# Flatness Preserves Instruction Following in Vision-Language-Action Models

**Anonymous Author(s)**
Affiliation
Address
email

**Abstract:** Vision-language-action (VLA) models have the potential for open-world generalization by leveraging pretrained vision-language representations, yet downstream finetuning on limited robot data often degrades these representations, leading to brittle policies that ignore language instructions in favor of visual shortcuts, a failure mode we term *instruction blindness*. We hypothesize that standard finetuning with limited data applies gradients to a sparse set of points, which manifests as a sharp loss landscape with high-curvature minima. We propose to address this directly through flatness-preserving optimization while finetuning on the exact same data, where learning a flatter landscape results in a model more robust to perturbations in the weight space. Specifically, we demonstrate that simply applying sharpness-aware minimization during VLA finetuning significantly improves instruction following by over 60% across multiple simulation and real-world benchmarks without additional data, architectural modification, or retraining. We further analyze the effect of selective sharpness, quantify its effects, and show that our approach is complementary to existing guidance techniques.

**Keywords:** Vision-Language-Action, Pretrained Representations, Regularization

## 1 Introduction

There is substantial debate in the robotics community as to how vision-language-action (VLA) models should leverage pretrained vision-language model (VLM) backbones [1, 2, 3]. For VLMs, internet-scale pretraining provides good coverage of the latent space for subsequent test samples. Leveraging these representations in VLMs provides the potential for robot policies to possess open-world generalization only if they can be transferred to robot action. However, numerous works have empirically verified the degradation of pretrained representations and extreme brittleness of VLA models to small input perturbations, especially when fine-tuned on low-data regimes [4, 5, 6, 7, 8, 6, 7, 9, 10, 11]. One significant, and until recently, underexplored failure mode is *instruction blindness* [8, 12], where the model disregards the language instruction due to vision shortcuts learned during finetuning. For example, in Fig. 1, given a scene trained along with the instruction "pick up the cream cheese", the model will disregard any new instructions such as "pick up the butter"and blindly carry out the training task [12, 10, 13, 3]. To mitigate this, existing methods often use new pre-training methods along with data augmentation [4, 14, 13, 15], guidance techniques [9, 16], or combinations of these [12] to preserve pretrained vision-language knowledge. We hypothesize that while such techniques are valid and indeed improve performance, they may simply be trying to remedy a more fundamental issue with regards to representation loss in the learned manifold caused by a mismatch between the scale and diversity of the original pre-training data and the limited domain-specific finetuning data.

Submitted to the 10th Conference on Robot Learning (CoRL 2026). Do not distribute.

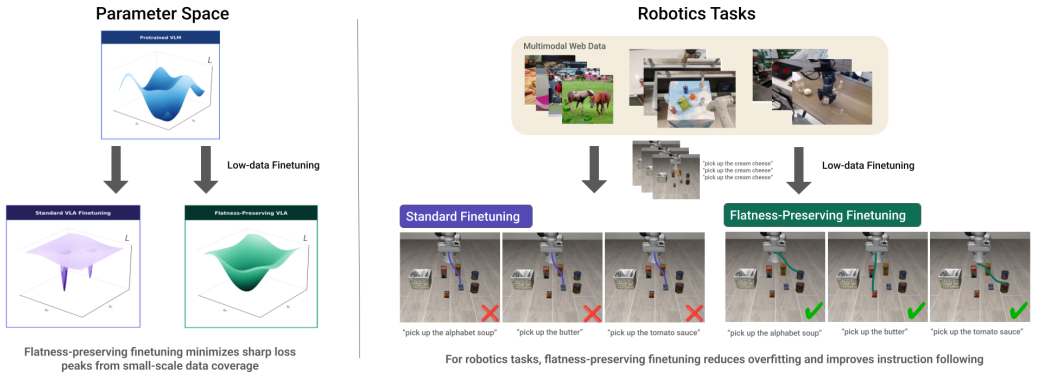


Figure 1: Left: Finetuning pretrained VLMs with limited data coverage results in sharp, narrow minima from overfitting on sparse data points (purple), while flatness-preserving optimization learns more stable minima (green). Right: Robot policies finetuned with limited deployment data exhibit instruction blindness, while flatness-preserving optimization with the same data mitigates this.

During VLM pretraining, internet-scale data provides dense coverage of the vision-language space such that for any new test sample, there is high probability that a nearby embedding was seen during training. Prior to deployment, VLA models are usually finetuned on a significantly smaller in-domain robot dataset, where data imbalance is common: language instructions are paired with multiple demonstrations, resulting in far less diversity. When finetuning, gradients are applied only to this sparse set of demonstration samples which easily causes overfitting. At the embedding level, smoothness is lost as the vast majority of the embedding space, including neighboring regions, have no guarantee of mapping to nearby representations. At the parameter level, we hypothesize that over-indexing on few samples produces a *sharp loss landscape* where the model converges to high-curvature minima and resulting policies are highly brittle to new inputs.

We therefore propose to explicitly enforce smoothness during finetuning to improve model generalization and language-following abilities, by leveraging *flatness-preserving optimization methods* widely used and shown to improve generalization in the machine learning (ML) and natural language processing (NLP) communities [17, 18, 19, 20]. Sharpness-aware minimization (SAM) is a form of bilevel optimization that enforces the model towards weight spaces in uniformly flatter regions by minimizing the worst-case loss within a region around the current parameters at each step. By enforcing flatter loss basins, the learned embedding function cannot be too sensitive to similar inputs in the neighborhood of seen data, since both would manifest as loss instability

Through both simulation experiments across the LIBERO-Pro Task, LIBERO-CF, and LangGap bencharks and real-world experiments, **we show that by simply applying SAM during the finetuning phase on the exact same in-domain data, the robustness and instruction-following ability of the VLA model dramatically improves on counterfactual instructions** by 60.2%, 70.2%, and 217% on LIBERO-PRO Task, LangGap, and LIBERO-CF respectively. Additionally, we argue that this method does not render other approaches obsolete, but instead provides a substantially higher baseline for the community to leverage in a complementary way. Importantly, we show that performance improves despite finetuning on the same limited and heavily biased training data. We then explore component-based application of SAM, quantify the sharpness, and analyze the effects of SAM optimization used in combination with other techniques. To the best of our knowledge, this is the first work to explore flatness-preservation techniques to large-scale robot policies.

