KUAVO-VLA-1.0: A Domain-Specialized Vision-Language-Action Foundation Model for Industrial Dual-Arm Humanoid Manipulation

Lejoin Intelligence

Vision-Language-Action (VLA) foundation models have demonstrated impressive open-domain manipulation, yet industrial deployment demands something different: sub-centimeter tolerances, long-horizon multi-stage assembly, heterogeneous tooling, and high, repeatable success rates under fixed line-side conditions. We argue these demands call for a level of specialization between a general-purpose foundation model and a single-task fine-tune, which we term a *domain-specialized foundation model* (DSFM). A general-purpose VLA model carries breadth in three things: embodiments, settings, and tasks. A DSFM drops the first two to raise success rate on the third. KUAVO-VLA-1.0 is trained for one embodiment, the Kuavo humanoid, and one setting, the industrial floor, and stays multi-task within them: new work in that setting is adapted from it rather than trained from scratch. We present KUAVO-VLA-1.0, a DSFM for dual-arm humanoid manipulation on the Kuavo platform, built on the LingBot-VLA-2.0 backbone in two stages. *Domain-specific retraining* on a self-collected dataset of 100 industrial tasks and 680 hours of teleoperated demonstrations yields the DSFM; *task-specific post-training* then specializes it to individual tasks. On a 25-task real-robot benchmark, KUAVO-VLA-1.0 raises average success on the 20 line-side industrial tasks from 14.0% to 49.7% against LingBot-VLA-2.0, the strongest baseline, under an identical 20-epoch per-task protocol, and progress score from 40.1% to 74.1%. The gain carries to 5 in-domain GM100 tasks held out of the dataset, where progress score is higher on every task, and success rate rises from 25.3% to 42.7% on a sample too small to settle that second margin. We release the report, project webpage, toolchain, checkpoints, and dataset.

Website
https://model.lejurobot.com/kuavo-vla-1

Code
https://github.com/LejuRobotics/LeTools-Learning

Models
https://openlet.openatom.tech/tools/KUAVO-VLA-1.0
https://www.modelscope.cn/models/lejurobot/LET-KUAVO-VLA-1.0-models
https://huggingface.co/LejuRobotics/LET-KUAVO-VLA-1.0-models

Dataset
https://openlet.openatom.tech/tools/KUAVO-VLA-1.0
https://www.modelscope.cn/datasets/lejurobot/LET-KUAVO-VLA-1.0-Dataset
https://huggingface.co/datasets/LejuRobotics/LET-KUAVO-VLA-1.0-Dataset

1 Introduction

Vision-Language-Action (VLA) foundation models are now the default starting point for general-purpose robot manipulation. Pretrained on web-scale vision-language datasets and ever-larger collections of robot trajectories, recent models demonstrate open-vocabulary instruction following, cross-embodiment transfer, and non-trivial generalization to novel objects and scenes [1, 2, 14]. Humanoid hardware has advanced in step, converging on dual-arm platforms purpose-built for the fine-grained bimanual manipulation these models are beginning to enable [24].

Industrial manipulation, however, is a different regime. Line-side tasks demand sub-centimeter precision when

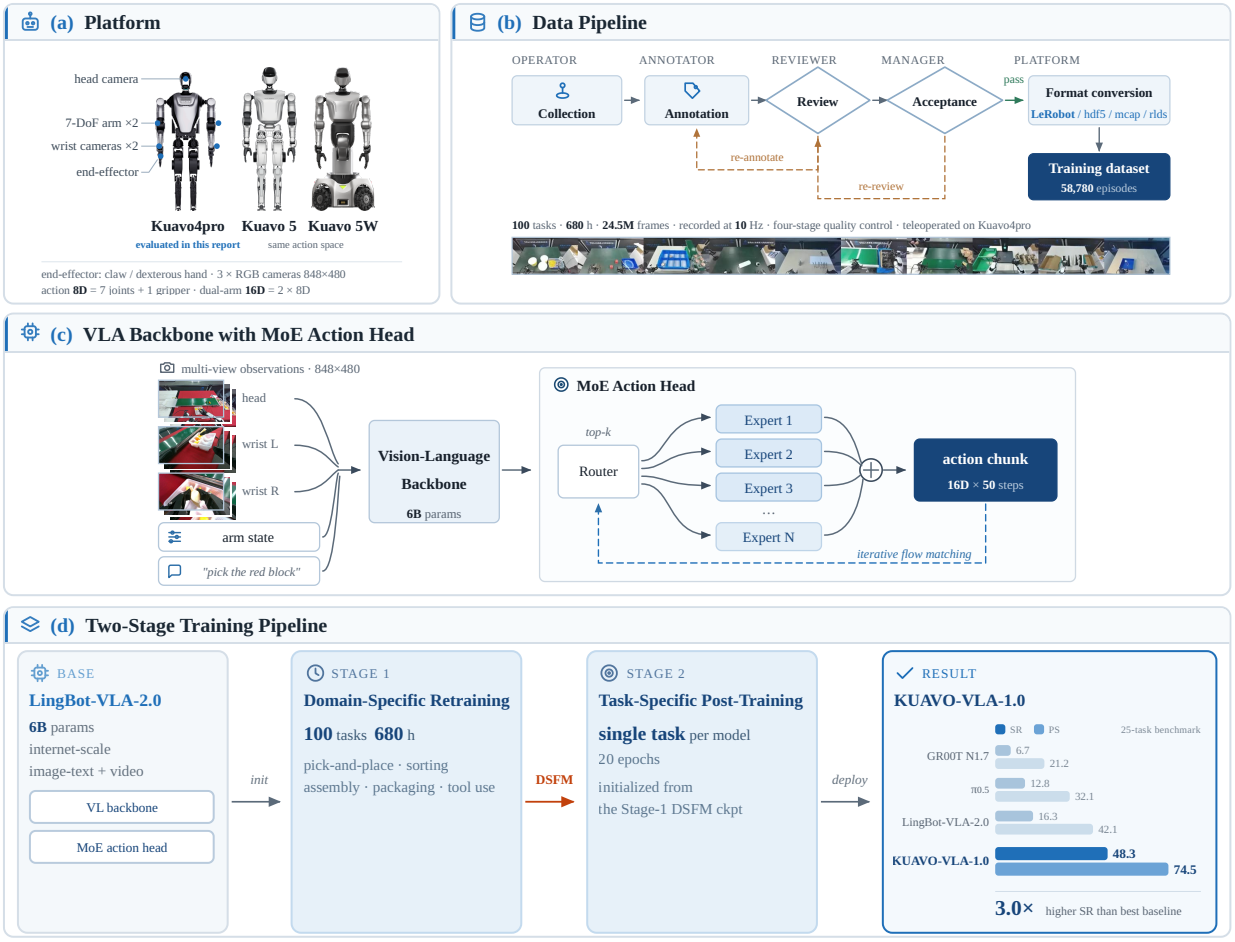


Figure 1. Framework of KUAVO-VLA-1.0. (a) KUAVO-VLA-1.0's configuration space on the Kuavo4pro platform: end-effector (`eef_type`, dexterous hand or parallel gripper) and which arm(s) are active (`which_arm`) are deployment-time configuration choices, with `which_arm` determining whether the action head predicts an 8-dimensional single-arm chunk or a 16-dimensional dual-arm chunk. Kuavo 5 and Kuavo 5W share that action space; all data and all results in this report are from Kuavo4pro. (b) Data collection and processing pipeline: teleoperated demonstrations are recorded as ROS bags and pass a four-stage quality-control chain — collection, annotation, review, and acceptance — with rejected data returned to the preceding stage; accepted episodes are then converted into standardized episodic trajectories (LeRobot format) that enter the training dataset. (c) Model architecture: a vision-language understanding backbone encodes multi-view images, proprioceptive state, and the language instruction into contextual tokens, which condition a Mixture-of-Experts action head that iteratively refines noised action tokens (flow-matching / diffusion-style) into a denoised action chunk. (d) Two-stage training pipeline: starting from a large-scale pretrained VLA foundation model, Stage 1 (domain-specific retraining) specializes the model to the industrial manipulation domain using our self-collected 100-task, 680-hour dataset, yielding the DSFM; Stage 2 (task-specific post-training) further specializes that DSFM on individual tasks from the 25-task benchmark, toward the production-grade reliability such deployments require.

mating parts, inserting connectors, or operating fixtures. They run long and in sequence, so a single early failure invalidates the entire episode. Tooling and packaging are built for human hands, not robot ones. And the bar is production-line reliability, not research-demo performance. A capable pretrained backbone is necessary but not sufficient here: closing the gap takes a deliberate, data-driven specialization strategy, validated on tasks that actually look like an industrial floor.

Domain-specialized foundation models. We frame the gap as a question of *where* specialization should happen, and argue for a level that VLA research has largely skipped over, though one with clear precedent outside robotics (Sec. 2). At one end sits the general-purpose VLA foundation model, carrying breadth in three things at once: embodiments, settings, and tasks. It is the right object to build on, but its capacity and its data budget are spread over a distribution

of which any single deployment touches only a sliver. At the other end sits the single-task fine-tune, which gives up all three: it buys reliability on one task at the cost of everything else, and must be paid for again, in demonstrations and training time, for every new task on the line. We term the level in between a *domain-specialized foundation model* (DSFM): a model that drops the first two kinds of breadth and keeps the third. KUAVO-VLA-1.0 is trained for one embodiment, the Kuavo humanoid, and one setting, the industrial floor, and stays multi-task within them. The narrowing buys success rate across those tasks, the number a production line is graded on, and not depth on any one of them. Operationally, a DSFM is still a foundation model: new tasks in its domain are adapted from it rather than trained from scratch. What it gives up is the open-domain breadth it will never exercise; what it buys is reliability inside the domain where it is actually deployed.

In this report we build one on the Kuavo humanoid platform, treating industrial deployment as a first-class design constraint rather than an afterthought. KUAVO-VLA-1.0 is produced by a two-stage pipeline on top of a large-scale pretrained VLA foundation model, shown in Fig. 1. The first stage, domain-specific retraining, retrains the backbone at scale on a self-collected dataset of 100 distinct industrial tasks totaling 680 hours of dual-arm teleoperated demonstrations, spanning pick-and-place, sorting, assembly and insertion, packaging, and tool-based operations. It is designed to build broad industrial-domain competence and positive cross-task transfer. This stage produces the DSFM. The second stage, task-specific post-training, specializes that DSFM on individual tasks that demand production-grade reliability, trading a small amount of within-domain generality for a substantial gain in per-task success rate.

We evaluate this recipe on a 25-task benchmark, pairing 20 line-side industrial operations with 5 unseen in-domain tasks from the public GM100 collection. We compare KUAVO-VLA-1.0 against a panel of state-of-the-art open VLA baselines, each specialized to the same task under the same per-task adaptation budget. Because the strongest of these baselines, LingBot-VLA-2.0, is also KUAVO-VLA-1.0's own backbone, the comparison holds architecture, pretraining, target task, and adaptation budget fixed, and varies only whether specialization is routed through a DSFM. On the industrial group KUAVO-VLA-1.0 reaches a 49.7% average success rate against 14.0% for the strongest baseline. On the held-out in-domain group it is ahead on progress score for every task, at 76.2% against 49.7%; success rate there rises from 25.3% to 42.7%, on a sample of five tasks too small to settle that second margin (Sec. 5.4).

