv.org/abs/2608.09771v1.

Linhan Wang, Zijian An, Mingyuan Zhang, Chen Dai, Yi Xu, Can Cui, Zichong Yang, Yinlin Chen, Lifeng Zhou, and Chang-Tien Lu. GlanceWAM: Sparse Test-Time Imagination for World-Action Models. *arXiv preprint arXiv:2608.23927*, 2026b. URL https://arxiv.org/abs/2608.23927v1.

Pengcheng Wang, Kaiwen Hong, Chensheng Peng, Katherine Driggs-Campbell, Masayoshi Tomizuka, Chenfeng Xu, and Chen Tang. DiscreteRTC: Discrete Diffusion Policies are Natural Asynchronous Executors. *arXiv preprint arXiv:2604.25050*, 2026c. URL https://arxiv.org/abs/2604.25050v3.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-VLA: Action-Aligned Visual Planning via Internalized Affordance. *arXiv preprint arXiv:2605.24203*, 2026d. URL https://arxiv.org/abs/2605.24203v1.

Xinkai Wang, Chenyi Wang, Yifu Xu, Mingzhe Ye, Fucheng Zhang, Jialin Tian, Xinyu Zhan, Lifeng Zhu, Cewu Lu, and Lixin Yang. LaMP: Learning vision-language-action policy with 3D scene flow as latent motion prior. In *European Conference on Computer Vision*, pp. 227–244. Springer, 2026e. doi: 10.1007/978-3-032-37013-6_13. URL https://link.springer.com/chapter/10.1007/978-3-032-37013-6_13.

Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, et al. OpenWAM: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026f.

Zixuan Wang, Yuxin Chen, Yuqi Liu, Jinhui Ye, Pengguang Chen, Changsheng Lu, Shu Liu, Bei Yu, and Jiaya Jia. VP-VLA: Visual prompting as an interface for vision-language-action models. *arXiv preprint arXiv:2603.22003*, 2026g. URL https://arxiv.org/abs/2603.22003.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2024.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic VLA foundation model. *arXiv preprint arXiv:2601.18692*, 2026a.

Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, et al. From foundation to application: Improving VLA models in practice. *arXiv preprint arXiv:2607.06403*, 2026b.

Guoyang Xia, Fengfa Li, Hongjin Ji, Lei Ren, Fangxiang Feng, Kun Zhan, and Yan Xie. VLAFlow: A Unified Training Framework for Vision-Language-Action Models via Co-training and Future Latent Alignment. *arXiv preprint arXiv:2607.01586*, 2026. URL https://arxiv.org/abs/2607.01586v2.

Junjin Xiao, Dongyang Li, Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Feng Xiong, Mu Xu, Xing Wei, Zhiheng Ma, Qing Zhang, and Wei-Shi Zheng. Learning Action Manifold with Multi-view Latent Priors for Robotic Manipulation. *arXiv preprint arXiv:2605.11832*, 2026. URL https://arxiv.org/abs/2605.11832v1.

Bin Xie, Erjin Zhou, Fan Jia, Hao Shi, Haoqiang Fan, Haowei Zhang, Hebei Li, Jianjian Sun, Jie Bin, Junwen Huang, et al. Dexbotic: Open-source vision-language-action toolbox. *arXiv preprint arXiv:2510.23511*, 2025.

Hengyi Xie, Chenfei Yao, Xianjin Wu, Yingying Zhu, Dingkang Liang, Xiang Bai, and Han Ding. TurboVLA: Real-Time Vision-Language-Action Model at 32 Hz on an RTX 4090 with $<$1 GB VRAM. *arXiv preprint arXiv:2607.27205*, 2026. URL https://arxiv.org/abs/2607.27205v2.

XPolicyLab Community, Tianxing Chen, Yue Chen, Tian Nian, Zijian Cai, Guangyu Chen, Wenwei Lin, Qiwei Liang, Zanxin Chen, Peicheng Xiang, et al. XPolicyLab: A unified standard and open ecosystem for robot policy evaluation and deployment. *arXiv preprint arXiv:2608.09892*, 2026. URL https://arxiv.org/abs/2608.09892v3.

Siyu Xu, Yunke Wang, Zijian Wang, Dihao Zhu, Chenghao Xia, Chengbin Du, Daochang Liu, Tao Huang, and Chang Xu. StellaVLA: In-Context Structured Demonstration for Generalizable Vision-Language-Action Models. *arXiv preprint arXiv:2608.11671*, 2026. URL https://arxiv.org/abs/2608.11671v1.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026a.

Heqing Yang, Yang Yi, Liyao Wang, Linqing Zhong, Donglin Yang, Ruipu Wu, Zitong Bai, Fengjiao Chen, Manyuan Zhang, Linjiang Huang, and Si Liu. EVA-Client: A Unified Data Collection, Inference, and Deployment Framework for Embodied Policies on Real Robots. *arXiv preprint arXiv:2607.02646*, 2026b. URL https://arxiv.org/abs/2607.02646v1.

Senqiao Yang, Chengyao Wang, Yuxin Chen, Zixuan Wang, Longxiang Tang, Haokun Gui, Jinhui Ye, Changsheng Lu, Xiaoyang Wu, Mingkang Zhu, et al. Beyond data scaling: Representation-centric continued pre-training for vision-language-action models. *arXiv preprint arXiv:2608.27550*, 2026c.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. ABot-M0: VLA foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026d.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Bin Yu, Shijie Lian, Xiaopeng Lin, Zhaolong Shen, Yuliang Wei, Changti Wu, Hang Yuan, Haishan Liu, Bailing Wang, Cong Huang, and Kai Chen. FrameSkip: Learning from Fewer but More Informative Frames in VLA Training. *arXiv preprint arXiv:2605.13757*, 2026a. URL https://arxiv.org/abs/2605.13757v1.

Bin Yu, Shijie Lian, Xiaopeng Lin, Yuliang Wei, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Xinming Wang, Bailing Wang, Cong Huang, et al. TwinBrainVLA: Unleashing the potential of generalist VLMs for embodied tasks via asymmetric mixture-of-transformers. *arXiv preprint arXiv:2601.14133*, 2026b. URL https://arxiv.org/abs/2601.14133.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient Large-Scale reinforcement learning via Macro-to-Micro flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 829–846. USENIX Association, 2026c. URL https://www.usenix.org/conference/osdi26/presentation/yu-chao.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025. URL https://proceedings.mlr.press/v270/zawalski25a.html.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-Arena: An open-source framework for benchmarking vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=qjeCH95rbF.

Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. CLAP: Contrastive latent action pretraining for learning vision-language-action models from human videos. *arXiv preprint arXiv:2601.04061*, 2026b. URL https://arxiv.org/abs/2601.04061.

Lingfeng Zhang, Xiaoshuai Hao, Yingbo Tang, Lei Zhou, Shuyi Zhang, Jinkun Liu, Hongsheng Li, Chenhao Zhang, Qiang Zhang, Hangjun Ye, Xiaojun Liang, Long Chen, and Wenbo Ding. OneVLA: A Unified Framework for Embodied Tasks. *arXiv preprint arXiv:2606.01241*, 2026c. URL https://arxiv.org/abs/2606.01241v2.

Tianle Zhang, Zhihao Yuan, Dafeng Chi, Peidong Liu, Dongwei Li, Kejun Hu, Likui Zhang, Junnan Nie, Ziming Wei, Zengjue Chen, et al. JoyAI-RA 0.1: A foundation model for robotic autonomy. *arXiv preprint arXiv:2604.20100*, 2026d. URL https://arxiv.org/abs/2604.20100.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Yunao Zheng, Guoyang Xia, Xiaojie Wang, and Lei Ren. Lngram: N-gram Conditional Memory in Latent Space. *arXiv preprint arXiv:2605.24869*, 2026. URL https://arxiv.org/abs/2605.24869v1.

Yiming Zhong, Yaoyu He, Zemin Yang, Pengfei Tian, Yifan Huang, Qingqiu Huang, Xinge Zhu, and Yuexin Ma. From noise to intent: Anchoring generative VLA policies with residual bridges. *arXiv preprint arXiv:2604.21391*, 2026. URL https://arxiv.org/abs/2604.21391.

Bin Zhou, Jialong Liu, Jianan Wang, Changhao Chen, and Kani Chen. CoRe-WAM: Correspondence-Aligned Temporal Residuals for World Action Models. *arXiv preprint arXiv:2609.27314*, 2026a. URL https://arxiv.org/abs/2609.27314v1.