## 2 Related Work

**Preserving Pretrained Representations in VLAs.** Existing works retain pretrained representations with various data, architecture, training, or inference techniques. LoRA [21] constrains finetuning to a lower-rank subspace of the weight space while [15] represents robot actions with language. Other methods [4, 14] leverage co-training data and architectural changes. Existing approaches thus

often relies on costly data collection, full re-training, or complex architectural changes. Most similar to our approach are methods that regularize representations during finetuning, such as knowledge insulation [22] or parameter merging [23]. Though such methods do not explicitly constrain the type of minima learned, we view these techniques as complementary to ours.

**Instruction Faithfulness.** Recently, ensuring that policies faithfully follow language instructions has emerged as a distinct focus. Data-driven approaches [24, 25] address this through targeted dataset construction while representation-level methods [16] up-weight language signals. Similarly, [13, 10, 9] decouples the visual prior and applies guidance at inference time to steer the policy. Our work differs from these in that we target the geometry of the loss landscape. Most similar to our approach, a concurrent work [11] focuses on directly regularizing the weight space during VLA fine-tuning in combination with inference-time guidance. Our approach is a distinct alternative, which provides a higher performance baseline that can be used in combination with other techniques.

**Optimization of Loss Landscape.** The relationship between loss landscape geometry and generalization has been extensively studied in ML literature [26, 27, 28, 29, 17, 19, 30, 31]. [26] first formalized the connection between flat minima and generalization, observing that solutions residing in wide regions of the loss surface tend to generalize better than those in sharp, narrow ones. Methods like stochastic weight averaging implicitly achieve flatness while sharpness-aware minimization (SAM) [17] is an explicit bi-level optimization which has been shown to improve generalization across vision, language, and joint tasks [30, 20, 19, 31]. Subsequent work has proposed a number of variants [18, 19, 32] to reduce parameter tuning, improving generalization, or boost efficiency.

# 3 Problem Formulation

## 3.1 Preliminaries

**Vision Language Action Models.** VLAs typically consist of a VLM backbone and a flow-matching or diffusion action head conditioned on vision and language inputs, denoted as $\pi(a|o, l)$. $a$ is the $k$-step future action chunk, $o$ is the current visual observation (e.g. RGB wrist camera and external camera images), and $l$ is the given language instruction. We assume access to robot finetuning data, $D_{FT}$, consisting of $(a, o, l)$ tuples, and $D_{PT} \gg D_{FT}$ for internet-scale pretraining data $D_{FT}$. The supervised finetuning (SFT) objective is typically: $\mathcal{L}(\theta) = \mathbb{E}_{t,x_0,x_1}\left[\|v_\theta(x_t, t, o, l) - (x_1 - x_0)\|^2\right]$ where a learned velocity field $v_\theta(x_t, t, o, l)$ at time $t$ transports Gaussian noise $x_0 \sim \mathcal{N}(0, I)$ to the target action distribution $x_1 = a$ along a linear path $x_t = (1 - t)x_0 + tx_1$.

**Instruction Blindness.** A key failure mode arising from vision bias is *instruction blindness*. We first define the broader notion of generalization: given a finetuning dataset $D_{FT}$, a test sample $(o, l)$ is *out-of-distribution* if the joint $(o, l)$ was not observed during training, even if $o$ and $l$ were seen independently. Instruction blindness is a practically significant case of OOD, in which a language instruction $l$ is paired with a new visual context. Formally, the model exhibits instruction blindness when: $\pi(a|o, l') \simeq \pi(a|o, l)$, $l' \neq l$ where $l$ is a previously observed instruction for the same scene but is semantically distinct from $l$. From a geometric perspective, instruction blindness occurs when the learned manifold $\mathcal{M}$ is insufficiently sensitive to semantically distinct language perturbations: the manifold has converged such that novel $(o, l)$, which are within support of the internet-scale pretrained VLM, are now poorly supported on $\mathcal{M}$. Instruction blindness has been shown to be a failure case of VLAs in numerous existing works [4, 9, 10, 13, 12, 11].

**Flatness and Generalization.** Learning *sharp minima* is common for large-scale, overparameterized models, where multiple minima can yield similar training loss while having significantly different test performance [19]. Formally, [17] defines the sharpness $\phi$, of a solution $\theta$, as the worst-case loss perturbation in a neighborhood of radius $\rho$: $\phi_\rho(\theta) = max_{||\epsilon||\leq\rho}\mathcal{L}(\theta + \epsilon) - \mathcal{L}(\theta)$. A solution is considered flat if $\phi_\rho(\theta)$ remains small for non-trivial $\rho$, indicating that the learned model is robust to parameter perturbations. We extend this notion to the manifold $\mathcal{M}$ learned by the VLA: a *smooth manifold* has curvature varies that gradually, such that nearby samples remain well-supported.

### 3.2 Assumptions

Our approach rests on two key assumptions. First, we assume that $(o, l)$ test pairs remain within the support of the pretrained action distribution (grasps, joint positions, etc.). Formally, we assume $a \in \text{supp}(p_\theta(a))$ for all test tasks, such that failures in instruction following are attributable to language grounding rather than to the absence of the required motor actions. Second, we assume that the capacity for instruction grounding exists within the VLM backbone and has not been fully lost during VLA pretraining. That is, the pretrained representations encode sufficient information to distinguish between language instructions, even if this signal is suppressed during finetuning.

### 3.3 Setup

Following [27], we quantify sharpness as a normalized version of $\phi$:

$$\mathcal{S}(\theta) = \frac{\phi_\rho(\theta)}{1 + \mathcal{L}(\theta)} \times 100 \tag{1}$$

Let $\mathcal{T}(o, l)$ denote a task with visual observation $o$ and language instruction $l$, and $\mathbf{1}[\pi_\theta, \mathcal{T}]$ be an indicator of task success under policy $\pi_\theta$. We define the instruction following performance as:

$$\mathcal{F}(\theta) = \mathbb{E}_{(o,l) \notin D_{FT}} \left[ \mathbf{1}[\pi_\theta, \mathcal{T}(o, l)] \right] \tag{2}$$

Given $D_{FT}$ with vision-language imbalance and lower language support, we seek to find parameters $\theta^*$ such that: $\mathcal{S}(\theta^*) < \mathcal{S}(\theta)$, where $\theta$ denotes the parameters obtained via standard finetuning. We hypothesize that this reduction in sharpness is associated with improved instruction following on OOC pairs: $\mathcal{S}(\theta^*) < \mathcal{S}(\theta) \implies \mathcal{F}(\theta^*) > \mathcal{F}(\theta)$, which we evaluate empirically in Section 5.