Contributions. We summarize our contributions as follows:

- We term the level between a general-purpose foundation model and a single-task fine-tune a domain-specialized foundation model (DSFM), and argue that it, rather than either end, is the unit around which industrial deployment should be organized. The pattern is established for language models in other domains; what is new here is carrying it into VLA, where the breadth given up across embodiments and settings comes back as success rate on the tasks that remain, and testing it on real hardware.
- We build one, KUAVO-VLA-1.0, a DSFM for dual-arm humanoid manipulation on the Kuavo platform, together with the two-stage recipe that produces it: domain-specific retraining on 100 industrial tasks and 680 hours of teleoperated demonstrations, followed by task-specific post-training. On the 20 industrial tasks of a 25-task real-robot benchmark it raises average success from 14.0% to 49.7% over the strongest open VLA baseline under a matched adaptation budget.
- We report how KUAVO-VLA-1.0 is deployed, and the runtime it is served under (Sec. 6). We release the 100-task, 680-hour industrial manipulation dataset that defines KUAVO-VLA-1.0's domain, together with model checkpoints and the LeTools-Learning toolchain, through the OpenLET open-data collection.

The rest of the report is organized as follows. Section 2 relates KUAVO-VLA-1.0 to prior VLA and domain-specialization work. Section 3 describes the dataset that defines its domain, and Sec. 4 the two-stage pipeline trained on that dataset. Section 5 reports the benchmark and its results. Section 6 covers how the model is served. Section 7 states the limits of the evidence, and Sec. 8 concludes.

2 Related Work

VLA foundation models. Building on the success of large vision-language models, a growing line of work trains transformer policies directly on robot trajectory data to produce general-purpose manipulation policies. Early systems such as RT-1 [3] and RT-2 [2] showed that web-scale vision-language pretraining transfers useful priors to closed-loop

robot control. Subsequent open-source efforts, including OpenVLA [12] and Octo [15], made large VLA backbones broadly accessible and spurred rapid community iteration on architecture, tokenization, and action representation, with the Open X-Embodiment collection [16] pooling the cross-embodiment trajectory data that OpenVLA and much subsequent open work train on. More recent systems adopt flow- or diffusion-based action heads for smoother, higher-frequency control, exemplified by π_0 and $\pi_{0.5}$ [1, 18], and scale pretraining data and model capacity substantially, as with Google DeepMind's Gemini Robotics [8] and NVIDIA's GR00T N1 family [14]. At the other end of the spectrum, SmolVLA [21] shows that a much smaller, community-trained VLA can remain competitive, underscoring that model scale is only one axis along which this design space is being explored. The two years since have pushed the line further on both capability and scope. Gemini Robotics 1.5 [7] interleaves action with multi-level reasoning in natural language and transfers motion across heterogeneous embodiments; $\pi_{0.7}$ [17] conditions a single generalist model on multimodal context beyond the language command, so that one model can be steered across strategies and embodiments; GR-3 [4] pairs a large VLA with a bimanual mobile platform and reports few-shot adaptation to new tasks; and EO-1 [19] folds embodied reasoning and control into a single model through interleaved vision-text-action pretraining. In parallel, several groups have released VLA foundation models specifically targeting bimanual and whole-body humanoid control, including LingBot-VLA-2.0 [24]. Every model above is general-purpose by design: many embodiments, many settings, many tasks, and evaluated on exactly that breadth. KUAVO-VLA-1.0 makes the opposite trade. It starts from one of them, LingBot-VLA-2.0, and gives up breadth on two of those three counts, building for one embodiment, the Kuavo humanoid, and one setting, the industrial floor. What it spends the saving on is the third: on the same industrial tasks, under the same per-task adaptation budget, it reaches 49.7% average success against 14.0% for LingBot-VLA-2.0 itself. That trade, and what it buys, is what this report is about.

Action representations and imitation learning. Underlying much of this progress is a body of work on effective action representations for robot imitation learning, including action-chunking transformers [27] and diffusion-based action heads [5], which substantially improved the ability of learned policies to reproduce smooth, multi-modal, temporally consistent behavior from teleoperated demonstrations. Modern VLA models typically inherit one of these action representations, coupling it with a large pretrained vision-language backbone.

Humanoid and dual-arm manipulation. Humanoid hardware has converged rapidly on dual-arm configurations capable of the kind of bimanual coordination required by real manipulation tasks. This has motivated policy-learning work that explicitly models whole-body and bimanual action spaces, rather than treating each arm independently, and that accounts for embodiment-specific constraints such as reachability, self-collision, and coupled base-arm motion [4, 22]. KUAVO-VLA-1.0 is deployed on the Kuavo dual-arm humanoid platform and inherits this line of whole-body-aware policy design.

Post-training and continual adaptation of VLA models. As pretrained VLA backbones have grown in scale, a separate line of work studies how to adapt them *after* pretraining, rather than how to pretrain them in the first place. Systematic studies of VLA fine-tuning recipes [11] and efficiency-oriented surveys [25] document the trade-offs between fine-tuning depth, sample efficiency, and downstream success rate. A related thread targets continual post-training specifically, aiming to add new tasks or domains to a VLA backbone over time without catastrophically forgetting previously acquired skills [26], a property that matters for any foundation model expected to keep absorbing new tasks over its deployment lifetime rather than being trained once and frozen. KUAVO-VLA-1.0's two-stage recipe sits within this line of work: domain-specific retraining is closer in spirit to large-scale continual adaptation across many tasks, while task-specific post-training is closer to conventional few-shot fine-tuning on a single task, and we evaluate both stages under one industrial-domain benchmark. What we add is a distinction this literature largely leaves implicit. Post-training is usually framed as a procedure applied to a model, parameterized by depth, data budget, and forgetting; we argue it also produces a distinct *kind* of artifact when the dataset spans an entire application domain, namely a DSFM, which is then the thing subsequent task-specific fine-tunes start from.

Domain-specialized foundation models outside robotics. The level of specialization we argue for is not new to machine learning. In language modeling it is well established: Gururangan et al. [10] showed that a second phase of pretraining on in-domain text improves downstream performance across four domains and eight classification tasks, in both high- and low-resource settings. The same recipe produced a family of domain-specific models rather than domain-specific fine-tunes: PubMedBERT for biomedicine [9], whose authors likewise argue that domain-specific pretraining belongs at the start of the pipeline rather than as an afterthought; SaulLM-7B for law [6], which continues pretraining a general 7B

model on thirty billion tokens of legal text; BloombergGPT for finance [23], whose corpus mixes a firm's own financial data with general text at comparable scale; and Code Llama for programming [20], the closest structural precedent to our own two-stage pipeline in that a general foundation model is retrained on a domain corpus and then specialized further per task. Across these verticals the pattern is the same: a domain earns its own foundation model when its data is large enough to move the model and distinct enough that general pretraining underrepresents it. Robotics has not had one, and the reason is cost. A robot domain fixes not only a subject matter but an embodiment, an action space, tooling, and physical scenes, so a dataset spanning it must be teleoperated rather than scraped. We built that dataset, and KUAVO-VLA-1.0 is what we trained on it.

Domain specialization for industrial robot learning. Most public VLA evaluation suites (household-scale simulation benchmarks and tabletop pick-and-place tasks) are only loosely representative of industrial line-side manipulation, which demands narrow tolerances, long-horizon multi-stage execution, and near-production success rates. A recent industrial-readiness survey makes this gap explicit, noting that most robotic foundation models are validated far from the reliability bar a real production line requires [13]. Within robotics the closest precedent is DeMaVLA [22], which pretrains on roughly 5,000 hours of real dual-arm demonstrations and then specializes to deformable manipulation; it narrows by task family where we narrow by embodiment and setting. KUAVO-VLA-1.0 targets this gap directly, and does so by building a DSFM for it: we apply domain-specific retraining on a self-collected 100-task, 680-hour industrial dataset, followed by task-specific post-training on individual tasks that require high reliability.

3 The Industrial Manipulation Dataset

A DSFM is defined by the domain its dataset spans, so the composition of that dataset is a design decision rather than an incidental one. For a robot the domain fixes the embodiment, here the Kuavo humanoid, and the setting, here the industrial floor, but not the task: the dataset is drawn to span the tasks that setting contains. This section describes what we collected, how it was collected, and how it was filtered.

3.1 Scale and Composition

The domain-specific retraining dataset comprises 100 distinct industrial manipulation tasks recorded on the Kuavo4pro dual-arm humanoid, totaling 58,780 demonstration episodes and 24,471,172 synchronized frames. At the native 10 Hz recording rate this is 680 hours of teleoperated manipulation. Every episode is recorded on the same embodiment: unlike cross-embodiment datasets assembled from heterogeneous robot fleets, all 680 hours share one kinematic structure, one action space, and one control stack, so the model is not asked to reconcile conflicting embodiment conventions during retraining. Tasks are assigned to one of five skill categories from their language prompt (Fig. 2): *pick-and-place / transfer* (move an item with no further shaping), *sorting* (route items by type or by an inspection outcome), *assembly / insertion* (fit one part into or onto another), *packaging* (place items into packaging and close or arrange for shipment), and *tool-based operations* (a hand tool is the means of acting on another object). The five categories are deliberately uneven, reflecting what industrial work cells actually ask for rather than a balanced benchmark design.

Figure 2. Task-category composition of the 100-task domain-specific retraining dataset, with representative head-camera frames from each category. Every frame comes from the same Kuavo4pro embodiment.

3.2 Observation and Action Format

Each frame stores three synchronized RGB streams and the robot's proprioceptive state. The cameras are one head-mounted view and one wrist-mounted view per arm, each recorded at 848×480 and 10 Hz, stored AV1-encoded. The

proprioceptive state and the action vector share a single 16-dimensional layout that interleaves the two arms:

$$\underbrace{[\,q^L_{1..7}\,]}_{\text{left arm}}\ \underbrace{[\,g^L\,]}_{\text{left eef}}\ \underbrace{[\,q^R_{1..7}\,]}_{\text{right arm}}\ \underbrace{[\,g^R\,]}_{\text{right eef}}$$

In the action vector each arm contributes seven joint targets and one end-effector command. In the proprioceptive state the same layout carries seven joint positions and one end-effector open/close state. The end-effector entry is interpreted according to the `eef_type` in use (Fig. 1a), which lets the same data format and the same action head to cover both the parallel gripper and the dexterous hand. Arm and end-effector channels are normalized separately, each with per-dimension mean/std statistics computed over the dataset.

Language is attached per episode as a single instruction describing the task being performed, rather than per-step subtask annotations.

3.3 Collection and Quality Control

All data is collected by human operators teleoperating physical Kuavo robots in industrial settings, and processed through our ROS-to-training pipeline (Fig. 1b), which records raw rosbags, screens them, and converts what survives into standardized episodic trajectories with the observation and action layout above.

