When Does Privileged Self-Distillation Help GUI Grounding? A Teacher-Gap Analysis of Visual SDPO

Ethan Morgan

CS224R Final Project, Spring 2026
June 8, 2026

Extended Abstract

Problem. GUI grounding means taking a screenshot and an instruction and outputting the exact pixel to click. It is the bottleneck for computer-use agents, and current vision–language models (VLMs) are bad at it, especially on dense, high-resolution software. I wanted to know whether a model could teach *itself* to ground better through privileged self-distillation (SDPO). The setup: during training, a "teacher" copy of the model sees the screenshot with the target region visually *spotlighted*, the "student" sees the clean screenshot, and I train the student to match the teacher using only its own unprivileged input. At test time the spotlight is gone and the student is on its own.

What I did. SDPO has only ever been done in text, and there is no public implementation of a *visual* version anywhere—so the first thing I had to do was build it. I implemented visual SDPO on Qwen3-VL-8B in `verl` from scratch: a same-weights teacher that re-scores the student's own tokens under a visually annotated image, on-policy, against a reverse-KL objective, with a sign-correctness test to verify the gradient pushes the student toward (not away from) the teacher. To the best of my knowledge this is the first visual privileged self-distillation for GUI grounding with an open implementation. To know whether it actually worked, I ran the same recipe as a ladder of methods—base, SFT, GRPO, SDPO from base, and SDPO used to refine an SFT model—all through one held-out evaluation harness on ScreenSpot (everyday UIs) and ScreenSpot-Pro (dense 4K software).

What I found. The primary finding is that visual SDPO underperformed SFT baselines. SFT beats SDPO on every benchmark cell, and even plain GRPO beats SDPO trained from base. SDPO from base improves ScreenSpot ($85.1 \to 90.2$) but actually *degrades* ScreenSpot-Pro ($46.7 \to 42.0$). The one place SDPO helps a strong model is as a refinement step on top of SFT ($92.2 \to 94.1$ on ScreenSpot), and that gain is real in the sense that plain SFT plateaus at exactly 92.16 even with 5× the compute—but on a single N=255 split the +1.96pp gap is not statistically significant ($p \approx 0.22$), so I only claim it as directional.

The result I am actually proud of is a cheap, eval-only **teacher-gap probe**. Before training anything, I check whether the spotlight teacher out-grounds the clean student. It does on ScreenSpot (+7.8 points) and it does *not* on ScreenSpot-Pro (−8.8 points), both significant at $p < 0.01$. That sign predicts the sign of the SDPO training effect on both benchmarks: SDPO helps exactly where the teacher is better and hurts where the teacher is worse. On hard targets the spotlight gets downsampled away and ends up misleading the teacher, so SDPO distills the student toward a policy that is worse than where it started. I also checked whether grounding training quietly damaged general capability; at full benchmark size it did not (MMLU, ARC, HellaSwag, GSM8K all within ±1pp of base, for both SFT and SDPO), so SDPO has no retention edge here either.

Why I think this happens. My explanation is that the problem is structural, not a bug. When the privileged signal is just the ground-truth label drawn onto the image, the best possible teacher

is one that already knows the label—which is exactly SFT. So on a task where the answer is a closed-form check ("is this point inside the box?"), privileged distillation collapses toward SFT, and a neural teacher is only a lossy version of it. The concurrent GUI-SD paper gets its visual teacher to work by engineering it up to near-perfect accuracy, which is the same thing as pushing it back toward the label oracle. The method should have real headroom where the answer is *not* closed-form—code, or multi-step trajectories—and the teacher genuinely knows something the student cannot reconstruct from a single label.

Contributions. (i) a controlled SFT/GRPO/SDPO comparison for VLM grounding under one harness; (ii) the teacher-gap probe, a cheap test that predicts before training whether privileged distillation will help; (iii) the closed-form-teacher argument for why it reduces to SFT here; and (iv) a reproducibility finding—one vLLM CUDA-graph flag silently corrupted the vision path and made the training reward read $\sim$6% while the model truly scored $\sim$84%.

