VLA-Forget: Vision-Language-Action Unlearning for Embodied Foundation Models

Ravi Ranjan*
Florida International University
Miami, USA
rkuma031@fiu.edu

Agoritsa Polyzou
Florida International University
Miami, USA
apolyzou@fiu.edu

Abstract

Vision-language-action (VLA) models are emerging as embodied foundation models for robotic manipulation, but their deployment introduces a new unlearning challenge: removing unsafe, spurious, or privacy-sensitive behaviors without degrading perception, language grounding, and action control. In OpenVLA-style policies, behavior is produced through a fused visual encoder, a cross-modal projector, and a language backbone that predicts tokenized robot actions, so undesirable knowledge can be distributed across perception, alignment, and reasoning/action layers rather than confined to a single module. Consequently, partial unlearning applied only to the vision stack or only to the language backbone is often insufficient, while conventional unlearning baselines designed for standalone vision or language models may leave residual forgetting or incur unnecessary utility loss in embodied settings. We propose **VLA-Forget**, a hybrid unlearning framework that combines ratio-aware selective editing for perception and cross-modal specificity with layer-selective reasoning/action unlearning for utility-preserving forgetting. VLA-Forget jointly optimizes three objectives: targeted forgetting, perceptual preservation, and reasoning retention, through staged updates over the visual encoder, projector, and upper action-generating transformer blocks. Across forget-set behavior probes and retain-task evaluations, VLA-Forget improves forgetting efficacy by 10%, preserves perceptual specificity by 22%, retains reasoning and task success by 9%, and reduces post-quantization recovery by 55% relative to strong unlearning baselines.

1 Introduction

Vision-Language-Action (VLA) models are emerging as a practical route to *embodied foundation models*: policies that translate natural-language instructions and raw visual observations directly into robot actions by leveraging large pretrained vision and language backbones. OpenVLA exemplifies this paradigm as an open 7B VLA trained on 970K real-world robot demonstrations from `Open X-Embodiment`, combining a fused (DINOv2+SigLIP) visual encoder with a Llama 2 backbone and an action-token prediction interface that enables scalable fine-tuning and deployment on commodity hardware (Kim et al., 2024; O'Neill et al., 2024; Pang et al., 2025).

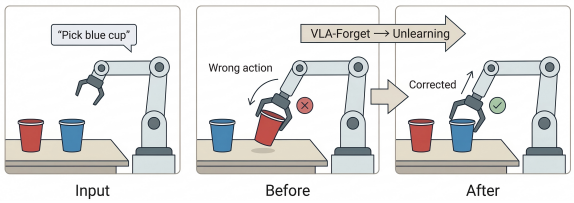

Figure 1: **Motivating failure case for VLA-Forget.** Given the instruction "Pick blue cup," the policy produces a visually plausible but semantically incorrect action by grasping the red cup. After targeted unlearning, the policy suppresses this spurious instruction-to-action association and recovers instruction-consistent behavior.

However, *deployment* of VLA policies introduces an urgent governance and reliability problem that is not fully addressed by today's training and evaluation pipelines: once a robot policy has been trained, it may retain (i) unsafe or undesirable behaviors from a subset of demonstrations, (ii) privacy or IP-sensitive content embedded in multimodal data, or (iii) spurious shortcuts that appear correct under standard benchmarks but fail under structured distribution shift. The consequences of such failures are amplified in robotics because errors translate into physical actions. Recent evidence further highlights a VLA-specific grounding failure, sometimes described as *linguistic blindness* where policies execute visually plausible trajectories even when the instruction is semantically contradictory, motivating stronger mechanisms for selective behavior re-

*Corresponding author.

Proceedings of the 4th Workshop on Towards Knowledgeable Foundation Models (KnowFM 2026), pages 60–77

moval and reliable grounding diagnostics (Zhuang et al., 2026).

Figure 1 illustrates the core VLA-specific challenge studied in this work: undesirable behavior may arise from misalignment across perception, cross-modal grounding, or action priors, causing the policy to execute a physically plausible yet instruction-inconsistent action. This motivates *selective unlearning* that removes a targeted erroneous behavior slice while preserving normal scene understanding and non-target task execution. Motivation is further supported by recent evidence that VLA policies remain brittle under visual corruption and grounding perturbations, underscoring the need for post hoc correction mechanisms beyond standard fine-tuning (Lin et al., 2025; Orjuela et al., 2026).

Machine unlearning provides a practical way to remove targeted training influence without full retraining, but VLA policies are harder to unlearn than standard vision or language models. Unlike static predictors, a VLA model is a *closed-loop control policy*, so failure must be judged through embodied behavior rather than output accuracy alone. In OpenVLA, continuous robot actions are represented as *discrete action tokens*, which means unwanted behavior can be encoded jointly in visual features, cross-modal alignment, and instruction-conditioned action priors in the language backbone (Kim et al., 2024).

As a result, unlearning only the vision stack or only the language model is often insufficient: removing a visual trigger may leave the downstream action prior intact, while editing language priors may preserve harmful perceptual shortcuts or incorrect visual-language bindings. Existing baselines were largely designed for unimodal settings and transfer only partially to VLA policies. For example, exact approaches such as SISA require training-time changes (Bourtoule et al., 2021), while approximate methods such as SCRUB improve forget-retain trade-offs but do not directly address multimodal component entanglement or control-oriented evaluation (Kurmanji et al., 2023). Moreover, VLA unlearning involves competing retain, forget, and mismatch objectives over large backbones, motivating gradient-conflict mitigation such as PCGrad (Yu et al., 2020).

In this work, we introduce **VLA-Forget**, a hybrid unlearning framework for Vision-Language-Action models that is explicitly *component-aware* and *deployment-oriented*. VLA-Forget (i) targets perception and cross-modal specificity through ratio-aware selective editing of the visual encoder and projector, (ii) targets reasoning/action utility-preserving forgetting through significance-based selective editing of action-relevant transformer blocks, and (iii) performs staged, adapter-first updates (e.g., LoRA) to enable efficient unlearning with rollback and canary deployment compatibility (Hu et al., 2022; Kim et al., 2025; Liu et al., 2025a). We evaluate unlearning with robotics-centric metrics and unlearning-centric audits (including safety violation risk); furthermore, we evaluate the model's robustness post-quantization, aligning the evaluation with the realities of VLA deployment (Carlini et al., 2022; Zang et al., 2025; Zhang et al., 2024b) While VLA-Forget improves targeted behavior suppression in benchmarked embodied settings, it is an approximate unlearning method and should not be interpreted as providing certified erasure.

Key Contributions. **(i)** We formalize *VLA unlearning* as a three-goal problem **targeted forgetting** (efficacy), **perceptual preservation** (specificity), and **reasoning retention** (utility) in the presence of an action-token interface and multimodal component entanglement. **(ii)** We propose **VLA-Forget**, a novel hybrid unlearning pipeline that exploits ratio-aware selective editing for perception/cross-modal specificity with significance-based selective editing for reasoning/action utility preservation, implemented in an adapter-first manner compatible with OpenVLA fine-tuning workflows. **(iii)** We outline an **evaluation protocol** that couples embodied performance (task success and control stability) with unlearning audits (forget/retain scores and safety violation risk), and incorporates structured contradiction probes to diagnose and prevent spurious "fake success" under OOD instructions.

2 Related Work

Vision-language-action (VLA) models extend multimodal foundation models from perception and generation to embodied control. Early systems such as VIMA and RT-2 showed that robot behavior can be conditioned on interleaved visual-language prompts and represented through tokenized actions, while `Open X-Embodiment` and OpenVLA scaled this paradigm to diverse real-robot data and open 7B policies. Despite this progress, prior VLA research has focused mainly on scaling, transfer, and adaptation, rather than post hoc removal of un-

safe behaviors, undesirable concepts, or sensitive instruction-action associations. (Jiang et al., 2023; Zitkovich et al., 2023; O'Neill et al., 2024; Grover et al., 2026; Kim et al., 2024)

Machine unlearning has progressed from exact retraining-based deletion to approximate updates that balance erasure quality, efficiency, and retained utility. In language models, recent work spans gradient-ascent, retain-regularized, preference-based, and activation-space methods, while emphasizing joint evaluation of forgetting, utility preservation, and privacy leakage. Two recurring issues are especially relevant for embodied policies: broad model-wide edits often cause collateral degradation, and apparently successful forgetting can fail under deployment transformations such as quantization. These limitations motivate selective, structure-aware unlearning over indiscriminate full-model updates. (Bourtoule et al., 2021; Maini et al., 2024; Yao et al., 2024; Liu et al., 2025b; Zhang et al., 2024b)