Yang Zhou, Xiaofeng Wang, Hao Shao, Letian Wang, Guosheng Zhao, Jiangnan Shao, Jiagang Zhu, Tingdong Yu, Zheng Zhu, Guan Huang, et al. DriveDreamer-Policy: A geometry-grounded world-action model for unified generation and planning. *arXiv preprint arXiv:2604.01765*, 2026b. URL https://arxiv.org/abs/2604.01765.

Brianna Zitkovich et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Release Scope and Model Details

The appendix documents the implementations, recipes, and evaluation conditions supporting the three contribution chapters. It includes model and research-use catalogs, data and training details, deployment interfaces, and benchmark protocols and results.

The implementation descriptions refer to the code snapshot inspected on September 17, 2026. The model repository catalog was inspected on the same date. Its 41 public repositories contain several kinds of resources, including pretrained components and outputs of separate research projects. This catalog count is not a count of independently evaluated robot policies. Reported results retain their respective model, checkpoint, and evaluation settings.

This manuscript uses source inspection, released configurations, model cards, and published reports. No new training or benchmark runs were performed during its preparation. Source-reported task and throughput values retain their original experimental scope. The code snapshot documents current implementations. It does not identify the training revision of every historical checkpoint or efficiency run.

Data integration and training recipes follow in Appendix B. Benchmark resources and results are grouped in Appendix C, and policy serving and robot deployment in Appendix D. Appendix E collects experiment configuration and training efficiency. Research uses and framework comparisons appear in Appendices F and G.

A.1 Policy Implementations

Table 2 lists one registered entry name for each implementation included in this catalog.

Policy implementation count. The policy implementation catalog covers 27 registered policy implementations: two in `VM4A`, 19 in `VLM4A`, and six in `WM4A`. Each registered class is counted once, and aliases are not counted as additional implementations. The count includes classes that inherit computation from another implementation. It does not establish an independently evaluated algorithm or released checkpoint for every entry.

The aliases `Pi0`, `Pi05`, and `QwenFM` resolve to `PI0`, `PI05`, and `QwenPI`, respectively, and are not counted again. These entries document implementation coverage. Their availability does not imply that every backbone and action head combination has been implemented or evaluated.

Visuomotor adaptation. Action Chunking with Transformers (ACT), adapted from LeRobot, uses visual representations for action-chunk prediction (Zhao et al., 2023; Cadene et al., 2026).

Initialization options. Compatible pretrained components from VLAct and ABot-M0 can initialize selected frameworks (Yang et al., 2026c;d). The framework and its modules handle the corresponding input conventions and representation connections, allowing subsequent policy development with applicable action heads and training recipes.

A.2 Model Composition and Interfaces

The following details expand the model interfaces in Section 3.1.

Compositional model construction. A framework combines pretrained components with action heads and defines the representations passed between them and the learning objectives. Model construction includes choosing components and specifying their connections. A method may select features from particular layers, combine representations from multiple components, or introduce a new computational branch. Existing modules provide reusable building blocks for these changes. The top-level computation keeps the path from observations to representations, actions, and losses in the method implementation, where it can be read alongside the architecture diagram. Figure 5 illustrates this computation and its shared training and prediction interfaces.

Shared observation interface. Training and prediction use inspectable observations, including pixel images, readable language, and optional robot state. Each method handles its required modali-

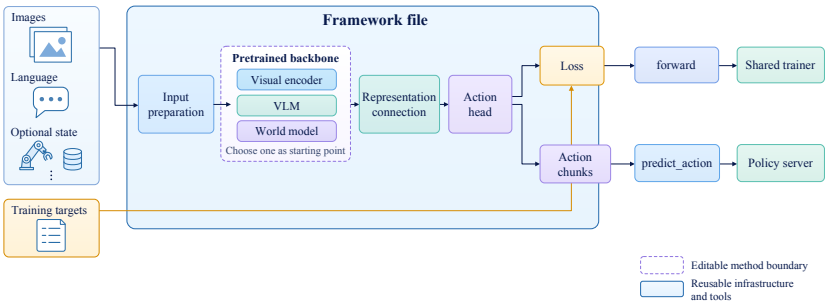


Figure 5: Model computation and shared interfaces. A framework prepares method-specific inputs, connects pretrained components to an action head, and defines its outputs and learning objectives. The trainer calls `forward` to obtain losses, while policy serving calls `predict_action` to obtain predicted action chunks. Internal representations and objectives remain method-specific.

ties and model-specific preparation within the framework and its modules. Keeping this preparation with the model helps maintain consistent processing between training and inference. The policy notation in Equation 1 accommodates method-specific modalities and auxiliary outputs. It does not require every modality or auxiliary output to be present in every implementation or exposed through the prediction entry point.

Framework training and prediction calls. The standard robot-training path calls `forward` to obtain the loss computed by the framework. Training samples include the supervision required by the selected objective. For prediction, `predict_action` returns predicted action chunks to the policy wrapper. The framework defines model-specific input construction, tokenization, encoding, and learning objectives, while the training entry point organizes parameter updates.

B Data Integration and Training Recipes

This section expands the data and training workflows in Section 4.

B.1 Data Preparation and Sampling

Representative robot sample. The standard LeRobot robot-data path constructs samples of the following form. This is an interface illustration rather than executable array construction:

```
{
    "image": [pixel_image_view_1, pixel_image_view_2],
    "lang": "pick up the object",
    "action": action_array,  # [time, action_dimension]
    "robot_tag": embodiment_identifier,
    "state": state_array    # optional
}
```

The collate function retains a list of sample dictionaries. Sampling, resizing, and configured numerical transforms occur before the model call. The framework and its modules prepare the selected model's token or feature inputs. In the inspected default packing path, one temporal entry is selected per camera. Supporting an image history requires a corresponding packing and model path even when lower-level readers can retrieve multiple offsets.

Data configuration and sampling. A local data configuration supplies modality keys and temporal offsets, plus state/action transforms. These settings specify camera views, language fields, ordered state and action components, and action windows. Figure 6 summarizes the source-specific preparation paths and their connections to learning objectives. Dataset mixtures provide dataset names, dataset sampling weights, and robot data-configuration identifiers. The sampler chooses a dataset, trajectory, and step. Its optional dataset-length and trajectory-length weighting flags affect

the resulting distribution. Experiments should record the actual mixture and flags. Data configurations can also provide a dataset-construction hook. These interfaces use existing LeRobot-format and GR00T data mechanisms (Cadene et al., 2026; NVIDIA et al., 2025). The corresponding source files preserve upstream attribution and modification notices.

Extending data sources. Local registries allow additional embodiment and dataset configurations, while dataset factories support specialized readers. Optional sample adapters arrange observations and available action or motion annotations for the selected framework. Robot and Universal Manipulation Interface (UMI) datasets (Chi et al., 2024) can use named dataset mixtures that associate each dataset with its configuration and dataset sampling weight.

Embodiment-specific configurations. The release provides separate configurations for LIBERO, RoboTwin, and RoboCasa-GR1. These define the camera, state, action, and language fields together with temporal indices and transformations. The examples illustrate alternative data configurations. No joint training result across these three embodiments is inferred from their presence in the release.

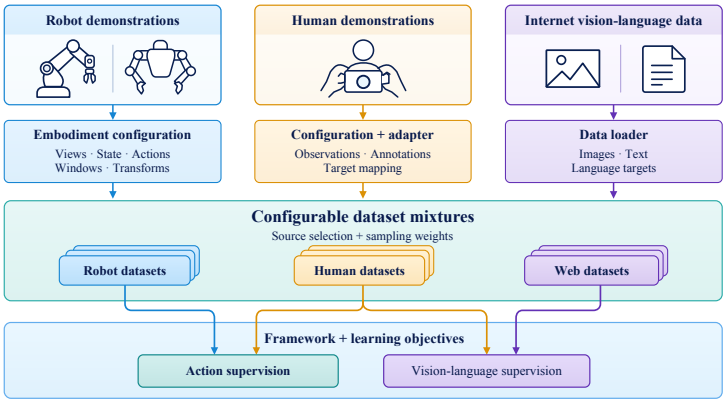


Figure 6: Source-specific data preparation and supervision. Robot demonstrations use embodiment configurations; human demonstrations use compatible configurations and adapters; vision-language data retain their own loaders. The mixture region summarizes source selection and sampling weights. Robot dataset mixtures and vision-language streams retain separate preparation and sampling settings, and supervision routes depend on the available annotations and selected framework.