Abstract

GUI grounding—mapping a screenshot and an instruction to the pixel to click—is the bottleneck for computer-use agents, and VLMs are weak at it on dense interfaces. I build the first visual privileged self-distillation (SDPO) for grounding: a same-weights teacher sees the target spotlighted, the student sees the clean screenshot, and the student is trained to match the teacher and then deployed without the spotlight. On Qwen3-VL-8B over ScreenSpot and ScreenSpot-Pro, visual SDPO does not beat supervised fine-tuning—SFT and even GRPO win, and SDPO from base degrades the hard benchmark. I explain why with a cheap, eval-only teacher-gap probe whose sign predicts the SDPO effect before training (the spotlight teacher beats the clean student by +7.8pp on ScreenSpot but is −8.8pp worse on ScreenSpot-Pro, both $p < 0.01$), and argue that when the privilege is the rendered label, privileged distillation reduces toward SFT on closed-form-verifiable tasks. The reusable result is the probe; the method's headroom is on tasks whose answers are not closed-form.

1 Introduction

GUI grounding is the task of mapping a screenshot x and an instruction like "click the Settings icon" to the pixel coordinate to click. It is the capability everything else in a computer-use agent sits on top of: if the model cannot reliably turn intent into the right click, no multi-step behavior works. Current VLMs are still weak at it, and they fall apart on dense professional software at 4K, where the target is a tiny fraction of the screen.

The method I set out to test is self-distillation with privileged feedback (SDPO) [1], adapted to grounding with a privilege that is purely visual. During training, a teacher copy of the model sees the screenshot with the answer region spotlighted, the student sees the clean screenshot, and I train the student to match the teacher. The privilege only exists at train time, which makes this a form of learning under privileged information [3]. The hope was that if the spotlight made the teacher localize better, the student would learn where to look from it and keep that ability once the spotlight was gone.

That is not what happened. Visual SDPO does not beat plain supervised fine-tuning, and on hard targets it makes the model worse. What I think is worth reporting is *why*, and a cheap way to predict it in advance. Concretely, I contribute:

- **The first visual SDPO implementation for GUI grounding.** SDPO has only been demonstrated in text, and no open-source visual version exists; I built the same-weights visual teacher, the annotated-image re-scoring path, and the on-policy reverse-KL objective in `verl` from scratch, with a sign test verifying correctness (§3).
- A controlled comparison—base, SFT, GRPO, SDPO from base, and SDPO-as-refinement—on ScreenSpot and ScreenSpot-Pro under one held-out harness (§5.1).
- The **teacher-gap probe**: an eval-only test of whether the privileged view out-grounds the clean view, whose sign predicts the sign of the SDPO training effect on both benchmarks (§5.3).
- The "closed-form-teacher ceiling" argument for why a label-derived privilege collapses toward SFT on this kind of task (§6; I present it as analysis, not a proven result).
- A reproducibility finding: the vLLM `enforce_eager` flag silently corrupted Qwen3-VL's vision computation, which cost me several early runs before I caught it (§5.5).

2 Related Work

SDPO [1] introduces self-distillation with privileged feedback for language models: a teacher re-scores the student's own tokens under feedback (a successful sibling rollout, environment feedback, or a correct solution provided *as in-context text*—not as a hard label target), minimizing a reverse KL with the full per-token distribution. Notably their off-policy variant that supervises on successes

directly underperforms the distillation form. **GUI-SD** [4] is concurrent work applying on-policy self-distillation to GUI grounding on the same model family, using a *visual* privilege (red ground-truth box, Gaussian soft-mask spotlight, and a text hint), reverse KL, a full-distribution per-token objective, and an EMA teacher. They report that a text-coordinate teacher collapses toward a one-hot distribution (entropy $\approx$ SFT) and engineer a stronger visual annotation to keep the teacher near-perfect; they do not report a head-to-head SFT-vs-distillation *accuracy* baseline, which I run here. **LUPI** [3] frames train-time-only privileged information.

How my setup differs, and why it is harder. Two things. First, I estimate the reverse-KL objective with a single sampled token per position (a score-function / REINFORCE estimator), where both papers use the full per-token vocabulary distribution. Mine is the same objective but a noisier, higher-variance way to estimate it. Second, my privilege is purely visual, with no text channel at all. Both choices put me in a strictly lower-information, higher-variance regime than the published work, which matters for how I read my own negative result (§6).

