ProGAL-VLA: Grounded Alignment through Prospective Reasoning in Vision-Language-Action Models

Supplementary Material

1. Theoretical Derivations and Proofs

This section expands the theoretical analysis from Section 4 and provides fully detailed derivations for the results used in the main paper.

1.1. Proof of Proposition 1 (Language Influence)

Proposition 1. Under the Verification Bottleneck assumption,

$$I(L; a_t \mid O_t, q_t) = I(L; g_t \mid O_t, q_t) - I(L; g_t \mid a_t, O_t, q_t). \tag{1}$$

Proof.

We first introduce a short notation to simplify expressions:

$$C_t = (O_t, q_t),$$

which denotes the complete perceptual and robot-state context available at time t.

Our goal is to relate the conditional mutual information between the language instruction L and the action a_t to the mutual information involving the verified goal g_t.

Consider the joint quantity $I(L; a_t, g_t \mid C_t)$. The chain rule for mutual information allows two valid decompositions:

$$I(L; a_t, g_t \mid C_t) = I(L; g_t \mid C_t) + I(L; a_t \mid g_t, C_t), \tag{2}$$

$$I(L; a_t, g_t \mid C_t) = I(L; a_t \mid C_t) + I(L; g_t \mid a_t, C_t). \tag{3}$$

These two expressions describe the same joint mutual information, but factor it in different orders.

The Verification Bottleneck states that:

$$a_t \perp L \mid (g_t, C_t).$$

This means: once the verified goal g_t is known (and the context C_t is fixed), the action no longer depends on the original instruction L. This architectural constraint is enforced because the fast controller receives only the verified goal g_t and does not have direct access to L.

In information-theoretic terms, this conditional independence gives:

$$I(L; a_t \mid g_t, C_t) = 0. \tag{4}$$

Replacing the last term in Eq. (2) using Eq. (4), we obtain:

$$I(L; a_t, g_t \mid C_t) = I(L; g_t \mid C_t).$$

Both chain-rule decompositions describe the same quantity, so we set Eq. (2) equal to Eq. (3):

$$I(L; g_t \mid C_t) = I(L; a_t \mid C_t) + I(L; g_t \mid a_t, C_t).$$

Rearranging the above expression gives:

$$I(L; a_t \mid C_t) = I(L; g_t \mid C_t) - I(L; g_t \mid a_t, C_t),$$

which is exactly the desired identity.

Finally, substituting back $C_t = (O_t, q_t)$ completes the proof.

1.2. Proof of Theorem 1 (GAC InfoNCE Bound)

Theorem 1 (InfoNCE style lower bound). Assume that, for each symbolic sub goal s, one positive grounded entity e^+ is drawn from $p(e \mid s)$ and $N-1$ negatives $\{e_i\}_{i \neq +}$ are drawn independently from the marginal $p(e)$. Let $\mathcal{L}_{GAC}$ be the contrastive loss

$$\mathcal{L}_{GAC} = -\mathbb{E}\left[\log \frac{\exp(f(s, e^+))}{\sum_{i=1}^{N} \exp(f(s, e_i))}\right], \tag{5}$$

where $f(s, e)$ is a critic function. Then

$$I(S; E) \;\geq\; \log N \;-\; \mathbb{E}[\mathcal{L}_{GAC}]. \tag{6}$$

Proof. We follow the standard construction used for InfoNCE. Define a joint distribution over $(S, E_1, \ldots, E_N, I)$ as

$$p(s, e_1, \ldots, e_N, i) = \frac{1}{N}\, p(s)\, p(e_i \mid s) \prod_{j \neq i} p(e_j), \tag{7}$$

where $I \in \{1, \ldots, N\}$ is the index of the positive example. Under this construction, the conditional posterior of I given $(s, e_1, \ldots, e_N)$ is

$$p(i \mid s, e_1, \ldots, e_N) = \frac{p(e_i \mid s)/p(e_i)}{\sum_{j=1}^{N} p(e_j \mid s)/p(e_j)} \tag{8}$$

$$= \frac{\exp\big(\log p(e_i \mid s) - \log p(e_i)\big)}{\sum_{j=1}^{N} \exp\big(\log p(e_j \mid s) - \log p(e_j)\big)}. \tag{9}$$

Given a critic $f(s, e)$, we define the model posterior

$$p_f(i \mid s, e_1, \ldots, e_N) = \frac{\exp(f(s, e_i))}{\sum_{j=1}^{N} \exp(f(s, e_j))}. \quad (10)$$

The GAC loss can then be written as the expected negative log likelihood of the true index I under this model:

$$\mathcal{L}_{GAC} = \mathbb{E}\big[-\log p_f(I \mid S, E_1, \ldots, E_N)\big]. \quad (11)$$

Now consider the Kullback–Leibler divergence between the true posterior and the model posterior, conditioned on $(S, E_1, \ldots, E_N)$:

$$\mathrm{KL}\big(p(\cdot \mid s, e_{1:N}) \,\|\, p_f(\cdot \mid s, e_{1:N})\big) \;\geq\; 0. \quad (12)$$

Taking expectation over $(S, E_1, \ldots, E_N)$ and expanding gives

$$0 \leq \mathbb{E}\Big[\mathrm{KL}\big(p(\cdot \mid S, E_{1:N}) \,\|\, p_f(\cdot \mid S, E_{1:N})\big)\Big] \quad (13)$$

$$= \mathbb{E}\big[-\log p_f(I \mid S, E_{1:N})\big] - \mathbb{E}\big[-\log p(I \mid S, E_{1:N})\big] \quad (14)$$

$$= \mathbb{E}[\mathcal{L}_{GAC}] - \mathbb{E}\big[-\log p(I \mid S, E_{1:N})\big], \quad (15)$$

where we used Eq. (11) in the last line.