Vision-language sample preparation. Dedicated dataloaders prepare image-text and multimodal instruction samples with language targets. Data settings select sources and control image pixel limits and maximum sequence length. Processing follows the selected VLM's input requirements, and the data stream retains its own batch structure. Source settings can combine vision-language datasets and optionally subsample each source. These loader settings remain separate from the robot dataset-mixture sampling settings.

B.2 Training Entry Points

The Policy Trainer corresponds to `train_starvla.py` and uses configured dataset mixtures with action supervision for supervised robot post-training and cross-embodiment training. These mixtures can include robot data and human-centered data with compatible supervision. The VLM Trainer corresponds to `train_starvlm.py` and trains on a multimodal data stream. The Co-training Trainer corresponds to `train_starvla_cotrain.py`. It combines a dataset mixture with action supervision and a multimodal data stream. Robot data and human-centered data can share the dataset mixture, with the selected framework handling their observations and targets. The training recipe controls source sampling and the contribution of each objective to parameter updates.

B.3 Representative Training Recipes

Robot-only training recipes. The LIBERO recipe combines demonstrations from four task suites under one embodiment configuration. Cross-embodiment training combines demonstrations from different robots (Open X-Embodiment Collaboration, 2024). The SimplerEnv recipe combines Bridge data from WidowX with RT-1 data from the Google robot. The selected framework determines how their observations and action targets enter the policy and learning objective. Such recipes can initialize from pretrained components or continue from a compatible robot checkpoint for task adaptation.

Combining robot and human supervision. Robot-human co-training combines complementary supervision (Kareer et al., 2025). Human-data recipes select objectives that the framework can compute from the available annotations, including action or motion targets from UMI or egocentric recordings. The recipe combines these objectives with robot action learning during parameter updates, allowing the sources to provide different forms of supervision. The source configuration and selected framework must retain compatible observation and target conventions.

Multimodal co-training. One instance of multimodal co-training combines robot action supervision with vision-language supervision (Black et al., 2025a). One co-training configuration selects QwenGR00T, LIBERO Goal robot data, and ShareGPT4V-COCO image-text data, with per-device batch sizes of 16 and 4 respectively. The trainer backpropagates the action loss and the Qwen language loss multiplied by `trainer.loss_scale.vlm`, set to 0.1 in this recipe, before the optimization update. The configuration also contains a `vla` loss-scale field, but the inspected training step does not read it. These details describe the released configuration and implementation. This manuscript does not report a new execution or performance result for that recipe. A separate launch example selects QwenFast, all four LIBERO suites, and the same vision-language source, with per-device batch sizes of two for each data stream. These are alternative configurations. Their model, dataset, and batch settings should be retained together.

Additional data source: UMI/DexWild. The released short-run configuration selects the following settings:

Setting	Value in the example
Framework	QwenOFT
Dataset mixture	`umi_dexwild_400`
Loader	`lerobot_datasets`
Action / state dimensions	23 / 23
Action horizon	8
VLM training	Frozen
Action-head learning rate	10^{-4}
Training steps	20

The registry maps this converted dataset to two camera fields and its action and state fields. A previously reported loader check covered 346 trajectories and 123,485 dataset steps. The reported short training check completed 20 optimizer steps and saved the model. The underlying run logs were not inspected for this manuscript. This reported execution is a short training check, with no associated human-to-robot transfer score.

An optional `umi_datasets` path adds validation and sample packing. It resolves action dimensions from the data and model configuration, checks inconsistent declared action semantics, and can return validity masks. QwenOFT consumes an action mask when provided by every sample in the batch. The standard-loader example above and this optional adapter configuration are separate paths. Masks returned by an adapter require a model that explicitly uses them. Observation-only manifests with null actions require a different learning objective and data path.

C Benchmark Integration and Reference Resources

C.1 Benchmark Coverage

Table 5 summarizes embodiment and task coverage, evaluation sizes, and available reference resources.

Table 5: Simulation benchmark coverage and reference resources. Evaluation sizes are rollouts for one documented protocol unless stated otherwise. Reference configurations are counted per benchmark integration. Policy checkpoint repositories are counted separately.

Benchmark	Embodiment	Task coverage	Evaluation size	Reference configs.	Checkpoint repositories
LIBERO	Franka Panda	40 tasks in four suites	2,000 rollouts	≥ 12	≥ 7
SimplerEnv	WidowX, Google robot	Four WidowX tasks, Google manipulation settings	Protocol-specific	≥ 7	≥ 9
RoboTwin 2.0	Aloha-AgileX	50 bimanual tasks	5,000 / condition	≥ 4	≥ 8
RoboCasa-GR1	GR1	24 tabletop tasks	1,200 rollouts	≥ 4	≥ 2
RoboCasa365	PandaOmron	50 target kitchen tasks	2,500 rollouts	≥ 4	–
RoboDojo	ARX X5	42 evaluation tasks	2,100 rollouts	≥ 3	≥ 4
BEHAVIOR	R1Pro	50 household challenge tasks	Instance-specific	≥ 1	≥ 4
MetaWorld	Sawyer	50 MT50 tasks	500 rollouts	≥ 2	≥ 2
CALVIN	Franka Panda	34 skills in five-task sequences	1,000 sequences	≥ 6	≥ 1
LIBERO-plus	Franka Panda	10,030 perturbation variants	Suite-specific	≥ 5	≥ 1
DOMINO	Aloha-AgileX	35 dynamic tasks	3,500 / setting	≥ 4	≥ 1
VLA-Arena	Franka Panda	11 suites, three difficulty levels	10 / task	≥ 1	≥ 1
VLN-CE	Navigation agent	Room-to-Room (R2R) and Room-Across-Room (RxR) navigation	Split-specific	≥ 1	≥ 1
Overall: 13 integrations, 10 robot embodiments				$\geq$ **54**	$\geq$ **41**

Reference configurations distinguish pretrained components, action heads, and specified state or initialization choices. Repeated metrics, training steps, and evaluation conditions do not add configurations. Policy checkpoint repositories are assigned to one benchmark integration for counting. Reuse across integrations does not increase the repository count. Initialization-only resources are excluded. – indicates that no policy checkpoint repository is catalogued for that integration.

C.2 Resource Catalog

Two inventories are distinguished here: a catalog of all repository types and the policy-only checkpoint inventory used for benchmark coverage. Their different inclusion rules are specified below.

The model repository catalog inspected on September 17, 2026 contains 41 repositories, of which 40 list weight files. Table 6 separates initialization components, robot-policy resources, policies from another research project, and a placeholder without weights. Counts refer to repositories, without deduplication of their weight contents or verification by loading. They therefore do not count distinct reference checkpoints or validated policies.

The categories assign each repository once. The two world-model policy resources use LIBERO, but are grouped by their model route in this catalog. Initialization components require subsequent policy training or adaptation. VLAct resources retain the method and result attribution of that work (Yang et al., 2026c). Resources published outside the inspected catalog are not included in these counts. Evaluation settings for selected checkpoint results are specified in Appendix C.6.

Table 5 uses a separate policy checkpoint repository inventory. It includes the 37 policy checkpoint repositories above and four additional repositories for MetaWorld, CALVIN, and VLN-CE, while excluding initialization-only components. Reuse of a policy across benchmarks does not increase the repository count. The reference-configuration counts distinguish model components, state in-

Table 6: Model repository catalog categories and counts. A listed weight file establishes artifact presence, not successful model loading or benchmark evaluation.

Resource category	Repositories	With weight files
Action-vocabulary-extended foundation models	2	2
VLAct pretrained representation backbone	1	1
BEHAVIOR policies	4	4
Bridge / RT-1 policies	9	9
LIBERO policies	5	5
RoboCasa-GR1 policies	2	2
RoboTwin policies	2	2
RoboDojo policies	3	3
World-model policy resources (WM4A)	2	2
VLAct downstream policies	10	10
Placeholder repository	1	0
Total	41	40

puts, and initialization choices. Repeated training steps, metrics, or evaluation conditions do not add configurations. These are benchmark-specific configurations, so a model used on several benchmarks contributes to each benchmark's coverage.

For reuse of a reported checkpoint, the relevant record consists of its code and weight revisions, initialization, dataset mixture, transforms and statistics, training configuration, environment version, and evaluation protocol. Task lists, rollout counts, seeds, and aggregation rules accompany performance claims where available. Missing fields require confirmation from the corresponding source. Results from separate algorithmic projects retain their original attribution. The public infrastructure can be extended while these resource-specific assumptions remain explicit.