Quality control runs as a four-stage chain, with a different role responsible at each stage (Fig. 1b). An operator records the episode (collection), an annotator labels it (annotation), a reviewer checks the recording and its labels against the intended procedure (review), and a manager signs the result off (acceptance). Review checks three things beyond whether the procedure was followed. The first is execution: an episode is sent back if the arm knocks into the workpiece, the fixture, or itself, since a demonstration containing a collision teaches the collision. The second is camera framing: the object being manipulated has to be framed in the wrist and head views through the part of the episode where it is manipulated, and it has to stay unoccluded at those moments, whether by the gripper, by the other arm, or by the operator. The third is coverage: object placements are expected to span the task's intended range rather than repeat a convenient configuration, the trajectories that reach them are expected to vary, and lighting and background are expected to change across a task's episodes instead of all coming from one session. Coverage is the reason review is not a formality: the variation a policy needs in order to generalize within a task gets into the dataset here, by decision, or it does not get in at all. The chain is not one-way: material rejected at review returns to annotation and material rejected at acceptance returns to review, so an episode may make more than one pass before it is either accepted or discarded. Alongside this human chain, an automated trajectory filter removes episodes with defects that review does not reliably catch, such as dropped or desynchronized camera frames, proprioception gaps, and trajectories whose length falls outside the expected range for the task. Only episodes that clear both the chain and the filter are converted and enter the dataset; the conversion targets LeRobot format, with HDF5, MCAP, and RLDS exports also supported. The 58,780 episodes reported above are the post-acceptance count.

Two properties of this pipeline matter for the rest of the report. First, because filtering is applied uniformly, the dataset is composed of successful demonstrations, and the retraining objective is behavior cloning on those demonstrations rather than any form of outcome-conditioned learning. Second, the same pipeline serves both training stages, so Stage 2 demonstrations are filtered on the same criteria as Stage 1.

The 100-task, 680-hour dataset, along with the task-specific post-training sets for the 25 benchmark tasks, will be released alongside this report (Sec. 8).

4 Method

KUAVO-VLA-1.0 targets a specific, practical problem: taking a strong, general-purpose VLA foundation model and turning it into a reliable manipulation policy for industrial dual-arm humanoid deployment. Rather than proposing a new backbone architecture, we treat this as a domain-specialization problem and design a two-stage pipeline, illustrated in Fig. 1d. Stage 1 (Sec. 4.2) converts the general-purpose backbone of Sec. 4.1 into a domain-specialized foundation model (DSFM) for industrial manipulation; Stage 2 (Sec. 4.3) specializes that DSFM on individual tasks. The dataset that defines the domain is described separately in Sec. 3.

4.1 Base Model

KUAVO-VLA-1.0 is initialized from LingBot-VLA-2.0 [24], a large-scale pretrained vision-language-action foundation model that follows the standard recipe in this family of models: a vision-language understanding backbone, pretrained on internet-scale image-text and video data, coupled with a Mixture-of-Experts (MoE) action head that iteratively refines noised action tokens, in a flow-matching process of ten denoising steps, into a denoised chunk of low-level robot actions, conditioned on contextual tokens from the understanding backbone (Fig. 1c). This backbone is pretrained across a broad mixture of embodiments and manipulation tasks, giving it general-purpose visual grounding and instruction-following priors before any domain specialization takes place. It is, in the terms of Sec. 1, a general-purpose foundation model: the starting point for a DSFM, not one itself. We deliberately treat the precise pretraining recipe of this backbone as an implementation detail rather than a contribution of this report. Our focus is the domain-specific retraining recipe described next, which we expect to transfer across comparably-sized VLA backbones with a similar understanding-backbone-plus-MoE-action-head design.

The backbone was chosen empirically rather than by preference. Before any domain-specific retraining, we post-trained each of the three candidate general-purpose models of Sec. 5.2 on the GM100 tasks that later form the public group of the benchmark (Tab. 2), under a common protocol; LingBot-VLA-2.0 came out ahead, and we built on the strongest starting point available to us. Those same tasks are later evaluated as the held-out group of Sec. 5.1, and we return there to what that reuse does and does not cost. Selecting the strongest candidate and then building on it makes the margin we report over it harder to obtain rather than easier.

This also fixes what the experiments can show. LingBot-VLA-2.0 is one of the baselines in Sec. 5, so comparing the two holds architecture, pretraining dataset, and parameter count constant and varies only the domain-specific retraining stage. The other baselines say how KUAVO-VLA-1.0 compares to the field; this one says what that stage did.

Concretely, the backbone pairs a Qwen3-VL-4B-Instruct vision-language understanding model with a token-level MoE action expert: 36 MoE layers, 32 routed experts per layer with top-4 routing plus a shared expert, for 6B parameters in total. Images are resized to 256 pixels per side before entering the understanding backbone, which patchifies them at a patch size of 16 with a 2×2 merge. The language instruction is truncated to 72 tokens. Action and state vectors are zero-padded to a fixed 55-dimensional slot so that one checkpoint serves every arm and end-effector configuration, with the active 16 channels selected per configuration.

KUAVO-VLA-1.0 is deployed on the Kuavo4pro dual-arm humanoid platform (Fig. 1a). The policy consumes a head camera and two wrist cameras, together with proprioceptive arm state, and outputs an action chunk over the dual-arm joint space (or single-arm, depending on task configuration), with the exact action dimensionality determined by which arm(s) are active and by the end-effector in use (Fig. 1a). The Kuavo 5 and Kuavo 5W platforms share the same action space (Fig. 1a), so a checkpoint trained on Kuavo4pro data is interface-compatible with them without a change of layout; no results on Kuavo 5 or 5W are reported here. All data and results in this report are from Kuavo4pro. Kuavo4pro supports two interchangeable end-effectors (a parallel gripper, `leju_claw`, and a dexterous hand, `qiangnao`) and KUAVO-VLA-1.0's action head and training recipe are configured, not hard-coded, to cover both, as well as single-arm and dual-arm operation (Fig. 1a).

4.2 Domain-Specific Retraining

The first stage exposes the pretrained backbone to industrial manipulation at scale, and produces the DSFM. We retrain the model on a self-collected dataset of 100 distinct industrial tasks, totaling 680 hours of dual-arm teleoperated demonstrations (Sec. 3), using the same action-prediction objective as backbone pretraining. Procedurally this is large-scale continual post-training, and it inherits that literature's concerns about retaining pretrained capability. What distinguishes it is the dataset: drawn to span an application domain, it fixes the domain the resulting model is specialized to.

The stage serves two purposes. First, it shifts the model's visual and behavioral distribution toward industrial scenes, meaning line-side fixtures, connectors, packaging, and tooling, which are underrepresented in general-purpose pretraining data. Second, because the 100 tasks span pick-and-place, sorting, assembly and insertion, packaging, and tool-based operations, retraining is designed to encourage skill transfer across them. That transfer is the difference between a DSFM and a multi-task checkpoint that happens to have been trained on industrial data; the unseen in-domain group of Sec. 5.4 speaks to it.

Appendix D lists the full training configuration. Most of it is carried over rather than tuned: the action-prediction objective and the auxiliary depth and future-frame losses come from the backbone's own recipe and keep their original weights, and the batch follows from the hardware at 32 sequences on each of 32 GPUs. Two settings are ours, and each carries an argument. The vision encoder is not frozen, so visual features adapt to the industrial distribution, but it runs at its own learning rate of 1×10^{-6}, fifty times below the rest of the stack, so industrial imagery reshapes those features without erasing the pretrained ones. The schedule is a constant 5×10^{-5} with no warmup and no weight decay, which keeps the stage as close as we can to a continuation of pretraining rather than a fresh optimization. The DSFM carried into Stage 2 is the tenth-epoch checkpoint.

4.3 Task-Specific Post-Training

Industrial deployment often demands per-task success rates that a single multi-task checkpoint cannot consistently deliver, particularly for tasks with tight tolerances or unusual tooling. For the 25 benchmark tasks we therefore add a lightweight task-specific stage: starting from the DSFM of Sec. 4.2, we post-train on demonstrations for a single task under a fixed budget of 20 epochs. The budget is fixed rather than tuned, every baseline is given the same one (Sec. 5.2), and the final-epoch checkpoint is the one evaluated in all cases. This is the sense in which a DSFM remains a foundation model inside its domain: it is what a per-task deployment is adapted from, not a finished policy.

5 Experiments

We design our evaluation to answer three questions: (1) How does KUAVO-VLA-1.0 compare against state-of-the-art open VLA baselines on real industrial manipulation tasks? (2) Under a matched per-task adaptation budget, what does routing specialization through a DSFM buy over specializing a general-purpose foundation model directly? (3) How much of the resulting evidence survives the sample sizes that real-robot evaluation allows?

5.1 Benchmark Setup

We construct a benchmark of 25 manipulation tasks on the Kuavo dual-arm humanoid platform, in two groups. The first group is 20 line-side industrial tasks (Tab. 1), covering the same broad skill categories present in our retraining dataset (pick-and-place, sorting, assembly and insertion, packaging, and tool-based operations). Every one of them also appears in the 100-task domain-specific retraining dataset (Sec. 4.2), and the relationship is tighter than mere appearance: the demonstrations used to post-train each of these tasks in Stage 2 are the same demonstrations that represent that task in the Stage 1 dataset. The *Hours* column of Tab. 1 is thus both quantities at once. Two consequences follow, and they pull in opposite directions. The first is reassuring: these tasks are not over-weighted in Stage 1, accounting together for 135.9 of the dataset's 680 hours, 20.0% of it for 20% of its tasks, and averaging the same 6.8 hours each as the 80 tasks outside the benchmark. The second is not: KUAVO-VLA-1.0 trains on each task's demonstrations twice, for ten epochs within Stage 1 and twenty more in Stage 2, whereas every baseline sees them only in Stage 2. The exposure gap on the target task's own data is thus 30 passes against 20, a factor of 1.5, on top of which KUAVO-VLA-1.0 has also seen the remaining 544.1 hours of the domain that no baseline has. On this group the benchmark therefore measures task-specific specialization on top of already-seen tasks, which is the regime a DSFM is built for, since the domain dataset is meant to cover the deployment distribution. The GM100 group below is the cleaner comparison on exactly this point.

The second group is 5 tasks drawn from the public GM100 task collection (Tab. 2). Their post-training demonstrations were teleoperated on Kuavo4pro in GM100 scenes and are absent from the Stage 1 retraining dataset, so every method here, KUAVO-VLA-1.0 included, sees each task's data exactly once and only in Stage 2. This group exercises the same dual-arm manipulation skills as the industrial dataset while removing the double exposure described above, which makes it both a probe of how well the DSFM adapts to in-domain tasks it was not trained on and the cleaner of our two groups on that point. These are also the tasks on which the backbone was selected (Sec. 4.1), which if anything favors the baseline here. We report the two groups separately throughout, and give the pooled 25-task average as the headline figure.