Rearranging yields

$$\mathbb{E}[\mathcal{L}_{GAC}] \;\geq\; \mathbb{E}\big[-\log p(I \mid S, E_{1:N})\big]. \quad (16)$$

The right-hand side can be evaluated in closed form for the joint in Eq. (7) (see for example Poole et al., 2019). One obtains

$$\mathbb{E}\big[-\log p(I \mid S, E_{1:N})\big] = \log N - I(S; E), \quad (17)$$

which substituted into Eq. (16) gives

$$\mathbb{E}[\mathcal{L}_{GAC}] \;\geq\; \log N - I(S; E). \quad (18)$$

Rearranging terms finally yields

$$I(S; E) \;\geq\; \log N - \mathbb{E}[\mathcal{L}_{GAC}], \quad (19)$$

That is, the GAC objective defines a variational lower bound on the mutual information between symbolic sub-goals and grounded entities.

2. Model Details

This section summarizes the architectures of all policies evaluated on LIBERO-Plus [4]. We focus on the high-level design of the backbones, modality encoders, and action parameterizations.

OpenVLA and OpenVLA-OFT Family: OpenVLA [6] is built around the Prismatic-7B vision–language backbone. Visual observations are encoded by a dual-stream vision module (SigLIP and DINOv2) whose feature maps are concatenated and passed through a lightweight MLP projector into the token space of a LLaMA-2–style language model. Text and image tokens are fused via cross-attention, and continuous control commands are discretized into 256 bins per dimension and represented as dedicated action tokens in the LLM vocabulary, enabling the policy to operate purely in token space.

The OpenVLA-OFT variants [7] share the same multimodal backbone but differ in the action head and input configuration. OpenVLA-OFT replaces the discrete-token head with a continuous MLP head that predicts all action dimensions in parallel and adds FiLM-style conditioning layers to strengthen language grounding. OpenVLA-OFT_w removes the wrist-camera stream and conditions only on the third-person view, while OpenVLA-OFT_m is used as a single multi-suite policy that shares parameters across all LIBERO suites. OpenVLA-OFT+ [7] further refines this family with an enhanced action head and improved conditioning, while keeping the same core VLM architecture.

π_0 and π_0-Fast: The π_0 architecture [1] follows a Transfusion-style design that couples a large multimodal backbone with an action-specialized transformer. A PaliGemma-based VLM encodes multiple RGB views, natural-language instructions, and proprioceptive state q_t into a token sequence; a subset of these tokens is routed to a smaller "action expert" transformer that is responsible for predicting future actions, while the VLM maintains semantic and visual understanding.

π_0-Fast [8] preserves the same backbone but changes the action representation. Instead of working in the raw time domain, trajectories are transformed into a sparse frequency-domain representation using a discrete cosine transform and then compressed with a BPE-style tokenizer (FAST). This yields shorter discrete action sequences while retaining high-fidelity control.

NORA: NORA [5] is a 3B-parameter VLA model built on the Qwen-2.5-VL-3B multimodal backbone. It processes a natural-language instruction together with a single RGB observation and outputs discrete action tokens using the FAST+ tokenizer, which discretizes continuous actions into a compact token space. The design targets a balance between strong visual–semantic reasoning and practical deployment

on moderate hardware.

WorldVLA: WorldVLA [3] unifies policy and world modeling in a single autoregressive transformer initialized from Chameleon. All modalities are tokenized: images via a VQ-GAN codebook, text via a BPE tokenizer, and robot actions via an action tokenizer that maps each control dimension to 256 discrete bins. These tokens are interleaved into one sequence and modeled jointly. A custom attention mask prevents the current action tokens from attending to previous actions, encouraging stable parallel prediction of action "chunks" while still conditioning on the full history of images and text.

UniVLA: UniVLA [2] aims for cross-embodiment generalization by operating in a discrete latent action space. The model uses the same Prismatic-7B VLM stack (SigLIP + DINOv2 encoders with a LLaMA-2 backbone) and extends the LLM vocabulary with latent action tokens drawn from a learned codebook. At deployment time, the policy autoregressively predicts sequences of latent tokens from visual observations and instructions. A lightweight robot-specific decoder head, implemented with multi-head attention over the latent token sequence, maps these latents into low-level control commands. UniVLA also supports history-augmented inputs where previously executed latent actions are fed back as context, which is particularly beneficial for long-horizon tasks.

RIPT-VLA: RIPT-VLA [9] starts from the continuous-action OpenVLA-OFT policy [7] and augments it with a distributional action head. The original MLP head produces the mean μ_θ of a factorized Laplace distribution over actions, and an additional lightweight head predicts the corresponding scale parameters σ_θ. This yields a stochastic policy with closed-form log-likelihoods, enabling sampling-based control while retaining the benefits of the OpenVLA-OFT backbone.

3. Implementation Details

3.1. Architecture Hyperparameters

We summarize the instantiated components of ProGAL-VLA. During inference, the total parameter count is dominated by the OpenVLA backbone; the prospective planner π_{slow} operates asynchronously and does not affect control-time latency.

4. Extended Experimental Results

We provide detailed breakdowns that complement the aggregate results in the main text and expose specific robustness properties and failure behaviors of ProGAL-VLA.

4.1. Granular Robustness Analysis

Table 2 decomposes failures according to the underlying perturbation type. This isolates whether degradation is caused by robot calibration errors, sensor noise, or geometric changes in viewpoint and field-of-view.

Robot Perturbations: Calibration Drift vs. High-Frequency Noise.