3 Method

3.1 Visual SDPO

The teacher and student are the *same weights*; the only thing that changes is the input image. The teacher does not generate anything—it re-scores the student's own sampled response y under the annotated image:

$$A_t = \big(\log \pi_\theta(y_t \mid x^{\text{ann}}, y_{<t}) - \log \pi_\theta(y_t \mid x^{\text{clean}}, y_{<t})\big).\text{detach}(), \tag{1}$$

a per-token advantage fed to an on-policy PPO surrogate. I enforce `ppo_epochs`=1 and `mini_bs`=`train_bs` at runtime, so $\pi_{\text{old}} \equiv \pi_{\text{student}}$ and the update is a score-function gradient of the reverse KL $D_{\text{KL}}(\pi_{\text{student}} \,\|\, \pi_{\text{teacher}})$. I clip $|A_t| \le 8$ for stability. I also gate distillation to "parsed-but-wrong" rollouts (reward band $0.05 < r < 0.95$), so the teacher signal is applied to genuine grounding misses and not to format errors (a spotlight cannot teach JSON syntax) or to already-correct clicks (distilling on those just drags a high-base model off-distribution).

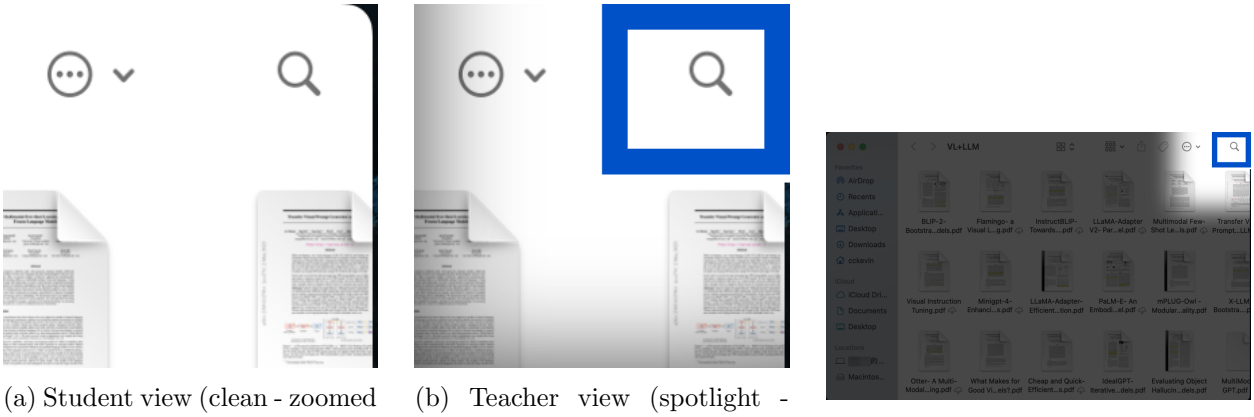
The gate bounds are deliberately loose rather than finely tuned: the lower bound 0.05 only needs to sit above the score of an unparseable output, and the upper bound 0.95 below the score of a clean hit, so any threshold separating the three reward bands (unparseable ≈ 0, parsed-but-wrong $= 0.1$, hit $= 1.0$) behaves the same. Early runs without the upper bound—distilling on *every* non-zero rollout, including correct ones—visibly drifted a high-base model off-distribution, which is what motivated gating to the middle band in the first place; the exact numeric cutoffs within that band did not matter.

3.2 The spotlight privilege

The teacher's screenshot has the ground-truth bounding box outlined and everything outside a window around it dimmed to $\sim$28% brightness, desaturated, and feathered (a soft vignette; Figure 1).

3.3 The teacher-gap probe

SDPO can only help if the privileged teacher is actually a better grounder than the student—otherwise there is nothing good to distill. I test this directly and without any training: I run the base model twice, once on the clean image (the student) and once on the spotlight image (the teacher), and let each generate a click. The gap $\Delta = \text{TeacherHit\%} - \text{StudentHit\%}$ tells me whether the privilege is helping at all. It turned out to be the most useful thing I built.