For evaluation, each task is executed over 15 real-robot rollouts under the same task-specific initialization protocol. We report success rate (SR) and progress score (PS) to capture end-to-end task completion and partial progress, respectively. SR is the fraction of rollouts that satisfy all task-specific scoring criteria within the 3-minute evaluation

horizon. PS assigns partial credit to observable task-specific scoring events. For task t, let K_t denote the maximum attainable score and $s_{t,i}$ the score obtained in rollout i; the rollout-level progress is $s_{t,i}/K_t$. For example, the egg-packing task (task 23) evaluates three eggs with one point for acquisition and one for stable placement of each egg ($K_t = 6$). A rollout that successfully places two eggs and acquires, but fails to place, the third therefore receives a progress score of $5/6$.

A rollout terminates upon reaching the 3-minute time limit, after three consecutive subtask failures, or upon a safety-critical event such as a collision. Progress is computed from scoring events completed before termination and normalized within each task before aggregation. Task-specific scoring criteria and full metric definitions are given in App. E.

Table 1. The 20 industrial tasks of the benchmark. Each task is initialized from the DSFM checkpoint (Sec. 4.2) and task-specifically post-trained for 20 epochs (Sec. 4.3). *Hours* is the volume of task-specific post-training demonstrations, which for this group is the same data that represents the task in the Stage 1 dataset (Sec. 5.1).

#	Category	Task prompt	Hours
1	Sorting	Pick up the specified part and place it into the appropriate compartment of the blue bin.	5.8
2	Pick & Place	Transfer the shock absorber buffer block from the conveyor belt to the shelf.	5.7
3	Sorting	Place parts of different sizes from the tabletop into their corresponding size-matched boxes.	5.5
4	Pick & Place	Grasp ring parts from the bin, place them on the conveyor belt and flip them over.	5.4
5	Packaging	Place canned products from the conveyor belt into the corresponding slots of the product tray while keeping the tray stable.	5.8
6	Assembly / Insertion	Pick up each bottle cap from the table and place it onto the corresponding laundry detergent bottle on the conveyor belt.	5.2
7	Pick & Place	Take the fast-moving consumer goods onto the conveyor belt one by one.	5.0
8	Pick & Place	Take fast-moving consumer goods from the conveyor belt to the basket.	5.1
9	Pick & Place	Invert the parts with the big end facing upwards on the conveyor belt.	5.4
10	Sorting	Pick up a PCB board, place it into the test fixture, and sort it into the pass bin or fail bin according to the indicator light.	5.4
11	Pick & Place	Pick up test tubes and place them on the conveyor belt with the correct alignment.	5.5
12	Tool-based	Use the pressing hammer to flatten the raised sealing flaps of the packaging boxes until they are fitted against the box body.	6.1
13	Pick & Place	Push small boxes on conveyor belt to align with partition for painting.	5.3
14	Pick & Place	Grasp parts from bin and place them on conveyor belt.	5.6
15	Sorting	Sort black switches into the right material bin and white switches into the left material bin.	13.2
16	Assembly / Insertion	Insert cylindrical column into ring-shaped column to complete assembly.	10.8
17	Pick & Place	Pull steel bars out from a constrained stack and place them on the table.	6.4
18	Pick & Place	Pick up SMT trays from the tray rack and place them into the storage bin.	15.5
19	Packaging	Pick up the cosmetic bottle and place it securely into the corresponding tray slot.	8.1
20	Sorting	Sort black and white switches from the conveyor belt into the storage bins on the corresponding sides.	5.1

Table 2. The 5 tasks drawn from the public GM100 collection. These exercise the same Kuavo4pro dual-arm manipulation skills but were held out of the Stage 1 retraining dataset, included as an unseen in-domain reference point (Sec. 5.1). Categories are assigned by analogy to the taxonomy of Tab. 1. These tasks have task-specific post-training demonstrations of their own, teleoperated on Kuavo4pro in GM100 scenes and absent from the Stage 1 dataset, 130 episodes per task. Their *Hours* are smaller than those of Tab. 1, averaging 2.0 against 6.8; all four methods receive the same data on a given task, so the comparison within a task is unaffected, but absolute rates should not be compared across the two groups.

#	Category	Task prompt	Hours
21	Tool-based	BM-107: Scan a QR code with the camera sensor.	1.8
22	Pick & Place	BM-96: Repeatedly squeeze a screaming-chicken toy to make it sound.	1.3
23	Packaging	BM-47: Pack eggs into egg-carton slots.	2.3
24	Sorting	BM-45: Sort and pack snacks by brand and package type.	3.3
25	Assembly / Insertion	BM-39: Replace a paper-towel roll.	1.4

5.2 Baselines

We compare KUAVO-VLA-1.0 against the following systems, all evaluated under a matched real-robot protocol on the same 25-task benchmark:

- **LingBot-VLA-2.0** [24], a publicly released VLA foundation model targeting cross-embodiment dual-arm and mobile manipulation. This is also the backbone KUAVO-VLA-1.0 is built from (Sec. 4.1), which makes it the one baseline for which everything but the domain-specific retraining stage is held fixed.
- $\pi_{0.5}$ [18], a flow-matching VLA model with open-world generalization.
- **GR00T N1.7** [14], from NVIDIA's Isaac GR00T family of generalist humanoid foundation models.[1]

[1]We evaluate the released GR00T-N1.7-3B checkpoint. NVIDIA has not published separate reports for the N1.5–N1.7 releases, so the citation is

- KUAVO-VLA-1.0, our DSFM after task-specific post-training on the task in question (Sec. 4.3).

For the three external baselines, we apply task-specific post-training on each benchmark task using the same protocol as KUAVO-VLA-1.0: 20 epochs per task, matching the demonstration budget and the number of optimization steps, so that every entry in Tab. 3 is compared under a matched adaptation budget. The baselines are general-purpose foundation models specialized directly to a task. KUAVO-VLA-1.0 reaches the same task through a DSFM. The comparison therefore holds the target task and the adaptation budget fixed and varies only whether the specialization is routed through a domain-specialized intermediate. What that isolates is the contribution of the retraining stage as a whole. It does not, on its own, say how much of that contribution comes from the breadth of the domain dataset and how much from the target task's own share of it (Sec. 5.1). LingBot-VLA-2.0 is a special case among the three: it is the backbone KUAVO-VLA-1.0 is built from, so architecture and pretraining are identical, and the only difference is the domain-specific retraining stage in between.

What "matched" covers here should be stated precisely, because it is narrower than the phrase suggests. We hold the target task, the demonstration set, the epoch count, and the number of optimization steps fixed. Each external model is post-trained under its released default recipe, adapted only as far as our observation and action layout requires. Model capacity is not matched either: the GR00T checkpoint we evaluate is a 3B model against KUAVO-VLA-1.0's 6B, so its results carry a capacity difference on top of everything else. Checkpoint selection is not a source of asymmetry in these numbers, because no selection was performed anywhere in the pipeline. Every entry in Tab. 3 is the checkpoint at the end of the twentieth post-training epoch, for KUAVO-VLA-1.0 and for all three baselines alike, and the DSFM those KUAVO-VLA-1.0 runs start from is itself the final, tenth-epoch checkpoint of Stage 1 (Sec. 4.2). Nothing anywhere is ranked by rollout outcome, so no method is tuned against the evaluation it is scored on. What remains is that a baseline may be under-tuned rather than incapable, and one pattern in Tab. 4 is consistent with that reading. Zero-success tasks are widespread among the baselines, at 20 of 25 tasks for GR00T N1.7, 15 for $\pi_{0.5}$, and 14 for LingBot-VLA-2.0, against 6 for KUAVO-VLA-1.0. Some of that is task difficulty, since KUAVO-VLA-1.0 also scores zero on six tasks. The numbers below are a comparison at default recipes.

All reported results are produced under one configuration: a 10 Hz policy loop interpolated to 100 Hz whole-body control, with action chunks of 50 steps executed open-loop between forward passes, and policy inference running in bfloat16 on a workstation-class NVIDIA RTX 4090 host connected to the robot. Section 6 details this runtime.

5.3 Main Results

Table 3. Results on the 25-task benchmark. **SR** and **PS** are defined in Sec. 5.1. Industrial covers the 20 line-side tasks of Tab. 1; Public covers the 5 unseen in-domain GM100 tasks of Tab. 2; Overall pools all 25. All methods are post-trained for 20 epochs per task under the protocol described in Sec. 5.2.

	Industrial (n=20)		**Public** (n=5)		**Overall** (n=25)	
Method	**SR**	**PS**	**SR**	**PS**	**SR**	**PS**
$\pi_{0.5}$	12.3	32.9	14.7	28.8	12.8	32.1
GR00T N1.7	8.3	22.5	0.0	16.2	6.7	21.2
LingBot-VLA-2.0	14.0	40.1	25.3	49.7	16.3	42.1
KUAVO-VLA-1.0	**49.7**	**74.1**	**42.7**	**76.2**	**48.3**	**74.5**

Figure 3 plots the group averages of Tab. 3. Per-task results for all 25 tasks are given in Tab. 4, and the same numbers aggregated by skill category in Tab. 5 and Fig. 5.

Table 3 compares KUAVO-VLA-1.0 against three general-purpose foundation models given the same 20-epoch per-task protocol on the same tasks. Over all 25 tasks KUAVO-VLA-1.0 reaches 48.3% SR and 74.5% PS, against 16.3% / 42.1% for the strongest baseline, LingBot-VLA-2.0. Restricted to the 20 industrial tasks the figures are 49.7% / 74.1% against 14.0% / 40.1%. The comparison against LingBot-VLA-2.0 stands apart, since it is also KUAVO-VLA-1.0's backbone: architecture, pretraining, and target task are identical, and the only difference is whether task-specific post-training starts from the general-purpose model or from the DSFM obtained by domain-specific retraining on it.

to the GR00T N1 white paper, which describes the architecture of the N1.x line.

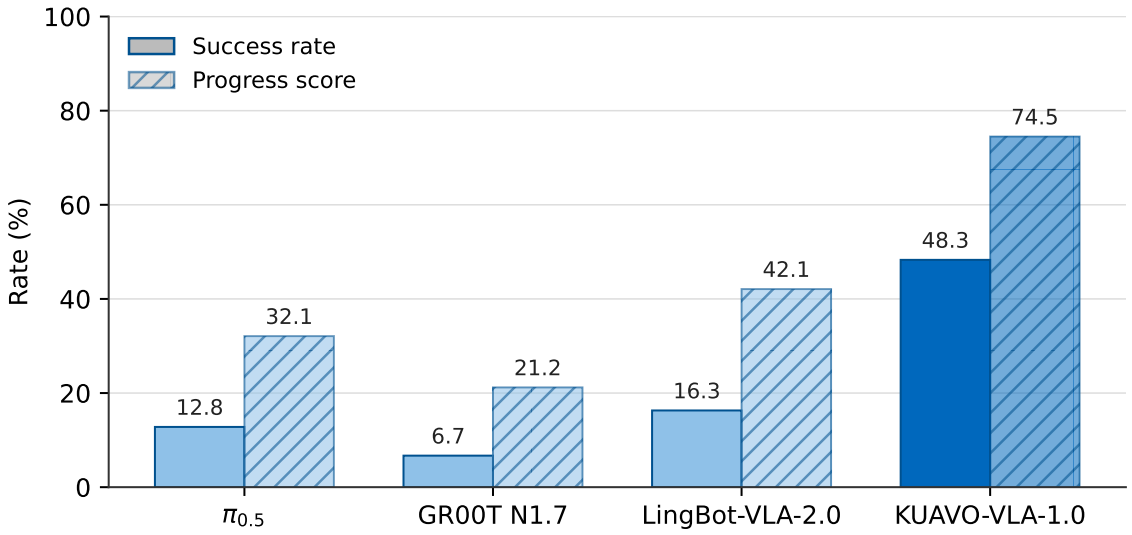


Figure 3. Average success rate (solid) and progress score (hatched) over the 25-task benchmark, visualizing Tab. 3. Every method is specialized to the target task under the same per-task adaptation budget.