- **Joint Drift (systematic offset).** OpenVLA collapses to 2.1% success under drift, indicating heavy reliance on precise proprioceptive–visual alignment. ProGAL-VLA retains 68.2% success, demonstrating that the verified goal embedding g_t serves as a stable 3D anchor that is independent of small calibration errors.
- **Joint Jitter (sensor noise).** ProGAL-VLA reaches 74.8% success, substantially outperforming both baselines. The GSM's temporal 3D graph filters out high-frequency noise that destabilizes conventional frame-conditioned policies.

Camera Perturbations: Geometric Invariance.

- **Viewpoint Shift.** OpenVLA degrades to 1.2% under moderate viewpoint changes, confirming a reliance on 2D pixel coordinates. ProGAL-VLA maintains 85.1% success, showing robustness to geometric camera motion due to reasoning in 3D entity space.
- **Field-of-View Variation.** ProGAL-VLA achieves 91.7% robustness under FOV changes (zoom in/out), indicating scale invariance in grounding and action generation.

4.2. Failure Mode Analysis

Table 3 dissects failure modes to quantify the "language-ignorance" issue and show how the Verification Bottleneck eliminates it.

Grounding Errors (Language Ignorance).

A grounding failure occurs when the agent interacts with a geometrically plausible but semantically incorrect object.

- OpenVLA exhibits 41.2% grounding failures, demonstrating strong bias toward visual shortcuts rather than instruction semantics.
- ProGAL-VLA reduces this failure type to 6.3%, confirming that symbolic-to-entity verification significantly improves semantic fidelity.

Ablation: Role of Contrastive Alignment.

The model without the GAC loss, `ProGAL (w/o` $\mathcal{L}_{GAC}$`)`, shows that architecture alone does not

Table 1. Hyperparameters for ProGAL-VLA Components.

Component	**Specification**
Prospective Planner (π_{slow})	
Model Backbone	Qwen-2.5-VL-Instruct-7B
Input Resolution	448×448
Output Format	Symbolic template (e.g., `grasp_green_mug`)
Invocation Frequency	Once per episode (or when grounding fails)
Grounded State Module (GSM)	
Object Detector	YOLO-World (L-sized pretrained model)
Depth Estimator	Metric3D (v2, ViT-L backbone)
Graph Memory Size	$N_{\max} = 16$ entity nodes (FIFO update)
Entity Embedding Dim.	$d = 1024$
Action Policy (π_{fast})	
Policy Backbone	OpenVLA-7B (Llama-2 based)
Semantic Input	Verified goal embedding g_t projected to 4096-d
Action Space	7-DoF end-effector motion + binary gripper command
Training Configuration	
Optimizer	AdamW
Learning Rate	2×10^{-5} with linear warmup
Batch Size	128
GAC Loss Weight λ	0.1
Temperature τ	0.07
GPUs	$4 \times$ NVIDIA A6000

Table 2. Robustness Breakdown (Success Rate %). "Drift" refers to systematic joint offsets; "Jitter" refers to high-frequency noise.

Model	**Robot Perturbation**		**Camera Perturbation**	
	Joint Drift	*Joint Jitter*	*Viewpoint Shift*	*FOV Change*
OpenVLA [6]	2.1	4.9	1.2	0.4
OpenVLA-OFT+ [7]	28.5	35.1	68.4	78.0
ProGAL-VLA (Ours)	**68.2**	**74.8**	**85.1**	**91.7**

guarantee correct grounding:

- Removing $\mathcal{L}_{GAC}$ increases grounding failures from 6.3% to 22.4%.
- This demonstrates that maximizing $I(S; E)$ is essential for binding symbolic sub-goals to the correct 3D entities.

Shift in Error Distribution.

The baseline is dominated by cognitive grounding failures, whereas ProGAL-VLA's remaining errors are primarily due to physical execution (grasping). This indicates that the Verification Bottleneck effectively resolves the core semantic alignment problem.

5. Formal Specification of the Verification Bottleneck

For completeness, we formalize the architectural constraint referred to as the Verification Bottleneck in the main paper. At each timestep t, let the prospective planner π_{slow} output a symbolic sub-goal s_t from the language-vision model. The Grounded State Module (GSM) maps the observation O_t into a set of perceptual entities $\{e_t^{(1)}, \ldots, e_t^{(K_t)}\}$ with associated 3D attributes.

The verification function

$$\phi(s_t, \{e_t^{(i)}\}) \to g_t \in \{e_t^{(1)}, \ldots, e_t^{(K_t)}\}$$

selects a single grounded entity consistent with the

Table 3. Failure Mode Distribution (Lower is better). Values are percentages of total episodes.

Model	Grounding Fail	Grasp Fail	Planning Fail	Success Rate
OpenVLA [6]	41.2%	12.5%	5.1%	41.2%
ProGAL (w/o $\mathcal{L}_{GAC}$)	22.4%	14.1%	4.8%	58.7%
ProGAL-VLA	**6.3%**	**7.1%**	**4.5%**	**82.1%**

symbolic sub-goal. The action policy π_{fast} receives only the verified entity g_t and does not have access to the instruction L or the symbolic template s_t.

This induces the conditional-independence constraint, $a_t \perp L \mid (g_t, O_t, q_t)$, which is the Verification Bottleneck.

This section clarifies that the bottleneck is architectural, not statistical: π_{fast} cannot bypass the grounding step, ensuring that semantic alignment is enforced structurally rather than implicitly.

6. Entity Memory Update Mechanism in GSM

The Grounded State Module maintains a bounded entity memory $M_t = \{e_t^{(1)}, \ldots, e_t^{(K_t)}\}$ with capacity $N_{\max} = 16$.

Given a new observation O_t, YOLO-World provides 2D detections and Metric3D provides depth estimates. Each detection is converted into a 3D entity embedding with appearance, geometry, and positional attributes.