(a) Student view (clean - zoomed in for demonstration)

(b) Teacher view (spotlight - zoomed in for demonstration)

(c) Spotlight, full screen

Figure 1: The visual privilege. The instruction targets the search icon. **(a)** The student sees the clean crop—the target sits among near-identical toolbar icons, so localization is genuinely ambiguous. **(b)** The teacher sees the *same* crop with the target boxed and the surroundings dimmed/desaturated. **(c)** The full screen shows how small the target is relative to the interface, which motivates the difficulty of dense, high-resolution targets (§5.3). Only the image differs between student and teacher; the weights and prompt are identical.

4 Experimental Setup

Model. Qwen3-VL-8B-Instruct, which emits clicks as a `computer_use` tool call with coordinates in a normalized 0–1000 space. **Benchmarks (held-out, no training overlap).** ScreenSpot (SS; everyday UIs, N=255; base $\approx$ 85%, already near-ceiling) and ScreenSpot-Pro (Pro; dense 4K professional software, tiny targets around 0.07% of screen area, N=317; base $\approx$ 47%). **Training.** SFT is next-token cross-entropy on the bbox-center click; GRPO and SDPO use the same data, batch size, learning rate (10^{-6}), and step budget so the only variable is the objective. SDPO runs on-policy (`ppo_epochs`=1) with the gate and advantage clip of §3. Infrastructure: `verl` 0.7.1, FSDP2, vLLM rollouts, 8×B200; `enforce_eager=True` is mandatory (§5.5). **Evaluation.** Greedy decode, scored by strict point-in-bbox, with a ±14px tolerance reported alongside. The teacher-gap probe (§3.3) runs the base model twice (clean vs. spotlight) on the same held-out splits.

5 Results

All numbers are held-out, strict point-in-bbox, greedy. SS N=255, Pro N=317.

5.1 Quantitative evaluation

SDPO from base improves ScreenSpot by +5.1 over base but degrades ScreenSpot-Pro by −4.7. The ScreenSpot gain is also modest in context: base is already near 85%, and this is a single seed. The only configuration where SDPO helps a model that is already strong is when I use it to refine an SFT checkpoint (SDPO-on-SFT, 94.12, the best ScreenSpot number in the table). Everywhere else, plain SFT is the thing to beat, and SDPO does not beat it.

5.2 Is the refinement gain just more SFT?

SDPO-on-SFT starts from an SFT checkpoint and trains further, so the obvious objection is that I just trained the winner longer—plain SFT might also reach 94 given more epochs. I checked this directly by training SFT longer and seeing whether it climbs to SDPO's level or flattens out below it (Table 2).

Model	Trained on	ScreenSpot	ScreenSpot-Pro
Base Qwen3-VL-8B	—	85.10	46.69
SDPO (spotlight, from base)	SS	90.20	41.96
SDPO (from base)	SS+Pro	84.31	41.96
GRPO	SS	91.76	50.47
SFT	SS	92.16	57.10
SFT	Pro	93.33	**65.30**
SFT	SS+Pro	93.33	64.67
SDPO-on-SFT	SFT-SS→SDPO	**94.12**	57.41

Table 1: Held-out strict accuracy. SFT $\geq$ GRPO $\geq$ from-scratch SDPO on every cell. From-scratch SDPO helps SS (+5.1) but hurts Pro (−4.7). SDPO only adds value as refinement (SDPO-on-SFT, best SS).

SFT-SS variant	ScreenSpot
1 epoch (baseline)	92.16
5 epochs (5× compute)	92.16
SDPO-on-SFT (SFT + 75 SDPO steps)	**94.12**

Table 2: Plain SFT *plateaus at exactly* 92.16 (235/255): five times the SFT compute yields the identical score, zero movement. So SDPO-on-SFT's 94.12 is *not* an under-training artifact reachable by longer supervised training—it is a teacher-signal effect. I state this carefully: the robust claim is "SFT does not improve past 92.16 with 5× epochs," and SDPO-on-SFT is the only arm to exceed it; the +1.96pp gap itself is inside the single-split noise band (N=255, two-proportion $z \approx 1.2$, $p \approx 0.22$), so I report it as directional rather than significant.

