Scaling Verification Can Be More Effective than Scaling Policy Learning for Vision-Language-Action Alignment

Jacky Kwok[1†], Xilun Zhang[1†], Mengdi Xu[1], Yuejiang Liu[1§]
Azalia Mirhoseini[1§], Chelsea Finn[1§], and Marco Pavone[1,2§]

[1] Stanford University, Stanford CA 94305, USA
[2] NVIDIA Research, Santa Clara CA 95051, USA
{jackykwok,xilunz,mengdixu,yuejiang.liu,azalia,cbfinn,pavone}@stanford.edu[⋆]

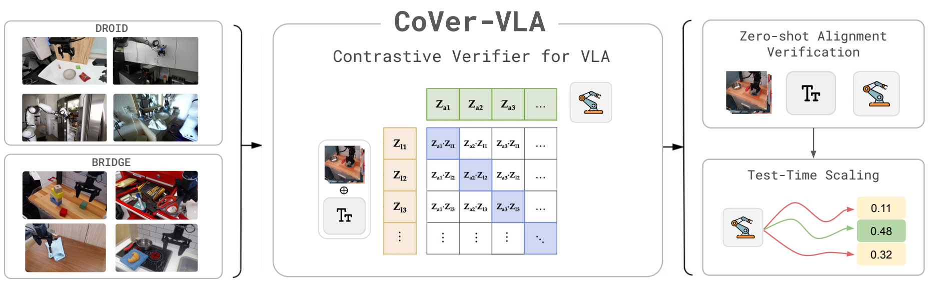

Fig. 1: We present **CoVer-VLA**, a contrastive verification framework for vision–language–action alignment. Our method trains a verifier on large-scale robotics datasets with contrastive learning, enabling zero-shot alignment verification for generalist robot policies out of the box. At test-time, the verifier can be used to perform instruction optimization and action verification, improving downstream performance for VLAs.

Abstract. The long-standing vision of general-purpose robots hinges on their ability to understand and act upon natural language instructions. Vision-Language-Action (VLA) models have made remarkable progress toward this goal, yet their generated actions can still misalign with the given instructions. In this paper, we investigate test-time verification as a means to shrink the "intention-action gap." We first characterize the test-time scaling laws for embodied instruction following and demonstrate that jointly scaling the number of rephrased instructions and generated actions greatly increases test-time sample diversity, often recovering correct actions more efficiently than scaling each dimension independently. To capitalize on these scaling laws, we present CoVer, a contrastive verifier for vision–language–action alignment, and show that our architecture scales gracefully with additional computational resources and data. We then introduce CoVer-VLA, a hierarchical test-time verification pipeline using the trained verifier. At deployment, our framework precomputes a diverse set of rephrased instructions from a Vision-Language-Model (VLM), repeatedly generates action candidates for each instruction, and then uses the verifier to select the optimal high-level prompt and low-level action chunks. Compared to scaling policy pre-training

[⋆] [†] Equal contribution. § Equal advising.

on the same data, our verification approach yields 22% gains in-distribution and 13% out-of-distribution on the SIMPLER benchmark, with a further 45% improvement in real-world experiments. On the PolaRiS benchmark, CoVer-VLA achieves 14% gains in task progress and 9% in success rate.

1 Introduction

For robots to be useful in human-centric environments, they must be able to interpret and act upon natural language instructions. Vision-Language-Action (VLA) models, pre-trained on large-scale robotic datasets, have made significant progress towards this goal [4, 20]. However, their widespread deployment is hindered by a critical "intention-action gap": the misalignment between generated actions and the given language instructions. When the policy fails to follow the instruction, this gap can result in costly errors. For instance, a robot tasked with "putting a plastic container into a drawer" might correctly grasp the container but then fail to discriminate between the oven and a nearby drawer, mistakenly placing the container inside the oven. The container could melt or even catch fire. Addressing this fundamental misalignment is essential for deploying robots in real-world settings.

Existing efforts to close this gap have largely focused on scaling policy pre-training, such as augmenting training data with rephrased instructions [42] or employing larger VLM backbones [2, 11]. However, these approaches typically yield only incremental gains, and performance still degrades severely under simple perturbations [10, 18]. Moreover, scaling policy pre-training often leads to *catastrophic forgetting*, where learning action generation diminishes the VLM's multimodal understanding and reasoning, hindering generalization and semantic understanding [10, 14]. In this paper, we argue that VLA alignment can be more effectively improved through test-time scaling. More specifically, we ask in this work:

Can we enable VLAs to leverage additional computation at test time to improve the alignment between their generated actions and the provided language instructions?

The implications of answering this question extend not only to the generalization capabilities of VLAs, but also to how practitioners should trade off pre-training and test-time compute in robotics. To this end, we first characterize the test-time scaling law for embodied instruction following. Assuming the presence of an oracle verifier, we observe that action error consistently decreases as we scale the number of rephrased instructions, establishing a clear relationship between linguistic diversity and performance gains. Moreover, we demonstrate that jointly scaling the number of rephrased instructions and the generated actions constructs a more diverse action proposal distribution. This hybrid sampling approach often recovers correct actions more efficiently than scaling each dimension independently.

To leverage these scaling laws, we seek to develop a robust verifier for both instruction optimization and action verification. Existing verifiers often focus on low-level dynamics [22, 28] and require costly interactions with the environment [25]. To address this, we draw insights from cross-modal alignment [32, 37] and introduce CoVer, a **co**ntrastive approach for **ver**ifying the alignment across vision, language,

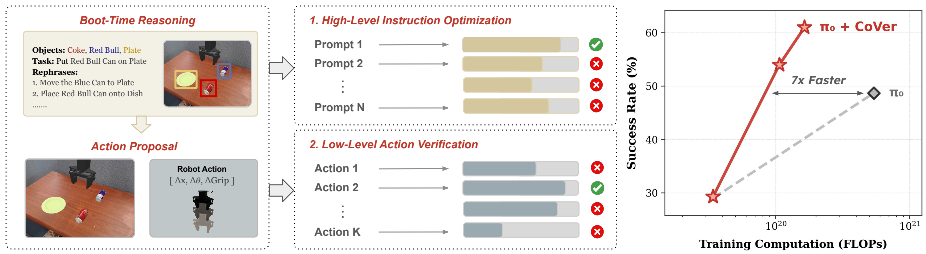

Fig. 2: Hierarchical Test-Time Verification Pipeline. Left: Given the initial observation and language instruction, a VLM performs structured reasoning over the scene and precomputes a set of rephrased instructions during boot time. At each step during deployment, our framework generates a batch of action candidates for each instruction using a VLA. **Middle:** CoVer then scores all instruction–action pairs and selects the optimal high-level instruction and low-level action chunk for execution. **Right:** Compared to prior work on scaling policy learning [3], our approach achieves stronger performance while requiring substantially less compute. The reported training compute for π_0 includes both pre-training and fine-tuning on augmented instruction sets, whereas π_0 + CoVer accounts for pre-training π_0 and training the CoVer verifier on the same data.

and action. Our architecture employs two key components: a text-aware visual encoder that selectively extracts task-relevant features, and an action encoder that captures long-range temporal dependencies within action chunks. The results show that scaling the number of synthetic instructions, model parameters, negative samples, and verifiers in an ensemble consistently improves verification and downstream retrieval accuracy of CoVer. We train CoVer on 20 million offline samples using a 1B parameter backbone, producing a robust verifier for test-time scaling.