The update procedure is:

1. Extract candidate entities from the current frame.
2. If $|M_t| < N_{\max}$, append all entities directly.
3. If memory is full, remove the oldest entries (FIFO) and insert new ones.
4. The resulting memory M_t forms the node set of the temporal 3D entity graph.

This memory is not used for long-horizon temporal reasoning; instead, it provides short-range stability and allows π_{fast} to operate on a temporally smoothed representation that suppresses frame-level noise.

7. Verified Goal Conditioning in π_{fast}

The action policy π_{fast} conditions exclusively on the verified goal g_t produced by the GSM and does not receive direct language input. Let $h(g_t)$ be the learned projection of the grounded entity embedding into a 4096-dimensional vector. The policy input at timestep t is, $x_t = h(g_t)$, and the control distribution is, $a_t \sim \pi_{fast}(x_t)$. Because x_t is derived solely from g_t, any semantic interpretation of the language instruction must flow through the verification step. This section clarifies how the architectural routing enforces the independence relation used in Proposition 1.

8. Symbolic Template Resolution

The prospective planner π_{slow} produces symbolic templates such as `grasp_green_mug`. These templates are not executed directly; instead, they index the grounding step.

Given a template s_t, a corresponding attribute filter is applied to the entities in memory:

$$\Gamma(s_t) = \{e \in M_t : e \text{ matches attributes in } s_t\}.$$

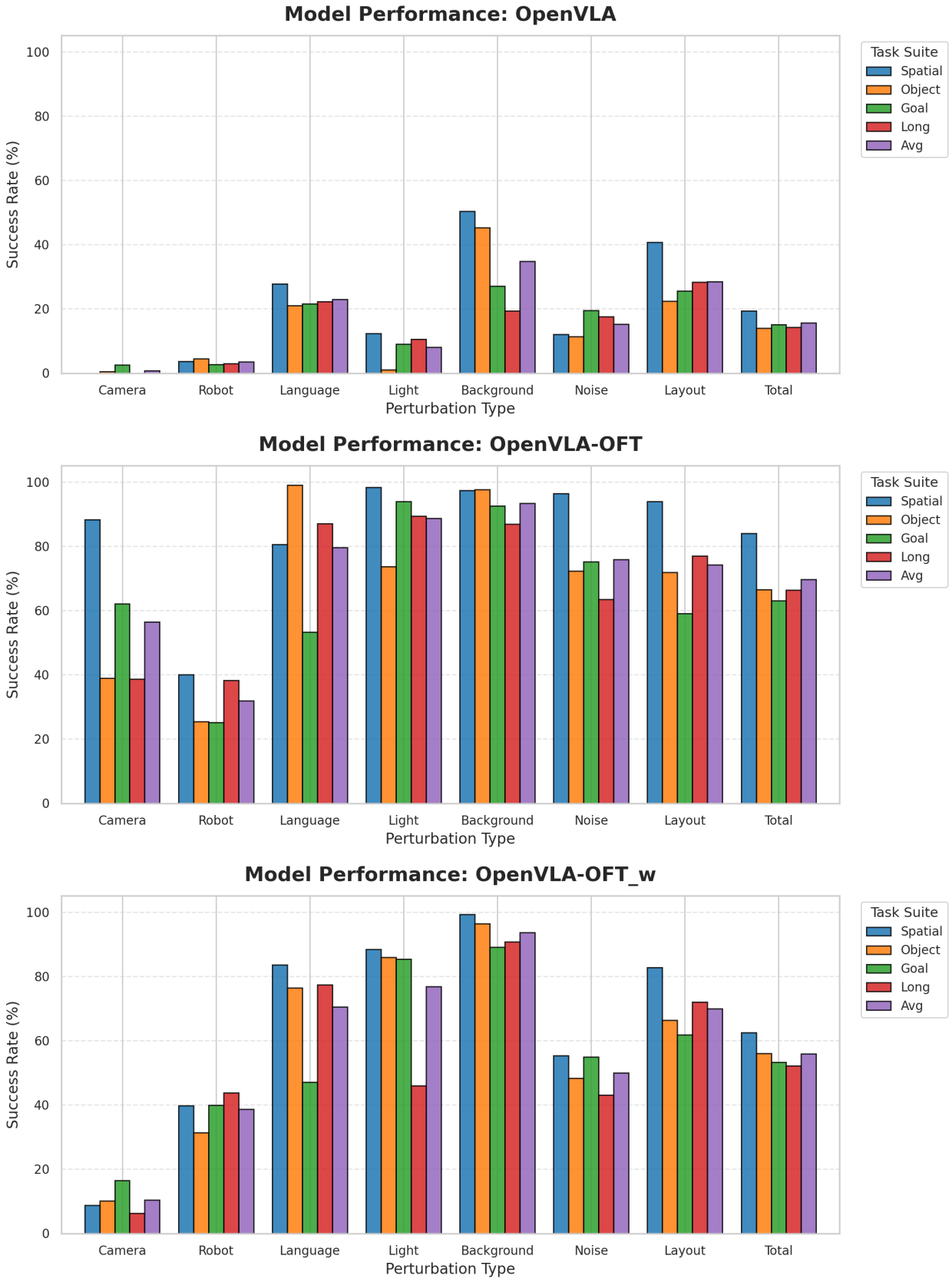
If multiple entities satisfy the template, a simple nearest-in-view heuristic (selecting the entity with largest detection confidence) is used.