5.3 The teacher-gap probe predicts the SDPO sign

Benchmark	student	teacher	gap Δ	discordant (T/S)	SDPO outcome
ScreenSpot	85.10	92.94	**+7.84**	30 / 10	helped (+5.1)
ScreenSpot-Pro	46.69	37.85	**−8.83**	28 / 56	hurt (−4.7)

Table 3: The probe's sign predicts the SDPO training effect on both benchmarks. Exact two-sided binomial test on discordant pairs (equivalent to McNemar's exact test): SS $p = 0.0022$, Pro $p = 0.0030$—both significant at $p < 0.01$. On Pro the spotlight makes the *base* model worse; median click error nearly doubles (0.019→0.037).

On ScreenSpot the spotlight helps the base model localize (+7.84); on ScreenSpot-Pro it hurts (−8.83). My read on the mechanism is that on tiny 4K targets the dimmed, feathered annotation destroys exactly the screenshot detail the model needs, and what survives gets downsampled away by the processor, so the hint ends up pointing the teacher at the wrong place. SDPO then pulls the student toward a teacher that is worse than the student to begin with, which is precisely why it degrades Pro. The useful part is that the probe is eval-only and needs no training, so it is a cheap precondition check before committing compute to any privileged-distillation run.

5.4 Capability retention

I also wanted to know whether grounding training quietly eroded the model's general ability, and whether SDPO forgets less than SFT (there is a common intuition that RL-style objectives preserve

prior capability better). I scored each checkpoint on text-only benchmarks at full size (multiple choice by answer-choice log-probability, GSM8K by generation), using a custom log-probability scorer because the standard harness refuses to load the Qwen3-VL config as a causal LM (§5.5).

Model	MMLU	ARC-easy	ARC-chal.	HellaSwag	GSM8K
base	74.83	85.40	58.45	53.70	77.86
SFT-SS	74.80	85.65	59.47	53.53	78.85
SDPO-SS	74.74	85.48	58.62	53.80	77.86
SDPO-on-SFT	74.86	85.61	59.30	53.53	78.85
SFT-mixed	74.90	85.65	59.81	53.70	78.77

Table 4: Full-N retention (MMLU 14042, ARC-easy 2376, ARC-challenge 1172, HellaSwag 3000, GSM8K 1319). **Honest null:** GUI-grounding fine-tuning (SFT, SDPO, and SDPO-on-SFT alike) causes no measurable off-task degradation—every checkpoint sits within ±1pp of base on every benchmark—and no model collapses to tool-call-only output. A small GSM8K gap seen at n=50 in a pilot did *not* survive at full N (all checkpoints cluster 77.9–78.9%). Thus SDPO shows *no* retention advantage over SFT here. This is itself consistent with the closed-form-teacher view (§6): when the privilege is the rendered label, SDPO and SFT learn—and forget—essentially the same thing.

5.5 Reproducibility: the broken ruler

This introduced significant debugging overhead and is documented here for reproducibility. With vLLM's default `enforce_eager=False`, CUDA-graph capture corrupted Qwen3-VL's vision/mrope computation. The symptom was that the training reward read about 6% while the exact same checkpoint, evaluated normally, scored about 84%. I was measuring the model with a broken ruler. Setting `enforce_eager=True` fixed it, with a byte-identical A/B to confirm. It invalidated several early runs before I figured out what was happening, so I flag it for anyone else training Qwen3-VL under vLLM rollouts.

5.6 Qualitative analysis

The quantitative sign-flip (teacher helps on SS, hurts on Pro) shows up in two places: the training dynamics and the annotated images themselves.

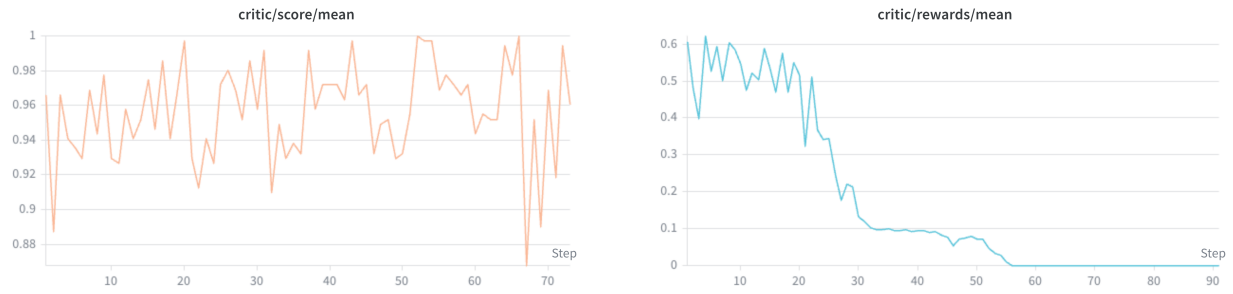
Looking at the annotated images the teacher actually sees makes the cause obvious.