That difference contains the whole of Stage 1, domain breadth and same-task data together. Starting from the DSFM yields a $3.0\times$ higher success rate overall, and $3.5\times$ on the industrial tasks alone.

Ratios of group means are not the whole picture, and at 15 rollouts per task the individual cells of Tab. 4 are noisy: a Wilson 95% interval on a single task spans between about 20 and 45 percentage points, depending on where the estimate falls, so no per-task difference in that table should be read as established on its own. The task, not the rollout, is the unit we draw conclusions from. Compared against LingBot-VLA-2.0 task by task, KUAVO-VLA-1.0 has the higher progress score on all 25 tasks, with no exception and a median gap of $+31.7$ points, and the higher success rate on 17 tasks, with 5 ties and 3 tasks where the backbone is ahead; the median gap across all 25 tasks is $+26.7$ points. Four of those five ties are tasks neither model completes at all, and the three the backbone wins are tasks 17, 22, and 23 (Tab. 4). A Wilcoxon signed-rank test rejects equality of success rates at $p = 2.2 \times 10^{-4}$ (normal approximation without continuity correction, over the 20 non-tied pairs; the exact test does not apply here because the non-zero differences contain ties in magnitude). That pooled test is carried by the industrial group, where it gives $p = 5.2 \times 10^{-4}$ over 20 tasks; on the 5 held-out tasks alone it does not reject ($p = 0.63$). What survives the noise, then, is not the exact value of any ratio but the direction and its consistency: the effect is present across the benchmark rather than concentrated in a few tasks.

The gap is wider in SR than in PS ($3.0\times$ versus $1.8\times$), and this is informative about where the benefit lands. PS credits partial progress, so a policy that reliably executes the opening stages of a task and then stalls still scores well. SR requires carrying every stage through to completion. The baselines are not failing to engage with these tasks, as their PS values show, but they break down before finishing. What domain specialization buys is disproportionately the ability to complete a task end to end, which is precisely the property an industrial deployment is graded on.

5.4 Analysis

Unseen in-domain tasks. The 5 GM100 tasks exercise the same Kuavo4pro dual-arm manipulation skills as the industrial dataset, but were held out of the 100-task Stage 1 dataset. They therefore probe how well the DSFM adapts to in-domain tasks it was not trained on. On this group KUAVO-VLA-1.0 reaches 76.2% PS against 49.7% for LingBot-VLA-2.0, its own general-purpose backbone, under the same 20-epoch protocol, and 42.7% SR against 25.3%. We read the two metrics differently here, because $n{=}5$ supports them unequally. The PS advantage holds on every one of the five tasks, from $+1.1$ to $+58.9$ percentage points, and leave-one-out over the five tasks keeps the ratio between $1.34\times$ and $1.78\times$; we treat it as the reliable form of this result. The SR advantage is not robust at this sample size: KUAVO-VLA-1.0 wins two of the five tasks, loses two, and ties one, and although the group mean is $1.7\times$,

leave-one-out spans $1.10\times$ to $2.56\times$, with task 24 alone accounting for most of the margin (Tab. 4). The conservative reading is that the DSFM carries these unseen tasks further through their stages than its backbone does, consistently and on every task, while whether it completes more of them outright cannot be settled by five tasks. Insofar as the effect is real, the natural explanation is a prior that retraining instilled, namely general manipulation competence (grasping, placing, sorting, packaging) that transfers within the domain rather than rote memory of the 100 tasks, since KUAVO-VLA-1.0 has no seen-task advantage here. These are public tasks with published demonstrations, adapted under the same 20-epoch budget, and they differ from the seen tasks in scene rather than in embodiment or skill. What they show is skill transfer to unseen in-domain tasks.

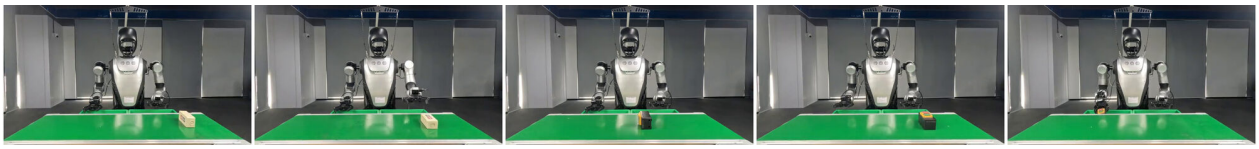
Offline metrics vs. closed-loop success. A recurring question is whether cheap offline metrics (training loss or open-loop action MSE) can stand in for real-robot rollouts when ranking checkpoints. Our experience on this training stack is that they cannot: sweeping open-loop action MSE across the epoch checkpoints of a domain-specific retraining run, we have repeatedly found that the MSE-minimizing checkpoint is not the one that performs best on the robot. We report this as an observation from our own training runs. Its consequence for us was a decision not to select. With no reason to trust the offline ranking, and not enough real-robot throughput to build a closed-loop one, we did not rank checkpoints at all and took the final epoch of each stage (Sec. 4.2, Sec. 4.3). Validation-loss curves on this stack should not be read as a quality ranking either. It is also why Sec. 7 identifies real-robot evaluation throughput, not training compute, as the bottleneck for extending this benchmark.

5.5 Case Studies

Two tasks illustrate what the aggregate numbers mean in practice, one from each group; Fig. 4 shows a rollout of each.

Conveyor switch sorting (task 20, industrial). The robot must track switches arriving on a moving conveyor, classify each by color, and place it into the bin on the corresponding side, repeating until the belt is clear. It is a long-horizon task in the benchmark's own terms, with eight scored stages, and it is unforgiving: a switch missed on the belt cannot be recovered. Starting from the DSFM and post-training on roughly five hours of task demonstrations, KUAVO-VLA-1.0 reaches 80.0% SR and 95.0% PS. The three general-purpose baselines given the same demonstrations reach 0.0%, 0.0%, and 26.7% SR respectively. The two that score zero still reach 35.0% and 31.7% PS, meaning they engage with the belt and pick items, but do not carry a full clearing sequence through.

Snack sorting (task 24, public GM100). This task exercises the same Kuavo4pro manipulation skills but lies outside the Stage 1 dataset: assorted snack packages must be sorted and packed by brand and package type, which requires recognizing deformable, visually similar packaging. With 130 demonstration episodes (3.3 hours), KUAVO-VLA-1.0 reaches 73.3% SR and 93.3% PS. Its own backbone, LingBot-VLA-2.0, given the same episodes, reaches 0.0% SR and 34.4% PS. That a model retrained on industrial line-side data adapts *better* to an unseen in-domain task than the general-purpose model it came from is what the DSFM account predicts. It is also the single task carrying most of that group's success-rate margin (Sec. 5.4), so we offer it as an illustration of the mechanism rather than as evidence for it.



(a) Task 20, conveyor switch sorting: 80.0% SR, 95.0% PS.

(b) Task 24, GM100 snack sorting: 73.3% SR, 93.3% PS.

Figure 4. Rollouts of KUAVO-VLA-1.0 on the two case-study tasks, sampled at even intervals across a single episode and ordered left to right. Both are successful episodes; App. C notes what this footage does and does not cover.

Qualitative behavior. Beyond the successful rollouts above, inspection of failed episodes reveals several recurring failure modes that remain challenging even after domain specialization. A common pattern is that the policy can execute an appropriate local manipulation for a substantial portion of an episode, yet fails when success depends on distinguishing visually similar task states, maintaining progress over repetitive behavior, satisfying a fine geometric condition, or coordinating a contact-rich transition.

Terminal-state ambiguity (task 21). BM-107 provides a representative example in which the beginning and end of the task can be visually difficult to distinguish. The robot must position a camera sensor and scan a QR code, but a successful scanning event does not necessarily produce a salient change in the RGB observations available to the policy. Consequently, the visual scene immediately before and after task completion may remain nearly identical. In failed rollouts, the policy may reach an appropriate scanning configuration but continue adjusting the sensor pose or repeat the scanning behavior instead of terminating. The difficulty is in distinguishing two semantically different states—"ready to scan" and "scan completed"—from highly similar observations. We refer to this as *terminal-state ambiguity*. Although KUAVO-VLA-1.0 achieves a relatively high success rate on BM-107, the remaining failures indicate that task completion can still be difficult to infer when the environment provides little observable evidence of the underlying event.

Temporal aliasing in repetitive tasks (task 22). BM-96 exposes a related ambiguity along the temporal dimension. The task requires the robot to repeatedly squeeze a screaming-chicken toy, such that consecutive portions of the trajectory contain nearly identical object configurations and action patterns. Once a stable grasp has been established, an individual squeezing motion can be executed repeatedly, but the policy must also determine how far the overall sequence has progressed and when it should terminate. Successive visual observations provide little direct evidence of this latent progress variable, making different points in the task horizon locally difficult to distinguish.

This failure mode is reflected by the gap between partial progress and full-task completion. KUAVO-VLA-1.0 obtains 65.6% PS but 0.0% SR on BM-96. The policy can therefore execute a substantial fraction of the intended behavior without completing the full sequence. The bottleneck is therefore not the local squeezing primitive but the ability to track task progress across repeated, visually similar states. We refer to this phenomenon as *temporal aliasing*.

Subtle geometric completion conditions (task 12). A related problem appears in the sealing-flap flattening task (task 12, Tab. 1). The robot uses a pressing hammer to flatten raised packaging flaps until they are fitted against the box body. Near completion, however, the distinction between a partially raised flap and a sufficiently flattened flap can correspond to only a small change in height or orientation. Camera perspective and partial occlusion by the hammer or end effector can further reduce the visibility of this difference. A policy may therefore terminate while the flap remains slightly raised, or continue pressing after the intended geometric condition has already been achieved.

Although KUAVO-VLA-1.0 performs well on this task overall, these remaining failures illustrate a more general limitation: industrial task completion can depend on visual differences that are small relative to the overall scene. The aggregate result for the task does not tell us whether the model represents the "flattened" state reliably; the failed rollouts are what expose fine-grained state discrimination as a source of error.