C.3 Benchmark Integration Details

Table 7 lists the 13 simulation benchmark integrations in the release snapshot, summarizing their task scope, embodiments, and integration materials. The descriptions below detail the training data, evaluation procedures, and integration support for each benchmark. Dataset choices and rollout counts refer to the documented recipes. Individual experiments specify their selected policy, checkpoint, and protocol.

RoboCasa-GR1 and RoboCasa365 are counted as distinct benchmark integrations, and LIBERO-plus is counted as an additional evaluation integration. The table describes integration coverage. Checkpoint availability and evaluation results remain specific to the corresponding policy and protocol.

LIBERO (Liu et al., 2023). StarVLA provides a joint training recipe for the spatial, object, goal, and long-horizon LIBERO suites. Training uses the IPEC-COMMUNITY LeRobot conversions with no-op actions removed, with a shared data configuration for external and wrist images, language instructions, and robot actions. The documented baseline trains one policy across all four suites. Evaluation uses each suite's task definitions and stored initial states, with 50 episodes per task and 10 tasks per suite, yielding 500 rollouts per suite. Success rates are reported separately for the four suites. The integration supplies data preparation, baseline training recipes and selected checkpoints, and a client that executes action chunks while retaining LIBERO's task-specific episode limits.

SimplerEnv (Li et al., 2025). The training workflow uses real-robot demonstrations from the Bridge and RT-1 sources in Open X-Embodiment, provided as LeRobot datasets. The `bridge_rt_1` dataset mixture associates these sources with their respective WidowX and Google robot configurations. Evaluation runs the corresponding simulated robot and task settings. Google robot scripts cover both visual matching and variant aggregation. Results retain the robot, task, and evaluation setting so that success under different visual conditions can be interpreted separately. Environment clients adapt rotation and gripper conventions, while the model runs through the shared

Table 7: Benchmark integration overview. Each row identifies a benchmark integration, its task or embodiment scope, and the available integration materials.

Benchmark	Task and embodiment scope	Integration content
LIBERO	Language-conditioned manipulation across spatial, object, goal, and long-horizon suites.	Data preparation, baseline training recipes, released policies, and task-suite evaluation.
SimplerEnv	WidowX and Google robot manipulation in visual-matching and variant-aggregation settings.	Open X-Embodiment data configurations, training recipes, control adapters, and evaluation and aggregation scripts.
RoboTwin 2.0	Bimanual manipulation with clean and randomized demonstrations.	Bimanual data configurations, training and evaluation scripts, and selected checkpoints.
RoboCasa-GR1	Humanoid tabletop manipulation with arm, hand, and waist control.	Embodiment configurations, training recipes, a rollout client, and selected checkpoints.
RoboCasa365	Household manipulation with the PandaOmron robot.	Data and training configurations, a simulator client, and evaluation scripts.
RoboDojo	Bimanual manipulation with the ARX X5 robot.	Data configurations, training recipes, selected checkpoints, and an external evaluation entry point.
BEHAVIOR	Full-length household manipulation tasks.	Data, training, and evaluation entry points, with parts of the snapshot documentation still in progress.
MetaWorld	Multi-task single-arm manipulation in the MT50 setting.	MT50 training and evaluation workflows with external checkpoint links.
CALVIN	Language-conditioned, long-horizon manipulation evaluated through task sequences.	Data conversion, training recipes, and sequential evaluation.
LIBERO-plus	Extended evaluation of LIBERO policies under changed conditions.	Evaluation adapters and zero-shot testing of LIBERO policies.
DOMINO	Dynamic manipulation with moving objects and changing scenes.	Training and evaluation workflows with history inputs, success rates, and manipulation scores.
VLA-Arena	Manipulation across task suites, difficulty levels, and constraint settings.	Training and evaluation entry points with suite-specific success and constraint metrics.
VLN-CE	Vision-and-language navigation in continuous environments.	A dedicated training entry point, checkpoint links, and external navigation evaluation.

policy server. Evaluation scripts check that server and evaluator seeds agree and record the checkpoint, episode counts, and per-episode success values alongside the aggregate results.

RoboTwin 2.0 (Chen et al., 2025a). StarVLA provides bimanual data configurations and multi-task training recipes for the 50 RoboTwin 2.0 tasks. The official data include 50 clean demonstrations and 500 randomized demonstrations per task. The documented baseline settings distinguish training with 50 demonstrations per task from the larger mixture combining clean and randomized data. These data budgets accompany the corresponding checkpoints. Evaluation selects the clean or randomized environment configuration and reports task success under that setting. A benchmark launcher pairs the simulator with a policy server and distributes tasks across available devices. The client handles the selected action representation and control conversion, allowing compatible policies to reuse the task runner while preserving the benchmark's environment configuration and success checks.

RoboCasa-GR1 (NVIDIA et al., 2025). This integration adopts the GR-1 tabletop benchmark introduced with NVIDIA GR00T N1, built on RoboCasa (Nasiriany et al., 2024). It uses the task-specific fine-tuning subsets of NVIDIA's PhysicalAI-Robotics-GR00T-X-Embodiment-Sim dataset. The provided downloader selects the `_1000` task folders, and the training recipe combines 24 tasks in one policy. Its data configuration expresses the humanoid's egocentric observations and arm, hand, and waist control fields. The documented evaluation performs 50 rollouts per task and reports per-task and average success rates. A simulator client maps observations and actions to the GR1

environment, with state input selected according to the evaluated checkpoint. Training recipes and selected Qwen-based checkpoints provide reference policies for this setting.

RoboCasa365 (Nasiriany et al., 2026). This integration uses the PandaOmron robot and the official `target/human` LeRobot bundles, comprising 18 atomic and 32 composite tasks. StarVLA provides single-task and joint target-50 training launchers with a PandaOmron data configuration. The documented full evaluation protocol uses 50 rollouts per task in target kitchens and task-specific horizons from RoboCasa v1.0.1. Results distinguish Atomic-Seen, Composite-Seen, and Composite-Unseen groups. These names refer to task coverage in Human300 pretraining data: a policy fine-tuned on all 50 target tasks has received demonstrations for the Composite-Unseen tasks as well. The selected training subset must therefore accompany any generalization result. The integration provides a simulator client and evaluation entry points. The short single-task walkthrough serves as a pipeline check, while complete benchmark runs use the full task list and its associated horizons.

RoboDojo (Chen et al., 2026c). The training recipes consume the official LeRobot v2.1 export directly, containing 3,500 episodes across 35 training tasks. The ARX X5 configuration selects head and two wrist views, robot state, and absolute joint actions. Reference QwenOFT, QwenGR00T, and QwenPI_v3 recipes and checkpoints share this observation/action convention. Their documented evaluation covers 42 tasks, including held-out and open tasks, with 50 episodes per task. Results retain both success rate and manipulation score (MS), with separate summaries for generalization, precision, long-horizon, memory, and open tasks. StarVLA supplies the data registration, training recipes, and evaluation launcher. XPolicyLab provides the external policy adapter and evaluation runtime, using RoboDojo's tasks, simulator, and scoring.

BEHAVIOR (Li et al., 2024). The integration targets the 50 full-length household tasks in the 2025 BEHAVIOR Challenge, using the released challenge demonstrations and task descriptions. Its training configuration registers the challenge data with the R1Pro embodiment, covering mobile, torso, bimanual, and gripper control. Evaluation delegates task execution and metric collection to the challenge's OmniGibson evaluator. The client selects task instances explicitly and records per-instance, per-episode metrics, including task success. Training-instance debugging and evaluation on test instances are separate settings. Observation wrappers also retain the challenge's distinction between standard observations and privileged information. Low-resolution images with red, green, and blue (RGB) channels are one supported standard-track option. Separate policy and simulator processes, together with task-wise scheduling, provide the integration path for these long-horizon tasks.

MetaWorld (Yu et al., 2020). StarVLA registers a LeRobot-format MT50 dataset mixture spanning 50 manipulation tasks, with a data configuration for a fixed camera and four-dimensional end-effector/gripper actions. The integration links to reference QwenPI_v3 checkpoints published by MINT-SJTU, including direct fine-tuning and language-action-pretrained initialization. The documented evaluation runs 10 episodes per task with a 400-step limit. It reports success rates for the easy, medium, hard, and very-hard task groups. Its overall score is the mean of these four group scores. This aggregation gives equal weight to difficulty groups, which contain different numbers of tasks. The simulator client supplies the image processing and control conventions used by these checkpoints and writes rollout videos and structured evaluation summaries.