I inspected the rollouts behind the numbers. On ScreenSpot the spotlight rescues clear cases: of the discordant examples, the teacher fixes 30 that the clean student misses and only breaks 10. On ScreenSpot-Pro the ratio inverts—the teacher fixes 28 but breaks 56—and the failure mode is visible in the clicks: under the spotlight the model often clicks into the dimmed region or at the edge of the lit window rather than the target, and its median click error nearly doubles ($0.019 \rightarrow 0.037$). This is the qualitative signature of distilling toward a misled teacher. I also confirmed the fine-tuned checkpoints still answer normally on text-only prompts (no collapse into emitting only `computer_use` tool-calls), which is why the retention numbers in §5.4 are meaningful rather than an artifact of format lock-in.

6 Discussion: the closed-form-teacher ceiling

This section is my interpretation of the results, not something I proved experimentally.

The way I came to think about it is this: a privileged teacher can only help to the extent it knows something the student cannot recover on its own *and* that you cannot just compute in closed form from the label. For grounding, the privilege is the ground-truth box drawn onto the image,



(a) Healthy run (stays ≈ 0.95)

(b) Pro collapse (reward $\to 0$)

Figure 2: Training reward over steps (W&B `critic` reward). **(a)** A run where the teacher is trustworthy holds a high reward throughout, fluctuating around 0.95. **(b)** The ScreenSpot-Pro spotlight run starts near 0.6, holds for $\sim$20 steps, then collapses: the reward decays through step $\sim$30, bottoms out, and *flatlines at* 0 from step $\sim$57 onward. This is the optimization signature of distilling toward a teacher that is worse than the student (§5.3)—once the policy is pulled off-distribution, format breaks and reward stays pinned at zero. The collapse is gradual and early, not a single bad step, which is why a cheap pre-training probe is worth more than watching the training curve.

and "is the click inside the box?" is a closed-form check. So the best teacher you could possibly have is one that already knows the label, and a teacher that already knows the label is just SFT. My neural spotlight teacher is a lossy version of that oracle, and on hard targets it is worse than no privilege at all (-8.83 on Pro). That predicts SFT $\geq$ SDPO, which is exactly what I see. It also lines up with GUI-SD's own analysis: their text-coordinate teacher collapses toward a one-hot distribution (effectively SFT), and the way they make it work is by engineering a stronger visual annotation, which is the same as pushing the teacher back toward knowing the label.[1]

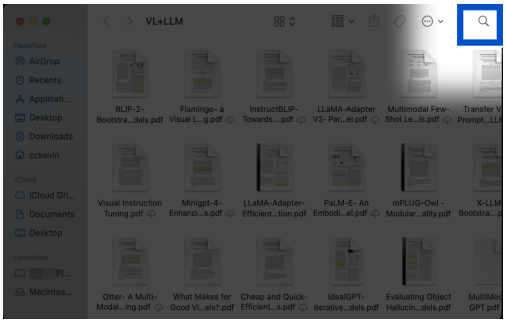
Pushed to its conclusion, this is a version of the Bitter Lesson [2]: the optimal teacher for a closed-form task is a hand-built verifier, which is exactly the kind of task-specific structure that does not scale. So I do not think single-frame grounding is the right place for privileged self-distillation. It should pay off where the answer is *not* closed-form and the privileged signal is something the student genuinely cannot reconstruct from one label—running the tests in code repair, or future steps in a multi-step trajectory. The teacher-gap probe carries over to those settings unchanged, as a cheap go/no-go check before you spend compute.