During deployment, our framework first leverages "boot-time compute" to let the robot reason offline. Given the initial observation and language instruction, a VLM performs structured reasoning over the scene—identifying relevant objects, spatial relations, and plausible task decompositions. The resulting reasoning traces are then used to precompute a diverse set of rephrased instructions, allowing the robot to avoid redundant rephrase generation during execution. At test time, we employ a hierarchical verification pipeline. This pipeline generates a batch of action candidates for each precomputed instruction with a VLA, scores all instruction-action pairs using CoVer, and then selects the optimal high-level instruction and low-level action chunks for execution. In summary, our contributions are as follows:

1. We characterize the test-time scaling law for embodied instruction following and propose a compute-efficient action sampling method.
2. We present a contrastive verifier for vision–language–action alignment and show that our architecture scales gracefully with additional computational resources and data.
3. We introduce boot-time compute for offline embodied reasoning and a hierarchical test-time verification pipeline that couples high-level prompt optimization with low-level action chunk selection.
4. We show that pairing VLAs with CoVer substantially improves downstream performance, achieving a 45% absolute improvement on real-world tasks, 18% on SIMPLER environments, and 9% on the PolaRiS benchmark.

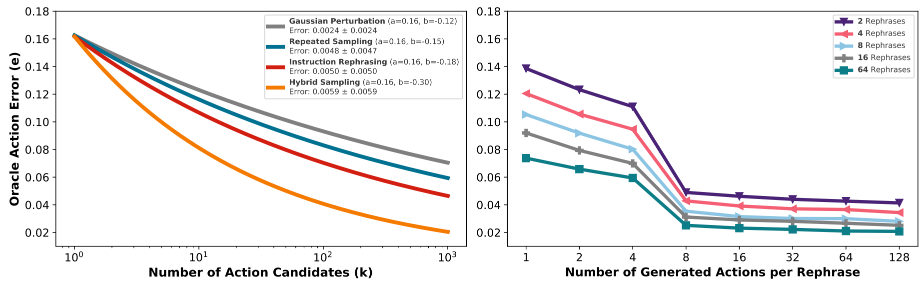

Fig. 3: Test-Time Scaling Law for Embodied Instruction Following. Compared to prior methods that construct an action proposal distribution through repeated sampling [28] or Gaussian perturbations [22], we find that instruction rephrasing produces a broader set of action candidates, leading to improved recovery of the correct action. Furthermore, a hybrid test-time scaling strategy that increases both the number of rephrases and the number of sampled actions per rephrase is more effective than either strategy alone. We characterize each sampling approach using a power law, where the logarithm of oracle action error e is a function of the number of action candidates k: $\log(e) \approx \log(a) + b \cdot \log(k)$.

2 Related Work

Vision-Language-Action Models. Recent VLA models, pre-trained on large-scale multimodal data and fine-tuned for visuomotor control, have demonstrated impressive generalization across tasks, objects, and environments [3, 20, 29, 34, 36]. Yet, they still struggle with instruction following: semantically equivalent rephrases can cause sharp drops in success [10, 18]. Some recent work seeks to mitigate this issue by scaling up model capacity [23], expanding training data [12, 43], and introducing auxiliary objectives to preserve linguistic knowledge [8, 21]. Orthogonal to these training approaches, our work takes a test-time perspective: we treat a user instruction as a distribution over phrasings and verify resulting actions before execution.

Test-Time Scaling. Inference with additional compute has emerged as a promising paradigm for tackling challenging problems across diverse domains, including language reasoning [5, 27, 33, 35], visual understanding [40], and agentic planning [45]. In the context of robot learning, recent studies have demonstrated the effectiveness of optimizing over multiple candidate action sequences to enhance performance [28, 41], consistency [26], and robustness [22]. Such sampling processes can be further accelerated via guidance mechanisms in the latent space [38, 46]. Despite these advances, existing approaches still struggle with instruction following and often incur substantial computational overhead. Our method addresses these challenges through an action verification mechanism explicitly designed for instruction following while enabling acceleration through pre-computation.

Action Verification. Early work on action verification derives signals directly from the policy itself, e.g., prediction uncertainty [13, 42] and temporal consistency [1, 26], yielding lightweight ways to convert prior knowledge into a quality estimator. More recently, a growing body of work has focused on training explicit models for action verification, such as value functions [7, 15] and preference models [22]. Another line of work decomposes verification into two stages: predicting future states with a dynamics model [31, 41], and then assessing task progress in the predicted states. However, these techniques are still largely centered on low-level dynamics, while high-level instruction following remains a challenge. We instead formulate action verification as a contrastive alignment problem between language and behavior, explicitly targeting instruction-following quality.

3 Test-Time Scaling Analysis

In this section, we characterize the test-time scaling law for embodied instruction following, revealing how linguistic diversity in instructions affects downstream robot policy performance. Following the scheme introduced by Kwok et al. [22], we uniformly sample 1,000 (s, a, I) tuples from the Bridge V2 dataset [39]. For each tuple, we scale the number of generated action candidates using different sampling strategies and compute the Normalized Root Mean Squared Error (NRMSE) between the ground-truth action a^* and each of the sampled actions $\{a_1, a_2, ..., a_m\}$.

We evaluate four sampling approaches: **Repeated sampling**: actions are repeatedly sampled from a robot policy $\pi(a|s, I)$ with a positive temperature. **Gaussian perturbation**: a small batch of actions is sampled from the policy $\pi(a|s, I)$, from which a Gaussian distribution is fit and used to draw all candidate actions. **Instruction rephrasing**: actions are sampled from the policy $\pi(a \mid s, I)$ conditioned on rephrased instructions $\{l_1, l_2, ..., l_k\}$ generated by a VLM. **Hybrid sampling**: instead of generating a single action candidate per rephrased instruction, we fan out and repeatedly sample multiple actions per rephrase. We also find that the relationship between action error and total inference FLOPs follows an exponentiated power law across these sampling methods. For power law fitting, we model the logarithm of action error e as a function of the allocated inference compute.

The results in Figure 3 reveal two key findings: (1) instruction rephrasing consistently yields lower action error compared to vanilla repeated sampling and Gaussian perturbation; and (2) the hybrid approach combining instruction rephrasing with repeated sampling achieves even greater diversity by exploring radically different actions rather than getting stuck in a local minimum.

4 Method

While prior works focus either on policy learning or on atomic-level action verification, our approach introduces a general hierarchical test-time verification and scaling framework (Section 4.1) that integrates *scalable verifier training* (Section 4.2) and *hierarchical instruction-action verification* (Section 4.3). Instead of treating the base model's output as final, we jointly select high-level language prompts and low-level action alignment through an optimized latency-aware inference pipeline.

4.1 Hierarchical Prompt-Action Optimization

We consider a sequential decision-making problem with observation space $\mathcal{O}$, action space $\mathcal{A}$, and natural-language instruction space $\mathcal{L}$. At timestep t, the robot receives an observation $o_t \in \mathcal{O}$ and a user instruction $l \in \mathcal{L}$. A chunk-based VLA policy π produces an action chunk $a_t \sim \pi(a_t \mid o_t, l)$, where a_t may correspond to multiple low-level control steps. Natural language permits many semantically equivalent rephrases, yet VLA policies are sensitive to phrasing, as documented by recent language-conditioned manipulation benchmarks [18, 44]. For a rephrased instruction l', the induced action $a'_t \sim \pi(a'_t \mid o_t, l')$ may deviate significantly from the intended behavior, revealing a brittleness to linguistic drift. This motivates treating the instruction itself as a decision variable that can be optimized at test time.

Language-level optimization. Rather than committing to a single phrasing, we construct a set with K number of rephrases:

$$\mathcal{L}_r(l') = \{l'_1, \ldots, l'_K\},$$

all expressing the same user intent. Each l'_k conditions a different action distribution under the fixed base policy. To formalize the objective, we use a conceptual reward function $r(o_t, a, l)$ that measures how well an action a fulfills the semantics of the original instruction l; this reward is *not* computed at test time but serves to define the ideal target behavior. We then aim to select the rephrase whose induced behavior best aligns with the original intent:

$$l^* = \arg\max_{l' \in \mathcal{L}_r} \mathbb{E}_{a \sim \pi(\cdot \mid o_t, l')} \left[r(o_t, a, l) \right].$$

This reformulates VLA inference as an optimization problem in *language space*, not parameter space.