## 4 Method

**Algorithm 1** SAM Finetuning for VLAs

**Require:** Pretrained VLA $\pi_\theta$, finetuning dataset $D_{FT}$, perturbation radius $\rho$, learning rate $\eta$, base optimizer AdamW
**Ensure:** Finetuned parameters $\theta^*$

**while** not converged **do**
  Sample batch $(a, v, l) \sim D_{FT}$
  **// Step 1: Compute perturbation at clean parameters**
  $\mathcal{L}(\theta) \leftarrow \mathbb{E}_{t,x_0,x_1} \left[ \| v_\theta(x_t, t, v, l) - (x_1 - x_0) \|^2 \right]$
  $g \leftarrow \nabla_\theta \mathcal{L}(\theta)$
  $\hat{\epsilon} \leftarrow \rho \cdot g \; / \; \|g\|$
  **// Step 2: Compute gradient at perturbed parameters**
  $\tilde{g} \leftarrow \nabla_\theta \mathcal{L}(\theta + \hat{\epsilon})$
  **// Step 3: Update original parameters with AdamW**
  $\theta \leftarrow \text{AdamW}(\theta,\ \tilde{g},\ \eta)$
**end while**
**return** $\theta^*$

We aim to mitigate instruction blindness and improve generalization in VLA finetuning without additional data collection, pre-training, or architectural modifications by applying sharpness-aware minimization (SAM) to VLA finetuning. SAM [17] showed that flat minima admit tighter PAC-Bayes generalization bounds, and thus solutions stable under parameter-space perturbations result in better generalization to test samples. We hypothesize that in the VLA finetuning setting, flatness discourages the model from learning the sharp, vision-dominated input features that are most predictive in small-scale datasets compared to standard gradient descent. We verify this hypothesis empirically through our experiments in Section 5.

SAM [17] reformulates standard gradient descent as a bi-level optimization problem that penalizes sharp minima. Rather than minimizing $\mathcal{L}(\theta)$ directly, SAM seeks parameters $\theta^*$ that minimize the

worst-case loss in a $\rho$-neighborhood of $\theta$, where $\rho$ is tuned as a hyperparameter:

$$\theta^* = \arg\min_{\theta} \max_{\|\epsilon\| \leq \rho} \mathcal{L}(\theta + \epsilon) \tag{3}$$

In practice, this is approximated via a two-step gradient computation at each training iteration. First, the maximizing perturbation is estimated via a single gradient ascent step:

$$\hat{\epsilon}(\theta) = \rho \cdot \frac{\nabla_\theta \mathcal{L}(\theta)}{\|\nabla_\theta \mathcal{L}(\theta)\|} \tag{4}$$

The model parameters are then updated using the gradient evaluated at the perturbed point $\theta + \hat{\epsilon}(\theta)$, rather than at $\theta$ itself:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta + \hat{\epsilon}(\theta)) \tag{5}$$

where $\eta$ is the learning rate. Algorithm 1 shows the pseudocode for the SAM algorithm on a VLA.

# 5 Evaluation in Simulation

Broadly, we are interested in understanding: **RQ 1**) Do the generalization benefits of SAM transfer to VLA models, specifically by mitigating instruction blindness? **RQ 2**) Is a flatter manifold indicative of better language following? **RQ 3**) How should SAM be applied on a large-parameter VLA model? **RQ 4**) Does SAM work in combination with other instruction-following techniques? We finetune a pre-trained VLA model using SAM and benchmark language-following ability compared with a variety of baselines finetuned on the exact same (or more) data. We evaluate the method on three counterfactual benchmarks: LIBERO-PRO [8], LIBERO-CF [9], and LangGap [25], consisting of a total of 138 new tasks. Details on the benchmarks can be found in the Appendix A.

## 5.1 Baseline Models

**Vanilla VLAs.** We first evaluate off-the-shelf VLA models finetuned on LIBERO, including OpenVLA-OFT [33], and $\pi_{0.5}$ [34]. OpenVLA-OFT uses a Prismatic 7B VLM [35] to decode action tokens. $\pi_{0.5}$ uses a PaliGemma [36] backbone with a flow-matching action head[1]. These models represent the effect of standard finetuning practices on biased data.

**Bayesian-Factorized Models.** In recent literature, methods explicitly factorize the VLA model into a prior $\pi(a|o)$ and likelihood model $\pi(l|o, a)$. These two models are then combined via the optimization objective [13] or with guidance at inference time [9, 12]. We compare with released models: BayesVLA [10] and CAG [9], as well as a finetuned classifier-free guidance [37] variant of the default VLA, denoted as $\pi_{0.5}$_cfg, trained with language dropout (details in Appendix C).

**Data-Augmented Models.** A data-centric approach to preserving pretrained language representations is to pretrain or finetune with more diverse language data, including counterfactual examples. We compare with the results released in [25], denoted as $\pi_{0.5}$_LangGap, where the model is finetuned with a diverse set of tasks and counterfactual language.

**Models with Representation Regularization.** Methods that regularize the loss or weight space are most similar to our approach. We compare against $\pi_{0.5}$_LORA, a LORA-finetuned version of $\pi_{0.5}$ that serves as the simplest method of retaining pretrained representations by learning adapters on top of the frozen model. Parameters for finetuning are in Appendix C.

## 5.2 Implementation and Metrics

We apply SAM to $\pi_{0.5}$_base during LIBERO finetuning with $\rho$ tuned as a hyperparameter. The base optimizer is AdamW [38] with gradient clipping. Each finetuned model was finetuned with 30k steps on the same LIBERO dataset. All evaluations are run zero-shot on new $(o, l)$ pairs with 50

[1]We note that "knowledge insulation" techniques [22] were already used in the pretrained $\pi_{0.5}$ checkpoint which we further finetune on.

Table 1: Success rate (%) across LIBERO-PRO [8] and LIBERO-LangGap [25]. † indicates the result is as reported in the original paper.

| | LIBERO-PRO | | | | | LIBERO-LangGap | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Goal** | **Spatial** | **Object** | **Long** | **Avg** | **Goal** | **Spatial** | **Object** | **Avg** |
| *Vanilla VLAs* | | | | | | | | | |
| OpenVLA-OFT [33] | 0.0 | 3.8 | 1.6 | 0.0 | 1.4 | 0.0 | 0.0 | 0.0 | 0.0 |
| $\pi_{0.5}$ [34] | 25.0 | 52.8 | 11.6 | 17.0 | 26.6 | 30.0 | 5.9 | 37.7 | 24.5 |
| *Factorized Guidance* | | | | | | | | | |
| BayesVLA† [10] | - | - | 10.0 | - | 10.0 | - | - | - | - |
| $\pi_{0.5}$_cfg | 25.4 | 49.4 | 28.0 | 18.0 | 30.2 | 43.8 | 8.5 | 35.8 | 29.4 |
| *Data Augmentation* | | | | | | | | | |
| $\pi_{0.5}$_LangGap† [25] | - | - | - | - | - | 27.2 | 10.2 | 37.0 | 24.8 |
| *Representation Regularization* | | | | | | | | | |
| $\pi_{0.5}$_LORA | 11.0 | 11.0 | 5.0 | 0.0 | 6.75 | 0.2 | 0.2 | 0.3 | 0.2 |
| $\pi_{0.5}$_SAM (Ours) | **43.4** | **54.2** | **42.4** | **30.6** | **42.6** | **56.7** | **19.7** | **48.7** | **41.7** |

rollouts. Additionally, we also apply SAM to OpenVLA-OFT and verify that instruction following improves on a different architecture but focus on the SOTA $\pi_{0.5}$ model for the remaining analyses as the base model provides a better baseline than OpenVLA-OFT (see Appendix D for details).