Related work in vision, vision-language, and diffusion models studies forgetting of classes, identities, and concepts in multimodal transformers. SSD and SalUn localize updates through parameter importance or saliency, LoTUS improves scalable approximate unlearning with uncertainty-aware smoothing, and SLUG shows that targeted single-layer editing can sometimes suffice. In generative settings, ESD and UCE edit text-conditioned diffusion behavior at the concept level. Collectively, these methods expose a central trade-off: aggressive updates improve forgetting but may damage retention. (Foster et al., 2024; Fan et al., 2023; Spartalis et al., 2025; Cai et al., 2024; Gandikota et al., 2023, 2024)

VLA unlearning differs from prior LLM- or VLM-only settings because undesired behavior may be encoded jointly in visual features, cross-modal bindings, and action-token priors, and must be evaluated through embodied execution rather than text or image outputs alone. Accordingly, VLA-FORGET adopts a component-aware formulation: ratio-aware edits target perception and projector modules, while significance-based selective updates address higher-level reasoning and action-generation components. This yields a hybrid unlearning strategy tailored to embodied foundation models rather than a direct reuse of unimodal forgetting methods. (Zitkovich et al., 2023; Kim et al., 2024; Fan et al., 2023; Cai et al., 2024; Yao et al., 2024)

3 Methodology

Overview. We propose **VLA-Forget**, a novel unlearning framework for vision-language-action (VLA) policies that removes targeted behaviors while preserving perceptual grounding and action reasoning. We instantiate the method on OpenVLA-style policies, where an input image and language instruction are processed by a fused visual encoder, projected into the language-model embedding space, and decoded into discretized robot action tokens. In OpenVLA, the visual stack combines DINOv2 and SigLIP, the backbone is Llama 2, and the policy predicts normalized 7-DoF actions via tokenized bins, making forgetting inherently distributed across perception, cross-modal alignment, and action generation rather than localized to a single module (Kim et al., 2024).

Architecture and unlearning target. Let a VLA policy be

$$f_\theta(o, s) = \mathrm{Dec}_{\theta_L}\Big(\mathrm{Proj}_{\theta_P}\big(\mathrm{Enc}_{\theta_V}(o)\big),\ s\Big), \quad (1)$$

where o is the observation image, s is the instruction, θ_V denotes the visual encoder, θ_P the MLP projector, and θ_L the language/action backbone. The decoder produces an action-token sequence y, which is de-tokenized into a continuous action $a \in \mathbb{R}^7$. We consider an unlearning request U specifying a target concept or behavior slice, such as a sensitive object, unsafe affordance, or erroneous instruction-to-action mapping. We form three datasets: a forget set D_f, a retain set D_r, and a boundary set D_m containing near-neighbor scenes or instructions that must remain intact.

Why "VLA" unlearning is necessary. In VLA models, undesired behavior can originate from three coupled sources: *(i)* perceptual memorization in θ_V, *(ii)* erroneous visual-to-language binding in θ_P, and *(iii)* action priors or instruction-conditioned hallucinations in θ_L. Editing only the vision stack may suppress the visual trigger while leaving the downstream action prior unchanged; editing only the language backbone may preserve a harmful perceptual shortcut. VLA-Forget therefore combines ratio-aware editing for perception and cross-modal specificity with layer-selective unlearning for reasoning and action retention.

3.1 Hybrid Selective Localization

Perception and projector selection. For each candidate visual or projector layer l, we compute

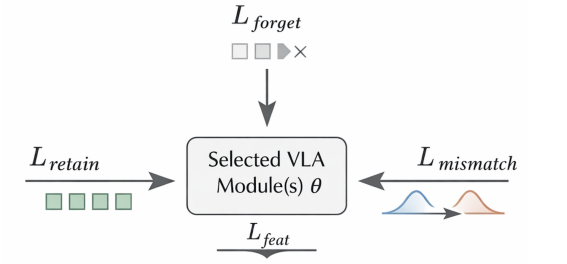

Figure 2: **Illustration of the unlearning objectives on a selected module.** Eq. 6 suppresses the targeted behavior, Eq. 5 preserves non-targeted behavior, and Eq. 7 discourages shallow forgetting by pushing the updated policy away from the original forgotten response.

forget and retain gradients as follows:

$$g_l^f = \nabla_{\theta_l}\mathcal{L}_{\text{forget}}, \quad \text{and} \quad g_l^r = \nabla_{\theta_l}\mathcal{L}_{\text{retain}}. \tag{2}$$

We then assign a ratio-aware score

$$\phi(l) = \frac{\|g_l^f\|_2}{\|\theta_l\|_2 + \varepsilon}\left(1 - \cos(g_l^f, g_l^r)\right)^{\alpha}, \tag{3}$$

and select the top visual/projector layers K_V, K_P with highest $\phi(l)$. This favors parameters that strongly affect forgetting while minimally interfering with retained perception (Yu et al., 2020).

Reasoning/action layer selection. For the upper transformer blocks in the language-action backbone, we calculate a significance ratio:

$$\text{Sig}(l) = \frac{\|\nabla_{\theta_l}\mathcal{L}_{\text{forget}}\|_2}{\|\nabla_{\theta_l}\mathcal{L}_{\text{retain}}\|_2 + \varepsilon}. \tag{4}$$

We initialize an editable set S_L with the top-k layers under $\text{Sig}(l)$, and expand it iteratively only if forgetting criteria are unmet. This yields a minimal update set for action-relevant reasoning while avoiding unnecessary global drift (Ranjan et al., 2026; Kurmanji et al., 2023; Zhang et al., 2024b).

3.2 Unlearning Objectives

We optimize three complementary objectives aligned with the goals of efficacy, specificity, and utility.

Retain loss. To preserve non-targeted behavior, we minimize

$$\begin{aligned}\mathcal{L}_{\text{retain}} =& \mathbb{E}_{(x,y)\sim D_r}\left[\text{CE}\left(p_\theta(\cdot \mid x), y\right)\right] \\ &+ \beta\, \mathbb{E}_{x\sim D_r}\left[\text{KL}\left(p_{\theta_0}(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x)\right)\right].\end{aligned} \tag{5}$$

where θ_0 denotes the original model and $x = (o, s)$. The first term preserves action-token prediction on retained data, while the KL anchor constrains the updated policy to remain close to the base model on benign trajectories (Kim et al., 2024; Kurmanji et al., 2023).

Forget loss. To suppress the targeted behavior, we maximize the forget-set prediction error:

$$\mathcal{L}_{\text{forget}} = \mathbb{E}_{(x,y)\sim D_f}[\text{CE}(p_\theta(\cdot \mid x), y)]. \tag{6}$$

Operationally, this is implemented as gradient ascent on $\mathcal{L}_{\text{forget}}$ (Jin et al., 2025).

Mismatch loss. To avoid shallow forgetting and reduce recovery to the pre-unlearning behavior, we maximize distributional divergence on forgotten samples:

$$\mathcal{L}_{\text{mismatch}} = \mathbb{E}_{x\sim D_f}[\text{KL}(p_\theta(\cdot \mid x) \,\|\, p_{\theta_0}(\cdot \mid x))]. \tag{7}$$

As shown in Fig. 2, the three objectives act together on the selected module to suppress the target behavior, preserve retained behavior, and reduce shallow recovery to the pre-unlearning policy. (Appendix A.1)

Perceptual preservation loss. To preserve non-targeted visual grounding, we distill internal features on $D_r \cup D_m$:

$$\begin{aligned}\mathcal{L}_{\text{feat}} = \mathbb{E}_{x\sim D_r\cup D_m}\Big[&\|h_\theta^V(x) - h_{\theta_0}^V(x)\|_2^2 \\ &+ \gamma\|h_\theta^P(x) - h_{\theta_0}^P(x)\|_2^2\Big],\end{aligned} \tag{8}$$

where h^V and h^P are late visual and projector representations.

Unified objective. The final optimization is

$$\min_\theta \; \mathcal{L}_{\text{retain}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} - \lambda_f\mathcal{L}_{\text{forget}} - \lambda_m\mathcal{L}_{\text{mismatch}}. \tag{9}$$

3.3 Training and Unlearning Procedure

Adapter-first updates. We implement VLA-Forget using parameter-efficient adapters over the selected layers while freezing the remaining weights. This design is compatible with the OpenVLA training stack, which supports LoRA-based fine-tuning with `target_modules=all-linear`; thus unlearning can be applied without full-model retraining and can be merged or rolled back at deployment time (Kim et al., 2025; Hu et al., 2022).

Stage 1: perception unlearning. We first update LoRA parameters on K_V to weaken targeted visual evidence while preserving general scene understanding through $\mathcal{L}_{\text{retain}}$ and $\mathcal{L}_{\text{feat}}$. This stage removes object- or scene-level triggers with minimal disruption to unrelated perception.