Action-level optimization. Given a selected rephrase l^*, sampling a single action from π is unreliable due to bias and noise. We therefore draw M candidate action chunks from the policy, conditioning on the current observation o_t and the selected instruction l^*:

$$a'_j \sim \pi(\cdot \mid o_t, l^*), \qquad j = 1, \ldots, M,$$

We then select the candidate that maximizes semantic alignment with the instruction:

$$a^*_t = \arg\max_{j \in [M]} \mathcal{V}_\theta\big(o_t, h_t, l^*, a'_j\big),$$

where $\mathcal{V}_\theta$ estimates vision–language–action alignment, and $h_t \in \mathcal{A}^W$ denotes the recent action history (e.g., the past C actions), providing temporal context to the verifier.

This view unifies language refinement and action verification: the system first searches for the rephrase whose induced action distribution aligns with the user intent, then verifies individual action candidates within that distribution. Overall, developing verifier $\mathcal{V}_\theta$ is essential. In the next section, we will describe how to develop a *robust* and *scalable* verifier from available robotics datasets.

4.2 Offline Verifier Training

The objective of verifier $\mathcal{V}_\theta$ is to assess the semantic alignment between visual observations, instruction language, and action sequences. A central challenge in

Algorithm 1 Verifier Training with Rephrase Augmentation

Require: Offline trajectories $\mathcal{D}=\{(o_t,h_t,l,a_t)\}_{t=1}^{T}$; batch size B; Augmented Instruction Set $\mathcal{I}$

```
Initialize augmented dataset D_aug ← ∅
Stage 1: Rephrase Augmentation
for (o_t, h_t, l, a_t) ∈ D do
    for l_n^oxe in I(l) do
        D_aug ← D_aug ∪ {(o_t, h_t, l_n^oxe, a_t)}
Stage 2: Verifier Training
Initialize parameters θ
while not converged do
    Sample minibatch {(o_i, h_i, l_i, a_i)}_{i=1}^B ~ D_aug
    for i = 1..B do
        F_i = F_combined(o_i, l_i)
        A_i = A(h_i, a_i)
        Normalize: f_i = F_i / ||F_i||_2,   a_i = A_i / ||A_i||_2
        Compute pairwise similarities s_{i,j} = ⟨f_i, a_j⟩
        L_i^{f→a} = −log [ exp(s_{i,i}) / Σ_{j=1}^B exp(s_{i,j}) ]
        L_i^{a→f} = −log [ exp(s_{i,i}) / Σ_{j=1}^B exp(s_{j,i}) ]
    L_InfoNCE = 1/(2B) Σ_{i=1}^B (L_i^{f→a} + L_i^{a→f})
    θ ← θ − η ∇_θ L_InfoNCE
```

Where the listing's formulas read, in full:

$\mathcal{D}_{\text{aug}} \leftarrow \mathcal{D}_{\text{aug}} \cup \{(o_t,h_t,l_n^{oxe},a_t)\}$

$\mathbf{F}_i=\mathbf{F}_{\text{combined}}(o_i,l_i)$, $\mathbf{A}_i=\mathbf{A}(h_i,a_i)$, $\mathbf{f}_i=\mathbf{F}_i/\|\mathbf{F}_i\|_2$, $\mathbf{a}_i=\mathbf{A}_i/\|\mathbf{A}_i\|_2$, $s_{i,j}=\langle \mathbf{f}_i,\mathbf{a}_j\rangle$

$\mathcal{L}_i^{f\to a}=-\log\frac{\exp(s_{i,i})}{\sum_{j=1}^{B}\exp(s_{i,j})}$

$\mathcal{L}_i^{a\to f}=-\log\frac{\exp(s_{i,i})}{\sum_{j=1}^{B}\exp(s_{j,i})}$

$\mathcal{L}_{\text{InfoNCE}}=\frac{1}{2B}\sum_{i=1}^{B}(\mathcal{L}_i^{f\to a}+\mathcal{L}_i^{a\to f})$

$\theta\leftarrow\theta-\eta\nabla_\theta\mathcal{L}_{\text{InfoNCE}}$

training a VLA verifier is that robotic datasets contain only successful demonstrations, providing no direct supervision indicating when an action is semantically *misaligned* with an instruction. Constructing negative examples is non-trivial [42]: synthesizing incorrect actions often produces unrealistic motions, while manually annotating failures is prohibitively expensive. Contrastive learning [32, 37] offers a natural solution by treating other actions in the batch as implicit negatives, allowing the model to learn alignment structure without curated failure labels. Our training pipeline consists of two stages: (i) augmenting the instruction space with diverse rephrases and (ii) contrastive learning on the augmented dataset. The detailed algorithm is shown in Algorithm 1.

Rephrase Augmentation. To address the linguistic sensitivity of VLA policies, we expand each original instruction solely in language space, leaving observations and actions fixed. The training language augmentation is obtained from Open-X Embodiment [6] datasets $\mathcal{I}$, where each original task instruction set $\mathcal{I}(l)$ corresponds to N rephrases. Each selected rephrase l_n^{oxe} from $\mathcal{I}(l)$ is then paired with the same observation o_t, and ground-truth action sequence that consists of short-term action history h_t and future action chunk a_t to form additional training tuples. Rephrases enable the verifier to encounter multiple linguistic realizations of the same underlying intent. This procedure enlarges the effective language coverage of the dataset without altering the action distribution, and equips the verifier with the ability to distinguish true semantic equivalence from phrasing-induced discrepancies that often mislead the base VLA policy. Though the same rephrase augmentation technique has been developed for policy learning [9, 10], we demonstrate that using the same data budget to train a verifier would be more effective than directly augmenting the policy training dataset.

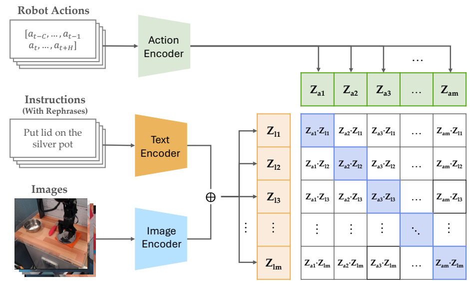

Fig. 4: Overview of CoVer Training Strategy. CoVer learns a joint embedding space aligning visual observations, language instructions, and robot actions through contrastive pre-training. A fused image–text encoder selectively extracts task-relevant visual features, while an action encoder projects action sequences into the same embedding space. This architecture enables cross-modal alignment between high-level instructions and executed behaviors.

Verifier Training and Architecture. The verifier aims to estimate the alignment between visual–textual and action representations. Visual inputs and language tokens are encoded with pre-trained SigLIP2 encoders [37], then fused via text-aware visual attention to obtain instruction-relevant features. Vision and text encoders are frozen during verifier training to preserve the web-scale knowledge [16]. The resulting fused representation $\mathbf{F}_{\text{combined}}$ captures visual–language context. The action sequence, which contains short-term history and future chunks, is processed by a transformer encoder to better capture the temporal features of low-level behaviors [26]. The fused vision–language representation $\mathbf{F}_{\text{combined}}$ and the action embedding $\mathbf{A}$ are then ℓ_2-normalized to get $\mathbf{f}$ and $\mathbf{a}$ respectively. Their similarity defines the alignment score: $s(\mathbf{f},\mathbf{a}) = \langle \mathbf{f},\mathbf{a} \rangle$. Given a minibatch of B tuples $\{(o_i,h_i,l_i,a_i)\}_{i=1}^{B}$, the verifier is trained with bi-directional InfoNCE [30] objective. This symmetrical formulation aligns vision–language embeddings $\mathbf{f}$ with action embeddings $\mathbf{a}$ in both directions. By treating all other pairs in the batch as *implicit negatives*, it leverages the diversity of each minibatch to learn robust fine-grained correspondences without requiring explicit failure labels or hand-crafted counterexamples. Such in-batch contrastive structure enables the verifier to discover meaningful distinctions between semantically aligned and misaligned behaviors, leading to more stable and cycle-consistent vision–language–action grounding during test-time verification. Notably, CoVer learns to recognize misalignment from *positive-only* demonstrations through in-batch self-supervision, sidestepping the difficulty of curating realistic failure trajectories; augmenting training with explicit near-miss or perturbed demonstrations is a complementary direction we leave for future work. The verifier structure is shown in Figure 4. Given a robust verifier that can score the alignment between intentions and actions, we develop a general verification framework that can adapt to any VLA policies without additional training.