7 Limitations

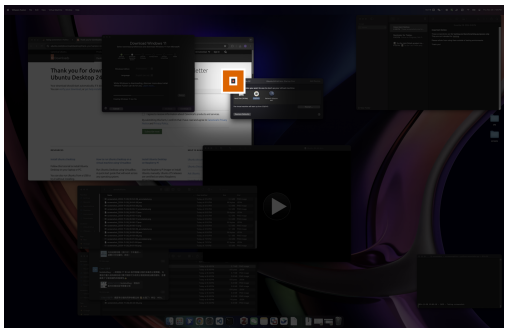
I want to be upfront about what this work does not establish. The RL runs are single-seed, and ScreenSpot base is near-ceiling, so the ScreenSpot deltas are small relative to noise. The SDPO-on-SFT advantage over SFT is about 1.2 SE—directional, not significant. The teacher-gap probe is only validated on two benchmarks, so its sign-prediction is suggestive, not a law. My estimator is single-sample and higher-variance than the full-distribution KL the published methods use, so part of "SDPO underperforms" is measuring my estimator rather than the method itself. The retention battery, while full-N at eval, is at a fairly light training scale. And the closed-form-teacher argument in §6 is interpretation, not something I measured.

The most direct next step, and the one I would run first given more compute, is 3–5 seeds on

[1] GUI-SD's specific teacher-accuracy and entropy figures cited here were read from an extracted-text copy and should be verified against the typeset PDF before being relied upon as exact numbers.



(a) ScreenSpot: target stays legible



(b) ScreenSpot-Pro: target swallowed

Figure 3: Why the teacher gap flips sign. **(a)** On a normal-resolution ScreenSpot screen, dimming the surroundings still leaves the target and enough local context legible, so the spotlight *helps* the teacher (+7.8pp). **(b)** On a dense 4K ScreenSpot-Pro screen, the same spotlight darkens almost the entire display and the tiny target sits in a small lit window that the processor's downsampling then erodes; the teacher loses the very detail it needs and is *worse* than the clean student (−8.8pp). The annotation strength that works at one resolution is destructive at another.

the SDPO-on-SFT refinement specifically: that is the single result whose effect size (+1.96pp) is real enough to matter but small enough that one seed cannot establish it ($p \approx 0.22$), and a handful of seeds would either confirm it as a genuine refinement gain or fold it into the noise. Everything else in the paper is either a clean null or already significant ($p < 0.01$ for the teacher-gap probe), so this is the one open statistical loop.

8 Conclusion

Visual privileged self-distillation did not beat supervised fine-tuning on GUI grounding for me, and I think I understand why: when the privilege is just the label drawn onto the image, the best possible teacher is the label itself, so the method collapses toward SFT—and on hard targets the neural teacher is actually worse than the student, so distilling toward it makes things worse. The piece I would actually reuse is the teacher-gap probe: a cheap, eval-only check whose sign told me, before training, whether privileged distillation would help or hurt. Where I think the method has real room to work is on tasks whose answers are not a closed-form check—code, or multi-step trajectories—where the teacher knows something you cannot get from a single label.

9 Team Contributions

This is a solo project, so all of the work is mine: implementing visual SDPO (the advantage estimator and the teacher-rescoring path in `verl`), the spotlight annotation, every training run (SFT, GRPO, SDPO, SDPO-on-SFT), the held-out evaluation harness, the teacher-gap probe, the retention battery, and the writeup. Relative to my proposal, the scope changed: I set out to show that visual SDPO improves grounding, but once the SFT and GRPO baselines were in place it was clear the method does not beat supervised fine-tuning here. So the project became about characterizing *when* privileged self-distillation helps, and the diagnostic plus the analysis became the real contributions.

References

[1] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas

Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[2] Richard S. Sutton. The bitter lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`, 2019.

[3] Vladimir Vapnik and Akshay Vashist. Learning using privileged information. In *Neural Networks*. 2009. see also Lopez-Paz et al., Unifying distillation and privileged information, ICLR 2016.

[4] Yan Zhang, Daiqing Wu, Huawen Shen, Yu Zhou, and Can Ma. Learn where to click from yourself: On-policy self-distillation for gui grounding. *arXiv preprint arXiv:2605.00642*, 2026.