We primarily evaluate instruction-following ability by reporting the task success rate (SR). To quantify the sharpness of the learned loss landscape, we use the Keskar Sharpness (Eqn. 1), $S$, and the eigenvalue spectrum of the Hessian $\nabla^2_\theta \mathcal{L}(\theta)$, where the largest eigenvalue $\lambda_{\max}$, provides a local measure of curvature at the solution. For real-world experiments, we also measuring the Grounding Rate defined in [9]. Further details on implementation and metrics are in Appendix B.

Table 2: Success rate (%) across LIBERO-CF [9].† indicates results reported in LIBERO-CF [9]

| **Method** | **CF-Spatial** | **CF-Object** | **CF-Long** | **CF-OOD** | **Average** |
|---|---|---|---|---|---|
| OpenVLA-OFT† [33] | 1.1 | 0.0 | 0.2 | 0.1 | 0.4 |
| OpenVLA-OFT CAG† [9] | 7.9 | 0 | 0.4 | 0.1 | 11.3 |
| $\pi_{0.5}$† [34] | 24.4 | 5.8 | 15.8 | 6.9 | 13.2 |
| $\pi_{0.5}$_cfg | 26.8 | 19.6 | 52.6 | 46.0 | 36.3 |
| $\pi_{0.5}$ CAG† [9] | 31.6 | 18.0 | 26.7 | 10.3 | 21.7 |
| $\pi_{0.5}$_LORA | 10.3 | 0.0 | 12.4 | 0.0 | 5.7 |
| $\pi_{0.5}$_SAM (Ours) | **55.3** | **26.2** | **55.4** | **54.1** | **47.8** |

### 5.3 RQ 1: Instruction Following

As shown in Table 1 and 2, applying SAM during finetuning with the exact same data consistently outperforms the default VLA and achieves SOTA performance across the LIBERO-PRO Task, LangGap, and LIBERO-CF benchmarks by 16%, 17.2%, and 28.7% respectively. Substantial improvements are achieved across all task suites including CF-OOD, which focuses on unseen objects. Specifically, we see substantial improvements in the "Object" subset of each benchmark, which involves correctly grounding the target object amongst many similar ones, suggesting that preserving flatness helps retain the vision-language coupling and minimizes spurious correlations between language and spatial locations or scene layouts. We note that Bayesian-Factorization of models and guidance techniques at inference-time also provide better instruction following compared to the default VLA, but still perform worse than our method. Despite not requiring any additional data during the finetuning process, our method still outperforms methods that use additional training data [25].

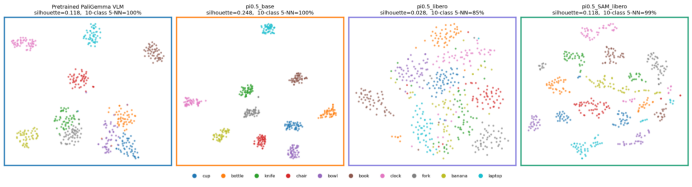

Figure 2: t-SNE plot of object-token representation probe for the Paligemma VLM, $\pi_{0.5}$_base, $\pi_{0.5}$, and $\pi_{0.5}$_SAM

Table 3: Sharpness and max. eigenvalue of the Hessian

| | $S$ (↓) | $\lambda_{max}$ (↓) |
|---|---|---|
| $\pi_{0.5}$ | 0.012 | 0.93 |
| $\pi_{0.5}$_SAM | 0.005 | 0.52 |

### 5.4 RQ 2: Verification of Representation Shifts

We verify the change in representation using SAM in two ways. First, we conduct a probe on the vision-language embedding space, using the method in [39] on the COCO dataset [40] and visualize using a t-SNE plot (details in Appendix B). As seen from Fig. 2, in the pretrained VLM and even the base VLA model, the object categories form coherent semantic clusters. In contrast, the representations are disrupted after standard finetuning on LIBERO, but are then implicitly remedied by finetuning with SAM. Subsequently, we quantify the curvature of the landscape with the a sharpness metric and the maximum eigenvalue of the Hessian $\lambda_{max}$. We use a Lanczos approximation to calculate this, following [17]. We verify in Table 3 that the sharpness value and max. eigenvalue of the Hessian are both lower when finetuning is SAM.

### 5.5 RQ 3: Component Ablation

We next investigate whether SAM should be applied globally or selectively to certain components. We perform a component-wise ablation by separating the $\pi_{0.5}$ model into it's vision, language, and action components. For each, the SAM algorithm is only applied to the filtered subset of gradients and we measure overall task success rate and sharpness on LIBERO-PRO Task (details in Appendix D). We find empirically that global application of SAM consistently outperforms component-wise application in task success. Specifically, applying SAM to an individual component at a time does not improve performance upon the default VLA, though applying it on the language component resulted in the highest average success of 23% and corresponding lowest sharpness in the LLM. We interpret this as evidence that the curvature induced by low-data finetuning is not localized to a single component but propagates through the network. To further verify this, see the global SAM-finetuning results on OpenVLA-OFT (Appendix D).

### 5.6 RQ 4: Sharpness and Guidance

Finally, we investigate the effect of combining SAM with a guidance-method inspired by the Bayesian-factorized methods empirically found to work well [10, 9, 13]. We apply inference-time guidance to the SAM-finetuned policy with the guidance scale $\lambda$ tuned empirically as a hyperparameter. We find that using a combination of both techniques can further boost instruction-following performance on top of $\pi_{0.5}$_SAM by an average of 17.8% on LIBERO-PRO Task with close performance on the other two benchmarks (details in D). Overall, the largest performance improvement is still due to SAM. Thus, using SAM provides a higher performance baseline which can be used in combination with other complementary techniques for instruction following.