CALVIN (Mees et al., 2022). The training workflow converts CALVIN demonstrations to LeRobot format and registers them as a configurable dataset mixture. The documented example uses the D-to-D setting. Evaluation retains the original CALVIN validation data and task oracle. The supplied evaluation file contains 1,000 sequences of five language-conditioned tasks. The client attempts each sequence in order, allowing up to 360 steps per task and ending a sequence when a task fails. Evaluation reports the fraction of sequences completing at least one through five consecutive tasks, together with the average number completed. StarVLA connects its policy server to this sequential evaluator, keeping the training-format conversion separate from the validation states and sequence protocol.

LIBERO-plus (Fei et al., 2026). The integration evaluates policies trained on LIBERO in a zero-shot setting, without additional training on LIBERO-plus. The extended evaluation covers 10,030 task variants under changed conditions. Its client reuses the policy-serving interface and the checkpoint's action statistics, while the benchmark supplies the task variants and initial states. Evaluation records success by perturbation category as well as the evaluated task suite. Parallel launchers partition the task list across workers. LIBERO-plus therefore contributes an additional benchmark integration to the coverage count, with the original LIBERO training recipe and checkpoint retained as part of the result's provenance.

DOMINO (Fang et al., 2026). StarVLA registers the clean and randomized dynamic demonstration splits and supports dataset mixtures that additionally include static demonstrations. The documented reference setting trains one policy on all 35 dynamic tasks using clean data with dynamic coefficient $\alpha = 0.1$. Evaluation selects the corresponding clean or randomized dynamic environment and retains DOMINO's native success checks and metrics. Success rate measures task completion, while manipulation score combines progress toward the target with penalties for events such as collisions and out-of-bounds motion. The integration provides training recipes, task launchers, and an environment client with configurable history inputs. This client can send previous RGB frames or an optical-flow representation through the serving interface when the selected model consumes temporal context.

VLA-Arena (Zhang et al., 2026a). Training uses the benchmark's LeRobot-format L0 demonstrations. The preparation script selects the Large release by default. The Small and Medium releases are nested subsets that can be selected for different data budgets. Evaluation spans 11 suites across safety, distractors, extrapolation, and long-horizon tasks, with difficulty levels L0–L2. The provided launcher defaults to 10 episodes per task and records the suite and difficulty level for each result. Success rate measures task completion, and safety suites additionally report constraint cost. StarVLA provides data preparation and training configurations, a client for the benchmark's environment, and parallel suite execution against policy servers. The selected data release, task set, and difficulty levels jointly define the evaluation setting.

VLN-CE (Krantz et al., 2020). The navigation workflow uses R2R and RxR training data distributed in the NaVILA dataset. An annotation converter represents image sequences, instructions, and target responses in the Qwen-VL conversation format, which is consumed by a dedicated training entry point. The integration connects a Qwen-VL WebSocket server to an external Habitat-based evaluator. Its supplied R2R and RxR configurations evaluate the unseen validation split, with the RxR configuration selecting English instructions. The evaluator aggregates navigation success, success weighted by path length (SPL), normalized dynamic time warping (nDTW), goal distance, and path length. This integration keeps navigation data preparation and environment execution explicit while exposing model inference through a separate server process.

C.4 SELECTED BENCHMARK RESULTS

Figure 3 presents ten selected benchmark settings, each with one STARVLA configuration and three representative reference methods. For each stated scope, the figure selects the highest reported aggregate among the collected STARVLA configurations. Table 8 identifies these variants. Training data, initialization, and budgets can differ. These selected references provide context for the available recipes rather than a complete leaderboard. Table 9 retains additional variants and evaluation scopes.

Table 8: Selected STARVLA configurations in Figure 3. SR is success rate in percent. Generalist denotes multi-benchmark training. MLP denotes a multilayer perceptron, and LA denotes language-action pretraining. Recorded training budgets are included where available.

Benchmark	Selected configuration	SR (%)
LIBERO	Generalist, Qwen3-VL-4B / MLP, batch 1024	98.8
SimplerEnv	Generalist, Qwen3-VL-8B / MLP	71.5
RoboTwin 2.0	Generalist, Qwen3-VL-4B / MLP, batch 1024	89.2
RoboCasa365	Qwen3-VL-PI, seen-34, 150k steps	40.64
MetaWorld	QwenPI_v3, Qwen2.5-VL-3B, LA initialization, 60k steps	70.9
RoboDojo	QwenOFT, VLAct initialization, 100k steps	8.17
LIBERO-plus	Qwen3-VL-PI	77.0
DOMINO	Qwen3-VL-OFT, static and dynamic data	12.89
VLA-Arena	Qwen3-VL-FAST, 8B, derived 33-entry mean	38.24
RoboCasa-GR1	Generalist, Qwen3-VL-4B / MLP, batch 1024	59.2

LIBERO uses the four-suite average, SimplerEnv uses WidowX visual matching, and RoboTwin 2.0 uses the randomized setting. RoboCasa365 uses target kitchens: the selected STARVLA policy is trained on 34 seen tasks and evaluated on 50 tasks, while the reference policies use the full target training set. GR00T N1.5 aggregates three task-group policies with 30 trials per task. STARVLA uses one policy with 50 trials per task. MetaWorld uses an equal-weight mean of four difficulty groups. For RoboDojo, the STARVLA score is a task mean over 1,260 rollouts on 42 tasks, while the reference values average five capability groups. LIBERO-plus retains the reported Total, and DOMINO uses Level 1 at $\alpha = 0.1$. VLA-Arena averages the reported success rates over all 11 suites and three levels with equal weight, $\overline{\mathrm{SR}} = \frac{1}{33}\sum_{s=1}^{11}\sum_{\ell=0}^{2}\mathrm{SR}_{s,\ell}$, for each complete policy configuration. This derived mean excludes the separate safety constraint metric. RoboCasa-GR1 uses the 24-task GR00T tabletop setting, with GR00T N1.5 and N1.6 shown as separate reference versions.

The LIBERO checkpoint with a reported four-suite score of 98.0% remains listed separately because its rollout details are unspecified. CALVIN retains its reported comparison without inferring the evaluation split from example scripts. MetaWorld uses the stated difficulty-group aggregation. CALVIN, BEHAVIOR 2025, and VLN-CE are not shown in the selected overview. Neither missing scores nor uncertainty estimates are inferred.

C.5 Additional Benchmark Results

The additional configurations below retain separate evaluation scopes and metrics. Record-specific training and evaluation details are provided in Appendix C.6.

Table 9: Additional reported benchmark results. Training configurations may differ, and evaluation scopes and metrics are kept separate. SR denotes success rate, MS denotes manipulation score, and LA denotes language-action pretraining. StarVLA training budgets default to 100k steps unless otherwise stated.