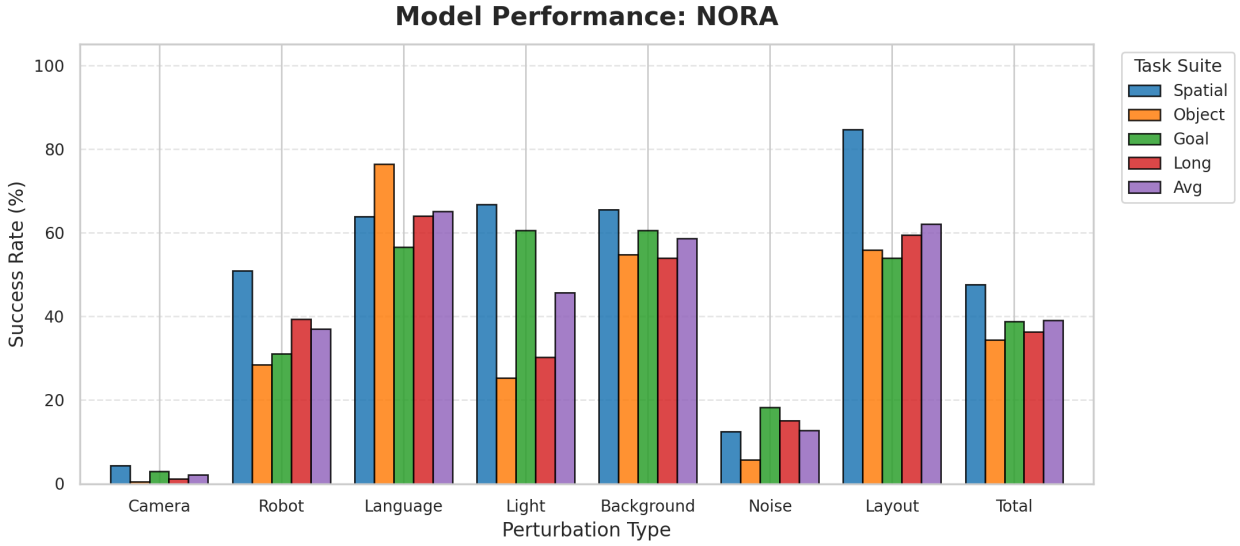
The verified goal is, $g_t = \arg\max_{e \in \Gamma(s_t)} \text{conf}(e)$. This template-resolution stage provides the symbolic-to-perceptual bridge that links π_{slow} and π_{fast} without exposing language features to the control policy.