4.3 Test-time Verification

In Section 4.2, we explored the advantages of contrastive training for vision-language-action alignment, which enables zero-shot verification for both instruction and actions. Such bidirectional features make the verification process more flexible. In this section, we propose CoVer-VLA, a test-time verification framework that is robust to language-induced action drift, while adding only minimal latency from proposal generation and verification. CoVer-VLA casts inference as a hierarchical verification problem as shown in Figure 5. The system first evaluates on the language level, selecting the instruction whose induced action distribution is most semantically reliable, and then selects the optimal action chunk conditioned on that instruction. This hierarchical structure enables the robot to update its active language prompt online and to filter action proposals using a learned alignment score, improving robustness without altering the underlying VLA policy. To support this procedure, we first introduce boot-time rephrase generation and caching that significantly boosts runtime efficiency by bringing scene reasoning offline. We follow with the details on batched action proposals that enable efficient search over both languages and actions. The resulting pipeline preserves robustness without compromising real-time control.

Boot-time rephrase generation and caching. To efficiently handle linguistic variability, we expand each free-form instruction l' into K rephrases using an off-the-shelf VLM. The VLM takes the initial scene image o_0 and user instruction l' as input. It generates both scene-level reasoning and rephrased command variants $\{l'_k\}_{k=1}^{K}$. Leveraging the VLM's reasoning capabilities incorporates web-scale knowledge into the rephrase generation process. Running the VLM on-the-fly, however, is computationally expensive and can introduce undesirable latency or motion discontinuities during robot control. Given that user intent is typically consistent throughout an episode, generating new rephrases mid-rollout offers limited benefits. Instead, we perform rephrase generation and embedding computation entirely at boot time. By caching rephrase embeddings before execution, we shift the heaviest computations off the critical path and ensure that retrieving rephrase features at inference time incurs negligible overhead. This allows the controller to evaluate paraphrastic variants efficiently at test time, enabling robust test-time optimization without compromising control smoothness.

Inference with batched action proposals. With rephrases cached and a verifier in place, we perform chunk-level optimization by jointly searching over rephrased instructions and candidate action chunks. Let $\{l'_k\}_{k=1}^{K}$ denote the K rephrases generated at boot time, with $l'_1 = l'$. At each chunk boundary, the base VLA policy induces a distribution over action chunks, $a \sim \pi(\cdot \mid o_t, l'_k)$, from which we sample M candidates for each rephrase. This yields $K \times M$ proposals:

$$a'_{k,j} \sim \pi(\cdot \mid o_t, l'_k), \qquad k=1,\dots,K,\ j=1,\dots,M.$$

Each proposal is then evaluated by the verifier ensemble with respect to the *user instruction* l',

$$s_{k,j} = \mathcal{V}_\theta(o_t, h_t, l', a'_{k,j}),$$

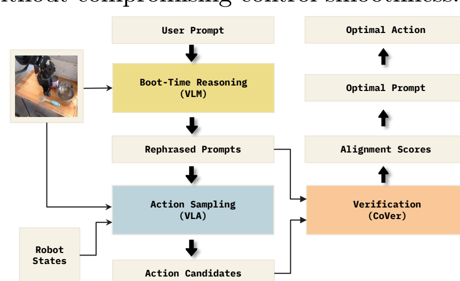

Fig. 5: Overview of Test-Time Verification Pipeline. At deployment, the system performs hierarchical optimization over language and action spaces. Given a user prompt and the initial observation, a VLM first reasons over the scene and generates a set of rephrased prompts at *boot time*. For each rephrase, a VLA samples action candidates conditioned on the corresponding instruction. The trained CoVer verifier then scores all instruction–action pairs and selects the optimal prompt and action for execution.

producing a semantic alignment score for every (rephrase, action) pair. To determine which rephrase induces the most reliable action distribution, we take the average scores across all M actions from the same language:

$$S_k = \frac{1}{M}\sum_{j=1}^{M} s_{k,j}, \qquad k^* = \underset{k}{\operatorname{argmax}} S_k.$$

The chosen rephrase l'_{k^*} becomes the active language for this chunk. Within the selected rephrase, the controller chooses the highest-scoring action candidate:

$$j^* = \underset{j}{\operatorname{argmax}} s_{k^*,j}.$$

The selected action chunk a'_{k^*,j^*} is executed, and the state $(o_{t+\Delta}, h_{t+\Delta})$ is updated accordingly. This procedure repeats at each chunk boundary, forming a closed-loop optimization that continually adapts both the instruction and the executed action.

5 Experiments

5.1 Verifier Scaling Results

In this section, we investigate the scaling behavior of the CoVer verifier. We conduct thorough studies to explore the impact of five key dimensions: model size, dataset size, batch size, training compute, and ensemble size.

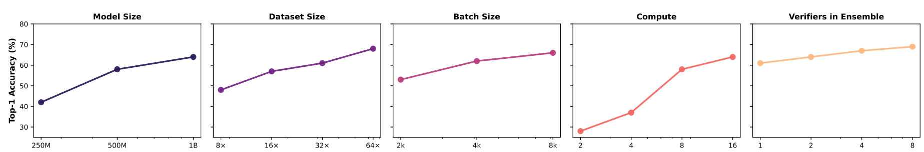

Fig. 6: Verifier Scaling Results. We show that our architecture scales gracefully with additional compute and data. The top-1 action-retrieval accuracy consistently improves as we scale the number of synthetic instructions, model parameters, negative samples, training compute, and the number of verifiers in the ensemble. This result strongly indicates that our approach benefits from scaling, which we exploit for training CoVer.

We first evaluate how scaling synthetic instructions and model parameters affects verifier performance. As shown in Figure 4, CoVer exhibits consistent scaling trends: every increase in dataset size (from 8× to 64×) or in model capacity (from 250M to 1B parameters) leads to steady improvements in top-1 retrieval accuracy. This provides strong empirical evidence that our contrastive approach effectively capitalizes on scaling.

We also investigate the effects of scaling batch size and training epochs. Because our verifier relies on contrastive learning, the number of in-batch negative samples is critical for learning robust decision boundaries. We find that larger batch sizes (scaling from 2,048 to 8,192) provide a richer set of negative examples, thereby facilitating better convergence. Similarly, extending training epochs exposes the model to more diverse negative samples, leading to improved results.

Finally, we explore test-time ensembling as a scaling dimension. Specifically, we train multiple verifiers with identical architectures and data budgets, differing only in their random seeds. During inference, we average the image, text, and action embeddings across these verifiers before computing the cosine similarity between modalities. We find that action retrieval accuracy consistently improves as the ensemble size increases (from 1 to 8). These gains stem from variance reduction, as the ensemble averages out individual model biases.

5.2 Implementation Details

Our final CoVer verifier is a 1B-parameter model trained with a batch size of 32,768 on the augmented Bridge V2 dataset [39] containing 16× synthetic instructions. Training was conducted for a total of 2k steps using 8 NVIDIA H200 GPUs. For deployment, we utilize an ensemble of 3 verifiers to balance robustness and computational overhead.