Model	Training / checkpoint	Metric	Value
LIBERO: four suites. Spatial, Object, Goal and Long, 40 tasks, 50 episodes/task. Different training budgets are retained.			
OpenVLA-OFT	175k steps	SR (%)	97.1
π_0	Not specified	SR (%)	94.1
GR00T-N1.5	20k steps	SR (%)	86.5
StarVLA-FAST / Qwen3-VL-4B	30k steps	SR (%)	95.4
StarVLA-OFT / Qwen3-VL-4B	30k steps	SR (%)	96.6
StarVLA-PI / Qwen3-VL-4B	30k steps	SR (%)	95.7
StarVLA-GR00T / Qwen3-VL-4B	30k steps	SR (%)	96.5
StarVLA-OFT / Cosmos-Predict2-2B	30k steps	SR (%)	95.8
StarVLA-PI / Cosmos-Predict2-2B	30k steps	SR (%)	95.5
StarVLA-GR00T / Cosmos-Predict2-2B	30k steps	SR (%)	95.2
StarVLA-FAST (Qwen2.5-VL)	30k steps	SR (%)	95.2
StarVLA-OFT (Qwen2.5-VL)	30k steps	SR (%)	96.1
StarVLA-PI (Qwen2.5-VL)	30k steps	SR (%)	95.4
StarVLA-GR00T (Qwen2.5-VL)	30k steps	SR (%)	95.4
LIBERO: additional released configurations. Four-suite checkpoint results with different training budgets. Rollout counts are unspecified.			
StarVLA-OFT / Qwen3-VL	50k steps	SR (%)	98
StarVLA-OFT / Wan2.2	60k steps	SR (%)	91.6
StarVLA-PI / Qwen3-VL (earlier)	100k steps	SR (%)	97.5
LIBERO: three-suite configuration. Three reported suites only. This value is not a four-suite average.			
StarVLA-GR00T / Cosmos-Predict2	50k steps	SR (%)	94.7
SimplerEnv: WidowX visual matching. Four tasks, five repeats of the official evaluation. GR00T N1.5 and the StarVLA GR00T-head implementation are listed separately.			
π_0	Not specified	SR (%)	27.1
GR00T N1.5	Not specified	SR (%)	61.9
StarVLA-FAST / Qwen3-VL-4B	15k steps	SR (%)	31.6
StarVLA-OFT / Qwen3-VL-4B	65k steps	SR (%)	64.6
StarVLA-PI / Qwen3-VL-4B	40k steps	SR (%)	60.9
StarVLA-GR00T / Qwen3-VL-4B	20k steps	SR (%)	65.3
StarVLA-OFT / Cosmos-Predict2-2B	30k steps	SR (%)	61.2
StarVLA-PI / Cosmos-Predict2-2B	30k steps	SR (%)	58.7
StarVLA-GR00T / Cosmos-Predict2-2B	30k steps	SR (%)	61.6
SimplerEnv: Google variant aggregation. Four-task Google variant-aggregation scope, not interchangeable with WidowX visual matching.			
GR00T N1.5	Not specified	SR (%)	44.5
StarVLA-OFT / Qwen3-VL-4B	100k steps	SR (%)	70.2
RoboTwin 2.0: clean. 50 tasks, 100 episodes/task. StarVLA recipe uses 50 clean plus 500 randomized demonstrations/task.			
$\pi_{0.5}$	Not specified	SR (%)	82.74
StarVLA-FAST / Qwen3-VL-4B	100k steps	SR (%)	72.5
StarVLA-OFT	150k steps	SR (%)	88.18
StarVLA-GR00T / Qwen3-VL-4B	100k steps	SR (%)	88

Continued on next page.

Table 9 continued.

Model	Training / checkpoint	Metric	Value
StarVLA-PI / Qwen3-VL-4B	100k steps	SR (%)	88.1
RoboTwin 2.0: randomized. 50 tasks, 100 episodes/task. Clean and randomized columns describe separate evaluations.			
$\pi_{0.5}$	Not specified	SR (%)	76.76
StarVLA-FAST / Qwen3-VL-4B	100k steps	SR (%)	83.2
StarVLA-OFT	150k steps	SR (%)	88.32
StarVLA-GR00T / Qwen3-VL-4B	100k steps	SR (%)	88.5
StarVLA-PI / Qwen3-VL-4B	100k steps	SR (%)	88.8
RoboCasa-GR1. GR00T tabletop benchmark with 24 tasks, 50 episodes/task. StarVLA training budget up to 100k steps.			
$\pi_{0.5}$	Not specified	SR (%)	37
GR00T-N1.6	Not specified	SR (%)	47.6
StarVLA-FAST	100k steps	SR (%)	39
StarVLA-PI	100k steps	SR (%)	43.9
StarVLA-GR00T	100k steps	SR (%)	47.8
StarVLA-OFT	100k steps	SR (%)	48.8
RoboCasa365: target kitchens. 50 tasks, 50 episodes/task. Training task collections differ, and all rows here use target-kitchen evaluation.			
StarVLA-OFT	220k steps	SR (%)	33.32
StarVLA-OFT (no state)	140k steps	SR (%)	30.92
StarVLA-PI	150k steps	SR (%)	40.64
StarVLA-GR00T	120k steps	SR (%)	39.24
RoboCasa365: pretraining kitchens. Official Human300 setting. This evaluation scope differs from the target-kitchen rows above.			
$\pi_{0.5}$	Human300 training	SR (%)	16.9
GR00T N1.5	Human300 training	SR (%)	23.9
GR00T N1.6	Human300 training	SR (%)	21.9
RoboDojo. 42 evaluation tasks, 50 episodes/task, 35 training tasks. Success rate (SR) and manipulation score (MS) are distinct.			
StarVLA QwenOFT	100k steps	SR (%)	4.86
StarVLA QwenOFT	100k steps	MS	8.01
StarVLA QwenGR00T	130k steps	SR (%)	3.81
StarVLA QwenGR00T	130k steps	MS	7.35
StarVLA QwenPI_v3	100k steps	SR (%)	6.19
StarVLA QwenPI_v3	100k steps	MS	9.6
MetaWorld: released configurations. 50 tasks, 10 episodes/task, seed 4042, horizon 400. Macro SR gives equal weight to four difficulty groups, not 50 tasks.			
StarVLA QwenPIv3 LA-pretrained	LA initialization, 60k	Macro SR (%)	70.9
StarVLA QwenPIv3 baseline	Baseline initialization, 60k	Macro SR (%)	59.1
MetaWorld: additional reported configurations. Reported aggregate over four difficulty groups. These values are separate reported configurations, not replacements for the released-checkpoint rows.			
StarVLA + LA	LA initialization	SR (%)	69.91
StarVLA without LA	Baseline initialization	SR (%)	58.39
CALVIN: reported five-task sequences. Exact evaluation split and actual sequence manifest are unspecified. Asterisk denotes a retrained baseline. Average length ranges from 0 to 5.			
PI0*	Not specified	Avg. length	2.954
PI0.5*	Not specified	Avg. length	3.885
StarVLA qwenpi (qwen2.5-vl-3B-instruct-action)	100k steps	Avg. length	3.576
StarVLA qwenpi (qwen3-vl-4B-instruct)	100k steps	Avg. length	3.472
StarVLA qwengr00t (qwen2.5-vl-3B-instruct)	100k steps	Avg. length	3.697

Continued on next page.

Table 9 continued.

Model	Training / checkpoint	Metric	Value
StarVLA qwengr00t (qwen2.5-vl-3B-instruct-action)	100k steps	Avg. length	3.786
StarVLA qwengr00t (qwen3-vl-4B-instruct)	100k steps	Avg. length	3.645
StarVLA qwengr00t (qwen3-vl-4B-instruct-action)	100k steps	Avg. length	3.757
LIBERO-plus. LIBERO-trained policies evaluated on seven perturbation categories. Total retains the native task weighting.			
OpenVLA-OFT	Not specified	SR (%)	69.6
π_0	Not specified	SR (%)	53.6
π_0-Fast	Not specified	SR (%)	61.6
StarVLA Qwen2.5-VL-FAST	100k steps	SR (%)	48.9
StarVLA Qwen2.5-VL-GR00T	100k steps	SR (%)	66.4
StarVLA Qwen2.5-VL-OFT	100k steps	SR (%)	67.2
StarVLA Qwen3-VL-OFT	100k steps	SR (%)	75
StarVLA Qwen3-VL-PI	100k steps	SR (%)	77
DOMINO: Level 1, $\alpha = 0.1$**.** 35 dynamic tasks, clean Aloha-AgileX, 100 episodes/task. StarVLA-GR00T uses a Qwen backbone.			
π_0 / PaliGemma	Not specified	SR (%)	8.17
π_0 / PaliGemma	Not specified	MS	23.96
$\pi_{0.5}$ / PaliGemma	Not specified	SR (%)	9.63
$\pi_{0.5}$ / PaliGemma	Not specified	MS	26.17
StarVLA-GR00T / Qwen3-VL	100k steps	SR (%)	6.1
StarVLA-GR00T / Qwen3-VL	100k steps	MS	28.6
StarVLA-Adapter	100k steps	SR (%)	4.4
StarVLA-Adapter	100k steps	MS	24.31
StarVLA-FAST / Qwen3-VL	100k steps	SR (%)	5.74
StarVLA-FAST / Qwen3-VL	100k steps	MS	20.66
StarVLA-OFT / Qwen3-VL	100k steps	SR (%)	10.86
StarVLA-OFT / Qwen3-VL	100k steps	MS	30.49
VLA-Arena: LongHorizon suite. One named suite only. L0/L1/L2 are separate difficulty levels. No benchmark-wide aggregate is implied.			
StarVLA Qwen3-VL-OFT	L0	SR (%)	98
StarVLA Qwen3-VL-OFT	L1	SR (%)	0
StarVLA Qwen3-VL-OFT	L2	SR (%)	0

C.6 Record-Specific Evaluation Settings

Table 10 separates previously reported baselines from released checkpoint reports. They retain their original training and evaluation settings. Neither group reports new runs for this manuscript.

Reported baselines. T1–T6 summarize previously reported baseline configurations. OFT denotes the parallel action-regression variant. The benchmark-specific settings are as follows.