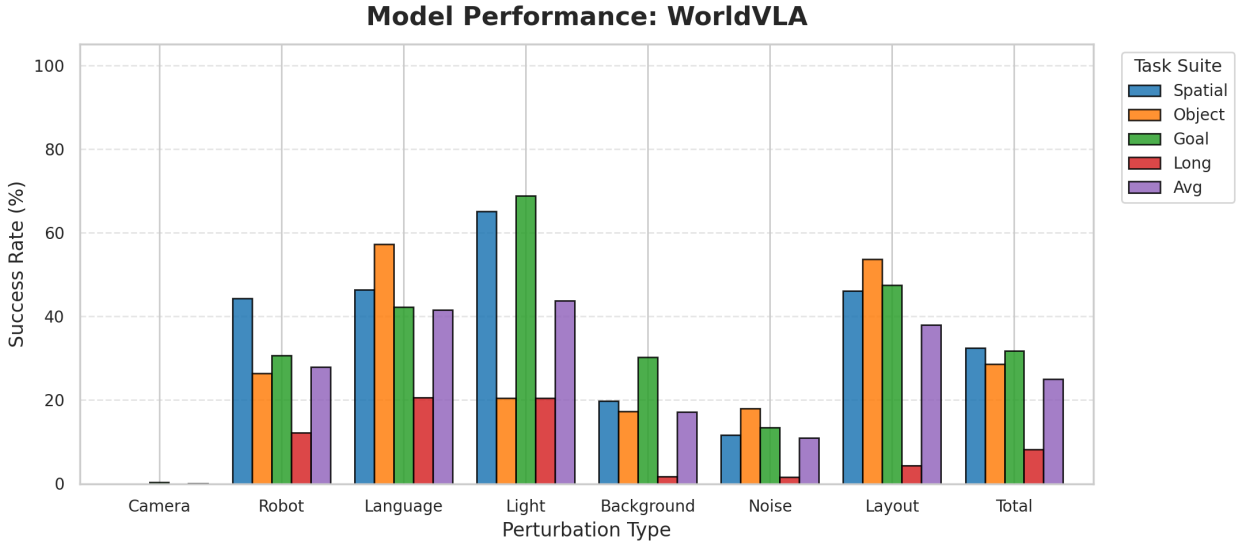
9. Detailed results of LIBERO-Plus

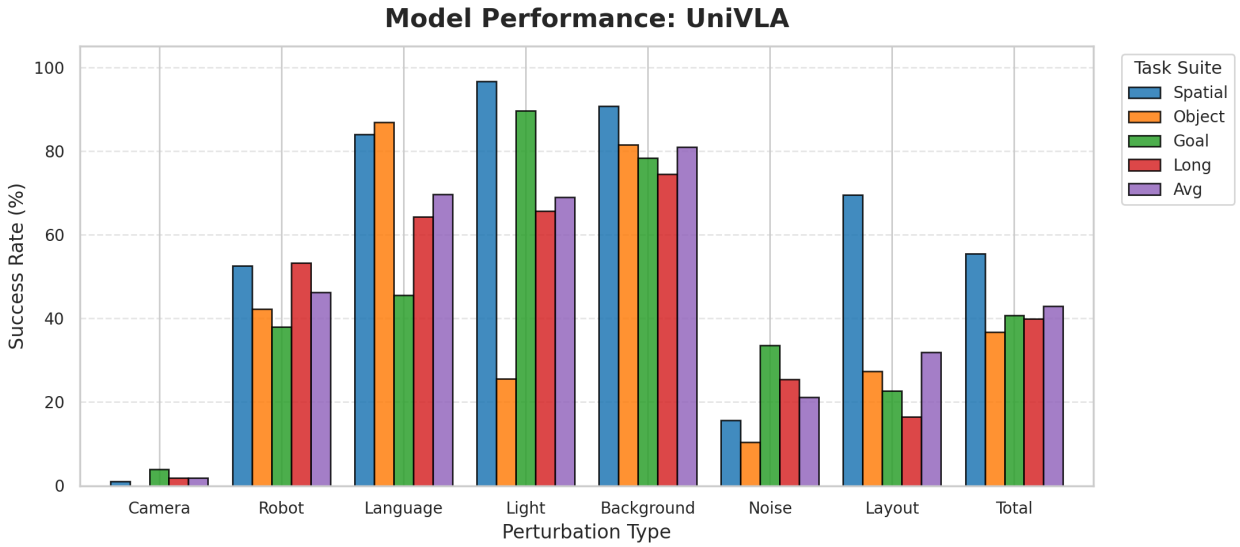
This section provides a detailed characterization of generalization under distribution shifts in the LIBERO-Plus benchmark. Figure 1 reports success rates under seven perturbation dimensions—Camera, Robot Initialization, Language Instruction, Lighting, Background, Sensor Noise, and Scene Layout—for a wide range of VLA policies, with each bar further decomposed by task suite (Spatial, Object, Goal, and Long-horizon). This factorized view exposes distinct robustness profiles: some methods are relatively stable to robot and layout shifts but degrade sharply under camera or language perturbations, while others show the opposite trend. Our ProGAL-VLA model achieves consistently strong performance across nearly all perturbation types and suites, indicating not only higher average success rates but also more uniform behavior under diverse sources of ambiguity and visual variation.

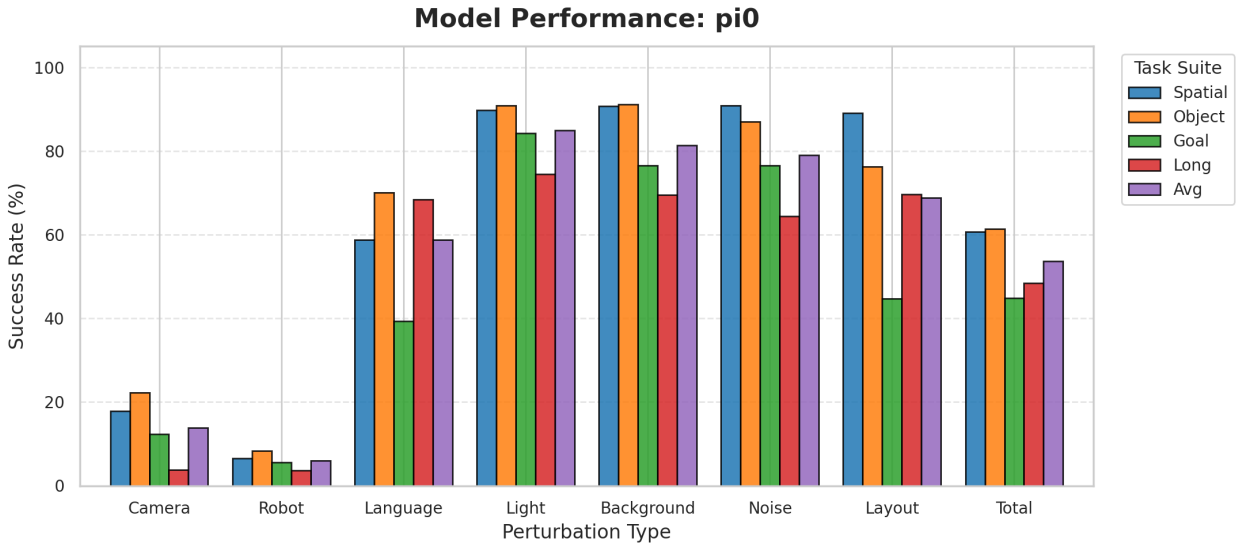
Figure 1. **Fine-grained Robustness Analysis on LIBERO-Plus.** We visualize the success rates across seven perturbation dimensions (x-axis) for four task suites (Spatial, Object, Goal, Long). While baselines such as OpenVLA and π_0 exhibit significant performance degradation under *Camera* and *Robot* perturbations, **ProGAL-VLA** maintains high robustness across most dimensions.