Precision-limited assembly (task 16). The cylindrical-column insertion task (task 16) provides a clearer example of where domain specialization helps the policy progress further without solving the final operation. The policy must first bring the two components into an appropriate coarse configuration and then align them accurately enough for insertion. In failed rollouts, the robot can approach the target and reach a visually plausible pre-insertion pose, but small translational or rotational errors prevent the cylindrical component from mating with the ring-shaped component. At this stage, the remaining difficulty is dominated by fine alignment and contact-sensitive correction rather than by coarse task interpretation.

This distinction is particularly visible in the quantitative results. All four evaluated methods obtain 0.0% SR on this task. However, KUAVO-VLA-1.0 reaches 50.6% PS, compared with 3.9%, 4.4%, and 5.0% for $\pi_{0.5}$, GR00T N1.7, and LingBot-VLA-2.0, respectively. Thus, KUAVO-VLA-1.0 consistently carries the assembly procedure substantially further through its scored stages, while the final precision-sensitive insertion remains unresolved. This case provides a concrete example of domain specialization improving coarse task execution and stage-wise progress without eliminating the tight geometric tolerance required for successful completion.

Bimanual coordination and handover. We also observe failures during object transfer between the two arms. Successful handover requires the receiving end effector to approach the object with an appropriate relative pose, establish a stable grasp, and close sufficiently before the delivering end effector releases it. Small errors in either spatial alignment or timing can lead to premature release, unstable transfer, object displacement, or conflicting motions between the two arms. These failures highlight the difficulty of synchronizing two coupled action streams during a contact-rich transition. Because we do not separately score handover quality in the benchmark, we treat this as a qualitative failure mode rather than making a quantitative claim about the relative advantage of any individual model.

What these five modes share is that the manipulation itself is not the obstacle. In each case the policy executes the intended behavior and then fails on a state transition, a completion condition, or a tolerance. This is the mechanism behind the SR/PS gap of Sec. 5.3: high progress does not imply reliable completion, and the assembly task (task 16) shows it most sharply, where domain specialization moves the policy through substantially more stages than the baselines without closing the final insertion.

6 Deployment

The practical value of a DSFM depends on whether the tasks in its domain can be executed on deployed hardware by operators who did not train the model. Every result in Sec. 5 uses a single serving configuration: KUAVO-VLA-1.0 in bfloat16 on a workstation-class NVIDIA RTX 4090 connected to the robot over the network. This section documents that runtime.

Policy execution is split across two processes connected by a synchronous request-reply socket: a robot-side process that owns the ROS interface, and a model-side process that owns the policy. Each request carries a fixed observation payload, namely three RGB views (one head camera and two wrist cameras, 848×480 on hardware) together with the 16-dimensional proprioceptive state and the language instruction. Each reply carries actions in the same 16-dimensional dual-arm space, seven arm joints plus one end-effector command per arm. The robot-side process contains no model-specific logic: all image layout, dtype, normalization, and arm-slicing conversions live behind a per-model adapter on the model side. The interface is deliberately stable, so that swapping the served model, or the end-effector and arm configuration of Sec. 4.1, requires no robot-side change. The same deployment stack therefore serves KUAVO-VLA-1.0, its task-specific variants, and the external baselines of Sec. 5.2 under an identical protocol, which is how the matched evaluation of Sec. 5 is enforced in practice.

Two rates should be distinguished. The policy loop runs at 10 Hz, and each emitted action is interpolated up to the 100 Hz whole-body-control command rate. The model itself is not evaluated at 10 Hz: it predicts an action chunk of $H = 50$ steps, which is executed open-loop before the next forward pass, so replanning occurs every 5 s of robot time. This amortizes a large backbone over many control steps, and is what makes a 6B-parameter VLA viable in a 10 Hz loop at all. The cost is that the policy is open-loop within a chunk, which bounds how quickly it can react to a disturbance mid-chunk.

7 Limitations

KUAVO-VLA-1.0 reaches 48.3% average success over the benchmark and 6.7% on assembly and insertion, well short of a policy a line can run unattended. The 20 industrial tasks appear in the Stage 1 dataset and their Stage 2 demonstrations are the same data, so on them we measure specialization on tasks the model has already seen. We do not measure what domain specialization costs in open-domain capability. Real-robot evaluation throughput, not training compute, bounds all of this, and it is why the benchmark is the size it is.

8 Conclusion

We term the domain-specialized foundation model (DSFM) the level of specialization between a general-purpose foundation model and a single-task fine-tune, and build one, KUAVO-VLA-1.0, for dual-arm humanoid manipulation on the Kuavo platform. It is trained in two stages: domain-specific retraining on a self-collected 100-task, 680-hour industrial dataset, then task-specific post-training. On a 25-task real-robot benchmark it raises average success on the 20 industrial tasks from 14.0% to 49.7% over LingBot-VLA-2.0, which is both the backbone it is built from and the

strongest of the open VLA baselines we compare against, under a matched per-task budget; on the 5 held-out GM100 tasks it is ahead on progress score for every task. Beyond the benchmark we describe the runtime KUAVO-VLA-1.0 is served under (Sec. 6).

To support reproducible research on domain-specialized foundation models for real industrial deployment, we release:

- this technical report
- a project webpage with qualitative rollouts and additional results, at https://model.lejurobot.com/kuavo-vla-1
- the training and retraining toolchain, built on our open-source LeTools-Learning code, at https://github.com/LejuRobotics/LeTools-Learning
- KUAVO-VLA-1.0 model checkpoints, at
  https://openlet.openatom.tech/tools/KUAVO-VLA-1.0
  https://www.modelscope.cn/models/lejurobot/LET-KUAVO-VLA-1.0-models
  https://huggingface.co/LejuRobotics/LET-KUAVO-VLA-1.0-models
- the underlying industrial manipulation dataset, at
  https://openlet.openatom.tech/tools/KUAVO-VLA-1.0
  https://www.modelscope.cn/datasets/lejurobot/LET-KUAVO-VLA-1.0-Dataset
  https://huggingface.co/datasets/LejuRobotics/LET-KUAVO-VLA-1.0-Dataset

Checkpoints and the dataset are published on all three platforms at once: OpenLET, an open-data community for real-robot embodied intelligence, ModelScope, and Hugging Face. The three carry the same files, so readers can take whichever suits their network. The dataset is released task by task as each is packaged, so the collection grows toward the full 100 tasks rather than appearing at once.

Looking ahead, we plan to expand the industrial task coverage of both the retraining dataset and the benchmark, and to test whether the DSFM level pays off in domains beyond industrial manipulation.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Chilam Cheang, Sijin Chen, Zhongren Cui, et al. GR-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[5] 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. *arXiv preprint arXiv:2303.04137*, 2023.

[6] Pierre Colombo et al. SaulLM-7B: A pioneering large language model for law. *arXiv preprint arXiv:2403.03883*, 2024.

[7] Gemini Robotics Team. Gemini Robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[8] Gemini Robotics Team and Google DeepMind. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[9] Yu Gu et al. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*, 2021.

[10] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[11] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

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

[13] David Kube, Simon Hadwiger, and Tobias Meisen. Robotic foundation models for industrial control: A comprehensive survey and readiness assessment framework. *arXiv preprint arXiv:2603.06749*, 2026.

[14] NVIDIA, Johan Bjorck, Fernando Castaneda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi 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.

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

[16] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. *arXiv preprint arXiv:2310.08864*, 2023.

[17] Physical Intelligence. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

[18] Physical Intelligence, Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[19] Delin Qu et al. EO-1: An open unified embodied foundation model for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[20] Baptiste Rozière et al. Code Llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[21] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[22] Taiyi Su, Jian Zhu, Tianjian Wang, Youzhang He, Zitai Huang, Jianjun Zhang, Chong Ma, Hanyang Wang, Tianjiao Zhang, Munan Yin, et al. DeMaVLA: A vision-language-action foundation model for generalizable deformable manipulation. *arXiv preprint arXiv:2605.31286*, 2026.

[23] Shijie Wu et al. BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

[24] Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, Yibin Liu, Shuai Yang, Tianxiang Zhou, Kejia Zhang, Lei Zhou, Cheng Su, Nan Xue, Bin Tan, Han Zhang, Youchao Zhang, Fei Liao, Xing Zhu, Yujun Shen, and Kecheng Zheng. From foundation to application: Improving vla models in practice. *arXiv preprint arXiv:2607.06403*, 2026.

[25] Zhaoshu Yu, Bo Wang, Pengpeng Zeng, Haonan Zhang, Ji Zhang, Zheng Wang, Lianli Gao, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on efficient vision-language-action models. *arXiv preprint arXiv:2510.24795*, 2025.

[26] Qixin Zeng, Shuo Zhang, Hongyin Zhang, Renjie Wang, Han Zhao, Libang Zhao, Runze Li, Donglin Wang, and Chao Huang. CRL-VLA: Continual vision-language-action learning. *arXiv preprint arXiv:2602.03445*, 2026.

[27] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Full Per-Task Benchmark Results

Table 4 reports per-task results for every method in Sec. 5.2 on every benchmark task, extending the aggregate numbers in Tab. 3. Each cell gives success rate / progress score. Within each task the highest SR and the highest PS are bolded independently (a metric on which every method scores zero is left unbolded). Task indices, descriptions, and category labels match Tab. 1.

Table 4. Per-task success rate (SR) and progress score (PS), in %, for every method in Sec. 5.2 on every benchmark task, split into the 20 industrial tasks and the 5 unseen in-domain GM100 tasks. LB2.0 = LingBot-VLA-2.0; GR00T = GR00T N1.7; Ours = KUAVO-VLA-1.0.