T1–T2: LIBERO. Qwen3-VL-4B and Cosmos-Predict2-2B use OFT, both at 30k optimization steps. The respective four-suite success rates are 97.8 / 98.6 / 96.2 / 93.8 and 98.6 / 97.6 / 95.0 / 91.8, yielding the reported means of 96.6% and 95.8%. Each suite contains 10 tasks evaluated with 50 episodes per task, or 500 rollouts per suite.

T3–T4: SimplerEnv. The WidowX visual-matching setting uses Qwen3-VL-4B with OFT at 65k steps and Cosmos-Predict2-2B with OFT at 30k steps. Their four-task rates are 90.3 / 38.5 / 29.7 / 100.0 and 66.8 / 62.6 / 25.3 / 90.2, with reported means of 64.6% and 61.2%. The reported protocol uses five repetitions of the full official evaluation. Per-repeat results and uncertainty estimates are not supplied here.

Table 10: Representative source-reported benchmark results. Previously reported baselines and released checkpoints retain their separate training and evaluation settings. Values are success rates (%) except for the separately labeled RoboDojo manipulation score. OFT denotes the parallel action-regression variant. Evaluation settings and protocol details appear in Appendix C.6.

Benchmark	Policy	Evaluation scope	Result	Record
Reported baselines				
LIBERO	Qwen3-VL-4B + OFT	Four suites	96.6	T1
	Cosmos-Predict2-2B + OFT	Four suites	95.8	T2
SimplerEnv	Qwen3-VL-4B + OFT	WidowX, visual matching	64.6	T3
	Cosmos-Predict2-2B + OFT	WidowX, visual matching	61.2	T4
RoboTwin 2.0	StarVLA-OFT	Clean / randomized	88.2 / 88.3	T5
RoboCasa-GR1	StarVLA-OFT	24 tasks	48.8	T6
Released checkpoint reports				
LIBERO	Qwen3-VL-OFT	Four suites	98.0	R1
	Cosmos-Predict2–GR00T	Three completed suites	94.7	R2
	Wan2.2–OFT	Four suites	91.6	R3
RoboDojo	Qwen3-VL-PIv3	42 tasks, success	6.19	R4
		Manipulation score	9.60	

T5: RoboTwin 2.0. StarVLA-OFT with Qwen3-VL-4B is trained using the clean and randomized demonstration sources. Success rates are 88.2% in clean and 88.3% in randomized conditions, with 50 tasks and 100 episodes per task in each condition. The stated training budget is 150k steps. The selected checkpoint step is unspecified.

T6: RoboCasa-GR1. StarVLA-OFT has a reported success rate of 48.8% over 24 tasks with 50 rollouts per task. The 24 reported task rates average to 48.75%, rounded to one decimal place. The training budget is 100k steps. The backbone revision and selected checkpoint step are unspecified.

Exact training revisions, complete run configurations, and matching released weights remain to be established for these baseline records.

Released checkpoint reports. R1–R4 summarize reported results for released checkpoints under distinct protocols, rather than a controlled comparison among the frameworks in this manuscript.

R1: Qwen3-VL-OFT. The 50k-step checkpoint has a reported success rate of 98.0% across four LIBERO suites, with an eight-step action horizon.

R2: Cosmos-Predict2–GR00T. The 50k-step checkpoint has a reported success rate of 94.7% over three completed suites. The long-horizon suite is not included in that mean.

R3: Wan2.2–OFT. The 60k-step checkpoint has a reported success rate of 91.6% across four suites, using one third-person camera at 224×224 and an eight-step action chunk.

R4: Qwen3-VL-PIv3 on RoboDojo. The 100k-step checkpoint has reported results of 6.19% success and a 9.60 manipulation score. Its setting has 35 training tasks and 42 evaluation tasks with 50 rollouts each. These are distinct metrics, with 2,100 evaluation rollouts in total.

D Policy Serving and Robot Deployment

D.1 Policy Serving Interfaces

Loading and metadata. `PolicyServerWrapper` loads the framework, configuration, and dataset statistics associated with a checkpoint. Handshake metadata includes the action chunk size, available normalization keys, and available state/action field information. Selected training image-size metadata can help clients check their inputs. Camera selection and ordering are configured on the client side. The service does not infer their correspondence to a new environment.

Prediction and action processing. Clients submit an `examples` list and, where needed, an `unnorm_key`. The wrapper calls the framework's prediction entry point, receives `normalized_actions`, and uses `PolicyNormProcessor` to apply the inverse training-time transform. The WebSocket response carries the result under `data["actions"]`. Environment adapters can then apply controller-specific rotation, gripper, or action-chunk scheduling logic. Normalization statistics and action field order must match the selected checkpoint and client path.

Additional protocol integration. The release provides a GR00T-compatible server using ZeroMQ (ZMQ) that reuses the same wrapper. Its observation adapter assembles named state groups in the configured order and splits predicted actions into named groups. The documented compatibility is with the N1.6 custom codec. State/video histories use their latest entry. This is a protocol-adaptation implementation, with no new interoperability or hardware-control measurement reported in this manuscript.

D.2 Robot Workflows

Franka and Realman have selected robot-workflow guides, while the Unitree G1 whole-body example is documented as a scaffold.

Representative embodiment paths include the single-arm Franka guide, the bimanual ARX X5 data configuration, and the RoboCasa-GR1 tabletop integration. The humanoid example concerns manipulation. It does not establish whole-body locomotion or control. These are representative settings, not a census of physical robots supported by every model.

E Experiment Configuration and Training Efficiency

E.1 Experiment Configuration Details

Combining experiment parameters. StarVLA combines built-in parameter defaults, YAML settings, and command-line overrides. Its `framework`, `datasets`, and `trainer` groups organize experiment parameters for model construction, data processing, and training. Components read experiment parameters from the shared experiment configuration. YAML settings override built-in defaults, and command-line values take precedence over matching YAML fields. Researchers can reuse experiment files while changing selected experiment parameters for a run.

Extending experiment configurations. StarVLA allows method-specific fields to be supplied through YAML or command-line overrides without declaring their names and types in a fixed global schema. For example, a new action-head option or objective weight can be added to its corresponding group. The relevant component reads the field and can provide defaults and checks. Merging retains additional fields so that existing experiment configurations can coexist with new options. Figure 7 follows a new setting from the experiment recipe through the component that uses it and the shared training tools.

Configuration snapshots. StarVLA saves the experiment parameters accessed during a run, covering model, data, and training choices. This configuration snapshot supports reusing experiment parameters with selected overrides and reconstructing the model when loading a checkpoint.

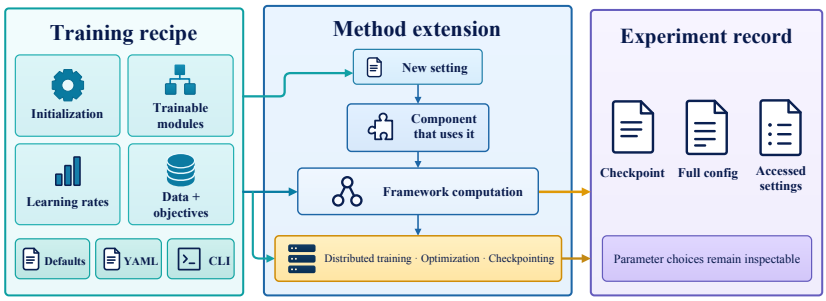


Figure 7: Editable recipes and method extension. Initialization, trainable modules, learning rates, and data objectives are selected through experiment settings. New settings are consumed by relevant components and enter framework computation while reusing shared training tools. Checkpoints and configuration records retain the experiment context; the accessed-settings snapshot records fields read during execution.

E.2 Shared Training Execution

Training loops use PyTorch with Accelerate and DeepSpeed for distributed execution. Frameworks compute method-specific losses, while training entry points organize data streams, backpropagation, and updates. Recipes select trainable modules and their learning rates. The backbone can remain frozen or be updated alongside the action head at a separate learning rate. Shared execution tools support bfloat16 mixed precision, gradient accumulation, and gradient clipping. Optimizers and learning-rate schedulers operate on selected parameter groups.