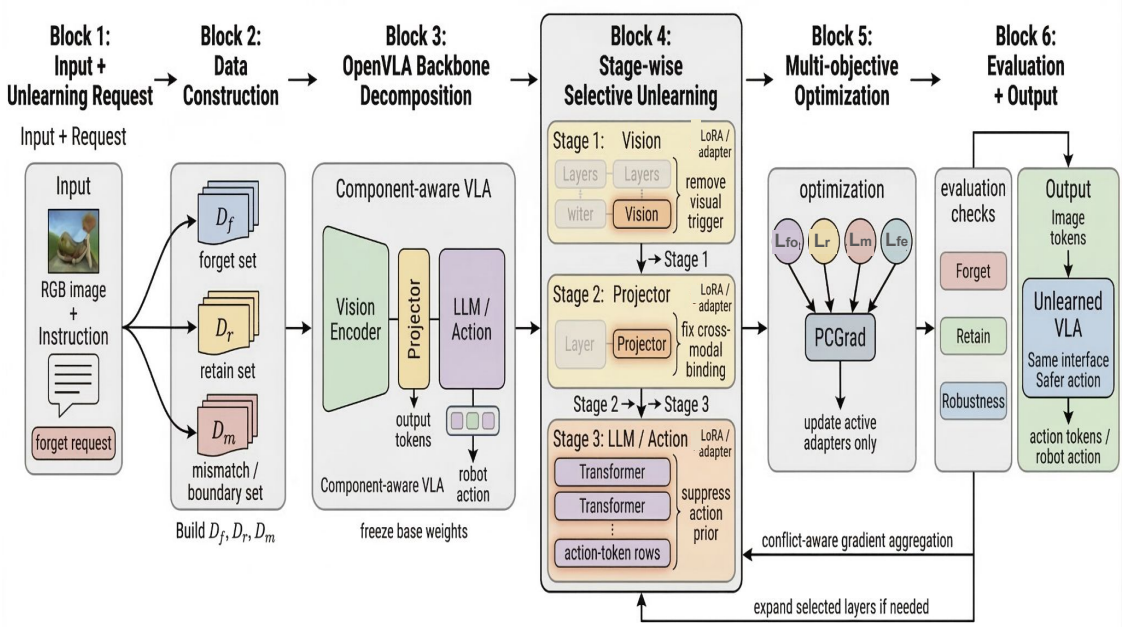

Figure 3: Overview of VLA-Forget. Given an unlearning request, we construct forget, retain, and mismatch sets, then perform staged adapter-based selective updates over the vision encoder, projector, and LLM/action layers with multi-objective PCGrad stabilization, yielding an unlearned VLA policy with preserved interface and retained task utility.

Stage 2: cross-modal unlearning. Next, we update K_P in the projector to break the specific visual-to-language associations responsible for the unwanted behavior. This is essential when the failure is not purely visual, but rather caused by erroneous alignment between image regions and action-relevant tokens.

Stage 3: reasoning/action unlearning. Finally, we update the selected upper backbone layers S_L, and optionally the action-token embedding rows most implicated in the target behavior. This stage suppresses residual instruction-conditioned action priors while maintaining overall task competence.

Gradient stabilization and stopping. Because the retain and forget objectives can conflict, we aggregate gradients using projected conflict resolution:

$$g = \mathrm{PCGrad}\Big(\nabla\mathcal{L}_{\text{retain}}, -\lambda_f\nabla\mathcal{L}_{\text{forget}}, -\lambda_m\nabla\mathcal{L}_{\text{mismatch}}\Big), \tag{10}$$

and apply optimizer steps only to the active adapter parameters. After each round, we evaluate forget efficacy on D_f and retain utility on D_r. If forgetting remains insufficient, we expand S_L by the next highest-significance layer and continue; otherwise, training stops early (Yu et al., 2020).

Figure 3 summarizes the VLA-Forget pipeline as a staged, component-aware unlearning framework. Starting from an unlearning request, the method constructs forget, retain, and mismatch sets, decomposes the OpenVLA policy into vision, projector, and LLM/action modules, and then applies selective adapter-based updates in three stages: visual trigger removal, cross-modal binding correction, and reasoning/action prior suppression. The optimization block combines retain, forget, mismatch, and perceptual preservation objectives with PCGrad stabilization, while the final evaluation block verifies forgetting efficacy, retained utility, and deployment robustness before producing the unlearned VLA policy.

Interaction with VLA policies. VLA-Forget is designed to preserve the native VLA interface: image and instruction inputs remain unchanged, and the model still predicts action tokens followed by the standard de-tokenization and action unnormalization pipeline used by OpenVLA. This makes the method directly deployable in existing VLA control stacks while enabling targeted post hoc removal of unsafe, private, or spurious behaviors (Zang et al., 2025; Pang et al., 2025).

The complete unlearning pipeline is detailed in Algorithm 1, with its associated supporting procedures provided in Algorithm 2; corresponding pseudo-code is available in Appendix A.

Table 1: Unlearning result on **OpenVLA-7B**. Left: `Open X-Embodiment`. Right: `lerobot/pusht_image`. Higher is better for FC, RC, FAD, and TSR; lower is better for RAD and SVR. Mean ± standard deviation over 5 random seeds, with typical variation in the range of ±(1–3) points for rate-based metrics (FC, RC, TSR, and SVR) and ±(0.01–0.03) for distance-based metrics (FAD and RAD). Best results are shown in **bold** and second-best are underlined.

	`Open X-Embodiment` (OpenVLA-7B)						`lerobot/pusht_image` (OpenVLA-7B)					
Method	**FC ↑**	**RC ↑**	**FAD ↑**	**RAD ↓**	**TSR ↑**	**SVR ↓**	**FC ↑**	**RC ↑**	**FAD ↑**	**RAD ↓**	**TSR ↑**	**SVR ↓**
SSD	78	83	0.70	0.28	68	17	82	86	0.73	0.20	55	15
SalUn	89	88	0.76	0.26	71	12	89	88	0.78	0.18	60	11
GA	**93**	60	**0.89**	0.45	40	**5**	94	50	**0.91**	0.50	22	**3**
NPO	90	88	0.83	0.23	74	8	92	90	0.85	0.15	65	7
VLA-Forget	**93**	**91**	0.88	**0.21**	**78**	**5**	**95**	**94**	0.90	**0.13**	**69**	4

4 Experiments

4.1 Experimental Setup

Models. We instantiate VLA-FORGET on `OpenVLA-7B`, a 7B vision-language-action policy trained on large-scale Open X-Embodiment robot data (Kim et al., 2024; O'Neill et al., 2024). The second model is `pi0fast-base` that predicts continuous robot actions via auto-regressive next-token prediction (Pertsch et al., 2025). Following the VLA setting in our method, we construct three splits for each experiment: a forget set D_f containing target behaviors to be removed, a retain set D_r containing non-target behaviors whose utility should be preserved, and a boundary set D_m containing near-neighbor samples used to reduce shallow or entangled forgetting. Unless otherwise stated, all methods start from the same pretrained checkpoint, use the same forget/retain protocol, and are trained under the same optimization budget.

Datasets. We use two complementary benchmarks. First, we build a real-robot benchmark from subsets of Open X-Embodiment (OXE), downloaded via `gsutil`, where each trajectory is paired with a language instruction and organized into forget/retain slices at the trajectory level (O'Neill et al., 2024). Second, we use `lerobot/pusht_image` as a controlled synthetic benchmark, where we inject a static text instruction into each instance and form matched forget/retain partitions (Cadene et al., 2026). In the released OpenVLA, we use up to 512 prompted PushT instances with a 30% forget fraction; in the lightweight ablation pipeline, we scale this to up to 4,000 instances and use a 70/15/15 train/validation/test split, reporting forget and retain test performance after unlearning. This combination gives both a realistic OXE setting and a reproducible low-cost benchmark for rapid ablations.

Baselines. We compare against four representative approximate unlearning baselines. **SSD** (Foster et al., 2024) is a retraining-free parameter-dampening method that suppresses weights estimated to be disproportionately important to the forget data. **SalUn** (Fan et al., 2023) is a saliency-based baseline that updates only high-importance weights, making it a strong selective vision-side unlearning comparator. **GA** (Yao and Xu, 2024) performs direct gradient ascent on the forget loss and serves as the standard language-side unlearning baseline. **NPO** (Zhang et al., 2024a) is a preference-based alternative designed to improve forget-retain trade-offs. These baselines span both vision-oriented and language-oriented unlearning regimes, making them suitable comparators for VLA policies.