#	Task	$\pi_{0.5}$	GR00T	LB2.0	Ours
1	Pick up the specified part and place it into the appropriate compartment of the blue bin.	0.0 / 20.0	0.0 / 11.1	0.0 / 5.6	0.0 / **31.1**
2	Transfer the shock absorber buffer block from the conveyor belt to the shelf.	20.0 / 53.3	0.0 / 26.7	0.0 / 31.1	**100.0 / 100.0**
3	Place parts of different sizes from the tabletop into their corresponding size-matched boxes.	0.0 / 2.9	0.0 / 16.2	0.0 / 8.6	0.0 / **19.1**
4	Grasp ring parts from the bin, place them on the conveyor belt and flip them over.	20.0 / 56.7	0.0 / 8.9	0.0 / 41.1	**60.0 / 84.4**
5	Place canned products from the conveyor belt into the corresponding slots of the product tray while keeping the tray stable.	0.0 / 22.5	0.0 / 1.7	0.0 / 28.3	**46.7 / 66.7**
6	Pick up each bottle cap from the table and place it onto the corresponding laundry detergent bottle on the conveyor belt.	0.0 / 8.3	0.0 / 0.0	0.0 / 38.3	**13.3 / 55.0**
7	Take the fast-moving consumer goods onto the conveyor belt one by one.	**80.0 / 94.4**	20.0 / 60.0	33.3 / 80.0	60.0 / 85.6
8	Take fast-moving consumer goods from the conveyor belt to the basket.	20.0 / 60.6	33.3 / 76.1	13.3 / 65.6	**73.3 / 82.2**
9	Invert the parts with the big end facing upwards on the conveyor belt.	0.0 / 33.3	0.0 / 2.2	0.0 / 28.9	**46.7 / 75.6**
10	Pick up a PCB board, place it into the test fixture, and sort it into the pass bin or fail bin according to the indicator light.	0.0 / 14.4	0.0 / 13.3	0.0 / 38.9	**60.0 / 83.3**
11	Pick up test tubes and place them on the conveyor belt with the correct alignment.	33.3 / 66.7	**60.0** / 77.8	**60.0** / 80.0	**60.0 / 85.6**
12	Use the pressing hammer to flatten the raised sealing flaps of the packaging boxes until they are fitted against the box body.	53.3 / 61.7	0.0 / 1.7	0.0 / 1.7	**86.7 / 88.3**
13	Push small boxes on conveyor belt to align with partition for painting.	13.3 / 61.7	40.0 / 73.3	66.7 / 90.0	**86.7 / 96.7**
14	Grasp parts from bin and place them on conveyor belt.	6.7 / 21.1	13.3 / 28.9	26.7 / 52.2	**60.0 / 87.8**
15	Sort black switches into the right material bin and white switches into the left material bin.	0.0 / 23.3	0.0 / 9.2	0.0 / 27.5	**60.0 / 88.3**
16	Insert cylindrical column into ring-shaped column to complete assembly.	0.0 / 3.9	0.0 / 4.4	0.0 / 5.0	0.0 / **50.6**
17	Pull steel bars out from a constrained stack and place them on the table.	0.0 / 2.2	0.0 / 0.0	**6.7** / 31.1	0.0 / **55.6**
18	Pick up SMT trays from the tray rack and place them into the storage bin.	0.0 / 0.0	0.0 / 1.1	46.7 / 77.8	**73.3 / 95.6**
19	Pick up the cosmetic bottle and place it securely into the corresponding tray slot.	0.0 / 16.7	0.0 / 5.6	0.0 / 7.8	**26.7 / 55.6**
20	Sort black and white switches from the conveyor belt into the storage bins on the corresponding sides.	0.0 / 35.0	0.0 / 31.7	26.7 / 63.3	**80.0 / 95.0**
Industrial average		12.3 / 32.9	8.3 / 22.5	14.0 / 40.1	**49.7 / 74.1**
Public GM100 tasks (unseen in-domain)					
21	BM-107: Scan a QR code with the camera sensor.	0.0 / 4.0	0.0 / 8.0	53.3 / 65.3	**80.0 / 88.0**
22	BM-96: Repeatedly squeeze a screaming-chicken toy to make it sound.	**13.3** / 48.9	0.0 / 3.3	6.7 / 54.4	0.0 / **65.6**
23	BM-47: Pack eggs into egg-carton slots.	0.0 / 6.7	0.0 / 4.4	**66.7** / 81.1	60.0 / **82.2**
24	BM-45: Sort and pack snacks by brand and package type.	60.0 / 84.4	0.0 / 40.0	0.0 / 34.4	**73.3 / 93.3**
25	BM-39: Replace a paper-towel roll.	0.0 / 0.0	0.0 / 25.0	0.0 / 13.3	0.0 / **51.7**
Public average		14.7 / 28.8	0.0 / 16.2	25.3 / 49.7	**42.7 / 76.2**
Overall average		12.8 / 32.1	6.7 / 21.2	16.3 / 42.1	**48.3 / 74.5**

B Per-Category Breakdown

Table 5 aggregates Tab. 4 by skill category, which isolates whether any method is systematically weaker on a particular kind of task.

KUAVO-VLA-1.0 leads every category on both metrics. On Assembly / Insertion and Packaging the baselines

Table 5. Per-category averages (success rate / progress score, in %) over the 25-task benchmark. n is the number of benchmark tasks in each category; the first five rows are the industrial group of Tab. 1, the last row the unseen in-domain GM100 group of Tab. 2. Within each row the highest SR and the highest PS are bolded independently.

Category	n	$\pi_{0.5}$	GR00T	LB2.0	Ours
Pick & Place	10	19.3 / 45.0	16.7 / 35.5	25.3 / 57.8	**62.0 / 84.9**
Sorting	5	0.0 / 19.1	0.0 / 16.3	5.3 / 28.8	**40.0 / 63.4**
Assembly / Insertion	2	0.0 / 6.1	0.0 / 2.2	0.0 / 21.7	**6.7 / 52.8**
Packaging	2	0.0 / 19.6	0.0 / 3.6	0.0 / 18.1	**36.7 / 61.1**
Tool-based	1	53.3 / 61.7	0.0 / 1.7	0.0 / 1.7	**86.7 / 88.3**
Public (GM100)	5	14.7 / 28.8	0.0 / 16.2	25.3 / 49.7	**42.7 / 76.2**

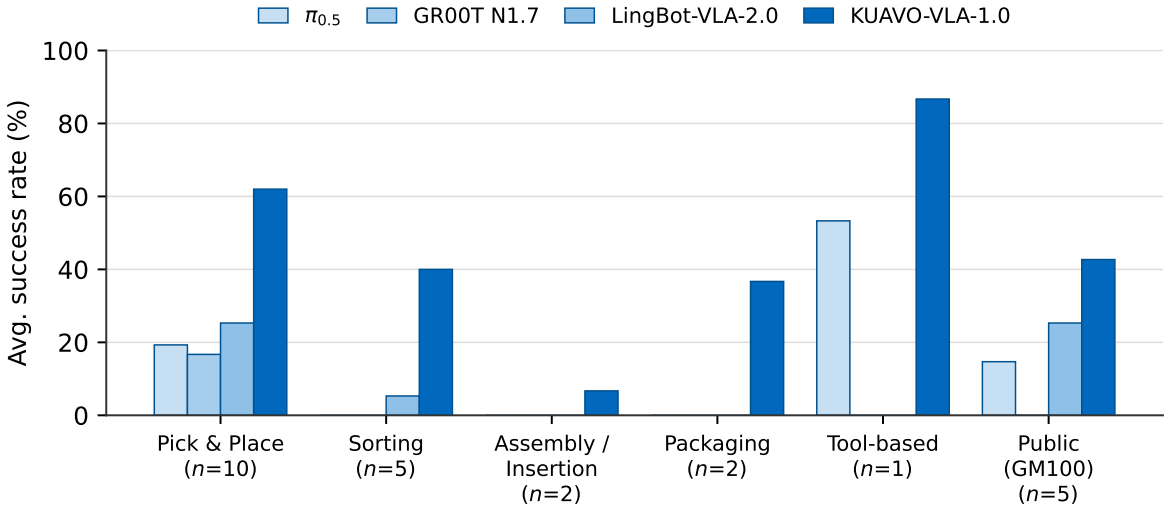


Figure 5. Average success rate by skill category, visualizing the SR column of Tab. 5. The three general-purpose baselines register no successful episode at all on Assembly / Insertion and Packaging; on Sorting only one of them scores at all, and only on one task. The rightmost group is the unseen in-domain GM100 set.

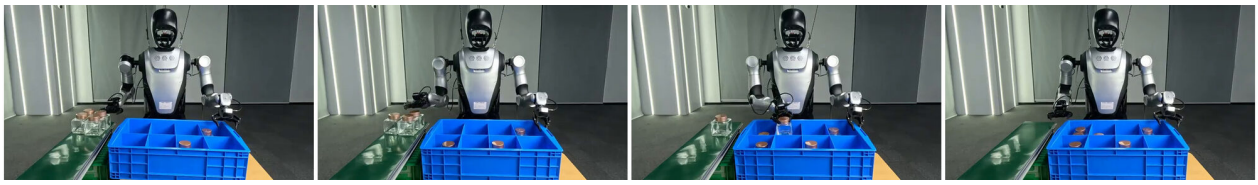
collapse entirely: none of the three general-purpose models completes a single episode of any task (0.0% SR throughout), while KUAVO-VLA-1.0 reaches 6.7% and 36.7%. Sorting is barely better for them: two of the three score zero across all five tasks, and the third, LingBot-VLA-2.0, owes its 5.3% entirely to one task (Tab. 4), against KUAVO-VLA-1.0's 40.0%. These are the categories that require a decision or a tolerance to be met rather than a motion to be executed, in that sorting conditions the placement on a perceived attribute, insertion demands sub-centimeter alignment, and packaging requires respecting slot geometry.

Pick & Place is where the baselines are most competitive (25.3% SR for LingBot-VLA-2.0) and correspondingly where KUAVO-VLA-1.0's relative advantage is smallest, though it still more than doubles the strongest baseline. The Assembly / Insertion row is the weakest in absolute terms for every method, KUAVO-VLA-1.0 included (6.7% SR against 52.8% PS), which is consistent with the tolerance argument of Sec. 1: these tasks are reliably *approached* but rarely *completed*. Tool-based rests on a single task and should not be read as a category-level result.

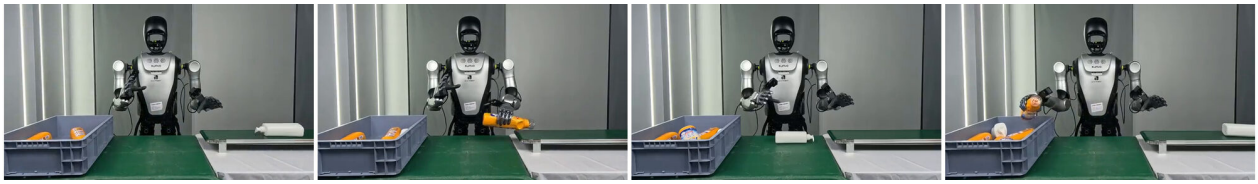
C Qualitative Rollouts

Figure 6 and Fig. 7 show KUAVO-VLA-1.0 executing ten benchmark tasks on the robot, five from each group. Each row is a single episode, sampled at even intervals and ordered left to right.

Every episode shown is a successful one, drawn from the deployment footage we recorded, so the figures illustrate the behavior behind the success rates of Tab. 4 rather than sampling those rates.

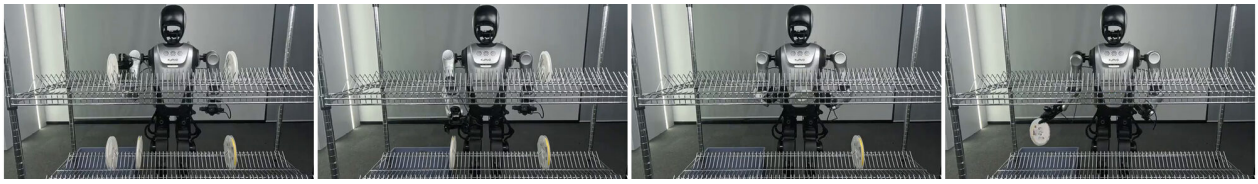


Task 5, Packaging: place canned products from the conveyor into tray slots.