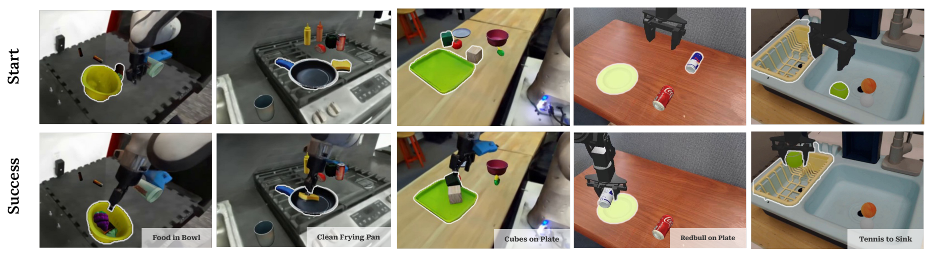

Fig. 7: Example tasks across DROID [19] and Bridge V2 [39] environments.

5.3 Evaluation Setup

We evaluate CoVer-VLA across both simulated and real-world settings, focusing on robustness to linguistic variation and generalization on out-of-distribution environments. Our primary benchmark is the SIMPLER benchmark [24], which includes four in-distribution (ID) manipulation tasks and three OOD variants containing distractor objects and clutter [9]. We evaluate on four representative tasks from the SIMPLER environment and adopt three challenging OOD tasks from Interleave-VLA [9], includes "Redbull on Plate", "Zucchini on Towel", and "Tennis in basket". The OOD environments contain multiple objects in the scene, including hard same-category distractors (e.g., a red Coca-Cola can next to the blue Redbull, or a ping-pong ball next to the tennis ball), where the VLA cannot rely solely on visual inputs and must also reason over the object information in the instructions. For real-world experiments, we use the WidowX robot to evaluate two tasks "pepto bismol on plate" and "redbull on plate". We use π_0 as the base model for tasks in BridgeV2. To assess how our approach performs with a stronger base policy, we also evaluate using $\pi_{0.5}$ and CoVer on the PolaRiS benchmark [17]. All evaluations are conducted under challenging red-teaming instructions generated by ERT [18]. Our framework samples 8 rephrased instructions and generates 5 action candidates per rephrase.

Baselines and Ablations. We compare CoVer-VLA against five variants built on the same π_0 backbone to disentangle the effects of training-time augmentation and test-time verification. (1) π_0 denotes the generalist robot policy fine-tuned on BridgeV2 without instruction augmentation or verification. (2) π_0 **(rephrase)** [10] represents π_0 finetuned on instruction-augmented datasets. (3) **RoboMonkey** [22] applies a

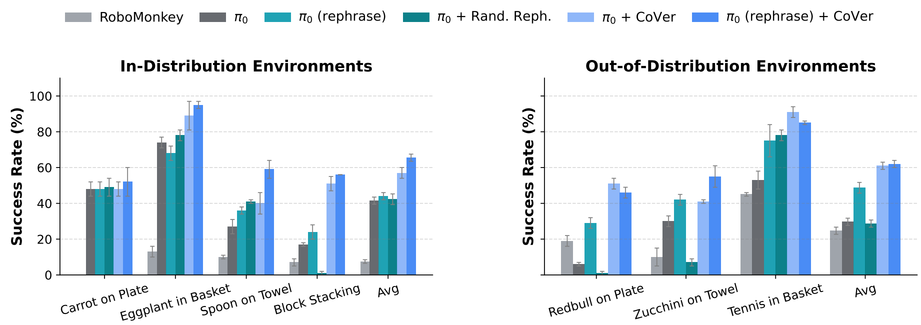

Fig. 8: SIMPLER Evaluation Results. We demonstrate that scaling test-time verification with CoVer significantly enhances the robustness of VLAs across diverse manipulation tasks. Compared to scaling policy pre-training on the same data, our verification-based approach achieves a 22% improvement on in-distribution tasks and a 13% improvement on OOD tasks.

7B-scale verifier with action resampling for test-time scaling, serving as the strongest prior method without hierarchical reasoning. (4)π_0+ **CoVer** introduces our verifier-based inference that jointly optimizes over rephrases and action chunks at test time. (5) π_0+**Rand. Reph.** uses a single random rephrase without verification to isolate the role of language selection. (6) π_0 **(rephrase)**+ **CoVer** combines both training-time augmentation and our hierarchical test-time verifier to examine their complementarity. Together, these baselines allow us to examine the effectiveness of (i) training-time instruction augmentation, (ii) test-time verification of instructions and actions, and (iii) verifier-guided hierarchical optimization. This allows us to systematically assess CoVer 's robustness and ability to generalize across tasks.

5.4 Simulation Evaluation Results

Figure 8 summarizes performance across four ID tasks and three OOD tasks under red-teaming instructions. Due to training distribution shift, RoboMonkey fails to select optimal actions given challenging instructions: its verifier is trained on OpenVLA action preferences that transfer poorly to π_0, and its step-level verification disrupts the temporally coherent action *chunks* produced by flow-based policies, degrading performance below the base policy. For all the other ablations, we observe different levels of performance gain over the base robot policy π_0. We highlight three key findings below:

(1) Training-time augmentation alone provides modest performance gain. We show that fine-tuning π_0 on augmented instruction sets can indeed improve robustness to challenging rephrases. However, this approach yields only minimal gains on in-distribution environments (41.5$\rightarrow$ 44) and provides modest improvements on OOD tasks.

(2) Random rephrases can improve performance on some tasks but lack consistency without language-level verification. Using a randomly generated VLM rephrase slightly improves ID performance over the base policy π_0 (41.5 $\rightarrow$ 42.3), confirming that rephrasing can enhance policy performance in some cases.

	Models	Task Prog. (%)	Succ. (%)
PanClean	$\pi_{0.5}$	48.4 ± 1.9	10.7 ± 0.9
	$\pi_{0.5}$ + CoVer	70.4 ± 4.0	33.3 ± 6.6
BlockStack	$\pi_{0.5}$	33.1 ± 1.3	0.0 ± 0.0
	$\pi_{0.5}$ + CoVer	44.3 ± 2.5	0.7 ± 0.9
FoodBussing	$\pi_{0.5}$	38.3 ± 2.4	0.7 ± 0.9
	$\pi_{0.5}$ + CoVer	47.0 ± 4.1	5.3 ± 1.9
Average	$\pi_{0.5}$	40.0 ± 6.4	3.8 ± 4.9
	$\pi_{0.5}$ + CoVer	**53.9 ± 11.7**	**13.1 ± 14.1**

Table 1: PolaRiS evaluation with $\pi_{0.5}$ (mean ± std, 50 episodes, 3 seeds). Pairing CoVer with $\pi_{0.5}$ improves task progress by 13.9% and success rate by 9.3% on average.

However, OOD performance declines (29.7 → 28.7), and the variance across tasks is substantial. For example, the model achieves a 78% success rate on *Eggplant in Basket* but only 1% on *Redbull on Plate*. This reveals a key insight: while certain rephrased instructions can be beneficial, others may catastrophically mislead the policy. These results underscore the potential of VLM-generated rephrasings, but also expose their inconsistency.