## 6 Evaluation in Real

To further evaluate the ability of SAM to mitigate visual bias in VLAs, we design a set of real-world evaluations usingi the DROID setup for five pick-and-place tasks shown in Fig. 3. For each scene, we collect demonstrations for an in-domain task and induce visual bias by only collecting demonstrations for that one task per scene layout. We then finetune a VLA on this biased dataset with and without SAM and evaluate with a counterfactual language instruction for each scene. While the objects remain the same, each test task involves picking up the object at a new location. We

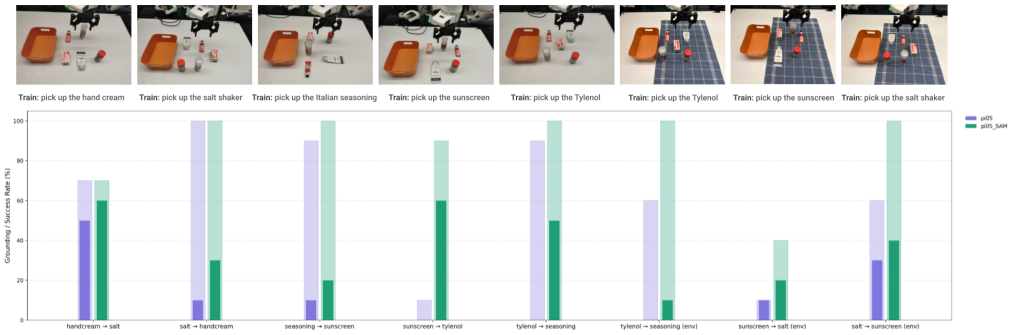


Figure 3: Real-world experiments on five object layouts and a counterfactual instruction at inference time. Each bar shows the grounding rate (translucent) and task success rate (solid).

also introduce environment perturbations where the background is changed for three of the tasks to evaluate robustness of vision-language grounding. Further details are in Appendix E.

We verify that finetuning with SAM results in better grounding success and task success for the model while finetuned on the exact same real-world pick-and-place dataset. While the default finetuned model achieves an average 13.8% success rate, the model finetuned with SAM achieves 36.3%, reflecting an improvement of 163%. We observe that the grounding gap is smaller on some real-world tasks, likely because of the objects being close to the original DROID training distribution. However, even on tasks when the default model can correctly ground the object, it fails to accurately position the end-effector to grasp it, reflecting some uncertainty in the action.

## 7 Limitations

While SAM significantly improves instruction following, several limitations remain. First, SAM requires two forward passes per training step, approximately doubling the compute relative to standard finetuning. While this overhead is modest compared to the cost of additional data collection or pretraining, it does require access to the full gradient of the model and enough memory. Parameter-efficient alternatives may alleviate this and we leave these methods to future work. Second, while SAM encourages the finetuned model to remain in flatter regions of the loss landscape, it does not explicitly prevent drift by constraining the model around pretrained initializations. Methods such as SWA [41] or parameter merging may preserve pretrained structure more explicitly. Finally, while our method yields substantial improvements in instruction following, absolute success rates remain low. This suggests that vision bias and sharp loss landscapes are a significant but not sole contributor to instruction blindness. Data diversity, task complexity, and fundamental limitations of the training paradigm likely play additional roles. We view our results as establishing a stronger baseline rather than a complete solution, and expect further gains when combined with complementary approaches.

## 8 Conclusion

Motivated by the observation that standard finetuning on biased data causes overfitting to training samples and instruction blindness, we investigate whether SAM applied while finetuning on the same limited dataset can work as a principled remedy. We find that preserving flatness while finetuning improves instruction following significantly across three benchmarks and in real-world experiments. Our analysis reveals that this specifically improves object grounding and that global application of SAM works better over selective alternatives. We further show that SAM is complementary to existing instruction-following techniques, suggesting that preserving flatness provides a more robust foundation from which to build. We hope this work encourages the robotics community to consider loss landscape geometry when seeking to develop generalist robot foundation models.

# References


[1] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.

[2] A. Goyal, H. Hadfield, X. Yang, V. Blukis, and F. Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

[3] J. Zhang, X. Chen, Q. Wang, M. Li, Y. Guo, Y. Hu, J. Zhang, S. Bai, J. Lin, and J. Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026.

[4] S. Grover, A. Gopalkrishnan, B. Ai, H. I. Christensen, H. Su, and X. Li. Enhancing generalization in vision-language-action models by preserving pretrained representations. *arXiv preprint arXiv:2509.11417*, 2025.

[5] J. Guo, Z. Wu, C. Tu, Y. Ma, X. Kong, Z. Liu, J. Ji, S. Zhang, Y. Chen, K. Chen, et al. On robustness of vision-language-action model against multi-modal perturbations. *arXiv preprint arXiv:2510.00037*, 2025.

[6] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[7] G. Wang, C. Zhang, Q. Liu, J. Zhang, J. Cai, J. Liu, and X. Liu. Libero-x: Robustness litmus for vision-language-action models. *arXiv preprint arXiv:2602.06556*, 2026.

[8] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

[9] Y. Fang, Y. Feng, D. Jing, J. Liu, Y. Yang, Z. Wei, D. Szafir, and M. Ding. When vision overrides language: Evaluating and mitigating counterfactual failures in vlas. *arXiv preprint arXiv:2602.17659*, 2026.

[10] K. Xu, Z. Zhu, A. Chen, S. Zhao, Q. Huang, Y. Yang, H. Lu, R. Xiong, M. Tomizuka, and Y. Wang. Seeing to act, prompting to specify: A bayesian factorization of vision language action policy. *arXiv preprint arXiv:2512.11218*, 2025.

[11] S. Huang, J. Shao, K. Wang, Q. Chen, J. Sun, Y. Guo, M. Schwager, and J. Bohg. Breaking lock-in: Preserving steerability under low-data vla post-training. *arXiv preprint arXiv:2604.23121*, 2026.

[12] Z. Zhan, Y. Chen, J. Zhou, Q. Lv, H. Liu, K. Wang, L. Lin, and G. Wang. Stable language guidance for vision-language-action models. *arXiv preprint arXiv:2601.04052*, 2026.

[13] S. Lian, B. Yu, X. Lin, L. T. Yang, Z. Shen, C. Wu, Y. Miao, C. Huang, and K. Chen. Bayesian-vla: Bayesian decomposition of vision language action models via latent action queries. *arXiv preprint arXiv:2601.15197*, 2026.

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

[15] A. J. Hancock, X. Wu, L. Zha, O. Russakovsky, and A. Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. *arXiv preprint arXiv:2509.22195*, 2025.

[16] N. Zhang, B. Zhu, S. Zhou, and J. Chen. Restoring linguistic grounding in vla models via train-free attention recalibration. *arXiv preprint arXiv:2603.06001*, 2026.