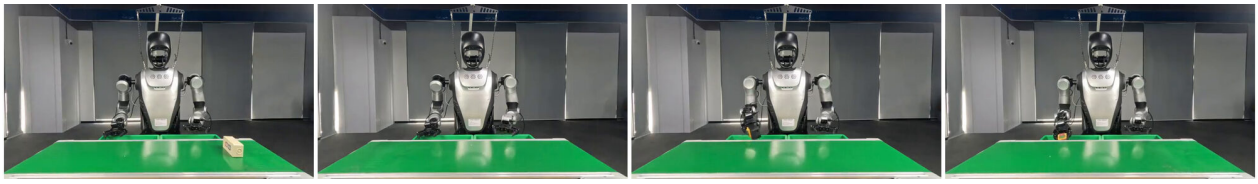


Task 8, Pick & Place: take consumer goods from the conveyor to the basket.

Task 10, Sorting: PCB into the test fixture, then to the pass or fail bin.

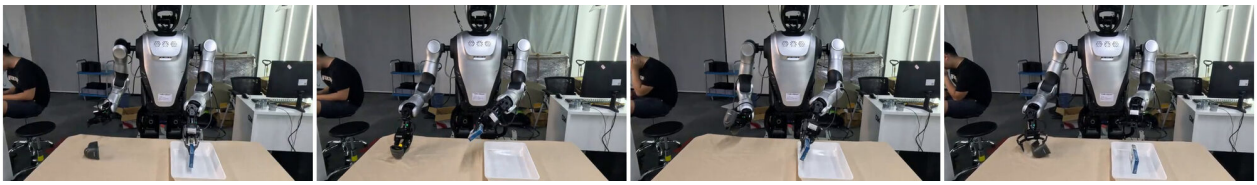


Task 18, Pick & Place: SMT trays from the rack into the storage bin.

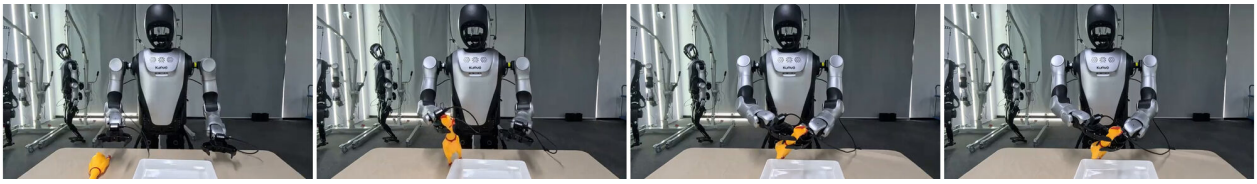


Task 20, Sorting: switches from the conveyor into the side bins.

Figure 6. KUAVO-VLA-1.0 on five of the 20 industrial benchmark tasks (Tab. 1). Each row is one successful episode.



Task 21, BM-107: scan a QR code with the camera sensor.



Task 22, BM-96: squeeze a screaming-chicken toy.

Task 23, BM-47: pack eggs into egg-carton slots.

Task 24, BM-45: sort and pack snacks by brand and package type.

Task 25, BM-39: replace a paper-towel roll.

Figure 7. KUAVO-VLA-1.0 on the 5 GM100 tasks (Tab. 2), which are held out of the Stage 1 dataset. Each row is one successful episode.

D Training Configuration

Table 6 lists the Stage 1 configuration in full. Section 4.2 explains which of these settings are inherited from the backbone's recipe and which are ours.

Stage 2 inherits this configuration unchanged, for every task. Optimizer, learning rates, batch, chunk size, auxiliary-loss weights, and the unfrozen vision encoder all carry over; what differs is the data, a single task's demonstrations instead of the 100-task dataset, and the budget, 20 epochs instead of 10. Because the global batch is unchanged at 1024, the step count of a Stage 2 run follows from that task's frame count: it is 2,300 steps for task 24, 3,620 for task 20, and 1,020 for task 25, each exactly 20 epochs over that task's demonstrations. Only the final checkpoint is written, which is the one evaluated (Sec. 5.2). The three external baselines are post-trained under their own released recipes rather than this one, so their learning rates and optimizers differ; Sec. 5.2 states what the matched budget does and does not cover.

Table 6. Stage 1 training configuration.

Setting	Value
Compute and parallelism	
GPUs	32 × NVIDIA H100
Sharding	FSDP2
Gradient checkpointing	enabled
`torch.compile`	enabled
Schedule	
Epochs	10 (238,970 steps)
Micro-batch	32 sequences per GPU
Gradient accumulation	none
Global batch	1024
Checkpoints	one per epoch; final epoch used
Optimizer	
Optimizer	Muon
Momentum	0.95, Nesterov
Newton–Schulz steps	5
Learning-rate scaling	matched to AdamW update RMS
Learning rate	5×10^{-5}, constant, no warmup
Vision-encoder learning rate	1×10^{-6}
Weight decay	0
Gradient clipping	1.0 (global norm)
Objective	
Action chunk	50 steps
Auxiliary losses	depth and future frame, weight 0.004 each
Vision encoder	not frozen

E Benchmark Scoring Protocol and Task-Specific Rubrics

E.1 Scoring Protocol

Each benchmark task is evaluated over $N = 15$ real-robot rollouts. To distinguish full task completion from meaningful partial progress, we report both the *success rate* (SR) and the *progress score* (PS).

Each task t is decomposed into a task-specific set of observable scoring events with a maximum attainable score K_t. These events correspond to task-relevant accomplishments such as successful object acquisition, target placement, orientation satisfaction, insertion or mating, correct classification and routing, or successful tool operation. Unless otherwise specified, each scoring event contributes one point.

Let $s_{t,i} \in \{0, \ldots, K_t\}$ denote the score obtained in rollout i of task t. The progress score for task t is

$$\mathrm{PS}_t = \frac{100}{NK_t} \sum_{i=1}^{N} s_{t,i}. \tag{1}$$

Thus, PS measures the average fraction of the task-specific scoring criteria completed across all evaluated rollouts, including rollouts that do not complete the full task.

A rollout is counted as a full success only when all scoring criteria are satisfied, i.e., when $s_{t,i} = K_t$. The task-level success rate is therefore

$$\mathrm{SR}_t = \frac{100}{N} \sum_{i=1}^{N} \mathbb{I}[s_{t,i} = K_t], \tag{2}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function.

For a set of benchmark tasks $\mathcal{T}$, we report macro-averaged SR and PS,

$$\mathrm{PS}_{\mathcal{T}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathrm{PS}_t, \qquad \mathrm{SR}_{\mathcal{T}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathrm{SR}_t. \tag{3}$$

This formulation normalizes progress within each task before averaging across tasks, preventing tasks with a larger number of scoring events from receiving greater weight in the aggregate metric. SR captures end-to-end task completion, whereas PS differentiates policies that make substantial progress from those that fail at an early stage.

E.2 Industrial Task Scoring Rubrics

The task indices below follow the ordering of the 20 industrial benchmark tasks in Table 1. The maximum score K_t and the corresponding task-specific scoring criteria are listed in Table 7.

Table 7. Task-specific scoring rubrics for the 20 industrial benchmark tasks. Unless otherwise specified, each scoring event contributes one point.

#	K_t	Scoring criteria
1	6	Three parts are evaluated independently. For each part, one point is awarded for successful object acquisition and one point for correct placement into the designated bin compartment.
2	6	Three workpieces are evaluated independently. Each contributes one point for successful pickup from the conveyor and one point for stable placement at the designated shelf location.
3	14	Seven parts are evaluated independently. Each contributes one point for successful object acquisition and one point for correct size-conditioned placement into the corresponding receptacle.
4	6	Three parts are evaluated independently. Each contributes one point for successful retrieval from the source bin and one point for placement on the conveyor in the required final orientation.

Continued on next page

Table 7 – continued from previous page

#	K_t	Scoring criteria
5	8	Four products are evaluated independently. Each contributes one point for successful pickup from the conveyor and one point for stable seating in the corresponding tray slot.
6	4	Two bottle caps are evaluated independently. Each contributes one point for successful cap acquisition and one point for correctly aligned placement onto the corresponding bottle opening.
7	6	Two bimanual transfer cycles are evaluated. Each cycle consists of successful right-arm object acquisition, successful inter-arm handover, and successful left-arm placement onto the conveyor belt, with one point awarded for each event.
8	12	Three transfer cycles are evaluated. For each cycle, left-arm acquisition, left-arm target placement, right-arm acquisition, and right-arm target placement are scored independently, yielding four points per cycle.
9	3	Three parts are evaluated independently. One point is awarded for each part successfully reoriented to the required big-end-up configuration on the conveyor belt.
10	6	Three PCB boards are evaluated independently. Each contributes one point for successful loading into the inspection fixture and one point for correct downstream routing to the pass or fail destination according to the inspection result.
11	6	Three test tubes are evaluated independently. Each contributes one point for successful acquisition and one point for placement on the conveyor with the specified alignment.
12	4	One point is awarded for successful acquisition of the pressing hammer, one point for successfully flattening the sealing flaps of the first package, one point for successfully flattening those of the second package, and one point for returning the tool to its designated resting location.
13	4	Four boxes are evaluated independently. One point is awarded for each box successfully moved into the specified alignment against the positioning partition.
14	6	Three parts are evaluated independently. Each contributes one point for successful extraction from the source bin and one point for stable placement onto the conveyor belt.
15	8	Four switches are evaluated independently. Each contributes one point for successful acquisition and one point for correct color-conditioned routing to the designated material bin.
16	12	Three assemblies are evaluated independently. For each assembly, one point is awarded for acquiring the cylindrical component, one for acquiring the mating ring component, one for successful insertion and mating, and one for placement of the completed assembly at the designated output location.
17	6	Three steel bars are evaluated independently. Each contributes one point for successful extraction from the constrained stack and one point for stable placement within the target tabletop area.
18	6	Three SMT trays are evaluated independently. Each contributes one point for successful removal from the tray rack and one point for placement into the designated storage bin.
19	6	Three bottles are evaluated independently. Each contributes one point for successful acquisition and one point for stable seating in the corresponding tray slot.
20	8	Four switches are evaluated independently. Each contributes one point for successful pickup from the moving conveyor and one point for correct color-conditioned placement into the corresponding side bin.

E.3 Held-Out GM100 Task Scoring Rubrics

The task indices below follow the ordering of the five held-out in-domain GM100 tasks in Table 2. Their task-specific scoring criteria are summarized in Table 8.

Table 8. Task-specific scoring rubrics for the five held-out GM100 benchmark tasks. Unless otherwise specified, each scoring event contributes one point.

#	K_t	Scoring criteria
21	5	One point is awarded for acquiring the target product, one point for acquiring and positioning the scanning device, one point for obtaining a valid QR-code read, one point for returning the target product to its designated location, and one point for returning the scanning device to its designated location.
22	6	One point is awarded for successful acquisition of the toy, one point for each of four successful compression cycles that actuate the toy, and one point for placement of the toy at the designated final location.
23	6	Three eggs are evaluated independently. Each contributes one point for successful acquisition and one point for stable placement into a valid carton slot.
24	6	Six snack packages are evaluated independently. One point is awarded for each package correctly classified and routed to its designated packing location according to brand and package type.
25	4	One point is awarded for removing the depleted roll from the holder, one point for placing the depleted roll in the designated discard area, one point for acquiring the replacement roll, and one point for correctly installing the replacement roll in the holder.