(3) CoVer-VLA substantially enhances generalization and complements policy learning. Pairing CoVer with π_0 significantly enhances robustness, yielding a 16% improvement on in-distribution tasks and a 31% gain in OOD environments. Notably, we find that scaling verification (π_0 + CoVer) outperforms scaling policy learning (π_0 fine-tuned with augmented instructions), achieving 15% gains on ID tasks and 12% on OOD, while requiring substantially less compute, as illustrated in Figure 2. Interestingly, our approach is complementary to scaling policy learning. Combining π_0 (rephrase) and CoVer achieves the strongest overall performance: 65.5% on ID tasks and 62.0% on OOD tasks. We further evaluate our method with a stronger base model, $\pi_{0.5}$, on the PolaRiS benchmark [17]. As reported in Table 1, pairing $\pi_{0.5}$ with CoVer leads to a 14% improvement in task progress and a 9% gain in success rate. By jointly selecting the semantically aligned instruction and verifying action chunks, our method reliably recovers correct behavior even under heavily perturbed instructions and in challenging OOD environments. The low absolute success rates with large variance on BlockStack and FoodBussing reflect their long-horizon nature, where a single early failure precludes completion; the complementary task-progress metric makes the consistent gains from CoVer evident. The remaining bottleneck is the proposal quality of the base policy: when no feasible action candidate is sampled, the verifier cannot recover the correct behavior, which we leave to future work.

Method	K	GFLOPs	Succ.
Temp. Sampling	4	6,280	31.0
Best-of-N	4	6,280	49.3
CoVer	4	7,936	55.3
Temp. Sampling	8	12,560	30.7
Best-of-N	8	12,560	47.0
CoVer	8	15,872	**74.0**

Table 2: Iso-compute comparison on OOD SIMPLER: at matched inference FLOPs, CoVer beats temperature sampling and best-of-N (BoN).

(4) Verification outperforms sampling at matched compute. To verify that our gains stem from verification rather than from additional inference compute, we compare

CoVer against temperature sampling and best-of-N (BoN) at a matched FLOP budget (Table 2). At K=1, CoVer adds only ~26% FLOPs (1,984 vs. 1,570 GFLOPs) yet yields no gain, confirming that verification requires sufficient candidate diversity to be effective. At K=4 and K=8, however, CoVer surpasses both baselines at matched compute by +23 to +43%, directly substantiating that scaling verification is more effective than scaling sampling under *deployment-time* compute, not merely training FLOPs.

(5) CoVer learns alignment, not scene classification. Because our OOD suites deliberately include hard same-scene negatives, i.e., distractors of the same category as the target (Coca-Cola vs. Redbull, zucchini vs. carrot, tennis vs. ping-pong ball), a model that merely recognized the scene or task would fail. The substantial OOD gains (+31%) therefore indicate that CoVer captures fine-grained instruction–action alignment rather than coarse scene membership.

5.5 Real-World Evaluation Results

We further evaluate CoVer-VLA in two real-world manipulation tasks as shown in Figure 9. CoVer-VLA substantially outperforms the baselines, improving the success rate by 30% and 60%, respectively. CoVer-VLA consistently shows the correct intention to accomplish the task, whereas the other baselines often fail to identify the correct object. We observe that the base π_0 model often failed to initiate motion under challenging scenes and instructions, resulting in 0% success. Overall, these results demonstrate that scaling test-time verification with CoVer provides an effective and scalable pathway toward building a robust robotics foundation model. These results highlight the critical role of test-time verification in ensuring reliable deployment of robots in real-world environments.

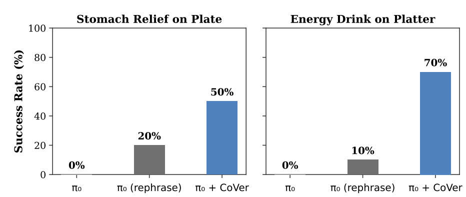

Fig. 9: Real-World Evaluation Results. π_0 + CoVer significantly outperforms the baseline π_0 (rephrase), achieving a 45% absolute improvement in task success rate over the baseline policy.

5.6 Latency Analysis and Optimizations

While our approach introduces additional computational overhead from action sampling and verification, we mitigate these costs through several key optimizations. Concretely, we decouple the image-text encoder and action encoder within our verifier architecture. This design enables the image-text embedding to be computed in parallel with the forward pass of the base robot policy. As a result, the end-to-end latency of our pipeline consists only of batched inference with π_0 (or $\pi_{0.5}$) and a lightweight action encoder from CoVer.

As shown in Table 3, the action encoder consistently adds only ~8 ms even at larger batch sizes. In addition, repeated sampling can exploit KV cache optimizations and batch processing to achieve higher throughput than greedy decoding, allowing CoVer-VLA to sample and verify 16 candidate actions in approximately 453 ms (~2.2 Hz). We also avoid online rephrase generation by shifting reasoning to boot time. Specifically, we precompute and cache a set of diverse rephrased instructions before deployment. This eliminates redundant runtime calls to the VLM, thereby minimizing inference-time latency.

We use an off-the-shelf VLM (GPT-4o) for boot-time reasoning, which requires $\sim$11 s to generate 8 rephrasings once per episode. Amortized over a 120-step episode running at 2.2 Hz (0.45 s/step), the effective control rate remains $\sim$1.85 Hz, since $\frac{11+120\times0.45}{120} \approx$ 0.54 s/step, which still exceeds competing VLA test-time scaling methods such as RoboMonkey [22] (1.5 Hz).

Batch Size	$\pi_{0.5}$ (ms)	CoVer (ms)	$\pi_{0.5}$ + CoVer (ms)
1	56	7	63
2	84	7	91
4	138	8	146
8	243	8	251
16	445	8	453
32	865	8	873

Table 3: Latency (milliseconds) across batch sizes running on RTX-5090 GPU. Since the image-text encoder can run in parallel with the forward pass of $\pi_{0.5}$, the end-to-end latency of our pipeline only consists of batched inference with $\pi_{0.5}$ and the lightweight CoVer action encoder.

6 Conclusion

In this paper, we present CoVer-VLA, a novel contrastive-based verifier and hierarchical test-time scaling framework that bridges the "intention–action gap" for generalist robot policies. CoVer-VLA achieves substantial performance improvements across both simulated and real-world settings, particularly under out-of-distribution conditions. Our findings demonstrate that allocating compute to reasoning and verification at deployment can be more effective than scaling policy training alone, providing a promising direction for robust policy deployment in the real world. CoVer-VLA operates at the action-chunk level and is platform-agnostic: it requires no access to the internal parameters of the base VLA, and the chunk size is configurable to allow finer temporal resolution for contact-rich tasks. We note that CoVer targets the intention–action gap, whereas localized intra-chunk motor deviations are an orthogonal failure mode traditionally handled by low-level dynamics verifiers [22, 28], with which our framework can naturally compose. While our study focuses on applying the verifier for test-time scaling, the same design and principle can extend beyond inference optimizations such as post-training with reinforcement learning or run-time monitoring. Future work could also explore more efficient architectures for both the base policy and verifier to further reduce latency and enable broader use of test-time scaling in real-world robotic settings.

Acknowledgements

We thank the members of the Stanford Autonomous Systems Lab, Scaling Intelligence Lab, and IRIS Lab for their constructive feedback and informative discussions. We gratefully acknowledge the support of DARPA; NASA ULI; Schmidt Sciences; Google DeepMind; Google Research; Google Cloud; SNSF; IBM and Felicis.