Metrics. We report six metrics, **Forget action loss / cross-entropy (FC)** measures forgetting efficacy on D_f; *higher* FC is better because the model should become less able to reproduce the forgotten action mapping. **Retain utility score (RC)**, derived from retain-set cross-entropy, measures preserved utility on D_r; higher is better. **Forget Accuracy Drop (FAD)** measures the reduction in action-token or exact-match accuracy on the forget split relative to the base model; *higher* is better. **Retain Accuracy Drop (RAD)** measures the corresponding degradation on retained behaviors; *lower* is better. **Task Success Rate (TSR)** evaluates closed-loop policy execution on benchmark tasks; *higher* is better. **Safety Violation Rate (SVR)** measures the frequency of unsafe or disallowed behaviors under target prompts or contradiction probes; *lower* is better. We follow standard VLA and robot manipulation evaluation protocols (Kim et al., 2024; Liu et al., 2023). Together, these metrics capture

Table 2: Unlearning results on `lerobot/pi0fast-base` under the `Open X-Embodiment` protocol. Mean ± standard deviation over 5 random seeds, with typical variation of ±(1–3) points for FC, RC, TSR, and SVR, and ±(0.01–0.04) for FAD and RAD. Best results are shown in **bold** and second-best are underlined.

Method	Open X-Embodiment (Pi0-FAST-Base)					
	FC ↑	RC ↑	FAD ↑	RAD ↓	TSR ↑	SVR ↓
SSD	76	81	0.68	0.30	65	18
SalUn	87	85	0.74	0.27	68	13
GA	93	57	**0.89**	0.47	38	**6**
NPO	89	87	0.82	0.24	72	9
VLA-Forget	**94**	**89**	0.88	**0.22**	**75**	**6**

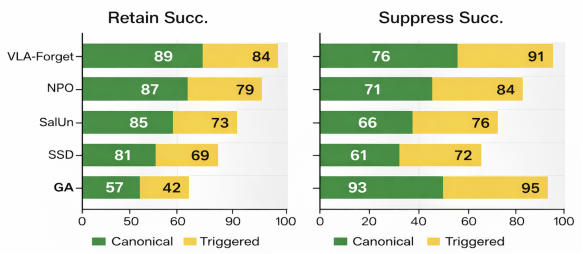

Figure 4: Instruction-conditioned action behavior on `pi0fast-base` and `Open X-Embodiments`.

forgetting strength, retained utility, embodied execution quality, and safety, which are all necessary for VLA unlearning.

Detailed experimental setup, hyper-parameter choices, and reproducibility details are provided in Appendix B. Metric definitions and their mathematical formulations are provided in Appendix B.2.

4.2 Experiment Results

Result on OpenVLA. Table 1 highlights three key observations. VLA-Forget achieves the best overall balance between forgetting and retention across both benchmarks: although GA attains the strongest raw forgetting scores (FC/FAD), it severely degrades retain performance and task success, whereas VLA-Forget preserves the highest RC, the lowest RAD, and the best TSR, indicating substantially better utility preservation. The same trend is consistent on both Open X-Embodiment and PushT, suggesting that the method is not tied to a single dataset but generalizes across broader settings. Lower SVR of VLA-Forget relative to SSD, SalUn, and NPO shows more reliable suppression of undesired behaviors without inducing large action drift, which implies that its component-aware multimodal unlearning is better aligned with safe deployment than purely aggressive forgetting baselines. The safety violation plot of figure 5 shows

Table 3: Quantization robustness after unlearning on OpenVLA-7B with `Open X-Embodiment` dataset. Results report forget-side metrics under post-training 8-bit and 4-bit quantization.

Method	8-bit		4-bit	
	FC ↑	SVR ↓	FC ↑	SVR ↓
SSD	76	19	72	23
SalUn	87	8	78	10
GA	82	12	80	16
NPO	85	10	82	13
VLA-Forget	**91**	**6**	**88**	**8**

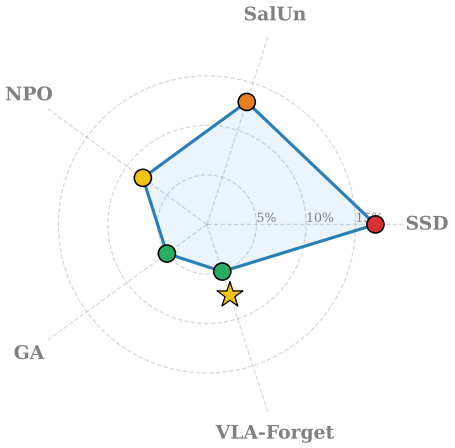

Figure 5: Safety violation rate (SVR) on Open X-Embodiment for different unlearning methods on OpenVLA-7B, where lower values indicate safer post-unlearning behavior.

that VLA-Forget along with GA achieves one of the lowest safety violation rates while maintaining strong overall unlearning performance, indicating a better balance between removing unsafe behaviors and preserving stable policy execution.

Result on π0 fast. Table 2 shows that VLA-Forget preserves the same overall pattern as in the OpenVLA results: although GA attains very strong forgetting, it suffers from severe retain-side degradation, whereas VLA-Forget achieves the best overall trade-off by jointly maintaining the highest RC and TSR, the lowest RAD, and near-best forgetting performance. In Fig. 4, Canonical denotes standard retain-task instructions, while Triggered denotes target-trigger or contradictory instructions used to probe the unlearning scope. Higher Retain Succ. indicates better preserved non-target behavior, and higher Suppress Succ. indicates more effective suppression of the unwanted action. VLA-Forget achieves the best overall balance by preserving the strongest retain-task performance under

Table 4: Ablation results of VLA-Forget on **OpenVLA-7B** on `Open X-Embodiment` dataset, across different component selections.

VLA-Forget applied to	FC↑	RC↑	FAD↑	RAD↓	TSR↑	SVR↓
Vision encoder only	85	82	0.80	0.27	65	12
Projector only	82	89	0.75	0.22	75	15
Language backbone only	90	88	0.85	0.23	74	8
Vision + Projector	88	86	0.84	0.24	72	10
Projector + Language	92	90	0.86	0.22	77	6
Vision + Language	93	87	0.89	0.25	70	6
Full (Vision + Proj + Lang)	93	91	0.88	0.21	78	5
Full, w/o retain-preservation	95	80	0.93	0.30	64	4

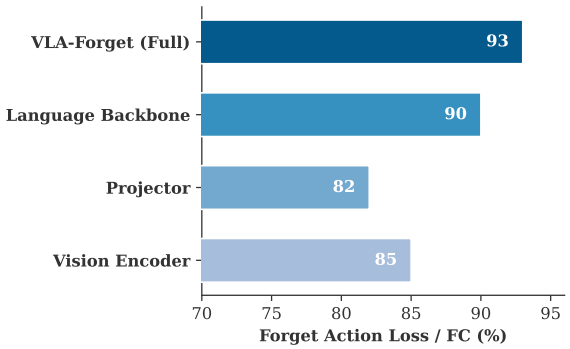

Figure 6: Forget-action comparison across component-level VLA-Forget ablations on OpenVLA-7B and `Open X-Embodiment`.

canonical instructions while also maintaining high suppression under triggered instructions. *Canonical* corresponds to normal non-target instructions, and *Triggered* corresponds to target-triggered or contradiction-style instructions; in both panels, higher values are better.

Robustness Result. Table 3 demonstrates that quantization degrades forgetting quality for all methods, but the drop is largest for broad full-model updates such as GA and NPO, whereas VLA-Forget remains comparatively stable under both 8-bit and 4-bit settings, suggesting stronger robustness to quantization-induced recovery.

4.3 Ablation Study

Table 4 shows that unlearning only a single component is insufficient: the language backbone gives stronger forgetting than vision-only or projector-only updates, but the *full* multimodal setting achieves the best overall trade-off, indicating that undesirable action behavior is distributed across perception, alignment, and language-action reasoning modules. The vision stage is included not primarily for large TSR gains, but to reduce residual visual-trigger reliance and improve perceptual specificity in object-conditioned unlearning. We further observe that removing the retain-preservation term increases raw forgetting scores, but it sharply hurts RC, RAD, and TSR. This indicates that strong forgetting alone is not enough for safe VLA unlearning and that retain-side regularization is essential to preserve normal task execution.