[17] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[18] J. Kwon, J. Kim, H. Park, and I. K. Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International conference on machine learning*, pages 5905–5914. PMLR, 2021.

[19] T. Sherborne, N. Saphra, P. Dasigi, and H. Peng. Tram: Bridging trust regions and sharpness aware minimization. *arXiv preprint arXiv:2310.03646*, 2023.

[20] Y. Liu, S. Mai, X. Chen, C.-J. Hsieh, and Y. You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12360–12370, 2022.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[22] D. Driess, J. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Ren, H. Walke, Q. Vuong, L. X. Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[23] Y. Yadav, Z. Zhou, A. Wagenmaker, K. Pertsch, and S. Levine. Robust finetuning of vision-language-action robot policies via parameter merging. *arXiv preprint arXiv:2512.08333*, 2025.

[24] C. Glossop, W. Chen, A. Bhorkar, D. Shah, and S. Levine. Cast: Counterfactual labels improve instruction following in vision-language-action models. *arXiv preprint arXiv:2508.13446*, 2025.

[25] Y. Hou and L. Zhao. Langgap: Diagnosing and closing the language gap in vision-language-action models. *arXiv preprint arXiv:2603.00592*, 2026.

[26] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[27] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[28] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[29] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[30] D. Bahri, H. Mobahi, and Y. Tay. Sharpness-aware minimization improves language model generalization. 2022.

[31] Y. Yang, Z. Zhang, R. V. Swaminathan, J. Liu, N. Susanj, and Z. Zhang. SharpZO: Hybrid sharpness-aware vision language model prompt tuning via forward-only passes. Oct. 2025.

[32] T. Li, P. Zhou, Z. He, X. Cheng, and X. Huang. Friendly sharpness-aware minimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5631–5640, 2024.

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

[34] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

[35] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

[36] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[37] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[38] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[39] N. Kachaev, M. Kolosov, D. Zelezetsky, A. K. Kovalev, and A. I. Panov. Don't blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*, 2025.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[41] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

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

# A Benchmark Details

We utilize three counterfactual benchmarks to evaluate instruction following, all based on the LIBERO simulation framework.

**LIBERO-PRO Task.** LIBERO-PRO [8] augments standard LIBERO evaluation across five dimensions: perturbed object positions, perturbed environment state, perturbed task instructions, perturbed object attributes, and perturbed initial positions. Specifically, we use the perturbed task instructions subset, where a counterfactual language instruction is given for a previously observed training scene layout. This benchmark consists of a total of 40 tasks (one counterfactual instruction for each LIBERO training task).

**LangGap.** LangGap [25] is a recent dataset that specifically targets instruction blindness evaluations. The benchmark consists of counterfactual instructions for the LIBERO goal, object, and spatial tasks, resulting in a total of 59 new tasks.

**LIBERO-CF.** LIBERO-CF [9] is another recent benchmark that specifically evaluates counterfactual failures of VLA models. They augment the LIBERO training scenes with counterfactual language instructions pertaining to different background objects, goal positions, and even out-of-distribution objects never seen during training (e.g. coke can). The benchmark consists of a total of 50 new tasks. Note: we were given access to the dataset by the authors.

# B Implementation of Metrics

**Keskar Sharpness.** Ideally, the Keskar sharpness would be calculated with all parameters of the model. However, in practice, the full VLA model has millions of parameters and perturbing the weights then calculating the gradient with respect to the loss over all of them requires a significant amount of memory. To make this calculation feasible, we calculate this only with respect to the action head as it's where high-dimensional features from the VLM backbone are compressed into task-specific parameters, and where we hypothesize the highest sharpness may occur.

**Eigenvalues of Hessian Spectrum.** Following the original SAM paper [17], we calculate the eigenvalues of the Hessian to quantify the local geometry of the learned loss landscape. First, we use the Lanczos algorithm to approximate the eigenvalues. Given the large parameter-scale VLA model, a full Hessian computation is constrained by compute and memory, thus we employ a more efficient function to compute the Hessian-vector products on the subset of parameters within the action head. This allows us to isolate the stability of the action control component of the VLA by examining the local geometry there.

**Object-level Representation Probe.** Inspired by [39], we employ a representation probe to analyze object representations across three PaliGemma variants: the base model (google/paligemma-3b-pt-224), the PaliGemma within the $\pi_{0.5}$_base VLA (pretrained, but not finetuned), the PaliGemma within the standard finetuning $\pi_{0.5}$_libero checkpoint, and the PaliGemma backbone within the SAM-finetuned $\pi_{0.5}$_libero. For each model, we extract Gemma's last-layer hidden states at object tokens by processing 50 COCO images per class across 10 object categories (cup, bottle, knife, chair, bowl, book, clock, fork, banana, laptop). The probe prompt takes the form ``<image> answer en Do you see [object]?'' and we extract the corresponding hidden state for the object token at the final transformer layer.

Qualitatively, we visualize the learned object representations using a t-SNE projection (perplexity$=$ 30, max_iter$=$ 1500, random_state$=$ 42) computed on the last-layer hidden states across all 500 samples, with points color-coded by class to assess whether object representations belonging to the same class remain together.

Quantitatively, we evaluate the quality of the object-class representation clusters using silhouette scores and the K-NN classification accuracy. These metrics are computed on held-out test images (10% of instances per class).

*Silhouette Score:* To quantitatively assess clustering, we calculate the silhouette score. For each instance $i$, we compute

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}, \quad S = \frac{1}{N}\sum_{i=1}^{N} s_i \in [-1, +1] \tag{6}$$

where $a_i$ is the mean distance to samples in the same class and $b_i$ is the mean distance to the nearest different class. Higher values indicate more separability between clusters.

*KNN Accuracy*: We additionally compute the KNN classification score with $k = 5$, measuring local neighborhood structure quality via majority vote among nearest neighbors.

**Grounding Success Rate.** Following [9], for real-world experiments, we manually annotate an additional metric: grounding rate, which is a binary score per rollout that measures whether the gripper makes contact with the specified target object. This effectively measure the vision-language grounding ability of the model through it's action "intent" while not penalizing grasp failures. We note this is an important metric for real-world experiments as full completion of an action trajectory is more challenging, while instruction blindness mainly concerns the ability to ground the natural language instruction within the visual observation.