Bibliography

[1] Agia, C., Sinha, R., Yang, J., Cao, Z.a., Antonova, R., Pavone, M., Bohg, J.: Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In: 8th Annual Conference on Robot Learning (2024)

[2] Beyer, L., Steiner, A., Pinto, A.S., Kolesnikov, A., Wang, X., Salz, D., Neumann, M., Alabdulmohsin, I., Tschannen, M., Bugliarello, E., et al.: Paligemma: A versatile 3b vlm for transfer. arXiv preprint arXiv:2407.07726 (2024)

[3] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L.X., Tanner, J., Vuong, Q., Walling, A., Wang, H., Zhilinsky, U.: \$$\pi$_0\$: A Vision-Language-Action Flow Model for General Robot Control. arXiv preprint arXiv:2410.24164 (Nov 2024)

[4] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M.G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.W.E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., Zitkovich, B.: Rt-2: Vision-language-action models transfer web knowledge to robotic control (2023), `https://arxiv.org/abs/2307.15818`

[5] Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q.V., Ré, C., Mirhoseini, A.: Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv preprint arXiv:2407.21787 (Jul 2024)

[6] Collaboration, O.X.E., Padalkar, A., Pooley, A., Mandlekar, A., Jain, A., Tung, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Garg, A., Brohan, A., Raffin, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Ichter, B., Lu, C., Xu, C., Finn, C., Xu, C., Chi, C., Huang, C., Chan, C., Pan, C., Fu, C., Devin, C., Driess, D., Pathak, D., Shah, D., Büchler, D., Kalashnikov, D., Sadigh, D., Johns, E., Ceola, F., Xia, F., Stulp, F., Zhou, G., Sukhatme, G.S., Salhotra, G., Yan, G., Schiavi, G., Kahn, G., Su, H., Fang, H.S., Shi, H., Amor, H.B., Christensen, H.I., Furuta, H., Walke, H., Fang, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Abou-Chakra, J., Kim, J., Peters, J., Schneider, J., Hsu, J., Bohg, J., Bingham, J., Wu, J., Wu, J., Luo, J., Gu, J., Tan, J., Oh, J., Malik, J., Booher, J., Tompson, J., Yang, J., Lim, J.J., Silvério, J., Han, J., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Oslund, K., Kawaharazuka, K., Zhang, K., Rana, K., Srinivasan, K., Chen, L.Y., Pinto, L., Fei-Fei, L., Tan, L., Ott, L., Lee, L., Tomizuka, M., Spero, M., Du, M., Ahn, M., Zhang, M., Ding, M., Srirama, M.K., Sharma, M., Kim, M.J., Kanazawa, N., Hansen, N., Heess, N., Joshi, N.J., Suenderhauf, N., Di Palo, N., Shafiullah, N.M.M., Mees, O., Kroemer, O.,

Sanketi, P.R., Wohlhart, P., Xu, P., Sermanet, P., Sundaresan, P., Vuong, Q., Rafailov, R., Tian, R., Doshi, R., Martín-Martín, R., Mendonca, R., Shah, R., Hoque, R., Julian, R., Bustamante, S., Kirmani, S., Levine, S., Moore, S., Bahl, S., Dass, S., Sonawani, S., Song, S., Xu, S., Haldar, S., Adebola, S., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Dasari, S., Belkhale, S., Osa, T., Harada, T., Matsushima, T., Xiao, T., Yu, T., Ding, T., Davchev, T., Zhao, T.Z., Armstrong, T., Darrell, T., Jain, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Wang, X., Zhu, X., Li, X., Lu, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Xu, Y., Wang, Y., Bisk, Y., Cho, Y., Lee, Y., Cui, Y., Wu, Y.H., Tang, Y., Zhu, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Xu, Z., Cui, Z.J.: Open X-Embodiment: Robotic Learning Datasets and RT-X Models. arXiv preprint arXiv:2310.08864 (Dec 2023)
[7] Dong, P., Mirchandani, S., Sadigh, D., Finn, C.: What Matters for Batch Online Reinforcement Learning in Robotics? arXiv preprint (May 2025)
[8] Driess, D., Springenberg, J.T., Ichter, B., Yu, L., Li-Bell, A., Pertsch, K., Ren, A.Z., Walke, H., Vuong, Q., Shi, L.X., Levine, S.: Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (Oct 2025)
[9] Fan, C., Jia, X., Sun, Y., Wang, Y., Wei, J., Gong, Z., Zhao, X., Tomizuka, M., Yang, X., Yan, J., et al.: Interleave-vla: Enhancing robot manipulation with interleaved image-text instructions. arXiv preprint arXiv:2505.02152 (2025)
[10] Fang, I., Zhang, J., Tong, S., Feng, C.: From intention to execution: Probing the generalization boundaries of vision-language-action models. arXiv preprint arXiv:2506.09930 (2025)
[11] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
[12] Grover, S., Gopalkrishnan, A., Ai, B., Christensen, H.I., Su, H., Li, X.: Enhancing Generalization in Vision-Language-Action Models by Preserving Pretrained Representations. arXiv preprint arXiv:2509.11417 (Sep 2025)
[13] Gu, Q., Ju, Y., Sun, S., Gilitschenski, I., Nishimura, H., Itkina, M., Shkurti, F.: Safe: Multitask failure detection for vision-language-action models. arXiv preprint arXiv:2506.09937 (2025)
[14] Hancock, A.J., Wu, X., Zha, L., Russakovsky, O., Majumdar, A.: Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting (2025), https://arxiv.org/abs/2509.22195
[15] Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J.G., Levine, S.: IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies. arXiv preprint arXiv:2304.10573 (May 2023)
[16] Huang, H., Liu, F., Fu, L., Wu, T., Mukadam, M., Malik, J., Goldberg, K., Abbeel, P.: Otter: A vision-language-action model with text-aware visual feature extraction. arXiv preprint arXiv:2503.03734 (2025)
[17] Jain, A., Zhang, M., Arora, K., Chen, W., Torne, M., Irshad, M.Z., Zakharov, S., Wang, Y., Levine, S., Finn, C., et al.: Polaris: Scalable real-to-sim evaluations for generalist robot policies. arXiv preprint arXiv:2512.16881 (2025)

[18] Karnik, S., Hong, Z.W., Abhangi, N., Lin, Y.C., Wang, T.H., Dupuy, C., Gupta, R., Agrawal, P.: Embodied red teaming for auditing robotic foundation models (2025), `https://arxiv.org/abs/2411.18676`

[19] Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M.K., Chen, L.Y., Ellis, K., Fagan, P.D., Hejna, J., Itkina, M., Lepert, M., Ma, Y.J., Miller, P.T., Wu, J., Belkhale, S., Dass, S., Ha, H., Jain, A., Lee, A., Lee, Y., Memmel, M., Park, S., Radosavovic, I., Wang, K., Zhan, A., Black, K., Chi, C., Hatch, K.B., Lin, S., Lu, J., Mercat, J., Rehman, A., Sanketi, P.R., Sharma, A., Simpson, C., Vuong, Q., Walke, H.R., Wulfe, B., Xiao, T., Yang, J.H., Yavary, A., Zhao, T.Z., Agia, C., Baijal, R., Castro, M.G., Chen, D., Chen, Q., Chung, T., Drake, J., Foster, E.P., Gao, J., Herrera, D.A., Heo, M., Hsu, K., Hu, J., Jackson, D., Le, C., Li, Y., Lin, K., Lin, R., Ma, Z., Maddukuri, A., Mirchandani, S., Morton, D., Nguyen, T., O'Neill, A., Scalise, R., Seale, D., Son, V., Tian, S., Tran, E., Wang, A.E., Wu, Y., Xie, A., Yang, J., Yin, P., Zhang, Y., Bastani, O., Berseth, G., Bohg, J., Goldberg, K., Gupta, A., Gupta, A., Jayaraman, D., Lim, J.J., Malik, J., Martín-Martín, R., Ramamoorthy, S., Sadigh, D., Song, S., Wu, J., Yip, M.C., Zhu, Y., Kollar, T., Levine, S., Finn, C.: DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. arXiv preprint arXiv:2403.12945 (Mar 2024)

[20] Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C.: Openvla: An open-source vision-language-action model (2024), `https://arxiv.org/abs/2406.09246`

[21] Kim, T., Lee, J., Koo, M., Kim, D., Lee, K., Kim, C., Seo, Y., Shin, J.: Contrastive Representation Regularization for Vision-Language-Action Models. arXiv preprint (2025)

[22] Kwok, J., Agia, C., Sinha, R., Foutter, M., Li, S., Stoica, I., Mirhoseini, A., Pavone, M.: Robomonkey: Scaling test-time sampling and verification for vision-language-action models. arXiv preprint arXiv:2506.17811 (2025)