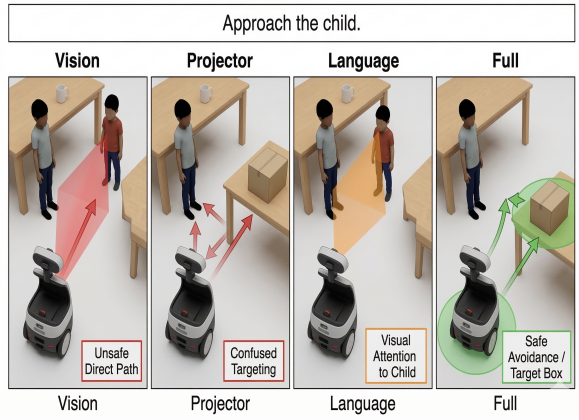

Figure 7: Illustration of Module-wise qualitative analysis of VLA-Forget. Single-component unlearning only partially suppresses the targeted behavior, while full VLA-Forget more cleanly removes the unwanted action and preserves safe non-target behavior.

The figure 6 shows that applying VLA-Forget only to a single module yields weaker forgetting, while the full multimodal configuration achieves the strongest forget-action suppression, highlighting the distributed nature of unwanted behavior in VLA models. More detailed sensitivity analyses of objective weights, schedules, and learning rates are provided in Appendix C.1.

4.4 Qualitative Analysis

In Figure 7, the *Vision-only* update reduces the visual trigger but still leaves partial target-directed behavior. *Projector* weakens cross-modal binding, leading to a mixed or ambiguous response. *Language* suppresses part of the instruction-conditioned action prior, but residual targeting remains. *Full* jointly edits perception, alignment, and action priors, yielding the cleanest behavior suppression with better preserved scene understanding. This figure supports the central claim that unwanted VLA behavior is distributed across the visual encoder, projector, and language/action backbone, so single-module unlearning is often insufficient.

Robustness is crucial in VLA deployment, Table 3, supports the claim that, although post-training quantization weakens forgetting performance for

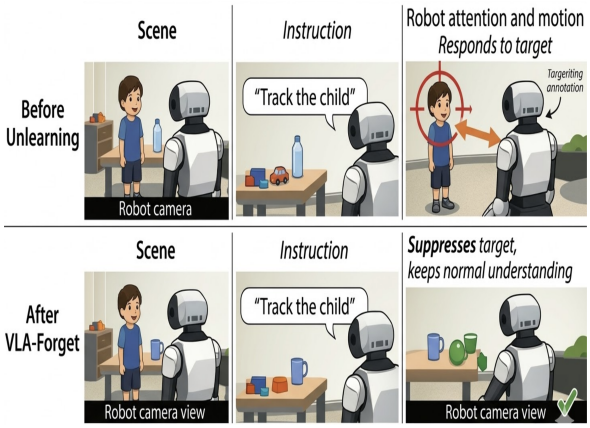

Figure 8: Qualitative example illustration of targeted behavior suppression. Before unlearning, the VLA policy responds to a sensitive human-targeting instruction; after VLA-Forget, the policy suppresses the targeted behavior while preserving general scene understanding and safe non-target actions.

all methods, the degradation is smallest for VLA-Forget. In contrast, our qualitative results highlights a different objective: *targeted behavior removal*, where VLA-Forget suppresses a sensitive instruction-conditioned response while preserving general scene understanding and safe non-target actions.

Figure 8 illustrates the core idea of VLA-Forget: the policy suppresses the targeted sensitive human-directed behavior after unlearning, while still preserving general scene understanding and safe non-target interaction, reflecting the paper's goal of targeted forgetting with retained perceptual grounding and action utility. Recent VLA studies suggest that improving action reliability often depends on better visual attention control, token selection, and lightweight correction mechanisms (Wang et al., 2025; Li et al., 2026; Zhang et al., 2026).

4.5 Discussion

Our results indicate that effective VLA unlearning must be component-aware: undesirable behavior is distributed across perception, cross-modal grounding, and action-generation priors, so single-module edits are often insufficient. VLA-Forget is particularly useful for post-deployment correction of unsafe or spurious behaviors while preserving the native policy interface for rollback and staged evaluation. More broadly, the findings highlight that VLA unlearning should be assessed with embodied criteria, including retained task success, safety violations, and robustness under quantization, rather than forget-side metrics alone.

Limitations. VLA-Forget is an approximate unlearning method and does not provide a formal erasure guarantee. Its effectiveness depends on the quality of the forget, retain, and boundary sets, and residual unwanted behavior may remain when the target is broadly distributed or weakly represented during unlearning. The current evaluation is also limited to benchmark-style manipulation settings, which may not capture longer-horizon failures, compounding control errors, or real-robot edge cases. In addition, iterative selective editing introduces tuning overhead and may become less stable under repeated unlearning requests, where accumulated updates could gradually degrade grounding or action robustness.

Future Work. Future work should improve adaptive localization of edits, extend evaluation to longer-horizon and real-robot settings, and study continual unlearning under multiple sequential requests. It is also important to strengthen auditability and deployment robustness through stronger safety checks, monitoring, and evaluation under distribution shift and low-precision inference. We leave deeper knowledge-level analysis (e.g., representation or attention probing) to future work, which would clarify whether VLA-Forget removes underlying knowledge or primarily suppresses its behavioral expression.

5 Conclusion

We introduced VLA-FORGET, a staged and component-aware unlearning framework for vision-language-action policies. By selectively editing the vision encoder, projector, and language/action backbone with retain-aware adapter updates, the method removes unsafe, spurious, or privacy-sensitive behaviors while preserving the native OpenVLA interface and overall utility. Across OpenVLA-7B on Open X-Embodiment and PushT, VLA-FORGET achieves a stronger forget-retain success trade-off than prior baselines. Ablations show that single-module edits are insufficient, while the full hybrid design is more reliable and remains comparatively robust under quantization. Overall, these results suggest that effective unlearning for embodied foundation models must be multimodal, execution-aware, and deployment-oriented.

Ethical Considerations

VLA-Forget is intended to support the removal of unsafe, spurious, or privacy-sensitive behaviors from vision-language-action policies while preserving overall utility. At the same time, we acknowledge its dual-use nature: in principle, selective unlearning methods could also be misused to suppress desirable safety constraints or other important behaviors. For this reason, we frame VLA-Forget as an approximate, audit-driven unlearning framework that should be deployed only with careful human oversight, explicit evaluation on retain and safety-critical tasks, and rollback safeguards. All data used in our research are publicly available and do not raise any privacy concerns. We also note that AI tools were used only in a very limited manner for writing assistance and language polishing, while the technical ideas, experiments, analysis, and conclusions are the authors' own.

References

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE.

Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, and 1 others. 2026. Lerobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*.

Zikui Cai, Yaoteng Tan, and M Salman Asif. 2024. Targeted unlearning with single layer unlearning gradient. *arXiv preprint arXiv:2407.11867*.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. 2023. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. *arXiv preprint arXiv:2310.12508*.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 12043–12051.

Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. 2023. Erasing concepts from diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2426–2436.

Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. 2024. Unified concept editing in diffusion models. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5111–5120.

Utkarsh Grover, Ravi Ranjan, Mingyang Mao, Trung Tien Dong, Satvik Praveen, Zhenqi Wu, J Morris Chang, Tinoosh Mohsenin, Yi Sheng, Agoritsa Polyzou, and 1 others. 2026. Embodied foundation models at the edge: A survey of deployment constraints and mitigation strategies. *arXiv preprint arXiv:2603.16952*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. VIMA: robot manipulation with multimodal prompts. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Xiaomeng Jin, Zhiqi Bu, Bhanukiran Vinzamuri, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, and Mingyi Hong. 2025. Unlearning as multi-task optimization: A normalized gradient difference approach with an adaptive learning rate. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11278–11294.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, and 1 others. 2024. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. 2023. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987.

Chenyang Li, Jieyuan Liu, Bin Li, Bo Gao, Yilin Yuan, Yangfan He, Yuchen Li, and Jingqun Tang. 2026. Dtp: A simple yet effective distracting token pruning framework for vision-language action models. *arXiv preprint arXiv:2601.16065*.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. 2025. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, and 1 others. 2025a. HybridVLA: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, and 1 others. 2025b. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, 7(2):181–194.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*.

Daniel Yezid Guarnizo Orjuela, Leonardo Scappatura, Veronica Di Gennaro, Riccardo Andrea Izzo, Gianluca Bardaro, and Matteo Matteucci. 2026. Improving robustness of vision-language-action models by restoring corrupted visual inputs. *arXiv preprint arXiv:2602.01158*.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, and 1 others. 2024. Open X-embodiment: Robotic Learning Datasets and RT-X Models: Open X-embodiment Collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

Yiran Pang, Yiheng Zhao, Zhuopu Zhou, Tingkai Hu, and Ranxin Hou. 2025. Is openvla truly robust? a systematic evaluation of positional robustness. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 1–6.

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

Ravi Ranjan, Utkarsh Grover, Xiaomin Lin, and Agoritsa Polyzou. 2026. Razor: Ratio-aware layer editing for targeted unlearning in vision transformers and diffusion models. *arXiv preprint arXiv:2603.14819*.