# C Implementation Details for Finetuned Methods

**$\pi_{0.5}$_CFG.** To represent a standard inference-time guidance baseline for improving language following, we finetune the $\pi_{0.5}$_base model on the LIBERO training data using classifier-free guidance (CFG) [37], denoted as $\pi_{0.5}$_cfg. Following the standard CFG method, at training time, the language input embedding is dropped out with a probability of $p = 0.1$ and replaced with a null embedding. At inference time, the output actions are calculated with the extrapolation equation:

$$\pi_{0.5}\_cfg(a|o,l) = \pi_{0.5}\_cfg(a|o,\emptyset) + \lambda(\pi_{0.5}\_cfg(a|o,l) - \pi_{0.5}\_cfg(a|o,\emptyset))$$

where $\lambda$ is the guidance scale. We sweep values of $\lambda = \{2, 3, 4\}$, measuring success rate on the LIBERO-PRO Task benchmark and empirically find the value of 3 to yield the highest task success rate.

**$\pi_{0.5}$_LORA.** To obtain the LORA-finetuned checkpoint on LIBERO, we follow the released LORA finetuning code from the official openpi codebase. We use the gemma_2b_lora variant of Paligemma as the backbone and turn off EMA decay.

**$\pi_{0.5}$_SAM.** The hyperparameters used for finetuning the $\pi_{0.5}$_SAM checkpoint are shown in Table 4. The model was finetuned on 4 x NVIDIA A100 Tensor Core GPUs with 80GB memory.

Table 4: Hyperparameters for finetuning $\pi_{0.5}$ with SAM

| **Hyperparameter** | **Value** |
|---|---|
| Learning Rate | $5 \times 10^{-5}$ |
| Batch Size | 16 |
| $\rho$ (Neighborhood Size) | 0.075 |
| Weight Decay | 0.1 |
| Base Optimizer | AdamW |
| Action Horizon | 10 |
| EMA Decay | 0.999 |
| Train Steps | 30,000 |

**$\pi_{0.5}$-SAM+CFG.** To test the combination of using an optimization technique like SAM along with inference-time techniques like guidance, we implement the $\pi_{0.5}$-SAM+CFG policy inspired by classifier-free guidance and Bayesian-factorized methods. We combine a conditional policy and unconditional policy using:

$$\pi(a|o,l) = \pi_{uncond}(a|o,\emptyset) + \lambda(\pi_{cond}(a|o,l) - \pi_{uncond}(a|o,\emptyset))$$

We use $\pi_{0.5}$-SAM as $\pi_{cond}$ and finetune a vision-only prior model on top of $\pi_{0.5}$-base using the same dataset as $\pi_{uncond}$. We sweep values of $\lambda = \{1.5, 2, 2.5, 3\}$ across the LIBERO-PRO Task benchmark and empirically find the value of 1.5 to yield the highest task success rate.

# D Additional Results

## D.1 Tuning $\rho$ Value for $\pi_{0.5}$-SAM

The value of $\rho$ for SAM finetuning represents the size of the neighborhood perturbed in order to calculate the SAM gradient. As each $\rho$ value involves fully finetuning the VLA again, we start with a value of 0.05 found to be the best value in the original SAM paper [17], and sweep over $\{0.05, 0.075, 0.1\}$. We evaluate the model trained with these $\rho$ values on the LIBERO-PRO Task benchmark for 10 rollouts per task and results are shown in Table 5.

Table 5: Success rate (%) across LIBERO-PRO Task for different values of $\rho$

| $\rho$ | Goal | Spatial | Object | Long | Average |
|---|---|---|---|---|---|
| 0.05 | 34.0 | 58.0 | 16.0 | 48.0 | 39.0 |
| 0.075 | 40.0 | 53.0 | 22.0 | 60.0 | **43.8** |
| 0.1 | 37.0 | 49.0 | 23.0 | 46.0 | 38.8 |

## D.2 Additional Qualitative Results for $\pi_{0.5}$-SAM

In Fig. 4, we visualize samples from the LIBERO-PRO Task suite object grounding tasks. We generate a heatmap visualization of the average first grasp position of the end-effector over 50 rollouts per task and compare between the standard finetuned model and ours. The first grasp position in each rollout is recorded as the position of the end-effector when the gripper first moves from fully open to closed, which is then projected onto the 2D external scene image.

## D.3 Component Ablation

We perform component-wise ablation of SAM by selectively applying the SAM algorithm to specific parameters of the model. For each of the vision, language, and action components, denoted as $\pi_{0.5}$-SAM$_{Language}$, $\pi_{0.5}$-SAM$_{Vision}$, and $\pi_{0.5}$-SAM$_{Action}$, respectively, we filter gradients with the following filters and apply the SAM algorithm individually:

Vision: `PaliGemma/img/`

Language: `PaliGemma/llm/`

Action: `(action_in_proj|action_out_proj|time_mlp)`

We then evaluate average task success on LIBERO-PRO Task benchmark with 50 rollouts per task. As seen in Table 6, the instruction following performance does not improve over the standard VLA model when SAM is applied selectively. We hypothesize that this is due to sharpness being induced across components in the model, and bottlenecked in certain subsets of parameters when selectively applied. We observe that applying the SAM update to the LLM component achieves slightly higher downstream success rate than the other two components, though the differences in success rate could be due to noise. Further, we observe that although applying SAM to the action head parameters reduces the sharpness significantly in the action head, the sharpness in the language model remains high, which could explain the worse performance.

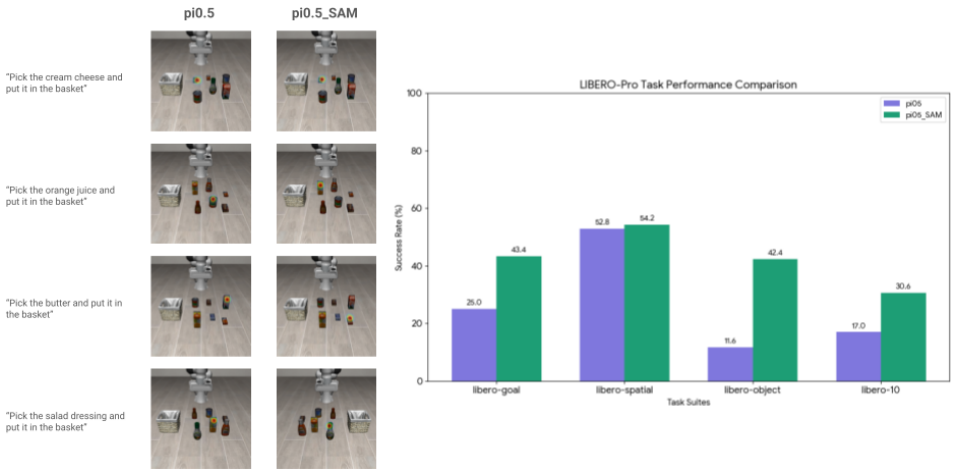


Figure 4: Left: heatmaps of the grasp position for different pick-object instructions between the standard model and our SAM-finetuned one. Qualitatively, the original model incorrectly biases towards the training object. Right: chart visualization of the performance improvement on LIBERO-PRO Task.