E.3 Training Measurements

The A100 sweeps in Figures 4 and 8 use StarVLA-GR00T with Qwen3-VL-4B on RoboCasa-GR1, running on A100 80GB GPUs. Figure 4(a) also includes reported H200 timings: StarVLA at batch sizes 8, 16, 32, and 64, and LingBot-VLA and Dexbotic at batch 32, all using eight GPUs. These comparison values retain the two-decimal precision of the source figure. Complete profiling settings for the comparison methods are not specified. At eight A100 GPUs and batch eight per GPU, the two sweep series report 1.131 and 0.735 seconds/step respectively. Their different baselines remain separate. Global batch grows from 16 to 192 in the batch sweep and from 64 to 2,048 in the GPU-scaling series. The measurements describe sample throughput and step latency. Time to a target policy quality was not measured. The training commit, complete runtime configuration, and profiling logs were not established by the available report.

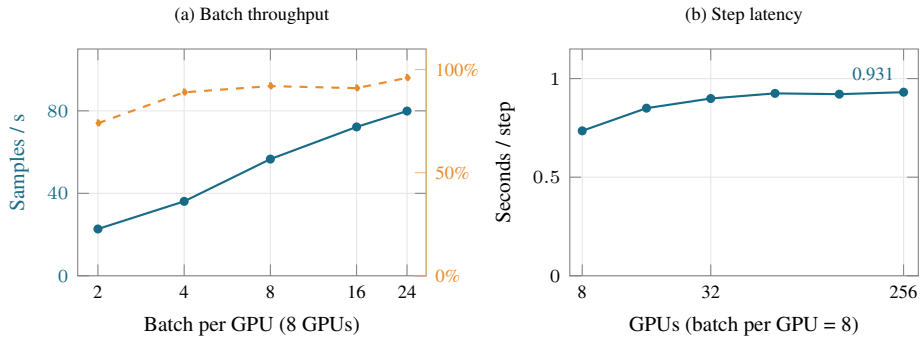


Figure 8: Additional efficiency measurements on A100 80GB GPUs. (a) Eight-GPU throughput (blue, left axis) and GPU utilization (orange, right axis) across per-GPU batch sizes. (b) Step latency on 8–256 GPUs at batch eight per GPU.

The additional sweeps in Figure 8 show how batch size affects throughput and how GPU count affects step latency, complementing the comparisons and throughput scaling in Figure 4. On eight A100 GPUs, increasing batch per GPU from 2 to 24 raises throughput from 22.7 to 79.9 samples/s. GPU utilization ranges from 74% to 96%. In the separate GPU-scaling series, batch per GPU stays at eight as GPU count increases from 8 to 256. Step latency changes from 0.735 to 0.931 seconds.

F Research Uses across Model Development, Training, and Evaluation

Table 11 describes how six research works use StarVLA across model development, training, and evaluation. A single work can extend a policy implementation, adapt its training pipeline, and use benchmark tools or reference baselines for evaluation. The examples are selected uses, not a count of community variants or independent adopting teams.

Table 11: Selected research uses of StarVLA. The cited works and their released implementations illustrate connections across model development, training, and evaluation. Each cell describes the specific use.

Work	Documented connection to StarVLA		
	Model	Training	Evaluation
LangForce (Lian et al., 2026c)	Implements latent action queries and paired vision-only and language-conditioned policy branches within StarVLA.	Uses the training entry and dataset configurations, adding objectives for instruction following.	Uses integrated workflows for LIBERO, SimplerEnv, and RoboCasa evaluation.
LaWAM (Chen et al., 2026b)	Extends Qwen-GR00T to condition action generation on predicted latent visual subgoals.	Adapts StarVLA dataloaders, training loop, and configurations for policy training.	Uses StarVLA-based policy serving and LIBERO / RoboTwin evaluation entry points.
EventVLA (Yang et al., 2026a)	Extends QwenOFT with event-driven keyframe memory for simulation policies.	Adapts the data and training pipeline to learn actions together with keyframe prediction.	Adapts policy serving and environment clients for RoboTwin-MeM and RM-Bench.
W^2-VLA (Pan et al., 2026)	Augments a StarVLA policy with task-conditioned future-wrist prediction and a flow-matching action head.	Adds action, chain-of-thought, and future-wrist prediction objectives to the StarVLA-based policy.	Uses StarVLA OFT / GR00T results as reference baselines on LIBERO and RoboTwin 2.0.
VLAct (Yang et al., 2026c)	Builds shared representations with StarVLA OFT, PI, and GR00T action heads.	Uses dataset mixtures and policy / VLM co-training for continued pretraining and downstream adaptation.	Uses benchmark configurations and evaluation launchers for LIBERO, VLA-Arena, RoboTwin, and DOMINO.
PhysBrain (Lin et al., 2025)	Adapts a backbone pretrained on human egocentric data with the GR00T-style action head used by Qwen-GR00T.	Follows StarVLA training configurations for VLM-to-policy adaptation on robot data.	Uses reported Qwen-GR00T / OFT / FAST results as reference baselines on RoboCasa-GR1.

The evaluation connections include execution tools and reference baselines, as specified in each row. PhysBrain's training connection concerns robot adaptation following human-data pretraining. These examples illustrate research uses. Implementation-effort savings have not been measured.

G REFERENCES FOR THE RESEARCH ECOSYSTEM TIMELINE

This section provides the paper-reference index for Figure 2, complementing the representative extensions discussed in Section 3.3. Research entries follow the figure's three time periods; associated papers for external frameworks are collected separately below.

G.1 RESEARCH WORKS

Table 12: Paper references for the research works in Figure 2.

Work	Reference	Work	Reference
October 2025–March 2026			
NoTVLA	Huang et al. (2025)	PhysBrain	Lin et al. (2025)
CLAP	Zhang et al. (2026b)	TwinBrainVLA	Yu et al. (2026b)
LangForce	Lian et al. (2026c)	LaRA-VLA	Bai et al. (2026a)
VLA-TMEE	Bai et al. (2026b)	VLA-JEPA	Sun et al. (2026b)
ABot-M0	Yang et al. (2026d)	LDA-1B	Lyu et al. (2026)
TempoFit	Sun et al. (2026c)	DiT4DiT	Ma et al. (2026a)
VP-VLA	Wang et al. (2026g)	LaMP	Wang et al. (2026e)
April–June 2026			
DriveDreamer-Policy	Zhou et al. (2026b)	AssemLM	Jing et al. (2026b)
JoyAI-RA 0.1	Zhang et al. (2026d)	CorridorVLA	Li et al. (2026a)
ResVLA	Zhong et al. (2026)	LoHo-Manip	Liu et al. (2026a)
DiscreteRTC	Wang et al. (2026c)	Keystone	Dai et al. (2026)
LoopVLA	Shen et al. (2026)	Multi-view VLA	Xiao et al. (2026)
FrameSkip	Yu et al. (2026a)	IntentVLA	Lian et al. (2026a)
PhysBrain 1.0	Lian et al. (2026b)	Afford-VLA	Wang et al. (2026d)
Lngram	Zheng et al. (2026)	Rethink VLA Init.	Lin et al. (2026b)
FineVLA	Hu et al. (2026)	OneVLA	Zhang et al. (2026c)
PHASER	Chen et al. (2026d)	SPARC	Blank et al. (2026)
Beyond English	Chen et al. (2026a)	LaWAM	Chen et al. (2026b)
EventVLA	Yang et al. (2026a)	Action Prior Learning	Jing et al. (2026a)
LA4VLA	Lin et al. (2026a)		
July–September 2026			
WALA	Liu et al. (2026b)	Anchor-Align	Dalal et al. (2026)
FoMoVLA	Li et al. (2026d)	JoyNexus	Sun et al. (2026a)
HiFi-UMI	Simple AI et al. (2026)	TurboVLA	Xie et al. (2026)
W^2-VLA	Pan et al. (2026)	SLIM	Wang et al. (2026a)
StellaVLA	Xu et al. (2026)	DeepInsight II	Li et al. (2026c)
FabriMAE	Aniri et al. (2026)	Latent Action Study	Bu et al. (2026)
GlanceWAM	Wang et al. (2026b)	VLAct	Yang et al. (2026c)
Temporal Forcing	Ding et al. (2026)	TacPAC	Ma et al. (2026b)
LayerRoute	Lu et al. (2026)	EffVLA	Sun et al. (2026d)
SCULPT-VLA	Li et al. (2026e)	ActiveArena	Li et al. (2026g)
THAW-VLA	Dao et al. (2026)	CoRe-WAM	Zhou et al. (2026a)

G.2 EXTERNAL FRAMEWORKS

The external frameworks with associated research papers are RLinf (Yu et al., 2026c), VLAFlow (Xia et al., 2026), EVA-Client (Yang et al., 2026b), and XPolicyLab (XPolicyLab Community et al., 2026).