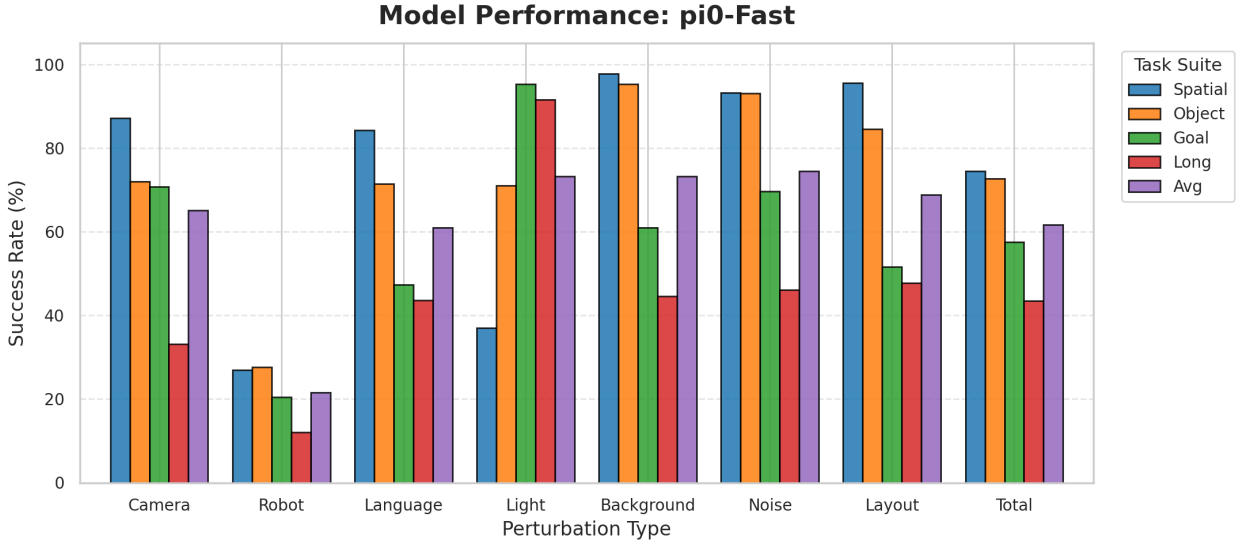


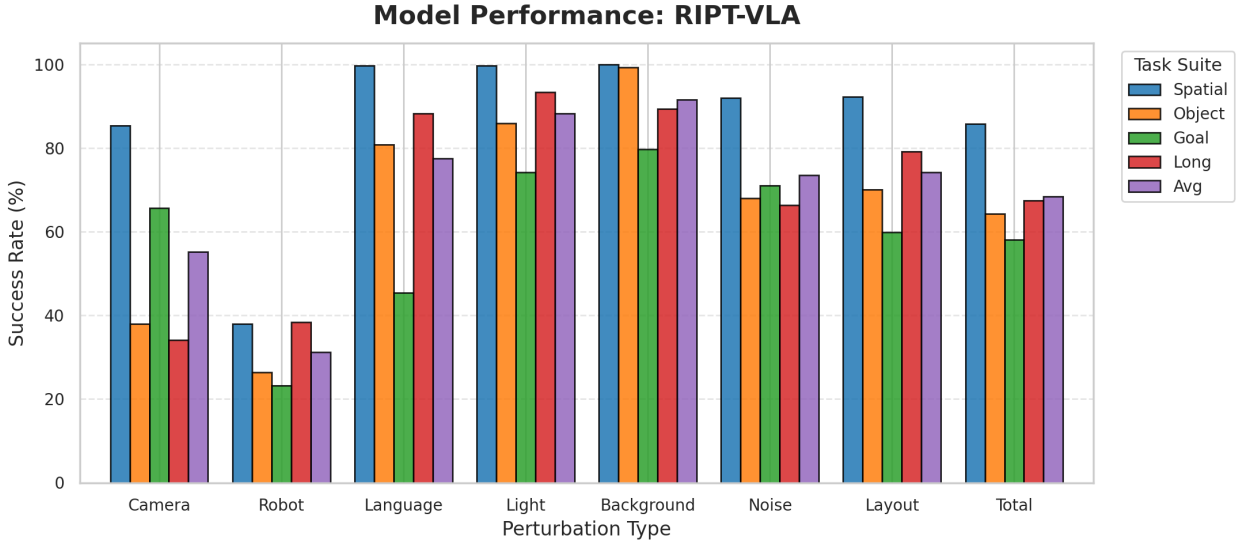

Model Performance: NORA
Success Rate (%)
Perturbation Type
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Task Suite
Spatial
Object
Goal
Long
Avg

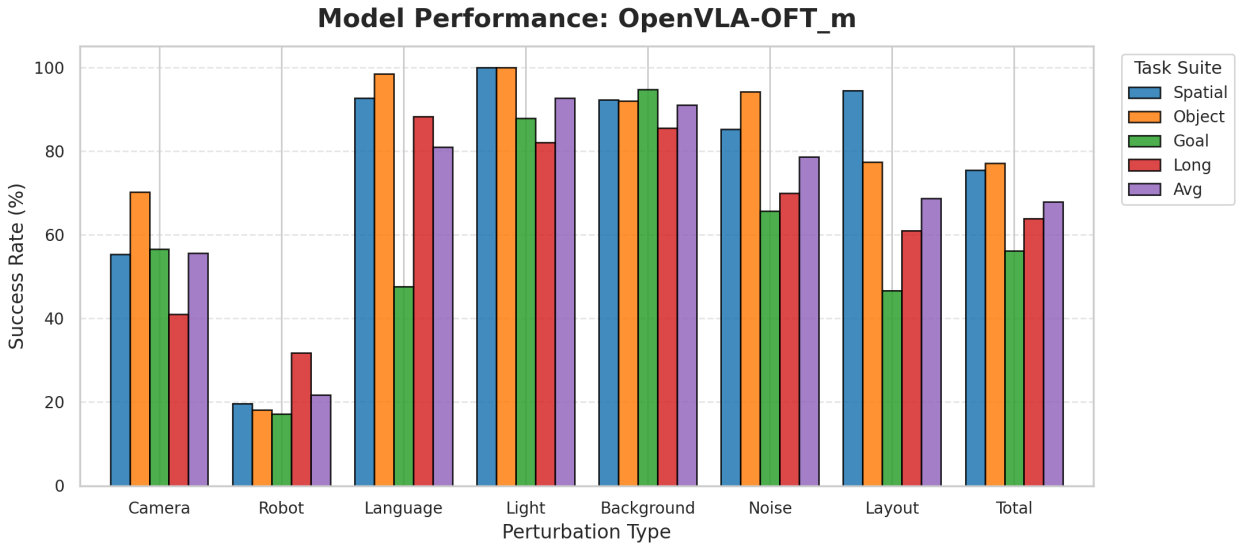

Model Performance: WorldVLA
Success Rate (%)
Perturbation Type
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Task Suite
Spatial
Object
Goal
Long
Avg