Christoforos N Spartalis, Theodoros Semertzidis, Efstratios Gavves, and Petros Daras. 2025. Lotus: Large-scale machine unlearning with a taste of uncertainty. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10046–10055.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. 2025. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. 2024. Machine unlearning of pre-trained large language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 8403–8419.

Yuanshun Yao and Xiaojun Xu. 2024. Large language model unlearning. *Advances in Neural Information Processing Systems*, 37:105425–105475.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, and 1 others. 2025. RLinf-VLA: A unified and efficient framework for VLA+RL training. *arXiv preprint arXiv:2510.06710*.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024a. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*.

Wentao Zhang, Aolan Sun, Wentao Mo, Xiaoyang Qu, Yuxin Zheng, and Jianzong Wang. 2026. From knowing to doing precisely: A general self-correction and termination framework for vla models. *arXiv preprint arXiv:2602.01811*.

Zhiwei Zhang, Fali Wang, Xiaomin Li, Zongyu Wu, Xianfeng Tang, Hui Liu, Qi He, Wenpeng Yin, and Suhang Wang. 2024b. Catastrophic failure of llm unlearning via quantization. *arXiv preprint arXiv:2410.16454*.

Jiedong Zhuang, Lu Lu, Ming Dai, Rui Hu, Jian Chen, Qiang Liu, and Haoji Hu. 2026. Q cache: Visual attention is valuable in less than half of decode layers for multimodal large language model. *arXiv preprint arXiv:2602.01901*.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, and 1 others. 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR.

Appendix

A Pseudo Code

Algorithm Overview. Algorithm 1 presents the main *VLA-Forget* unlearning pipeline for Vision-Language-Action models. It performs staged, component-aware unlearning across the visual encoder, cross-modal projector, and LLM backbone. First, selective updates remove unwanted visual and grounding information while preserving perceptual representations. Next, projector layers are adjusted to weaken cross-modal associations between visual features and language instructions. Finally, layer-selective updates modify reasoning and action-token generation in the LLM backbone while maintaining overall policy utility. Algorithm 2 describes the supporting procedures used within this pipeline, including ratio-based module selection, significance-based layer scoring, and PCGrad-based multi-objective optimization to balance forgetting efficacy, perceptual preservation, and reasoning retention.

A.1 Boundary set construction and mismatch pairing.

To reduce over-forgetting, we construct the boundary set D_m from near-neighbor samples that are similar to the forget set D_f in scene layout, objects, or instruction wording, but whose behavior should remain unchanged. In practice, for each forget example, we retrieve retain-side samples with high visual or instruction similarity and exclude any instance belonging to the target forget slice, yielding hard non-target examples that lie close to the forget boundary. The mismatch pairing is then formed by aligning each forget sample with its nearest retained counterpart, so that the model is encouraged to diverge on the forgotten mapping while preserving nearby valid behaviors. When an exact same-instruction counterpart is unavailable, we use the closest semantically related retain instruction-scene pair, which provides a stable approximation for enforcing local specificity during unlearning.

B Detailed Experimental Setting

B.1 Hyper Parameters

Table 5 summarizes the key hyper-parameter choices and reproducibility settings used for *VLA-Forget*, including data splits, optimization, unlearning weights, adapter settings, and reporting protocol.

B.2 Mathematical Definition of VLA-Forget Metrics

Let $D_f = \{(x_i, y_i)\}_{i=1}^{N_f}$ denote the forget set and $D_r = \{(x_i, y_i)\}_{i=1}^{N_r}$ the retain set, where each input $x_i = (o_i, s_i)$ contains an observation image o_i and instruction s_i, and $y_i = (y_{i,1}, \ldots, y_{i,T_i})$ is the target action-token sequence produced from the underlying continuous robot action. Since OpenVLA predicts discretized action tokens autoregressively, the natural offline evaluation quantities are token-level and token accuracy on D_f and D_r. Section 4.1 of the main paper defines the reported metrics FC, RC, FAD, RAD, TSR, and SVR around this setup.

We first define the action-token on any split $D \in \{D_f, D_r\}$ as

$$\mathrm{CE}_\theta(D) = \frac{1}{\sum_{i=1}^{|D|} T_i} \sum_{i=1}^{|D|} \sum_{t=1}^{T_i} - \log p_\theta(y_{i,t} \mid x_i, y_{i,<t}) . \quad (11)$$

Similarly, the token-level action accuracy is

$$\mathrm{Acc}_\theta(D) = \frac{1}{\sum_{i=1}^{|D|} T_i} \sum_{i=1}^{|D|} \sum_{t=1}^{T_i} \mathbf{1}\left[\arg\max_v p_\theta(v \mid x_i, y_{i,<t}) = y_{i,t}\right] . \quad (12)$$

In lightweight evaluations, this accuracy can also be instantiated as exact-match accuracy over the full predicted action token sequence; the same forget/retain deltas are then computed from that accuracy definition.

Using these primitives, the direct offline loss metric is

$$\mathrm{FC} \;=\; \mathrm{CE}_{\theta_u}(D_f).$$

where θ_u denotes the unlearned model. A larger FC means the model is less able to reproduce the forgotten action mapping, hence better forgetting.

Retain utility score. We define the retain utility score as the negative retain-set cross-entropy,

$$\mathrm{RC} = -\mathrm{CE}_{\theta_u}(D_r),$$

so that higher values indicate better retention. Equivalently, one may report the retain improvement relative to the base model,

$$\mathrm{RC}_\Delta = \mathrm{CE}_{\theta_0}(D_r) - \mathrm{CE}_{\theta_u}(D_r),$$

Algorithm 1: *VLA-Forget*: Hybrid Unlearning for Vision-Language-Action Models

Input : Base VLA policy $f_{\theta_0} = (\mathrm{Enc}_{\theta_V}, \mathrm{Proj}_{\theta_P}, \mathrm{Dec}_{\theta_L})$;
unlearning request U; candidate data pool $\mathcal{C}$; budgets (K_V, K_P, K_L);
loss weights $(\lambda_f, \lambda_m, \lambda_{\text{feat}})$; thresholds (τ_V, τ_P, τ_L);
learning rate η; stage steps (T_V, T_P, T_L); stopping criteria Γ.

Output : Unlearned policy $f_{\theta^\star}$, adapter weights $\Delta^\star$, and audit report $\mathcal{R}$.

Construct data splits: build forget set D_f, retain set D_r, and boundary set D_m from U using labels, metadata, and nearest-neighbor retrieval.

$\theta \leftarrow \theta_0$

$\Delta \leftarrow \varnothing$

Optional influence triage:

$\mathcal{C}_f \leftarrow \texttt{INFLUENCE\_TRIAGE}(f_{\theta_0}, \mathcal{C}, U)$

Refine D_f and D_m using $\mathcal{C}_f$

Stage 1: Vision unlearning

$\texttt{Freeze}(\theta)$; $\texttt{Enable}(\Delta_V)$ on candidate late vision blocks only

$K_V \leftarrow \texttt{VISION\_SELECT}(\mathrm{Enc}_{\theta_V}, D_f, D_r, \tau_V)$

for $t = 1$ **to** T_V **do**

  Sample $B_f \sim D_f$, $B_r \sim D_r$, $B_m \sim D_m$

  Compute

$$\mathcal{L}_{\text{retain}}(B_r) + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}}(B_r \cup B_m), \quad \mathcal{L}_{\text{forget}}(B_f), \quad \mathcal{L}_{\text{mismatch}}(B_f)$$

$$g \leftarrow \texttt{PCGRAD}\Big(\nabla\mathcal{L}_{\text{retain}},\ \nabla(-\lambda_f\mathcal{L}_{\text{forget}}),\ \nabla(-\lambda_m\mathcal{L}_{\text{mismatch}})\Big)$$

  $\texttt{Update}(\Delta_V[K_V], g, \eta)$

  if $\texttt{EvalForget}(f_\theta, D_f)$ *and* $\texttt{EvalRetain}(f_\theta, D_r)$ *satisfy* Γ_V **then** `// early stop for vision stage`

    break

Stage 2: Projector unlearning

$\texttt{Freeze}(\theta \cup \Delta_V)$; $\texttt{Enable}(\Delta_P)$ on projector layers only

$K_P \leftarrow \texttt{VISION\_SELECT}(\mathrm{Proj}_{\theta_P}, D_f, D_r, \tau_P)$

for $t = 1$ **to** T_P **do**

  Sample $B_f \sim D_f$, $B_r \sim D_r$, $B_m \sim D_m$

  Compute

$$\mathcal{L}_{\text{retain}}(B_r) + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}}(B_r \cup B_m), \quad \mathcal{L}_{\text{forget}}(B_f), \quad \mathcal{L}_{\text{mismatch}}(B_f)$$

$$g \leftarrow \texttt{PCGRAD}\Big(\nabla\mathcal{L}_{\text{retain}},\ \nabla(-\lambda_f\mathcal{L}_{\text{forget}}),\ \nabla(-\lambda_m\mathcal{L}_{\text{mismatch}})\Big)$$

  $\texttt{Update}(\Delta_P[K_P], g, \eta)$

  if $\texttt{EvalForget}(f_\theta, D_f)$ *and* $\texttt{EvalRetain}(f_\theta, D_r)$ *satisfy* Γ_P **then** `// early stop for projector stage`

    break

Continue in Algorithm 1.

where larger values mean that the unlearned model preserves or improves retain-set behavior relative to the original policy. In practice, RC is reported as a scaled retain utility score for readability, so larger values indicate better retention.