[23] Li, X., Li, P., Liu, M., Wang, D., Liu, J., Kang, B., Ma, X., Kong, T., Zhang, H., Liu, H.: Towards Generalist Robot Policies: What Matters in Building Vision-Language-Action Models. arXiv preprint arXiv:2412.14058 (Dec 2024)

[24] Li, X., Hsu, K., Gu, J., Pertsch, K., Mees, O., Walke, H.R., Fu, C., Lunawat, I., Sieh, I., Kirmani, S., Levine, S., Wu, J., Finn, C., Su, H., Vuong, Q., Xiao, T.: Evaluating real-world robot manipulation policies in simulation. arXiv preprint arXiv:2405.05941 (2024)

[25] Liu, J., Gao, F., Wei, B., Chen, X., Liao, Q., Wu, Y., Yu, C., Wang, Y.: What can rl bring to vla generalization? an empirical study. arXiv preprint arXiv:2505.19789 (2025)

[26] Liu, Y., Hamid, J.I., Xie, A., Lee, Y., Du, M., Finn, C.: Bidirectional Decoding: Improving Action Chunking via Guided Test-Time Sampling. In: The Thirteenth International Conference on Learning Representations (ICLR) (2025)

[27] Muennighoff, N., Yang, Z., Shi, W., Li, X.L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., Hashimoto, T.: S1: Simple test-time scaling. arXiv preprint arXiv:2501.19393 (Mar 2025)

[28] Nakamoto, M., Mees, O., Kumar, A., Levine, S.: Steering your generalists: Improving robotic foundation models via value guidance. Conference on Robot Learning (CoRL) (2024)

[29] NVIDIA, Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L.J., Fang, Y., Fox, D., Hu, F., Huang, S., Jang, J., Jiang, Z., Kautz, J., Kundalia, K., Lao, L., Li, Z., Lin, Z., Lin, K., Liu, G., Llontop, E., Magne, L., Mandlekar, A., Narayan, A., Nasiriany, S., Reed, S., Tan, Y.L., Wang, G., Wang, Z., Wang, J., Wang, Q., Xiang, J., Xie, Y., Xu, Y., Xu, Z., Ye, S., Yu, Z., Zhang, A., Zhang, H., Zhao, Y., Zheng, R., Zhu, Y.: GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. arXiv preprint arXiv:2503.14734 (Mar 2025)

[30] Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)

[31] Qi, H., Yin, H., Zhu, A., Du, Y., Yang, H.: Strengthening Generative Robot Policies through Predictive World Modeling. arXiv preprint arXiv:2502.00622 (May 2025)

[32] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021)

[33] Saad-Falcon, J., Lafuente, A.G., Natarajan, S., Maru, N., Todorov, H., Guha, E.K., Buchanan, E.K., Chen, M.F., Guha, N., Re, C., Mirhoseini, A.: Archon: An Architecture Search Framework for Inference-Time Techniques (Oct 2024)

[34] Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., Cadene, R.: SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. arXiv preprint arXiv:2506.01844 (Jun 2025)

[35] Snell, C., Lee, J., Xu, K., Kumar, A.: Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. arXiv preprint arXiv:2408.03314 (Aug 2024)

[36] Team, G.R., Abeyruwan, S., Ainslie, J., Alayrac, J.B., Arenas, M.G., Armstrong, T., Balakrishna, A., Baruch, R., Bauza, M., Blokzijl, M., Bohez, S., Bousmalis, K., Brohan, A., Buschmann, T., Byravan, A., Cabi, S., Caluwaerts, K., Casarini, F., Chang, O., Chen, J.E., Chen, X., Chiang, H.T.L., Choromanski, K., D'Ambrosio, D., Dasari, S., Davchev, T., Devin, C., Palo, N.D., Ding, T., Dostmohamed, A., Driess, D., Du, Y., Dwibedi, D., Elabd, M., Fantacci, C., Fong, C., Frey, E., Fu, C., Giustina, M., Gopalakrishnan, K., Graesser, L., Hasenclever, L., Heess, N., Hernaez, B., Herzog, A., Hofer, R.A., Humplik, J., Iscen, A., Jacob, M.G., Jain, D., Julian, R., Kalashnikov, D., Karagozler, M.E., Karp, S., Kew, C., Kirkland, J., Kirmani, S., Kuang, Y., Lampe, T., Laurens, A., Leal, I., Lee, A.X., Lee, T.W.E., Liang, J., Lin, Y., Maddineni, S., Majumdar, A., Michaely, A.H., Moreno, R., Neunert, M., Nori, F., Parada, C., Parisotto, E., Pastor, P., Pooley, A., Rao, K., Reymann, K., Sadigh, D., Saliceti, S., Sanketi, P., Sermanet, P., Shah, D., Sharma, M., Shea, K., Shu, C., Sindhwani, V., Singh, S., Soricut, R., Springenberg, J.T., Sterneck, R., Surdulescu, R., Tan, J., Tompson, J., Vanhoucke, V., Varley, J., Vesom, G., Vezzani, G., Vinyals, O., Wahid, A., Welker, S., Wohlhart, P., Xia, F., Xiao, T., Xie, A., Xie, J., Xu, P., Xu, S., Xu,

Y., Xu, Z., Yang, Y., Yao, R., Yaroshenko, S., Yu, W., Yuan, W., Zhang, J., Zhang, T., Zhou, A., Zhou, Y.: Gemini Robotics: Bringing AI into the Physical World. arXiv preprint arXiv:2503.20020 (Mar 2025)

[37] Tschannen, M., Gritsenko, A., Wang, X., Naeem, M.F., Alabdulmohsin, I., Parthasarathy, N., Evans, T., Beyer, L., Xia, Y., Mustafa, B., et al.: Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786 (2025)

[38] Wagenmaker, A., Nakamoto, M., Zhang, Y., Park, S., Yagoub, W., Nagabandi, A., Gupta, A., Levine, S.: Steering Your Diffusion Policy with Latent Space Reinforcement Learning. arXiv preprint arXiv:2506.15799 (Jun 2025)

[39] Walke, H.R., Black, K., Zhao, T.Z., Vuong, Q., Zheng, C., Hansen-Estruch, P., He, A.W., Myers, V., Kim, M.J., Du, M., et al.: Bridgedata v2: A dataset for robot learning at scale. In: Conference on Robot Learning. pp. 1723–1736. PMLR (2023)

[40] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully Test-Time Adaptation by Entropy Minimization. In: International Conference on Learning Representations (Sep 2020)

[41] Wu, Y., Tian, R., Swamy, G., Bajcsy, A.: From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. In: Robotics: Science and Systems (RSS) (2025)

[42] Xu, C., Nguyen, T.K., Dixon, E., Rodriguez, C., Miller, P., Lee, R., Shah, P., Ambrus, R., Nishimura, H., Itkina, M.: Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. arXiv preprint arXiv:2503.08558 (2025)

[43] Yang, S., Li, H., Chen, Y., Wang, B., Tian, Y., Wang, T., Wang, H., Zhao, F., Liao, Y., Pang, J.: InstructVLA: Vision-Language-Action Instruction Tuning from Understanding to Manipulation. arXiv preprint (Jul 2025)

[44] Zhang, S., et al.: VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2025)

[45] Zhang, X., Lin, H., Ye, H., Zou, J., Ma, J., Liang, Y., Du, Y.: Inference-time Scaling of Diffusion Models through Classical Search. arXiv preprint arXiv:2505.23614 (May 2025)

[46] Zhang, Y., Wang, C., Lu, O., Zhao, Y., Ge, Y., Sun, Z., Li, X., Zhang, C., Bai, C., Li, X.: Align-Then-stEer: Adapting the Vision-Language Action Models through Unified Latent Guidance. arXiv preprint (2025)