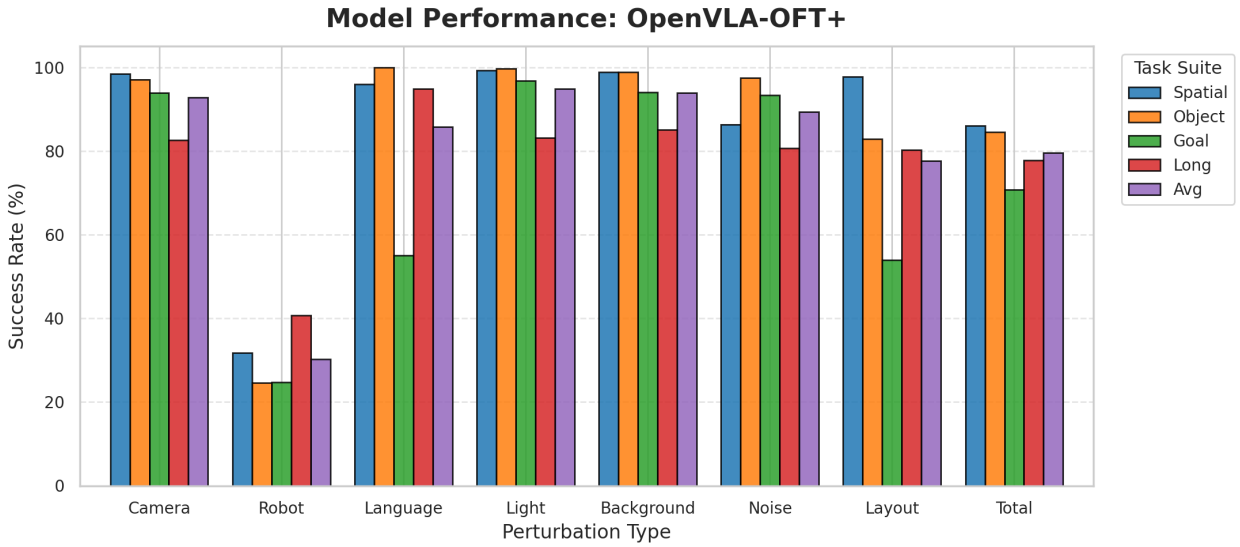

Model Performance: UniVLA
Success Rate (%)
Perturbation Type
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Task Suite
Spatial
Object
Goal
Long
Avg

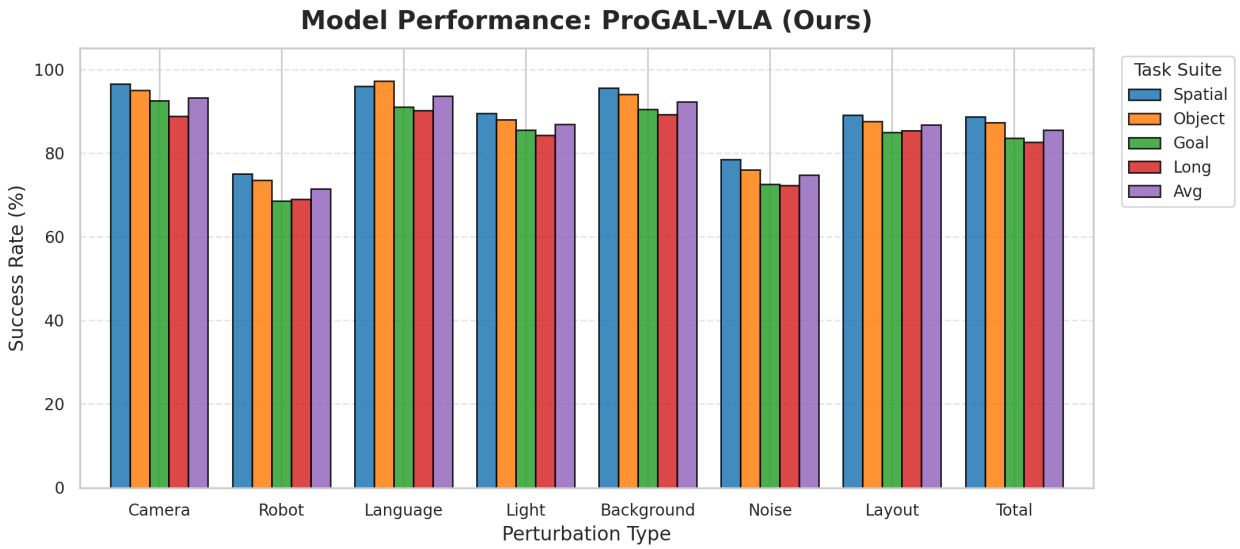

Model Performance: pi0
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg


Model Performance: pi0-Fast
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg


Model Performance: RIPT-VLA
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg


Model Performance: OpenVLA-OFT_m
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg


Model Performance: OpenVLA-OFT+
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg


Model Performance: ProGAL-VLA (Ours)
Success Rate (%)
0
20
40
60
80
100
Camera
Robot
Language
Light
Background
Noise
Layout
Total
Perturbation Type
Task Suite
Spatial
Object
Goal
Long
Avg

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*. 3

[2] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025. 4

[3] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025. 4

[4] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. 3

[5] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. Nora: A small open-sourced generalist vision language action model for embodied tasks, 2025. 3

[6] 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. 3, 5, 6

[7] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 3, 4, 5

[8] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 3

[9] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models, 2025. 4