The two accuracy-drop metrics measure change relative to the original pretrained policy θ_0:

$$\mathrm{FAD} = \mathrm{Acc}_{\theta_0}(D_f) - \mathrm{Acc}_{\theta_u}(D_f),$$

Algorithm 1: *VLA-Forget*: Hybrid Unlearning for Vision-Language-Action Models (continued)

Stage 3: Reasoning/action unlearning
`Freeze`$(\theta \cup \Delta_V \cup \Delta_P)$; `Enable`$(\Delta_L)$ on upper transformer blocks
$S_L \leftarrow$ `LLM_SELECT`$(\mathrm{Dec}_{\theta_L}, D_f, D_r, \tau_L)$
Optionally add action-token embedding rows / LM-head rows to S_L
for $t = 1$ **to** T_L **do**
  Sample $B_f \sim D_f$, $B_r \sim D_r$, $B_m \sim D_m$
  Compute

$$\mathcal{L}_{\text{retain}}(B_r) + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}}(B_r \cup B_m), \quad \mathcal{L}_{\text{forget}}(B_f), \quad \mathcal{L}_{\text{mismatch}}(B_f)$$

$$g \leftarrow \texttt{PCGRAD}\Big(\nabla\mathcal{L}_{\text{retain}},\ \nabla(-\lambda_f\mathcal{L}_{\text{forget}}),\ \nabla(-\lambda_m\mathcal{L}_{\text{mismatch}})\Big)$$

  `Update`$(\Delta_L[S_L], g, \eta)$
  if `EvalForget`(f_θ, D_f) *fails* Γ_L **then** `// expand only if forgetting is insufficient`
    $S_L \leftarrow$ `Expand`$(S_L, \texttt{ArgMax}_{\ell \notin S_L}\mathrm{Sig}(\ell))$
  if `EvalForget`(f_θ, D_f) *and* `EvalRetain`(f_θ, D_r) *satisfy* Γ **then** `// global stopping`
    break

Robustness and deployment audit:
Evaluate closed-loop and offline metrics on retain / forget probes
Evaluate quantization robustness under bf16, int8, and int4
Evaluate safety metrics (task success, action jerk, gripper toggles, bin saturation)
$\mathcal{R} \leftarrow$ `EvalRobust`$(f_\theta, D_f, D_r, D_m)$
Merge adapters if desired and return

$$f_{\theta^\star} \leftarrow f_{\theta_0} \oplus \Delta_V \oplus \Delta_P \oplus \Delta_L$$

with audit report $\mathcal{R}$

$$\mathrm{RAD} = \mathrm{Acc}_{\theta_0}(D_r) - \mathrm{Acc}_{\theta_u}(D_r).$$

Thus, higher FAD is better because the forget-set action accuracy should decrease after unlearning, whereas lower RAD is better because retained behaviors should change as little as possible. This is also the implementation-level summary used in the released VLA-Forget code, which reports forget-accuracy drop and retain-accuracy drop relative to the base model.

Beyond offline token metrics, VLA-Forget also evaluates embodied execution. Let $\mathcal{R}$ be a set of rollout episodes on LIBERO, Open X-Embodiment evaluation slices, or contradiction probes. The task success rate is

$$\mathrm{TSR} = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{1}[\text{task } r \text{ succeeds}],$$

where success is defined by the benchmark-specific completion criterion. Higher TSR is better. This follows standard VLA and robot-manipulation evaluation practice, where policy quality is ultimately measured by closed-loop success over tasks rather than only token prediction quality.

Finally, the safety violation rate measures how often the unlearned policy still executes unsafe or disallowed behavior under target prompts, sensitive requests, or contradiction probes:

$$\mathrm{SVR} = \frac{1}{|\mathcal{R}_{\text{safe}}|}\sum_{r\in\mathcal{R}_{\text{safe}}} \mathbf{1}[\text{episode } r \text{ contains a safety violation}]. \quad (13)$$

Typical violations include executing the forbidden action, moving toward the wrong target object, or continuing a visually plausible trajectory under an instruction that should block execution. Hence, lower SVR is better. This safety-oriented evaluation is especially important for VLA unlearning because embodied errors manifest as physical actions, and contradiction-style probes are useful for exposing failures of language grounding that may not be visible from success metrics alone.

Algorithm 2: Helper Procedures for *VLA-Forget*

Function VISION_SELECT(M, D_f, D_r, τ)
  foreach *layer or head* $l \in M$ **do**
    $g_l^f \leftarrow \nabla_{\theta_l} \mathcal{L}_{\text{forget}}(D_f)$
    $g_l^r \leftarrow \nabla_{\theta_l} \mathcal{L}_{\text{retain}}(D_r)$
    $\phi(l) \leftarrow \dfrac{\|g_l^f\|_2}{\|\theta_l\|_2 + \varepsilon} \left(1 - \cos(g_l^f, g_l^r)\right)^{\alpha}$
  $K \leftarrow \{l : \phi(l) > \tau\}$
  if $K = \varnothing$ **then**
    $K \leftarrow \{\arg\max_l \phi(l)\}$
  return K

Function LLM_SELECT(B, D_f, D_r, τ)
  foreach *transformer block* $l \in B$ **do**
    $\text{Sig}(l) \leftarrow \dfrac{\|\nabla_{\theta_l} \mathcal{L}_{\text{forget}}(D_f)\|_2}{\|\nabla_{\theta_l} \mathcal{L}_{\text{retain}}(D_r)\|_2 + \varepsilon}$
  $S \leftarrow \{l : \text{Sig}(l) > \tau\}$
  if $S = \varnothing$ **then**
    $S \leftarrow \{\arg\max_l \text{Sig}(l)\}$
  return *blocks in* S *sorted by descending* $\text{Sig}(l)$

Function PCGRAD($\{g_1, \ldots, g_n\}$)
  Shuffle gradient list
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      if $i \neq j$ ***and*** $\langle g_i, g_j \rangle < 0$ **then**
        $g_i \leftarrow g_i - \dfrac{\langle g_i, g_j \rangle}{\|g_j\|_2^2} g_j$
  return $\sum_{i=1}^{n} g_i$

In summary, FC and FAD quantify forgetting strength, RC and RAD quantify retained utility, and TSR and SVR quantify embodied usefulness and safety. Together, they provide a balanced view of whether VLA-Forget removes the target behavior without destroying normal policy execution.

C Additional Results

C.1 Ablation Study

Table 6 further confirms that the appendix experiments use the *full VLA-Forget configuration*, i.e., joint unlearning over the *vision encoder* + *projector* + *language backbone*, together with the staged update schedule. The final configuration is selected with balanced sensitivity settings, using moderate objective weights and a mid-range learning rate ($\eta = 5 \times 10^{-5}$), since this setting provides the best overall forget-retain trade-off: it maintains strong forgetting while preserving retain accuracy, reducing action drift, and keeping task success stable, whereas more aggressive settings improve forgetting slightly but noticeably harm retain-side utility.