Table 6: Average success rate (%) on LIBERO-PRO Task and sharpness in each model component for component-wise applications of SAM

| | Average SR | $S_{lang}$ | $S_{vis}$ | $S_{act}$ |
|---|---|---|---|---|
| $\pi_{0.5}$_SAM$_{Language}$ | 23.0 | 0.100 | 0.015 | 0.025 |
| $\pi_{0.5}$_SAM$_{Vision}$ | 22.7 | 0.120 | 0.011 | 0.025 |
| $\pi_{0.5}$_SAM$_{Action}$ | 21.0 | 0.124 | 0.009 | 0.004 |

### D.4 Sharpness and Guidance

Results across all three benchmarks with 50 rollouts per task are shown in Tables 7 and 8 for the $\pi_{0.5}$_SAM+CFG policy. As observed from the tables, the combination policy improves performance on LIBERO-PRO Task, matches average success rate on LangGap, and has a slightly lower success rate on LIBERO-CF. It is worth noting that the the comparison between this policy and the default SAM policy also vary across task suites. For example, on LIBERO-PRO Task, this combination policy does significantly better on the Spatial subset. Further analysis on the effects of inference-time techniques compared to finetuning and other paradigms on different task types may be a direction for future work.

Table 7: Success rate (%) across LIBERO-PRO [8] and LIBERO-LangGap [25] benchmarks comparing the effect of inference-time guidance on $\pi_{0.5}$_SAM.

| | **LIBERO-PRO** | | | | | **LIBERO-LangGap** | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Goal** | **Spatial** | **Object** | **Long** | **Avg** | **Goal** | **Spatial** | **Object** | **Avg** |
| $\pi_{0.5}$ | 25.0 | 52.8 | 11.6 | 17.0 | 26.6 | 30.0 | 5.9 | 37.7 | 24.5 |
| $\pi_{0.5}$_SAM | 43.4 | 54.2 | 42.4 | 30.6 | 42.6 | 56.7 | 19.7 | 48.7 | **41.7** |
| $\pi_{0.5}$_SAM+CFG | 54.2 | 63.6 | 54.6 | 28.4 | **50.2** | 55.1 | 18.6 | 51.4 | **41.7** |

### D.5 Results on OpenVLA-OFT

To test whether SAM works on an alternate VLA model architecture that's more unified, we conduct additional experiments with the OpenVLA-OFT model, which uses a Prismatic-VLM to parallel decode action tokens directly. We adopt their LoRA finetuning scheme on LIBERO and apply

Table 8: Success rate (%) across LIBERO-CF [9] comparing the effect of inference-time guidance on $\pi_{0.5}$_SAM.

| Method | CF-Spatial | CF-Object | CF-Long | CF-OOD | Average |
|---|---|---|---|---|---|
| $\pi_{0.5}$ | 24.4 | 5.8 | 15.8 | 6.9 | 13.2 |
| $\pi_{0.5}$_SAM | 55.3 | 26.2 | 55.4 | 54.1 | **47.8** |
| $\pi_{0.5}$_SAM+CFG | 56.4 | 31.6 | 35.6 | 55.5 | **44.8** |

SAM finetuning with $\rho = 0.03$. We finetune on 2 x NVIDIA H100 GPUs with 80GB memory. Hyperparamters for finetuning OpenVLA-OFT with SAM are shown in Table 9.

Table 9: Hyperparameters for finetuning OpenVLA-OFT with SAM

| Hyperparameter | Value |
|---|---|
| Learning Rate | $2 \times 10^{-4}$ |
| LR Decay Factor | 0.1 |
| Batch Size | 10 |
| $\rho$ (Neighborhood Size) | 0.03 |
| Weight Decay | 0.1 |
| Base Optimizer | AdamW |
| LoRA Rank | 32 |
| LoRA Dropout | 0.1 |
| Action Horizon | 8 |
| Train Steps | 150,000 |

From the results in Table 10, we see that finetuning on the same dataset with SAM improves instruction following compared to standard finetuning. Notably, for LangGap, the performance increased significantly from zero success rate to 12.2%. Overall, however, the success rates are significantly lower than $\pi_{0.5}$. Because flatness preservation is meant to preserve representations during finetuning on top of the pretrained backbones, the effects are limited by the original capabilities of the OpenVLA-OFT base model. For this reason, we focus on $\pi_{0.5}$ for the majority of our analysis but show that even weaker initial models benefit from this technique.

Table 10: Success rate (%) across LIBERO-PRO [8] and LIBERO-LangGap [25] benchmarks for OpenVLA-OFT

| | LIBERO-PRO | | | | | LIBERO-LangGap | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Goal** | **Spatial** | **Object** | **Long** | **Avg** | **Goal** | **Spatial** | **Object** | **Avg** |
| OpenVLA-OFT | 0.0 | 3.8 | 1.6 | 0.0 | 1.4 | 0.0 | 0.0 | 0.0 | 0.0 |
| OpenVLA-OFT_SAM | 0.04 | 2.0 | 0.0 | 9.4 | **2.9** | 1.3 | 0.4 | 34.9 | **12.2** |

# E Implementation of Real World Evaluations

For real-world experiments, we collect 50 demonstrations for each of five tabletop pick-and-place tasks to use as a limited in-domain finetuning dataset. Inspired by the LIBERO-Object task format, our tasks consist of specific object layouts of five household objects: salt shaker, seasoning bottle, sunscreen, tylenol, and a tube of hand cream. For each layout, we collect demonstrations for picking up one of the objects and putting it in the basket. Across all demonstrations, each object has demonstrations for how to pick it up. We collect demonstrations using an Oculus VR controller for teleoperation with the DROID [42] set up, consisting of a 7-DoF Franka Panda with a Robotiq-2F85 gripper, a ZED X wrist camera and a ZED X left external camera. We then finetune the $\pi_{0.5}$_base checkpoint on the suite of five tasks altogether, for 10k steps using a batch size of 32. We finetune models with $\rho$ values across $\{0.025, 0.05, 0.075\}$ and evaluate on five rollouts for three counterfactual tasks. We empirically find that a value of 0.05 works best for the real-world setting.

During evaluation, we come up with a counterfactual instruction for each scene configuration to pick up a different object. To introduce some visual variation and test the robustness of vision-language object grounding, we additionally change the background for three additional counterfactual tasks, where a towel is laid across the table. Each counterfactual task at test time is then evaluated zero-shot with 10 rollouts.