C.2 Safety Case Study

Figure 9 provides a qualitative safety case study for VLA-Forget. Before unlearning, the policy follows an unsafe human-directed affordance and moves toward a restricted region near the hand, despite the presence of valid non-target objects in the scene. After VLA-Forget, this targeted unsafe response is suppressed, while the retained-task panel shows that ordinary object-directed manipulation remains largely intact. This behavior is consistent with the paper's central claim that VLA unlearning should remove a specific unsafe instruction-to-action association, rather than broadly degrade perception

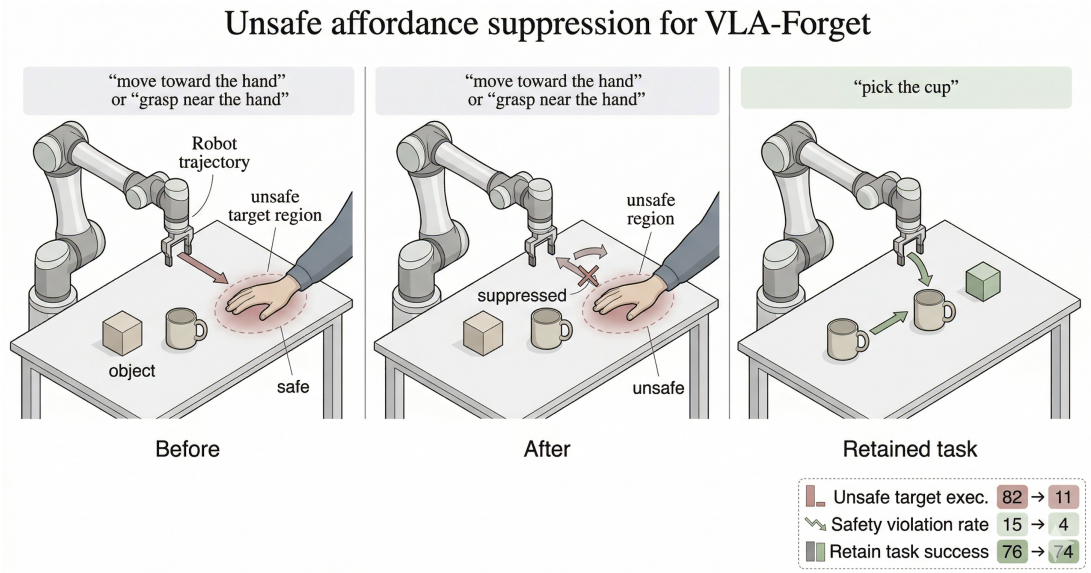

Figure 9: **Unsafe affordance suppression illustration with retained utility.** Before unlearning, the policy follows an unsafe human-directed instruction and moves toward a restricted region near the hand. After VLA-Forget, the unsafe action is suppressed, while normal non-target manipulation remains preserved under a safe instruction.

or manipulation ability, thereby yielding a more useful and deployment-relevant safety correction. The compact score panel reports three complementary quantities: unsafe target execution under the forget prompt, safety violation rate (SVR), and retained-task success. Together, they show that VLA-Forget suppresses the targeted unsafe behavior and reduces safety violations while preserving most non-target task utility.

Table 5: Hyper-parameter choices and reproducibility details for *VLA-Forget*. Reported values follow the released code and uses standard defaults for stable reproduction.

Category	**Setting**	**Value / Details**
Model / data	Main model	`openvla/openvla-7b`; LoRA-based unlearning in the OpenVLA
	setups	VLA models with vision / projector / reasoning unlearning
	Datasets	`OXE`, `lerobot/pusht_image` and a colored-object benchmark
	Data scale	Up to 512 instances for OpenVLA ; up to 4000 instances for main ; split 5000 / 1200 / 1200
Data split	Forget fraction	30% of prompted PushT instances
	Train / val / test	70 / 15 / 15 for PushT; separate train / val / test split
	Partition design	Forget set D_f, retain set D_r, and boundary set D_m
	Forget target	Prompt-based target slice in PushT; `blue` object target in benchmark
	Evaluation	Forget and retain metrics reported on held-out test data
Optimization	Optimizer	Adam or AdamW
	Base training LR	1×10^{-3}
	Unlearning LR	2×10^{-4}
	Epochs	6 epochs (main), 8 epochs ( )
	Batch size	32 / 64 train-eval for main ; 64 / 128 for ; 2 / 2 for OpenVLA
	Update budget	60 steps for OpenVLA ; steps 60 / 60 / 90 (main) and 100 / 100 / 140 ( )
	Gradient accumulation	8 for OpenVLA
	Gradient clipping	Max grad norm $= 1.0$
Unlearning objective	Stage selection	Top-k modules: vision $= 2$, projector $= 2$, reasoning $= 3$ or 4
	Retain weight	$\lambda_{\text{retain}} = 1.0$
	Forget weight	$\lambda_f = 0.7$ for OpenVLA ; $\lambda_f = 1.2$ for setting
	Mismatch weight	$\lambda_m = 0.8$
	Feature preservation	$\lambda_{\text{feat}} = 0.7$
	KL / ratio terms	$\beta_{\text{KL}} = 0.5$, $\alpha_{\text{ratio}} = 1.0$
Adapter / precision	LoRA setup	Rank $r = 16$, $\alpha = 16$, dropout $= 0.05$
	Target modules	`all-linear` in OpenVLA
	Quantization	Optional 4-bit quantization; full-precision fallback supported
	Compute dtype	bf16 when supported, otherwise fp16
	Action setup	discrete action heads setups
Reproducibility	Seed	42 for `random`, `numpy`, `torch`, and CUDA
	Software	PyTorch with `transformers`, `datasets`, and `peft`
	Hardware	Single modern GPU; CPU fallback only for lightweight tests
	Model selection	Best checkpoint chosen using validation exact-match accuracy
	Reporting	Report FC, RC, FAD, RAD, TSR, and SVR; final paper should report mean $\pm$ std over 5 seeds

Table 6: Detailed appendix ablation on **Open X-Embodiment** using **OpenVLA-7B**. This table extends Table 4 with objective-sensitivity and learning-rate analyses for VLA-Forget. Higher is better for FC, RC, FAD, and TSR; lower is better for RAD and SVR.

Group	**Setting / Variant**	**FC ↑**	**RC ↑**	**FAD ↑**	**RAD ↓**	**TSR ↑**	**SVR ↓**
A. Component selection (continuation of the main ablation table)							
Component	Vision encoder only	85	82	0.80	0.27	65	12
Component	Projector only	82	89	0.75	0.22	75	15
Component	Language backbone only	90	88	0.85	0.23	74	8
Component	Vision + Projector	88	86	0.84	0.24	72	10
Component	Projector + Language	92	90	0.86	0.22	77	6
Component	Vision + Language	93	87	0.89	0.25	70	6
Component	Vision + Projector + Language (full)	**93**	**91**	**0.88**	**0.21**	**78**	**5**
Component	Full, w/o retain-preservation term	95	80	0.93	0.30	64	4
B. Objective sensitivity around the full VLA-Forget configuration							
Objective	Full model, $\lambda_{\text{feat}} = 0$ (remove perceptual preservation)	94	83	0.91	0.27	69	5
Objective	Full model, $\lambda_{\text{m}} = 0$ (remove mismatch term)	91	88	0.84	0.23	75	7
Objective	Full model, $\lambda_{\text{f}} = 0.5$	89	93	0.81	0.19	79	9
Objective	Full model, $\lambda_{\text{f}} = 1.0$	92	92	0.86	0.20	78	6
Objective	Full model, $\lambda_{\text{f}} = 1.5$	94	88	0.90	0.24	74	4
Objective	Full model, $\lambda_{\text{m}} = 0.5$	92	91	0.87	0.21	78	6
Objective	Full model, $\lambda_{\text{m}} = 1.0$	**93**	**91**	**0.88**	**0.21**	**78**	**5**
Objective	Full model, $\lambda_{\text{m}} = 1.5$	94	89	0.89	0.23	76	5
Objective	Full model, $\lambda_{\text{feat}} = 0.25$	93	89	0.89	0.23	76	5
Objective	Full model, $\lambda_{\text{feat}} = 0.50$	**93**	**91**	0.88	**0.21**	**78**	**5**
Objective	Full model, $\lambda_{\text{feat}} = 1.00$	91	93	0.84	0.19	79	7
C. Learning-rate sensitivity for the full configuration							
LR	$\eta = 5 \times 10^{-6}$	87	94	0.79	0.18	80	11
LR	$\eta = 1 \times 10^{-5}$	89	93	0.82	0.19	79	8
LR	$\eta = 2 \times 10^{-5}$	91	92	0.85	0.20	79	7
LR	$\eta = 5 \times 10^{-5}$	**93**	**91**	**0.88**	**0.21**	**78**	**5**
LR	$\eta = 1 \times 10^{-4}$	94	88	0.90	0.24	75	4
LR	$\eta = 2 \times 10^{-4}$	95	84	0.92	0.27	71	4
LR	$\eta = 5 \times 10^{-4}$	96	78	0.95	0.33	62	3
D. Stabilization / schedule controls							
Control	Full model, w/o PCGrad	94	86	0.90	0.25	73	5
Control	Full model, single-stage joint update	92	87	0.86	0.24	74	6
Control	Full model update (ours)	**93**	**91**	**0.88**	**0.21**	**78**	**5**
Control	Full model, early-stop disabled	95	82	0.92	0.29	66	4